ソフトウェア工学 64－6
（１９８９．２．２）

自動プログラミングにおける自然言語仕様からオペレーションを中心とした表現への変換

白井勝利† 山田宏之†† 馬場口登† 手塚慶一†
†大阪大学工学部 ††愛媛大学工学部

自動プログラミングシステムにおいて，ユーザの記述のしやすさ，読みやすさの点から，自然言語仕様を受理できることが望まれている．そのためには，仕様に含まれる意味を形式的な表現に変換する必要がある．そこで我々は，仕様の各文の意味を表現するための枠組みとして，対象領域における基本的な操作を実現する上で必須の情報が直接記述できる，オペレーションフレーム（ＯＦ）を導入する．仕様記述における情報の欠落は，ＯＦにおける必須情報の欠落に対応するので，ＯＦの集合に対して対象領域のオペレーションに関する知識を適用することにより，単語レベル，及び文レベルの欠落情報が容易に補完できる．本稿では，対象領域を待ち行列シミュレーションに設定し，ＯＦが自然言語仕様に含まれる情報の一表現形式として有用であることを明らかにする．

Transformation of Natural Language Specification into Operation-centered Frame in Automatic Programming

†Katsutoshi SHIRAI, ††Hiroyuki YAMADA, †Noboru BABAGUCHI and †Yoshikazu TEZUKA

†Faculty of Engineering, Osaka University, 2-1 Yamadaoka, Suita-shi, 565, JAPAN

††Faculty of Engineering, Ehime University, 3 Bunkyo-cho, Matsuyama-shi, 790, JAPAN

It is desirable to realize an automatic programming system that can deal with natural language specifications in order to improve users' facility. Such a system must transform the information in the specification into the formal representation. For this purpose, we introduce an useful scheme for representation, called an operation-centered frame (OF). An OF represents the prerequisites to perform one basic operation in the domain. Since ellipsis in the specification corresponds to that in the OF, a series of OFs enables us to complement various ellipsis for words or sentences. In this paper, focusing on queueing simulations, we demonstrate that OFs provide the versatile functions in automatic programming.

1．まえがき

ソフトウェアの生産性を向上させる一手段として，自然言語仕様を受理できる自動プログラミングシステムの開発が挙げられる．自然言語による仕様記述には，人間が日常的に行っている思考過程を反映させやすく，また，記述の簡便さ，直感的な理解容易性等，他の手段による表現にはない特徴があり，記述効率が向上する[1]．しかし，これらの特徴は仕様に不完全性を許すことになる．ソフトウェア開発の下流行程において，この不完全性が検出されるとそれまでの作業を無効にすることがあり，開発効率が低下する．したがって，仕様中の不完全性はできるだけ早い段階で検出し，排除する必要がある．

従来から，自然言語仕様中の不完全性を排除する方法として，対象領域固有の知識を利用する方法があるが，対象領域固有の知識の表現形式，及び利用法が明確でない．そこで本研究では，対象領域の知識を“一つの機能を実現するために必要とされる情報”とし，それを表現するための枠組み（記述形式）を考察する．本記述形式により記述された情報には対象領域に現れる基本的操作を実現する上で必須の情報が直接的に記述されるため不完全性のうち情報の欠落に対処できる．

本稿では，具体的な対象領域を待ち行列シミュレーションとし，①自然言語仕様からの自動プログラミングシステムの概要，②対象領域の操作に対する必要情報の記述形式，③仕様中に現れる欠落情報の分類，及び各々に対する補完の方法，④ソフトウェアモデルに関する考察，⑤システムの動作例を述べる．

2．システムの概要[2]

本研究で考察するシステムの構成を図１に示し，各処理モジュールの概略，及び自然言語仕様記述に課した制約を以下に述べる．

2．1　自然言語処理部

自然言語処理部では，動詞を中心とした解析に適した格文法に基づいて要求仕様記述中の各文を処理し，以下の３つの操作を行う．

①文法に従って要求仕様を検査することによる自然言語仕様に含まれる構文的誤りの排除．

②一般的な世界の知識を適用することによる一般的な意味の誤りの排除．

③動詞をフレーム名とする表層的な格フレーム構造（以下，文フレームと呼ぶ）への変換．

システムの構成上は，自然言語処理部とモデル形式化部の変換過程は一つのモジュールとして統合することが可能である．しかし，自然言語処理の研究で得られる新しい成果を自然言語処理部に反映するために，構文上の曖昧性，多義性の解決など自然言語の知識に依存する部分を自然言語処理部とし，対象領域の知識に依存する部分と分離している．

2．2　モデル形式化部

モデル形式化部では，モデル形式化知識ベースにある対象領域におけるオペレーションに関して体系化された知識を利用して以下の３つの操作を行う．

変換　：ソフトウェアの仕様に対する表層的な記述である文フレームの集合から，個々の文フレームに対応した対象領域に依存する意味を表すフレーム（オペレーションフレーム，以下ＯＦと略記）の集合に変換する．

補完　：ＯＦの集合内の欠落情報を補完し，完全なＯＦの集合にする．

構造化：ＯＦの集合を形式モデルに変換する．

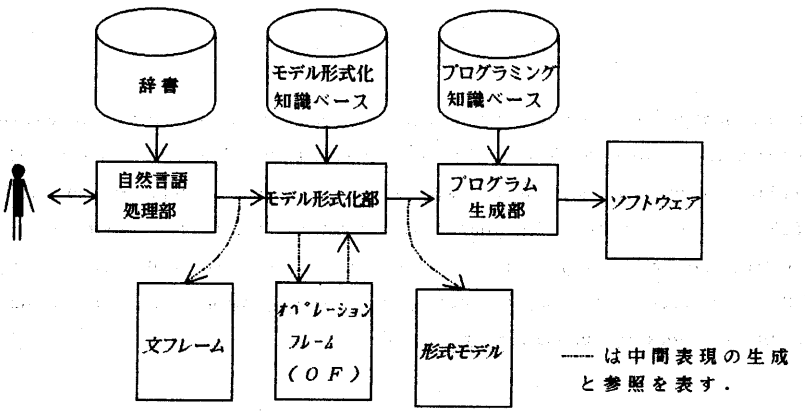

図１　システム構成

検証　：作成された形式モデルをユーザに提示し，ユーザの意図したプログラムモデルと一致するかどうかを検証する．

2．3　プログラム生成部

プログラミング知識ベースに含まれる目標言語の構文や意味に関する知識を用いて抽象的な形式モデルを詳細化し，具体的なソースコードに変換する．

2．4　自然言語仕様記述に課した制限

本システムで受理する自然言語仕様に対して課した制約は次のとおりである．

言語：日本語

構文：単文，条件文

3．オペレーションフレーム

自然言語仕様からソフトウェアを作成するには，自然言語仕様に含まれる意味をシステムが処理できるように表現することが重要である．そこでユーザが仕様記述で，「どのようなことを行いたいのか．」といった機能的な操作を中心に記述することに着目して，仕様記述の意味を対象領域におけるオペレーションを中心とした表現に変換する．この表現形式をＯＦと呼ぶ．

ＯＦは図２で示すようにプリミティブタスク（以下ＰＴと略記），格役割，格要素で構成される格フレームである[3]．また，格要素となる単語が持つべき属性としてケースプロパティ（以下ＣＰと略記）がある．

以下，各構成要素を説明し，文献４）の待ち行列シミュレーションの１４個の例題に対する仕様記述に基づいた設定を示す．

なお，条件付き選択や，条件付き反復を表す条件文，また，その条件部で用いる数値比較の演算文も同じＯＦの枠組みで扱う．

3．1　プリミティブタスク

同一の内容もしくは処理について自然言語で表現するとき多様な表現が可能となり，意味理解をきわめて困難にしている．この困難さを取り除くには自然言語での多様な表現をシステム内の一意の表現に変換しなければならない．そこで対象領域における処理を実現するために必要とされる基本的操作を設定し，対象領域の用語を用いて記述したものをＰＴと呼ぶ．

待ち行列シミュレーションのＰＴを表１に示す．

```
(ＰＴ
    (格役割1  格要素1)  ------  ＣＰ1
    (格役割2  格要素2)  ------  ＣＰ2
        :         :          :
    (格役割n  格要素n))  ------  ＣＰn
```

図２　ＯＦの構造

表１　ＰＴ

ＰＴ	説明
施設のタスク	
start_at	開始する
end_at	終了する
初期リソースの定義	
init_exist	初期状態で存在する
客のタスク	
客の到着定義	
arrive	到着する
リソース無関係のタスク	
exit	行うタスクがなくなり退去する
waste_time	時間を消費する
リソース関係のタスク	
ask_resource	リソースの有無を調べる
get_resource	リソースを獲得する
rel_resource	獲得したリソースを解放する
make_resource	リソースを生成する
条件によるタスクの制御	
if	条件付き選択を行う
while	条件付き繰り返しを行う
=,<,<=,>,>=,¯=	数値比較の演算を行う

3．2　格役割

ＯＦは「どのような行為主体がどのような規範に基づいてＰＴで表されるオペレーションを行うのか．」を表現する．したがって，それぞれの単語がＰＴに対してどのような役割をもって結合しているのかを表すものが格役割である．

シミュレーションにおいて重要な役割をなす時間，リソース，数量等に重点をおいて設定した，それぞれのＰＴが取り得る格役割を表２に示す．

3．3　ケースプロパティ

格要素となる単語とＰＴとの意味的な整合性を検査するために，格要素の満たすべき意味的な制約条件としてＣＰを設定する．

シミュレーションの仕様記述に現れる単語の意味的な制約条件を検討した結果に基づき設定したＣＰを表３に示す．

4．欠落情報の補完

自然言語仕様記述では単語レベルや文レベルの情報の欠落が生じる．これはＯＦにおいては格要素の欠落や，ＯＦ自体の欠落となって現れる．この結果，ユーザの意図に合致したソフトウェアが作成できない．そこで本システムでは，ＯＦの情報をもとにして省略要素（格要素，文）を検出し，対象領域に関する体系化された知識を適用してその要素を補完する．

本研究では，文献４）の例題に対する自然言語仕様記述を対象として，仕様中で起こり得る省略について考察し，その補完方法に応じて４種に分類した．なお，各分類の命名については文献５）を参考にした．

ここでは各省略の型について，省略の背景（ユーザの意図），省略の例，補完に用いる知識，補完法を述べる．さらに，各型の適用戦略を述べる．

表２　ＰＴに対する格役割

格役割	説明
agent	行為者
time	時刻
duration	時間
place	場所
quantity	数量
resource_name	対象リソース
priority	優先権
condition	条件
then	成立時のタスク
else	不成立時のタスク

表３　ＣＰの例

ＣＰ	説明
resource_user	リソース使用者
number	数
resource	リソース
time_a	時刻（一回）
time_i	時刻（反復）
of	ＯＦ

4．1　単語の省略補完

4．1．1　連想型（Ⅰ，Ⅱ型）

ある文中の単語から，その文中の他の単語が容易に連想される場合には，ユーザはその連想される単語を省略することがある．

＞補完法：

Ⅰ型：格要素としてＯＦに存在している単語から連想される種々の情報をその語の付属情報から取り出して，欠落情報を埋める．

例１：「客は３０分の指数分布で到着する．」
→“指数分布”に対し３０分は平均である．
→「客は平均３０分の指数分布で到着する．」と補完する．

Ⅱ型：欠落している格要素のＣＰをユーザに提示し，その属性を持つ単語を入力してもらうことで補完する．

4．1．2　文脈型（Ⅰ，Ⅱ，Ⅲ型）

既に記述した情報であり，かつ文脈を参照することによりその情報が復元できるときには，ユーザは冗長な繰り返しを避けるために，その情報を省略することがある．

＞補完法：

省略を含むＯＦを処理する以前のＯＦを参照し，欠落情報を補完する．

Ⅰ型：agent省略の場合→前文のagentが候補となる
ただし，ＰＴがifまたはwhileの場合のcondition部のagent省略の場合は，then部またはelse部のagentが候補となる．

Ⅱ型：ＰＴ省略の場合→前文のＰＴが候補となる

例２：①と②が連続して入力されるとする．
①「客１が来る．」
→(arrive (agent 客１) …)
②「間隔は平均３０分の指数分布である．」
→(PT? (time_i exp …)…)

省略されたPT?を arriveと補完し，２つのＯＦを合成する．

Ⅲ型：agent以外の格要素の省略の場合→

欠落している格要素のＣＰを属性として持つような単語を文脈中に探索し候補として提示する．

これは連想型Ⅱ型において，ＣＰにより指定された属性を持つ単語が文脈中に存在する場合である．

４．１．３　デフォルト型

対象世界で常識的な数値，あるいは特に注意を払わない数値に対してはユーザは省略することがある．

＞補完法：

欠落情報をあらかじめシステムで有するデフォルト情報により補完する．

例３：リソースの有無の問い合わせ(ask_resource)の場合

ＰＴがask_resourceであるＯＦにおいて

quantityの記述が省略されていれば

→ユーザは「とりあえずその資源があるかどうかを知りたい．」のだと判断する．

→quantity＝1と補完する．

例４：リソース要求の優先度の場合

リソースの要求においてpriorityは省略される可能性があるが，その場合ユーザは「特に優先度を問題としていない．」と解釈してpriorityを最も優先度の低い０にする．

４．２　文の省略補完

４．２．１　関係利用型

客が一連のタスクによりある機能を達成する場合，ユーザは中心的なタスクを記述し，それに付随するタスクを省略する場合がある．そのため仕様記述において，付随的なタスクを記述する文が抜けてしまうことになり，モデル形式化部では省略された文に相当するＯＦを補完することが必要となる．

＞補完法：

ＰＴ間の順序関係を対象領域の知識として規定しておき，この順序関係を用いて省略されたＯＦを補完する．

例５：客のトップレベルの行動

［arrive→タスク群→exit］

一般にシミュレーションにおいて客は到着後に一連のタスクを行い，その後退去するが，退去に関する記述を忘れることがある．そこで，客が退去するかどうかをユーザに問い合わせ，退去する場合にはexitをＰＴとするＯＦを補完する．

例６：タスク群の順序関係

リソースの獲得に関するＰＴの組合せには，

① ［ask_resource→get_resource→waste_time→rel_resource］

② ［get_resource→waste_time→rel_resource］

③ ［get_resource］

のように３つのパターンがあるが①，②においてユーザはリソース獲得の中心となるget_resourceについては記述するが，付随的なタスクであるask_resourceについては省略する場合がある．そこでask_resourceをタスクとして持つのか否かを問い合わせ，①，②のどちらのパターンであるかを決定する．

４．３　補完の戦略

上述の各補完方法の適用順序を効率の面から検討し，以下のような順序とする．

①連想型Ⅰ型を用いる．

②デフォルト型を利用して，数値的な欠落情報を補完する．

③文脈型Ⅱ型を利用して，２つのＯＦの合成を行うことで省略を補完する．

④文脈型Ⅰ型を利用して，省略された格要素のうちagentを補完する．

⑤文脈型Ⅲ型を利用して，省略された格要素のうちagent以外の省略を補完する．

⑥⑤のＣＰを持つ単語を文脈内に検索できなかった時は連想型Ⅱ型を利用してユーザに問い合わせて補完する．

⑦関係利用型で候補とされるＯＦを提示し，ユーザがＯＦを必要とした時には上述の①～⑥を繰り返してそのＯＦを埋める．

この方法により，欠落情報が補完され，モデル形式化部の次の操作であるモデルへの構造化に進む．

５．形式モデル

形式モデルはユーザが記述した自然言語仕様に含まれる情報をソフトウェアのモデルとして構造化したものである．

5．1　オブジェクト指向のモデリング

オブジェクトとは，対象とその対象を操作する手続きを一まとめにした概念単位である．オブジェクト・モデルとは，計算すべき問題領域に存在する対象やその問題を説明するのに必要な概念を，それぞれオブジェクトおよび，それらのオブジェクトが解釈しうるメッセージとしてモデル化したものである．

このオブジェクト・モデルの特徴を以下に挙げる．

・対象領域をシミュレーションとした場合，さまざまな“もの”がシミュレーションの世界のオブジェクトとして素直に表されるのでモデル構築性に優れている[6]．

・ある一つの個体に関する情報をまとめて記述するので可読性の点で優れている．

・目標言語がオブジェクト指向型言語の場合には，制御文，変数を付加することにより容易にソフトウェア化が可能であり，移行性が高い．

以上のことから，本研究において形式モデルにはオブジェクトモデルを採用する．

5．2　形式モデルの記述形式

形式モデルはフレームの集合で表され，個々のフレームは次のような記述形式をとる．

```
(object_name
    (condition_tag_1   task_1)
                :
                :
    (condition_tag_n   task_n))
```

object_nameには客の名前または施設名を記述する．condition_tagにはif,then,else,endif,while,wend,null（空白）のうちいずれかを記述する．task には客または施設の振舞いをＯＦの形式またはその agentを省略した形式で記述する．

object_name を見ることでシステムがどのような要素で構成されているかを理解でき，conditon_tagにより，条件によるタスクの制御が読み取れる．

また，object_name が施設名であるフレームは，施設の初期状態や施設に対する客の到着（施設が呼を生起したものと見なす）を表現する．

object_name がその他であるフレームは客のタスクを表現する．

5．3　ＯＦから形式モデルへの構造化

複数のＯＦの情報をオブジェクト毎（ここでは agentになる名詞毎）にまとめる．その際の手続きを次に示す．

それぞれのＯＦについて，ＰＴが，

①start_at,end_atであるとき，agentをobject_nameとするフレームのtaskに，そのＯＦのagentを除いて記述する．

②arriveであるとき，placeをobject_nameとするフレームのtaskに，そのＯＦの placeを除いて記述する．

③ifまたはwhileであるとき，格役割thenのＯＦ中のagentをobject_nameとするフレームに対して，condition_tagがifであるtaskにconditionのＯＦを，thenであるtaskにthenのＯＦを，elseであるtaskにelseのＯＦをそれぞれ記述する．

④その他の時，agentをobject_nameとするフレームのtaskに，そのＯＦのagentを除いて記述する．

上述の手続きのうち，agentをobject_nameとするフレームが形式モデルに存在しなければそのフレーム（ただし，空きフレーム）を作成してから，手続きを継続する．

例として，客１が平均１２分の指数分布で店に到着し，その後のタスクの一部が「もし本が３冊あるならば，客は２冊買う，さもなければ，１５～２０分の一様分布で待つ．」の場合の形式モデルを図３に挙げる．

```
(店
    :
  (arrive   (agent 客１)
    (time_i (exponential (mean 12)))
    :
)

(客１
    :
  (if
     (ask_resource
        (quantity 3)
        (resource_name 本)))
  (then
     (get_resource
        (quantity 2)
        (resource_name 本)))
  (else
     (waste_time
        (duration
           (uniform (from 15) (to 20))))
  (endif)
    :
)
```

図３　形式モデルの例

6．具体例

本システムの動作を具体的な例で説明するため，ハンバーガショップに来店する３タイプの客と納入業者の振舞いをシミュレートするための自然言語仕様記述を形式モデルに変換するまでの過程を考える．ここでは，図４において，客３の振舞いに関する記述である①，②，③，④の文がどのように変換されていくかを中心に説明する．

6．1　仕様から文フレームへの変換

まず格文法に基づく解析により⑤，⑥，⑦，⑧のように動詞をフレーム名とする文フレームに変換する．

6．2　文フレームからＯＦへの変換

次に，モデル形式化知識ベースに含まれる，動詞をＰＴに変換する規則（例：来る→arrive），文フレームの格役割をＯＦの格役割に変換する規則（例：主格→agent,時刻→time）の適用，並びに時間単位の変換（ここでは分を基本単位と仮定してので，時間・秒から分への変換）により⑨，⑩，⑪，⑫になる．

6．3　ＯＦの欠落情報補完

ＯＦの格要素で??で表されるのが欠落している情報である．⑨において"??facility"は placeの格要素としてfacilityを属性としてもつものが要求されることを表している．これに対し文脈型Ⅱ型を用い，既に処理したＯＦの格要素を検索し，“店”がfacilityを属性として割り当てられているので“店”を補完する．⑩で時刻を表す表現のtime_dが欠けているが，連想型Ｉ型により一様分布を表す "uniform"を補完する．また⑩，⑪，⑫で agentの格要素が抜けているが，文脈型Ｉ型を用いて⑨で用いられている“客３”を順次補完する．⑪でリソースの個数を表す??numberが抜けているがデフォルト型により１を補完する．

6．4　ＯＦ自体の補完

客３のタスクにおいて退出を表すタスクが見られない．そこで関係利用型によりユーザに問い合わせ，退出のタスクを表すＯＦを⑬のように補完する．

6．5　ＯＦの集合から形式モデルへの変換

客３のＯＦのうち，到着を表すＰＴつまりarriveをフレーム名とするＯＦ⑨を⑭のように形式モデルの施設を表すフレームに記述し，その他のＯＦすなわち，⑩，⑪，⑫，⑬を客３がフレーム名であるフレームに⑮，⑯，⑰，⑱のように記述する．

以上の操作により自然言語仕様は形式モデルに構造化される．

7．まとめ

本稿では，自動プログラミングシステムにおける仕様に含まれる情報の表現形式として，対象領域でのオペレーションを中心とした表現（ＯＦ）を提案した．つぎに，ＯＦの集合に現れる単語レベル，文レベルの欠落情報を分類し，各分類に対する補完法を述べた．また，ＯＦからソフトウェアモデルに構造化する際にオブジェクト中心に構造化することによりモデルの可読性・移行性が向上することを示唆した．

今後の課題として，今回提案した補完法の評価，さらに複雑な自然言語仕様の受理が挙げられる．

参考文献

1)辻井，上原：“ソフトウェア工学と自然言語処理”，情報処理，Vol.28,No.7,pp.913-921(1987).

2)白井，山田，手塚：“待ち行列シミュレーションモデル作成のための自然言語インタフェース”，第36回情全大，6N-3(1988).

3)白井，山田，馬場口，手塚：“自然言語仕様からモデル表現への変換－シミュレーションソフトウェアを対象とした場合－”,第37回情全大，2J-1(1988).

4)Goldberg, A. and Robinson, D.:"Smalltalk-80 The Language and Its Implementation", Addison Wesley,pp.439-537 (1983).

5)唐沢，松永，小川，田村：“省略表現を含む不完全テキストを知識を用いて復元する英文補完システム”，情学論，Vol.26,No.2,pp.374-381(1985).

6)山本：“オブジェクト指向とシミュレーション”，情報処理，Vol.29,No.7,pp.374-381(1988).

自然言語仕様記述

店は午前７時に開店する．
午後１０時４５分に閉店する．
客１は平均４分の指数分布で来店する．
１０～２５秒でメニューを見る．
ハンバーガを２個買う．
５～１６分かけて食べる．
客２は平均１５分の指数分布で店にくる．
もし，ハンバーガが３個あるならば，３個買う，
さもなければ，店を出る．
客３は午前１２時から午後１時まで
平均２分の指数分布で店にくる．…①
１０～２５秒メニューを見る．…②
ハンバーガを買う．…③
５分～２５分かけて食べる．…④
納入業者は平均２時間の指数分布で店にくる．
ハンバーガを２００～３００個納入する．

文フレーム

```
(開店する
   (主格  店)
   (時刻  午前7時))
(閉店する
   (時刻  午後１０時４５分))
(来店する
   (主格  客１)
   (時刻  (平均４分の指数分布))
          :
          :
(来る
   (主格  客３)
   (時刻  (午前１２時から午後１時まで
           平均２分の指数分布))                …⑤
(見る
   (時間  １０～２５秒)
   (対象  メニュー))                          …⑥
(買う
   (対象  ハンバーガ))                        …⑦
(食べる
   (時間  ５～２５分))                        …⑧
          :
          :
(納入する
   (対象  ハンバーガ)
   (個数  ２００～３００))
```

ＯＦ

```
(start_at
       (agent 店)
       (time 7*hour))
(end_at
       (agent ??facility)
       (time 22*hour+45))
(arrive
       (agent 客１)
       (time (exponential (mean 4)))
       (place  ??facility))
                :
                :
(arrive
       (agent 客３)
       (time (exponential
              (mean 2)(starting 12*hour)
              (ending 13*hour))
       (place ??facility))                     …⑨
(waste_time
       (agent ??resource_user,
              ??resource_producer)
       (time (??time_d
              (from 10*second)
              (to 25*second))))                …⑩
(get_resource
       (agent ??resource_user)
       (quantity ??number)
       (resource_name ハンバーガ))             …⑪
(waste_time
       (agent ??resource_user,
              ??resource_producer)
       (duration (??time_d (from 5) (to 25))))  …⑫
                :
                :
(make_resource
       (agent ??resource_producer)
       (quantity
         (discrete_uniform
           (from 200)
           (to 300))
       (resource_name ハンバーガ))

------- 関係利用型により補完したＯＦ ---------
(exit
       (agent 客１))
(exit
       (agent 客３))                           …⑬
(exit
       (agent 納入業者))
```

形式モデル

```
(店
    (start_at
          (time 7*hour))
    (end_at
          (time 22*hour+45))
    (arrive
          (agent 客１)
          (time (exponential (mean 4))))
    (arrive
          (agent 客２)
          (time (exponential (mean 15)))))
    (arrive
          (agent 客３)
          (time (exponential
                 (mean 2)(starting 12*hour)
                 (ending 13*hour)))            …⑭
    (arrive
          (agent 納入業者))
          (time (exponential
                 (mean 2*hour))))
(客１
    (waste_time
          (duration (uniform
                     (form 10*second)
                     (to 25*second))))
    (get_resource
          (quantity 2)
          (resource_name ハンバーガ))
    (waste_time
          (duration (uniform (from 5) (to 16))))
    (exit))
(客２
    (if
      (ask_resource
          (quantity 3)
          (resource_name ハンバーガ))
    (then
      (get_resource
          (quantity 3)
          (resource_name ハンバーガ))))
    (else  (exit))
    (endif))
(客３
    (waste_time
          (time (uniform
                 (from 10*second)
                 (to 25*second))))             …⑮
    (get_resource
          (quantity 1)
          (resource_name ハンバーガ))           …⑯
    (waste_time
          (duration (uniform (from 5) (to 25))))…⑰
    (exit))                                     …⑱
(納入業者
    (make_resource
          (quantity
             (discrete_uniform
             (from 200)
             (to 300))
          (resource_name ハンバーガ))
    (exit))
```

図４　具体例