ソフトウェア工学 114－2
（1997．5．23）

業務プロセスのフォーマルなモデリング"Formaler"の概要

新田稔 nitta@rd.enicom.co.jp
新日鉄情報通信システム(株) 技術部

Formalerは、目的ソフトウェアの必要性や正当性に重点を置いた記述を行なう技法である。発注されたソフトウェアをそのまま開発するのではなく、対象業務に対する提案活動などへの適用を目指している。また、業務用語を、常識的・直観的意味を奪い去った記号として扱うことで、ソフトウェア開発工程を、業務知識が要求される工程とそうではない工程に二分することをねらっている。Formalerの静的モデリングは、ERモデリングに近い表記で、そのような形式的な用語を宣言することができる。

Overview of "Formaler": a Formal Method to Describe Business Processes

NITTA Minoru
NIPPON STEEL Information and Communication Systems Inc.
Systems Technology Department

Formaler is a formal method which emphasizes to describe the necessity and reasonability of the target software. By using Formaler, engineers may propose business process improvements rather than develop ordered applications. Formal descriptions strip business terms of ordinary and intuitive meanings, and make software development process into two parts; one needs engineers have business domain knowledge and another dose not. Such formal terms can be declared by the static modeling of Formaler which is very similar to the ER modeling.

1. はじめに

企業内において情報システムを戦略的に構築するには、すべての業務アプリケーションを開発する前に、本当にそれが必要なものであるのかを再度問うてみる必要がある。業務アプリケーションはなんらかの業務プロセスを支援するソフトウェアであるから、この問いは、その業務プロセスが本当に必要なのか、どのような業務目的に貢献しているのか、を問うことになる。

従来のソフトウェア開発では、ソフトウェアを作るということが大前提であり、そこで用いられる分析や設計技法は、何を作るか(What)やどのように実現するか(How)の記述を目的としていた。

しかし、そのような技法だけを用いていると、なぜそのソフトウェアが必要であるのか(Why)が不明なまま残り、1)業務プロセスに対してソフトウェア技術者は提案ができない、2)目的の明確でないソフトウェアを作ってしまう、3)業務目的の変更に際してソフトウェアの再構築が非常に困難である、などの弊害が起きる恐れがある。

筆者らが提案しているFormalerは、Whyの明確化に重点を置いて業務の目的をフォーマルに記述し、その記述から業務を支援するソフトウェアを導出することを目指した技法である。Formalerの記述によって、業務の目的とソフトウェアを紐付けすることができるので、ソフトウェア技術者による業務改善の提案、本当に必要とされるソフトウェアの開発、容易なソフトウェア再構築などが期待される。

本報告は、Formalerの目的や特徴と、Formalerのモデリングのひとつである静的モデリングによる用語の定義を中心に、Formalerの概要を紹介するものである。

2. Formalerの方針

2.1. フォーマリズムの必要性

業務アプリケーションの開発においては、開発者がその業務分野の知識を持っていることが望まれる。しかし、すべての業務分野の知識を習得することは不可能であり、通常、該当業務の知識が豊富な技術者が上流工程を担当し、下流工程は業務知識の乏しい者に任されることになる。

このような状況において、ビジネス・アプリケーションの開発はフォーマルな技法を必要としている。ここで言う「フォーマル」とは、記述に用いられる用語から一切の常識的・直観的意味を奪

い去り、それらを単なる記号、無定義語として扱うことである。ソフトウェアの仕様書をフォーマルに記述しておけば、業務知識がなくとも、仕様書だけからソフトウェアを作ることができる。

逆にフォーマルな記述ができなければどこまでも用語の意味を引きずることになり、意味の取り違い・思い違いによる誤りが発生したり、用語の意味をいちいちユーザーに尋ねることになったり、業務知識のある技術者をいつまでも配置しておかなければならないという事態になる。

フォーマルな記述は、ソフトウェア開発の工程を、業務知識を駆使する工程と、ソフトウェア構築技術を駆使する工程とに分ける壁になる。とくに、大規模なシステムを多人数で開発する場合など、このフォーマルな壁は非常に有効である。

2.2. 平易な構文と用語

業務の目的をフォーマルに記述するには、その構文と用語を定めなければならない。構文は技法として固定できるが、用語は対象業務によって異なってくる。したがって、記述全体は、用語の宣言と宣言された用語を用いた記述の二段階になる。

フォーマルな記述には上に述べたようなメリットがあるが、一方、ユーザーの理解が困難であるような数学的な表現を用いた仕様書は受け入れ難い。数学的に完全であっても、そこに書かれているものが欲しているソフトウェアであるか否かをユーザー自身が判断できないような仕様書では意味がないからである。

では、どのようにすればユーザーに受け入れられる記述になるだろうか。フォーマルな仕様記述技法としては、Z[1]のように集合論と述語論理によって仕様を記述するものや、OBJ[2]のように多ソート代数系と等式論理を用いて代数的に仕様を記述するものがあるが、Formaler は前者に属する。集合論と述語論理による記述では、すべて言明はなんらかの集合を指し示す。その言明が、数学的な表現ではなく(実際には数学的な表現なのだが)、日本語に近い表現で目的の集合を指すメタファーになっていれば、受け入れられやすいだろう。

情報モデルやデータモデルから自然言語風の語彙への変換は、英語に関しては実用的なものがある。しかし、それらの英単語を和訳しただけでは日本語としてはいささか不自然なところもあり(たとえば[3])、読み易さのためには日本語の語順を考慮する必要があると思われる。

2.3. 業務を支援するソフトウェアの形態

多種多様な形態のソフトウェアを自在に生成するのは不可能であるので、ソフトウェアを自動生成するためには、その形態を定めておく必要がある。そこで、Formaler では、次のようにソフトウェアの形態を定めている。

個々のユーザーは、目的の業務を遂行するために、世界の変化を記憶し、あるいは書きとめておき、その記録から情報を検索したり、数値を集計したり、他のユーザーとデータを交換したりしている。ユーザーが関知すべき世界が小さく単純であれば、これらは人間の手作業だけで行なうことができるが、世界が大きく複雑なものになると、この作業は物理的に人間の能力を超え、コンピューター・システムによる支援が必要になる。

そこで必要とされるソフトウェアは、ユーザーに代わって世界の変化を記録し、ユーザーに現在の状態を教えたり、次になすべき作業を指示するという形で業務を支援するソフトウェアである。

ソフトウェアが記録する情報は、物理的な制約がなければユーザー自身が持っていたはずの情報であり、したがってユーザーは、ソフトウェアへ問い合わせを発するだけではなく、世界の変化をソフトウェアに通知しなければならない。

ソフトウェアからの応答が、世界の変化をすべて記憶できる仮想的なスーパーユーザーの判断と常に一致するなら、そのソフトウェアはユーザーの望むソフトウェアであると言える。

2.4. 従来の技法との連続性

新しい技法が現場に受け入れられるには、既存の手法との連続性が必要であり、部分的にでもすぐに使えるものであることが望ましい。とくに「フ

ォーマル」な技法は敬遠・拒絶されがちであるので、この点を重要視しなければならない。これは技術的な事柄というよりは、社会的あるいは心理的な事柄であり、記述の見た目の親近感や、現在の技法の不備が改善できることをアピールする必要がある。

Formaler の用語の宣言は、RDB のデータモデリングに多用される ER モデリング[4]に近い表記で行われ、見た目の親近感がある。また、従来の ER モデリングには、詳細な制約はコメントで書かなければならなかったり、エンティティーやリレーションシップの名称の意味を知らないと正しく解釈できないなどの不備があったので、データモデルをフォーマルに記述できるということだけでもユーザーや技術者の食指が動くことが期待される。

3. Formaler による記述のあらすじ

Formaler による業務目的の記述は、1)業務の対象世界の構造はこうであり、2)世界の状態は出来事によってこのように変わり、3)この業務の目的は世界の状態をこのようにすることであるから、4)そのためにこのような順で出来事を発生させる、というあらすじになる。これを、静的モデル、動的モデル、機能モデルの三つのモデルで構成する。

3.1. 三つのモデリング

静的モデリングの目的は、業務の対象世界の静的な構造を記述し、動的モデリングや機能モデリングで用いられる用語を宣言することである。静的モデリングでは、対象世界が『エンティティー』で構成されるとみる。ただし、個々のエンティティーは時間とともに発生・消滅を繰り返すので、それらの集まりである『セット』がモデルの構成要素となる。静的モデルは図式で表記されるが、そこで宣言されたセットは、テキスト形式で、動的モデルや機能モデル中で引用される。

動的モデリングの目的は、対象世界の変化の仕方を、変化前後の差異で記述することである。動的モデリングでは、対象世界は、業務のコントロールの下に引き起こすことができる『アクション』と、業務の外部に起因し、その発生を観測することはできるがコントロールすることはできない『イベント』とによって変化するとみる。ただし、いつどのようなイベントやアクションが起きるのかを予測することはできないので、イベントやアクションそのものではなく、それらの『テンプレート』が記述される。

機能モデリングの目的は、Why の記述、すなわち、業務目的とそれに貢献する業務手順を記述することである。業務目的は、エンティティーの増減などによる『オペレーショナル・ポリシー』で示される。また業務手順は、「あるイベントを観測したときにはあるアクションを起こす」ことによってオペレーショナル・ポリシーに貢献する『プロシージャ』で示される。関連するオペレーショナル・ポリシーとプロシージャの組み合わせが『ファンクション』である。

3.2. 業務目的の明示とソフトウェアの導出

機能モデルの記述から、「ユーザーからのイベントの発生の通知を受けると、それに対して必要な処理を行なうとともに、実行すべきアクションをユーザーに指示する」という形式のソフトウェアを導くことがことができる。従来の仕様化や設計の技法が、あるソフトウェアを目標にして、「何を作るか」や「どのように実現するか」を考え、記述するものであるのに対し、Formaler は、最初からなんらかのソフトウェアを目標にするのではなく、対象業務の目的をモデル化していけばその業務を支援するソフトウェアが自然に導かれるという技法である。

したがって、静的モデルには、目的ソフトウェアで管理すべきデータの構造ではなく、業務の担当者が見ている世界の構造が記述される。また動的モデルには、ソフトウェア内部の状態変化ではなく、世界の変化の仕方が記述される。

ソフトウェア内のデータ構造や処理は、ユーザーがどのような問い合わせをソフトウェアに発し、どのような応答を期待するかという記述から導かれる。

3.3. 記述の順序

Formalerの三つのモデルは、静的モデルで宣言されたエンティティーのセットが動的モデルや機能モデルで引用され、動的モデルで宣言されたイベントやアクションが機能モデルで引用される、という関係にある。したがって基本的には

静的モデル → 動的モデル → 機能モデル

の順で記述される。しかしそれは単純なウォーターフォールではなく、動的モデルで必要になった用語を静的モデルに溯って宣言するというようなスパイラルな洗練が行なわれる。

業務目的の記述と業務プロセスの記述は、鶏と卵のようである。なんらかの業務プロセスを仮定しないと、業務の世界観は構成できないので、静的モデルは記述できない。しかし、静的モデルがないと用語が宣言できないので動的モデルや静的モデルは記述できない。したがって、モデルの作成工程は、アプリオリに与えられた業務の目的からすべてを導くという工程ではなく、現状の業務プロセスを記述してその目的を明確にし、その目的のために業務プロセスを最適化していくという工程になる。

このようにして従来は明記されることのなかった業務目的を記述しようとすると、納得のいく目的が書けなかったり、意外なことが業務目的になったりすることもある。

4. 静的モデリング

4.1. エンティティー

静的モデルには、対象世界がエンティティーで構成されるとみなし、その静的な構造が記述される。エンティティーには、独立して存在する『自立エンティティー』と、ひとつまたは複数の『ロール』によって形成される『集約エンティティー』とがある。それぞれのロールはひとつのエンティティーによって担当される(図 1)。

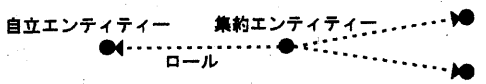

図 1 エンティティーとロール

集約エンティティーが、一個のエンティティーとして、他の集約エンティティーのロールを担当することもある。集約エンティティーは、ロールを担当するエンティティーがひとつでも消滅するとそれ自身も消滅する。

4.2. セット

個々のエンティティーは時間とともに発生・消滅するので、静的な構造を記述するために、同じ性質を持つエンティティーの集まりである『セット』を考え、これをモデルの構成要素とする。

ある時点で見れば、あるセットには、複数のエンティティーが属することもあれば、ただひとつのエンティティーが属することも、またエンティティーがひとつも属さないこともある。しかし、そのセットは常に存在している。

セットには、モデル中で他のセットに依存せずに宣言される『基盤セット』と、他のセットに依存して宣言される『派生セット』がある。しばらく基盤セットのみについて話を進める。

基盤セットは無定義概念であり、どのようなエンティティーの集まりであるのかは、モデル中には示されない。ただし次のルールがある。

(1) すべてのエンティティーは必ずただひとつの基盤セットに属し、発生から消滅まで属する基盤セットは不変
(2) 自立エンティティーと集約エンティティーが混在する基盤セットはない
(3) 同じセットに属するエンティティーは、同じロールを担当できる(エンティティーのセットをそのロールの『ターゲット』と呼ぶ)
(4) 同じ基盤セットに属する集約エンティティーは、それを形成するロールが共通
(5) 集約エンティティーは、それが属するセットと、ロールを担当するすべてのエンティティーが決まれば、ひとつに特定される

4.3. セットとロールの例

例として、「先生が科目を学生に教える」という世界(図 2)を、個々の先生、科目、学生を自立

エンティティーとし、ある先生がある科目をある学生に教えるという個々の事実を集約エンティティーとしてモデル化してみる。

モデルには、すべての先生が属するセット「先生」、すべての科目が属するセット「科目」、すべての学生が属するセット「学生」、ある先生がある科目をある学生に教えるというすべての事実が属するセット「教える」が登場する。

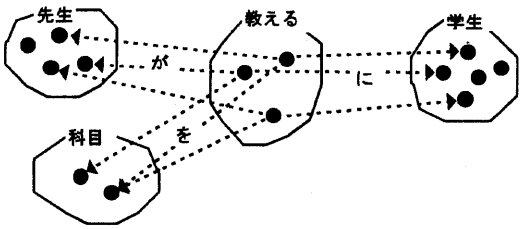

図 2 「先生が科目を学生に教える」

セット「教える」に属する集約エンティティーには三つのエンティティー、セット「先生」に属しロール「(どの先生)が」を担当するエンティティー、セット「科目」に属しロール「(どの科目)を」を担当するエンティティー、および、セット「学生」に属しロール「(どの学生)に」を担当するエンティティー、が参加する。別の言い方をすれば、ロール「が」のターゲットはセット「先生」、ロール「を」のターゲットはセット「科目」、ロール「に」のターゲットはセット「学生」である。

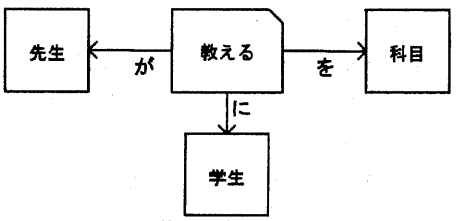

図 3 静的モデルの図式表記

4.4. 図式表現

静的モデルは図式で表記する。自立エンティティーのセットは矩形で表わし、集約エンティティーのセットは右上角が欠けた矩形で表わす。ロールは、集約エンティティーのセットからターゲットへの矢印で表わす。それぞれの図形には、セットやロールの名称を付記する。たとえば、図 2の世界は図 3で表記される。

4.5. ロールの全射と単射

あるロールのターゲットに属するエンティティーは、そのロールを担当する可能性があるが、必ずしもすべてのエンティティーがロールを担当しているわけではない。もし、ターゲットに属するすべてのエンティティーがそのロールを担当するのなら、そのロールは『全射』であると言う。

また、ひとつのエンティティーが複数の集約エンティティーにおいて同じロールを担当する可能性もある。もし、ひとつのエンティティーが複数の集約エンティティーにおいて同じロールを担当することがないのなら、そのロールは『単射』であると言う。全射でありかつ単射であるロールもある。

全射／単射の別は、ロールの矢印の形で示される(図 4)。

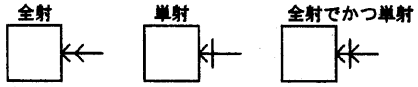

図 4 全射と単射のシンボル

4.6. 部分セット

他のセットに依存せずに宣言される基盤セットに対し、他のセットに依存して宣言されるセットが派生セットである。依存されるセットを、派生セットの『親セット』と呼ぶ。派生セットには、『部分セット』、『結合セット』、『共通セット』などがあるが、ここでは部分セットを紹介する。

部分セットは、親セットに属するエンティティーを、あるロールを担当している/していないで更に分類したエンティティーの集まりである。

部分セットは、親セットの矩形内に部分セットへの分類を示す破線の矩形を置き、その中にそれぞれの部分セットを示す矩形を置いて表す。部分セットへの分類が二通り以上ある場合には、複数の破線の矩形を置く。

エンティティーを部分セットに分類する条件を示すため、同じ分類に属する部分セットのすべてか、あるいはひとつを残して、全射のロールの矢

印を入れる。全射のロールの入っている部分セットが、そのロールを担当しているエンティティーの集まりである(図 5)。

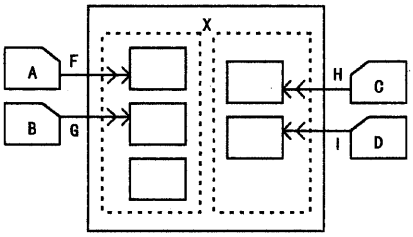

図 5 部分セットの図式表記

4.7. ロールの組み合わせ制約の表現

部分セットでロールの組み合わせの制約が表現できる。たとえば図 5は、「セット X に属するエンティティーはロール F と G とを同時には担当しない」および「ロール H と I とを同時には担当しない」という排他的なロールの組み合わせを表している。なお、部分セットへの分類を分けることで「ロール F や G を担当する」と「ロール H や I 担当する」との独立(直交)が示されている。

また、図 6左は「ロール F とロール G を同時に担当し、どちらか一方だけを担当することはない」というモデルであり、図 6右は「ロール G を担当するときには必ずロール F と同時に担当する(ロール F だけを担当することはあるが、ロール G だけを担当することはない)」というモデルである。

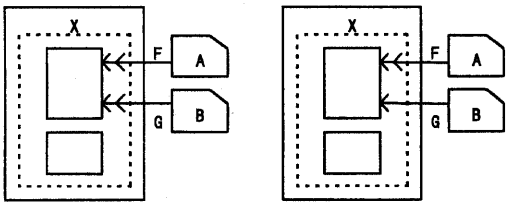

図 6 部分セットによる協働的ロールの表現

5. 用語の引用

5.1. 基本セットの引用

静的モデルに登場する基盤セットおよび派生セットを『基本セット』と呼ぶ。自立エンティティーの基本セットは、形式 1 で、動的モデルや機能モデル中で引用できる。また集約エンティティーの基本セットは、形式 2 で引用できる。

[自立エンティティーのセットの名称]　(形式 1)

ターゲット 1 引用　ロール 1 の名称 ...
ターゲット n の引用　ロール n の名称
< 集約エンティティーのセットの名称 >
(形式 2)

たとえば、静的モデルが図 7であれば、自立エンティティーの基本セットは

[本]　　[利用者]

と引用され、集約エンティティーの基本セットは

[本]を[利用者]が<借りている>
[本]を[利用者]が<予約している>

と引用される。

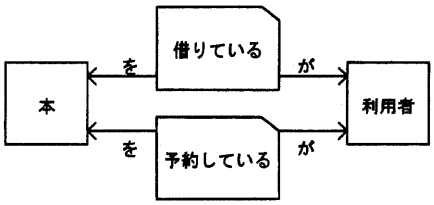

図 7 語彙が引用されるモデルの例

自立エンティティーのセットの名称を名詞に、集約エンティティーのセットの名称を動詞に、ロールの名称を助詞にすることにより、形式 2 の集約エンティティーの基本セットの引用は、日本語の文して読むことができる。

5.2. 導出セットの引用

静的モデルに直接的には登場しないエンティティーの集まりを『導出セット』と呼ぶ。たとえば、あるロールのターゲットに属するエンティティーは、そのロールを担当する可能性はあるが、必ずしもすべてのエンティティーがそのロールを担当しているわけではない。ロールを担当しているエンティティーだけの集まりは、静的モデルには登場しないが、形式 3 で引用することができる。

ターゲット 1 の引用　ロール 1 の名称 ...
ターゲット n の引用　ロール n の名称
< 集約エンティティーのセットの名称 > :
該当のロールのターゲットの引用
(形式 3)

たとえば

[利用者]が<借りている>:[本]

は、セット「本」に属するエンティティーのうち、セット「借りている」のいずれかの集約エンティティーにおいてロール「を」を担当しているエンティティーの集まりを指す(図 8)。

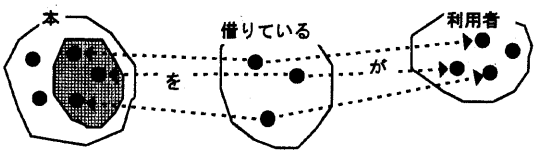

図 8 利用者が借りている本

形式 3 の引用は、日本語のひとつの句として読むことができ、かつ、該当のエンティティーの集まりを示すメタファーになっている。

5.3. 引用中の引用

形式 2 の中で、ターゲットの引用を、そのセットのエンティティーの部分な集まりを示す引用で置き換えることができる。たとえば

[本]を[利用者]が<予約している>

は、集約エンティティーの基盤セット「予約している」の引用であるが、この中のセット「本」の引用を、セット「借りている」のいずれかの集約エンティティーにおいてロール「が」を担当しているエンティティーの集まりの引用

[利用者]が<借りている>:[本]

で置き換えて

[利用者]が<借りている>:[本]
を[利用者]が<予約している>

を得る。

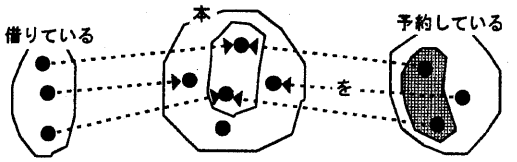

図 9 借りている本を予約している

これは、セット「本」に属するエンティティーのうち、セット「借りている」のいずれかの集約エンティティーにおいてロール「を」を担当しているエンティティーの集まり X を想定し、セット「予約している」の集約エンティティーのうち、X に属するエンティティーによってロール「を」が担当されている集約エンティティーの集まりを指す(図 9)。

また同様に、形式 3 の中のロールを担当しているエンティティーのセットの引用も、そのセットのエンティティーの部分な集まりを示す引用で置き換えることができる。たとえば

[利用者]が<予約している>:
([利用者]が<借りている>:[本])

は、セット「本」に属するエンティティーのうち、セット「借りている」のいずれかの集約エンティティーにおいてロール「を」を担当しているエンティティーの集まり X を想定し、X に属するエンティティーのうち、セット「予約している」のいずれかの集約エンティティーにおいてロール「を」を担当しているエンティティーの集まりを指す(図 10)。

このエンティティーの集まりは、言い換えれば、セット「本」に属するエンティティーのうち、セット「借りている」のいずれかの集約エンティティーにおいてロール「を」を担当し、かつ、セット「予約している」のいずれかの集約エンティティーにおいてロール「を」を担当しているエンティティーの集まりである。

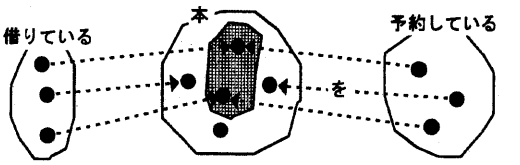

図 10 借りられ、かつ、予約されている本

なお、引用中の語句の結合の順序は左から右であるが、ある部分の結合を優先するため、あるいは結合の範囲を明確にするために、その部分を「()」で囲むことができる。

6. 動的モデリングと機能モデリング

次に、静的モデルで宣言されたエンティティーのセットが動的モデルや機能モデル中でどのように引用されるかを簡単に紹介する。

6.1. 動的モデリング

動的モデルには、対象世界の変化の仕方が、変化前後の差異で記述される。テキスト1に動的モデルの例を示す。

```
event 利用者が本を借りに来た({A}, {B})
      {A} => [利用者]; {B} => <本棚にある>:[本];
{     make {A}が{B}を<借りたい>;
      delete {A}が<本棚にある>
}

action 利用者に本を貸し出す({A}, {B})
   {A} => [本]を<借りたい>[利用者];
   {B} => {A}が<借りたい>[本];
{     delete {A}が{B}を<借りたい>;
      make {A}が{B}を<借りている>;
}
```

テキスト1 本の貸出に関するイベントとアクション

テキスト1の動的モデルでは、イベント「利用者が本を借りに来た」は、「[利用者]」および「<本棚にある>:[本]」に属する二つのエンティティーをパラメターとして持ち({A}と{B})、イベントが観測されたときには、エンティティー「{A}が{B}を<借りたい>」が発生し、エンティティー「{A}が<本棚にある>」が消滅すると述べられている。またアクション「利用者に本を貸し出す」では、エンティティー「{A}が{B}を<借りたい>」が消滅し、エンティティー「{A}が{B}を<借りている>」が発生する。ただし静的モデルは図 11であるとした。

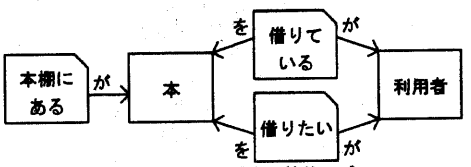

図 11 本の貸出に関する静的モデル

6.2. 機能モデリング

機能モデルには、業務目的とそれに貢献する業務手順が記述される。機能モデルの例をテキスト2に示す。

テキスト2の静的モデルでは、セット「[利用者]が[本]を<借りたい>」に属するエンティティーを減らすことがオペレーショナル・ポリシーであり、イベント「利用者が本を借りに来た」に対してアクション「利用者に本を貸し出す」を行うというプロシージャがそれに貢献すると述べられている。

「[利用者]が[本]を<借りたい>」に属するエンティティーが、イベント「利用者が本を借りに来た」で発生し、アクション「利用者に本を貸し出す」で消滅することから、このプロシージャがオペレーショナル・ポリシーに貢献していることが確認される。またこのモデルから、「利用者が本を借りに来た」が通知されれば「利用者に本を貸し出す」の指示を出すソフトウェアが導かれる。

```
function 貸し出し窓口 {
     ope-policy
          decrease [利用者]が[本]を<借りたい>
     when 利用者が本を借りに来た({A}, {B}) {
          do 利用者に本を貸し出す({A}, {B});
     }
}
```

テキスト2 貸し出し窓口のファンクション

7. おわりに

以上、技法の目的や特徴と静的モデリングによる用語の定義を中心に Formaler の概要を紹介した。Formaler は、現在、静的モデリングと用語の引用の方式がほぼ完了したものの、動的モデリングと機能モデリングについてはまだ作業中のため、Formaler 本来の目的である Why の記述方法については、今回は充分に述べることができなかった。また静的モデリングについても、ER モデリングより Z などのフォーマルな技法に近づく方がよいのではとの考えもあり、並行して作業を進めている。皆様からコメントやご意見を頂ければ幸いである。

参考文献など

[1] B. Potter, J. Sinclair and D. Till, *An Introduction to Formal Specification and Z*, Prentice-Hall, 1991

[2] J. A. Goguen and T. Winker, *Introducing OBJ3*, SRI International, 1988

[3] Composer, テキサス・インスツルメンツ社の CASE

[4] 保鷹良介「データベースシステムとデータモデル」1989、オーム社