Vol.2020-IFAT-138 No.5
Vol.2020-DC-116 No.5
2020/3/27

オブジェクト指向プログラミングによる意味的クラス継承に関する考察

～造り酒屋オントロジモデルの検討から得られた可能性と限界～

大野 邦夫†

日本酒醸造における発酵管理を支援するセンサーシステムであるKojimoriのデータの意味的な管理を目指して、一連の業務を分析し、UMLによるクラス図に基づき、CLOS（Common Lisp Object System）によるシミュレーションシステムを実現したが、次の段階としてC、C++による実用的な実装手法を検討した。先ずクラス図の把握しやすさを目的として、視認的な表示法を洗練すると共に、クラス図の継承を表形式で管理する手法の検討を行った。さらにクラス階層を、分類型と統合型に区別して管理する手法を考案した。その結果、クラス図を要素還元して要素部材として中間的なクラスを生成管理し、その要素部材クラスを統合してシステム化する手法を実現した。さらにこの手法に基づき、クラス図を実装と分離し、意味的な構成を把握するための参照手段として活用する手法を検討すると共に、意味的なクラス継承の適用限界を考察した。

キーワード　クラス階層, CLOS, 多重継承, オブジェクト分析設計, オントロジ, 日本酒醸造, センサーシステム

A Study on Semantic Class Inheritance by Object Oriented Programing

～ Posibilities and Limitations through Sake Factory Ontology Model ～

Kunio OHNO†

Semantic management of Kojimori which is a sensor system for fermentation management in sake brewing, has been analyzed and a CLOS system has been developed based on class inheritance diagram. For the next stage, we are planing to consider C and C ++ system as practical implementation. First, refinement of class diagram has been conducted in order to understand the class structure easily. Then a method of dividing class hierarchies into analytical classifying type and synthetic designed type. Due to the analytical classification, systematize element classes should be created,and should contribute to organize synthetic classes. Finally, based on this method, we have separated class diagram concept from the system implementation, and considered it as a reference tool for semantic understanding.

Keywords class architecture, CLOS, multiple inheritance, object analysis & design, ontology, sake brewery, sensor system,

1. はじめに

オブジェクト分析設計手法のUMLを用いて、日本酒製造プロセスをモデル化する手法に関して以前報告したが[1][2]、本報告ではその後の検討経緯を紹介すると共に、その経験から得られたクラス構築手法に関する考察を述べる。プロトタイピング向けのモデル記述言語のCLOS（Common Lisp Object System）から具体的な実装言語であるCやC++に移行させるには、両言語に精通したプログラマを必要とするがこれは大変なコストを生じることになる。そのための工夫として、クラス図をテンプレート化し、それを系統的に表形式で管理する手法を考案した。従来プログラムにおける変数名、関数名、クラス名、メソッド名は、アルファベットで記述するのが一般的であったが、日本語で記述することにより認知・把握能力が向上する。国内におけるアプリケーションであればその方が便利である。

さらにCやC++においては、大量のクラス群で系統的に管理するよりは、データ管理に特化したデータベースを用いる方が実用的である。オブジェクト指向データベースが使用実績を重ねていればこのような状況に適合し使用されたと思われるが、演算定義が明確なRDBの進化には敵わなかった。

そのような検討を通じて判明したのは、クラス継承を、分類・分析を通じた要素還元的に行う手法と、具体的な設計・実装を通じた統合型に区分すべきであるという教訓である。

† (株) モナビITコンサルティング
Monavis IT Consulting Co. LTD.

本稿ではその経緯を日本酒製造プロセスの具体的なクラス継承を例にして紹介する。

2. 酒造プロセスのモデル化

2.1 ITによる支援とモデルの設定

酒造事業をITが支援する場合、このプロセスに関わる関係者を支援することが要求される。従来の勘と経験に基づく知識や技能を論理化・定量化することが期待される。そこで、今回は公益財団法人・日本醸造協会による「最新酒造講本」のデータを参照モデルとして、先に検討したモデル[2]の詳細化を試みた。図1は、先の検討で作成したアクティビティ図であるが、日本酒製造における蒸米から清酒の生成までの経過を示している。図の楕円はオブジェクトを示し、矩形は処理を示す。標準的なUMLのアクティビティ図は、始点、終点の明記、方向の矢印などが規定されているが、この図ではそれらを簡略化している。

清酒は消費者ニーズ、販売見込み数量に従い、計画を立てて合理的に製造される。その手法は、「最新酒造講本」における製造計画に記されているが、そのコアとなる作業は、蒸米からもろみの製造までの作業である。

「最新酒造講本」に日仕舞いのもろみ製造（1日1本のもろみを仕込む）の場合の仕込み配合の事例が具体的な数値と製造のスケジュールに関して詳細に記されている（同18～19ページ）。それに基づく具体化的なモデルについて検討した。表1と表2に仕込の際の蒸米、麹米（麹）、汲み水、醸造アル

Vol.2020-IFAT-138 No.5
Vol.2020-DC-116 No.5
2020/3/27

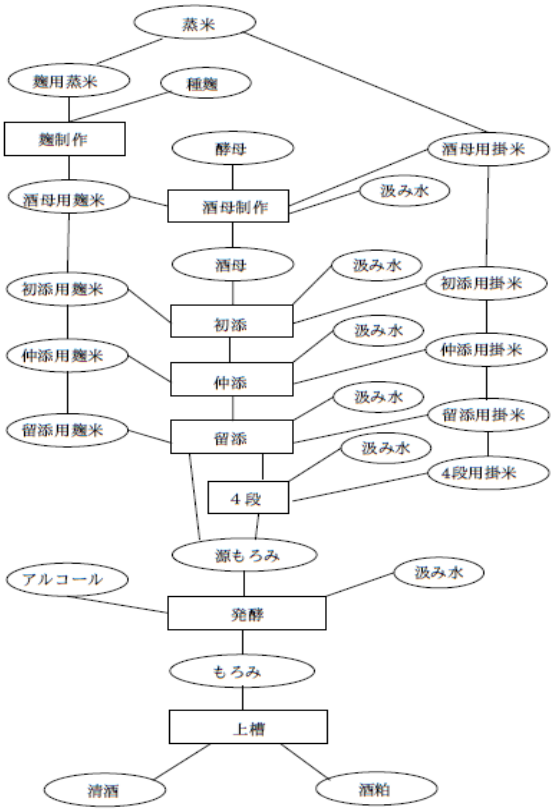

図1　日本酒醸造プロセスんおアクティビティー図

コールの配合を示す。表1が普通酒で表2が増醸酒である。普

表1　普通酒の仕込み配合

普通酒	酒母	初添	仲添	留添	4段	合計	備考
総米	140	300	600	960	200	2200	kg
蒸米	100	200	480	780	200	1760	kg
麹米	40	100	120	180		440	kg
くみ水	160	300	740	1400	300	2900	リッター
アルコール						1100	リッター(30%)

表2　増醸酒の仕込み配合

増醸酒	酒母	初添	仲添	留添	4段	合計	備考
総米	140	300	600	960		2000	kg
蒸米	100	200	480	780		1560	kg
麹米	40	100	120	180		440	kg
くみ水	160	300	740	1400		2600	リッター
アルコール						6000	リッター(24%)

通種は4段仕込を行うのに対し、増醸酒はそれを行わない。増醸酒は普通酒に対して大量のアルコールを用いて増量することに特徴がある。

2.2 蒸米からもろみまでの詳細プロセス

表1・表2の事例はこのフローの典型例と考えることができる。表3は、表1・表2の配合事例の実施計画である。この計画に添ってアクティビティ図を詳細に説明する。

2.2.1 製造ロットの製造履歴

9月1日に、「イ1」記号のロットが記されている。これは酒母の麹のための蒸米40kgが製造され、麹室に搬入され麹として発酵され始めたことを示す。その際に、発酵の素となる種麹が使用される。その後の、「イ1」記号のロットを追跡してみよう。

9月4日に、100kgの掛米が記されているが、これは酒母生成のための蒸米が、酒母用の容器に加えられたことを示す。その時に9月1日から麹室で3日間で発酵生成された麹も汲み水160リッターと共に酒母用の容器に加えられる（表1参照）。その際に酵母も加えられる。その後、9月15日まで、酒母室の容器中ででんぷんが糖化発酵する状況に置かれる。

2.2.2 製造ロットの仕込プロセス

9月15日に出来上がった酒母は、もろみ用の容器に移され、初添、仲添、留添の３段仕込みのプロセスに入る。なお３段仕込みは、掛麹と掛米の両者について行われる。掛麹については、9月15日に初添、1日おいて17日に仲添、翌日の18日に留添が行われ、その同日が掛米の初添になる。掛米の仲添は1日おいて20日、留添は翌日の21になる。この2段階の3段仕込みで、汲み水も用意されており、初添、仲添、留添について各々、300、740、1400リッター使用される。

2.2.3 仕上げのプロセス

その後、もろみは糖化とアルコール化の並行発酵が行われるが、10月4日に4段仕込みとして掛米200kgと汲み水300リッターが追加される。さらに10月10日には、醸造アルコールが1100リッター追加されもろみの発酵が停止される。その3日後に上槽が行われ、目指す清酒が誕生する。

2.2.4 別のロットの履歴

以上は「イ1」記号のロットの時系列の上方であるが、「イ2」、「イ3」、「イ4」は、１日、２日、3日後に「イ1」の経過を追随してロットとして製造されている。その次の「ロ5」は表1の普通酒ではなく表2の増醸酒であり、プロセスが4段以降の最後の場面で異なっている。増醸酒では、4段の仕込みは無いが、発酵停止の際の醸造アルコールの量が6000リッターとなっている。

2.3 クラス図

図2に、図1のアクティビティー図に基づくクラス図を示す。この場合は、継承関係のみを記述しているので、クラスはその名称が記された単なる矩形の箱で記述されている。クラス図の基本的な役割は、継承関係を明確化することにある。そのためにはクラス名を書いた箱が線や矢印で結ばれたマップとして示されていることが重要であり、システムの基本的な構造を示す情報となっている。

一般にクラス図では、上が抽象的なクラスで下が具体的なクラスである。UMLのクラス図は、継承関係を矢印で記すように規定されているが、矢印は一般的には上を向いている。従って矢印は無くても直感的に継承関係は把握できる。この図では矢印を省略し、スーパークラスが継承関係を表す線の上方としている。

クラス図の記法としては、図2のように上下関係に基づき矢印を省略する方法以外に、図3のようにクラスを縦横に並べて整列させて関係付ける方法も考えられる。クラス階層は集合論的な概念と経験的意味概念の融合の産物であり多少の個人的主観は包含される。クラス図は個人が把握しやすいように記述するのが重要なノーハウである。

クラスの詳細を記述する場合は、図4に示すようにクラス名称の下に2つの欄が追加される。それらは、クラスの性質を記述するインスタンス変数の欄とメソッドの名称の欄であり、これらの情報を用いてプログラムが作成される。

Vol.2020-IFAT-138 No.5
Vol.2020-DC-116 No.5
2020/3/27

表3　日本酒製造の実施計画サンプル

	酒母用		もろみ																				
			掛麹用								掛米用												
月日	記号	麹	記号	掛米	記号	初添	記号	仲添	記号	留添	記号	初添	記号	仲添	記号	留添	記号	4段	白米総量	記号	アルコール	上槽	検定
9月1日	イ1	40																	40				
9月2日	イ2	40																	40				
9月3日	イ3	40																	40				
9月4日	イ4	40	イ1	100															140				
9月5日	□5	40	イ2	100															140				
9月6日	イ6	40	イ3	100															140				
9月7日	イ7	40	イ4	100															140				
9月8日	イ8	40	□5	100															140				
9月9日	イ9	40	イ6	100															140				
9月10日	□10	40	イ7	100															140				
9月11日	イ11	40	イ8	100															140				
9月12日	イ12	40	イ9	100															140				
9月13日	イ13	40	□10	100															140				
9月14日	イ14	40	イ11	100															140				
9月15日	□15	40	イ12	100	イ1	100													240				
9月16日	イ16	40	イ13	100	イ2	100													240				
9月17日	イ17	40	イ14	100	イ3	100	イ1	120											360				
9月18日	イ18	40	□15	100	イ4	100	イ2	120	イ1	180	イ1	200							740				
9月19日	イ19	40	イ16	100	□5	100	イ3	120	イ2	180	イ2	200							740				
9月20日	□20	40	イ17	100	イ6	100	イ4	120	イ3	180	イ3	200	イ1	480					1220				
9月21日	イ21	40	イ18	100	イ7	100	□5	120	イ4	180	イ4	200	イ2	480	イ1	780			2000				
9月22日	イ22	40	イ19	100	イ8	100	イ6	120	□5	180	□5	200	イ3	480	イ2	780			2000				
9月23日	イ23	40	□20	100	イ9	100	イ7	120	イ6	180	イ6	200	イ4	480	イ3	780			2000				
9月24日	イ24	40	イ21	100	□10	100	イ8	120	イ7	180	イ7	200	□5	480	イ4	780			2000				
9月25日	□25	40	イ22	100	イ11	100	イ9	120	イ8	180	イ8	200	イ6	480	□5	780			2000				
9月26日	イ26	40	イ23	100	イ12	100	□10	120	イ9	180	イ9	200	イ7	480	イ6	780			2000				
9月27日	イ27	40	イ24	100	イ13	100	イ11	120	□10	180	□10	200	イ8	480	イ7	780			2000				
9月28日	イ28	40	□25	100	イ14	100	イ12	120	イ11	180	イ11	200	イ9	480	イ8	780			2000				
9月29日	イ29	40	イ26	100	□15	100	イ13	120	イ12	180	イ12	200	□10	480	イ9	780			2000				
9月30日	□30	40	イ27	100	イ16	100	イ14	120	イ13	180	イ13	200	イ11	480	□10	780			2000				
10月1日	イ31	40	イ28	100	イ17	100	□15	120	イ14	180	イ14	200	イ12	480	イ11	780			2000				
10月2日	イ32	40	イ29	100	イ18	100	イ16	120	□15	180	□15	200	イ13	480	イ12	780			2000				
10月3日	イ33	40	□30	100	イ19	100	イ17	120	イ16	180	イ16	200	イ14	480	イ13	780			2000				
10月4日	イ34	40	イ31	100	□20	100	イ18	120	イ17	180	イ17	200	□15	480	イ14	780			2000				
10月5日	□35	40	イ32	100	イ21	100	イ19	120	イ18	180	イ18	200	イ16	480	□15	780			2000				
10月6日	イ36	40	イ33	100	イ22	100	□20	120	イ19	180	イ19	200	イ17	480	イ16	780			2000				
10月7日	イ37	40	イ34	100	イ23	100	イ21	120	□20	180	□20	200	イ18	480	イ17	780			2000				
10月8日	イ38	40	□35	100	イ24	100	イ22	120	イ21	180	イ21	200	イ19	480	イ18	780	イ1	200	2200				
10月9日	イ39	40	イ36	100	□25	100	イ23	120	イ22	180	イ22	200	□20	480	イ19	780	イ2	200	2200				
10月10日	□40	40	イ37	100	イ26	100	イ24	120	イ23	180	イ23	200	イ21	480	□20	780	イ3	200	2200	イ1	1100		
10月11日	イ41	40	イ38	100	イ27	100	□25	120	イ24	180	イ24	200	イ22	480	イ21	780	イ4	200	2200	イ2	1100	イ1	
10月12日	イ42	40	イ39	100	イ28	100	イ26	120	□25	180	□25	200	イ23	480	イ22	780			2000	イ3	1100	イ2	
10月13日	イ43	40	□40	100	イ29	100	イ27	120	イ26	180	イ26	200	イ24	480	イ23	780	イ6	200	2200	イ4	1100	イ3	イ1
10月14日	イ44	40	イ41	100	□30	100	イ28	120	イ27	180	イ27	200	□25	480	イ24	780	イ7	200	2200	□5	6000	イ4	イ2

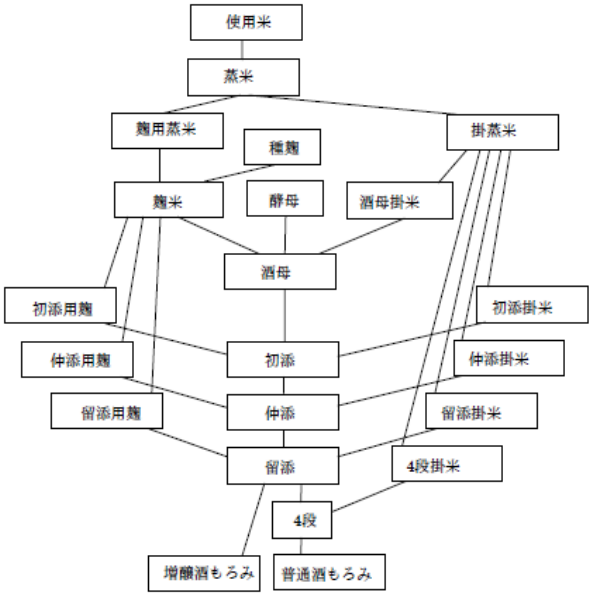

図2　アクティビティー図に基づくクラス図

3. Common Lispによる実装

3.1 属性としてのインスタンス変数

前項で述べたクラス図に基づき、CommonLispによるクラス定義（CLOS）を用いてプロトタイプ構築を行う。個々のクラスについて、図4に示す記法を用いてCLOSプログラムを実装する。

3.1.1 不変な属性のインスタンス変数

インスタンス変数は、固有の属性として不変なものと、センサーデータのように変化するものがある。前者は、:initformによるスロット値として代入するか、設定メソッドにより利用者から手操作で入力される。

3.1.2 変化するインスタンス変数

変化するインスタンス変数は、今のところセンサーデータを想定している。このデータは、蓄積される必要があるので、リストデータとして追記可能とした。さらに取得時刻は常に必要と思われるので、測定時刻を自動的に取得し、その時刻とデータのペアでリストとして管理するようにした。なお、この時間とのペアによる累積データはセンサー以外の手操作入力の場合にも適用可能である。温度以外のボーメ、糖分、酸度、アミノ酸度などは、サンプルを取り出して測定器や装置を用いてデータを取得し、手操作で入力されるが、これらのデータもこのカテゴリである。

3.1.3 変化する値の入力としてのメソッド記述

メソッドとしては、今述べたセンサーデータ、ボーメ、糖分、酸度、アミノ酸度の入力をクラス図に記述した。なお、実際のメソッド定義は今後状況に応じて追加されると思われ

Vol.2020-IFAT-138 No.5
Vol.2020-DC-116 No.5
2020/3/27

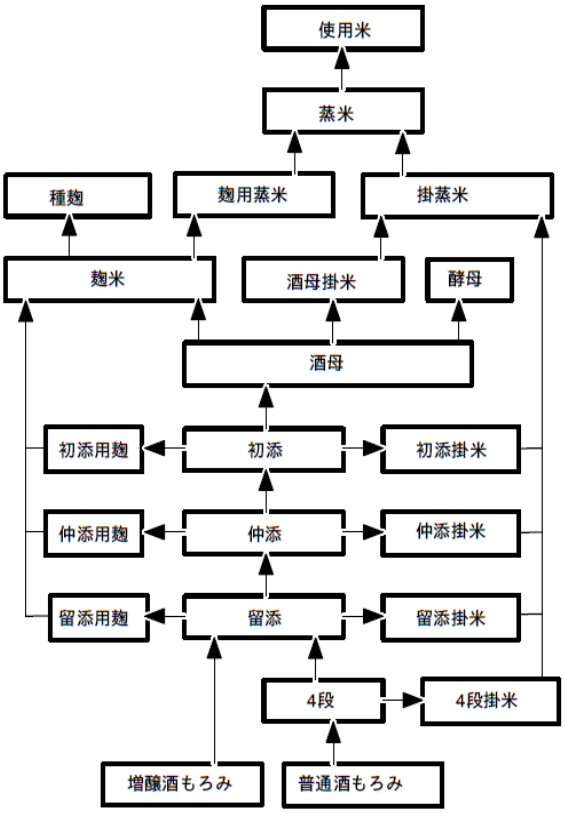

図3　表示を矩形的な継承に改訂したクラス図

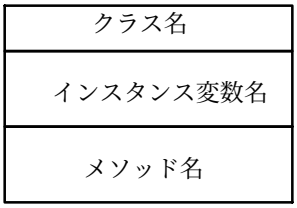

図4　クラス図におけるクラスの詳細記法

る。このような部分的なカスタマイズはオブジェクト指向プログラミングの特徴である。

3.2 クラス定義の例

具体的な定義例として、麹米クラスを取りあげる。クラス図を図5に示す。

クラス定義は、下記のとおりである。

```
(defclass 麹米 (種麹 麹用蒸米)
  ((酒母麹日時 :accessor 酒母麹日時?)
   (酒母麹量 :initform 40 :accessor 酒母麹量?)
   (酒母麹温度 :accessor 酒母麹温度?)
   (酒母麹室温度 :accessor 酒母麹室温度?)))
```

3.3 メソッド定義

メソッド定義は、一定の値のインスタンス変数入力と、変化するデータを随時取得する場合の2種類に大別される。前者については下記のようになる。

```
(defmethod 酒母麹日時入力 ((obj 麹米))
  (progn
```

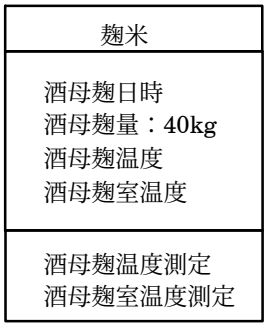

図5　麹米のクラス図

```
    (format t "酒母麹日時を入力してください？：")
    (setf (slot-value obj '酒母麹日時) (read-line)) ))
```

後者については、麹室における麹米の温度取得のメソッドの例を紹介する。

```
(defmethod 酒母麹温度入力 ((obj 麹米))
  (let* ((old-data (slot-value obj '酒母麹温度))
         (add-data
          (list (get-time)
            (progn
              (format t "酒母麹温度を入力してください。：")
              (read-from-string (read-line))))))
    (setf (slot-value obj '酒母麹温度)
          (cons add-data old-data))))
```

現状のレベルでは、上記の2種類のメソッドでデータを管理する。今後、さらに具体的な課題が生じる段階で、メソッドを追加していくことになる。

4. クラス・テンプレートによる表形式モデル

4.1 基本的考え方

クラス定義、メソッド定義は、図4、図5に示すクラステンプレート相互の関係からパターン化して体系化することが可能である。一通り、プログラムコードを記述し、動作することを確認したならば、クラステンプレートを表形式で再定義しておくと、後のデバッグやシステム更新のために有効である。なお、この手法に関しては、昨年ベトナムのダナンで開催されたIEVC2017[3]並びに第5回画像関連学会秋季大会[4]で紹介したが、ここではその手法を要約して述べる。

4.2 使用米・蒸米のクラス群

表4は、使用米・蒸米関連のクラス群のテンプレートである。テンプレートには、クラス名、スーパークラス、インスタンス変数名、メソッド名を記述する。一般のクラステンプレートは、スーパークラスは記述しない。それはクラステンプレート自体が、クラス図の詳細記述となっていて継承関係は明白だからである。この場合は、表形式でテンプレートを記述するので、継承関係を明確化するためにスーパークラスを記述し、その項目を設けている。この項目が無いクラスは、何も継承しないクラスで、バニラクラスと呼ばれる。使用米のクラスは、バニラクラスであり、インスタンス変数としては、品種と等級を有する。同様に、メソッドとしては、品種と等級の入力機能が定義される。

蒸米クラスは、使用米を継承する。インスタンス変数としては、精米歩合、精米日時、洗米日時を管理し、その入力機

Vol.2020-IFAT-138 No.5
Vol.2020-DC-116 No.5
2020/3/27

表4　使用米・蒸米関連のクラス群のテンプレート

Class-name	使用米	蒸米	麹用蒸米	掛蒸米	酒母掛米
Superclass		使用米	蒸米	蒸米	掛蒸米
instance variable	品種 等級	精米歩合 精米日時 洗米日時	(mix-in)	(mix-in)	(mix-in)
method	品種入力 等級入力	精米歩合入力 精米日時入力 洗米日時入力			

能がメソッドとして定義される。麹用蒸米、掛蒸米、酒母掛米は、その後の継承関係で異なるクラスとなっているが、当面管理すべき属性としてのインスタンス変数は蒸米と同様であり、新たなインスタンス変数を持たないmix-inとして定義されている。

4.3 麹米関係のクラス群

表5は、麹米関係のクラス群に関するテンプレートである。麹米は、種麹と蒸米により生成される。種麹はスーパークラスを持たないバニラクラスである。麹米生成に用いられる。麹米は酒母の生成だけでなく、その後の仕込み（初添、仲添、留添）の際にも逐次追加される。

表5　麹米関係のクラス群のテンプレート

Class-name	種麹	麹米	初添用麹	仲添用麹	留添用麹
Superclass		麹用蒸米 種麹	麹米	麹米	麹米
instance variable	種麹品名 種麹メーカー 種麹製造年月日 種麹購入年月日	酒母麹日時 酒母麹量 酒母麹温度 酒母麹室温	初添麹日時 初添麹量 初添麹温度 初添麹室温	仲添麹日時 仲添麹量 仲添麹温度 仲添麹室温	留添麹日時 留添麹量 留添麹温度 留添麹室温
method	種麹品名入力 種麹メーカー入力 種麹製造年月日入力 種麹購入年月日入力	酒母麹日時入力 酒母麹量入力 酒母麹温度入力 酒母麹室温入力	初添麹日時入力 初添麹量入力 初添麹温度入力 初添麹室温入力	仲添麹日時入力 仲添麹量入力 仲添麹温度入力 仲添麹室温入力	留添麹日時入力 留添麹量入力 留添麹温度入力 留添麹室温入力

4.4 掛米関係のクラス群

表6は、掛米関係のクラス群に関するテンプレートである。酵母は、酒母生成に用いられ、日本酒の元となる原液の制作に用いられる。掛米は蒸米を麹によりブドウ糖に変化させる触媒である。酵母はブドウ糖をアルコールに変える触媒であり、酒母はその両者と掛米としての蒸米を混合した日本酒の原料である。この原料を仕込むために別の容器に移し変えるのであるが、その時点から原料はモロミと呼ばれるようになり、日本酒として仕込まれるようになる。仕込みには、初添、仲添、留添の3段階があり、3段仕込みと呼ばれる。添えというのは、追加する意味であり、麹米と掛米、水が各々の段階で追加される。近年は上記の3段仕込みに追加して、調整用のアルコールを加えて掛米を加える4段仕込になっている。なお4段では麹米は加えない。

4.5 仕込み工程におけるクラス群

表7は、仕込み工程関係のクラス群に関するテンプレートである。初添、仲添、留添の3段と、追加された4段のクラス、さらに製品としての2種類のもろみで構成される。

5. プロトタイプから実装への課題

5.1 生成されるインスタンスとその活用

以上のクラス定義に基づき具体的に生成されるインスタンスのインスタンス変数は、84項目になる。これらの項目の値は、定数の場合と連想リストの場合に大別され、そのデータを活用するアプリケーションにとっては明確に識別された製造ロットのオブジェクトとなる。このオブジェクトへのデータ入力や参照の具体例は先の報告[2]で示した。

以上のようにして、表3に示した多様なロットに関して、そのロットの属性の種別と、時間的な履歴がオブジェクトとして個別に管理され、清酒になった際の種々の要因を遡って検討・分析することが可能となる。これを手続き型の言語で系統的なデータを管理するには、事前に十分な予備的な検討が必要になり、大変な手間を要することになる。

その点CLOSによるオブジェクト指向技術を用いると、以上のアクティビティ図、クラス図を経由して、プロトタイプシステムを構築し、シミュレートすることが可能になる。さらに想定外の問題が生じるような場合も、容易に修正・改訂することが可能であり、オブジェクト指向のメリットを生かすことが可能となる。

5.2 Common Lispによるシステムの問題

以上は、シミュレーションまでは良いが、実際のC言語やC++、さらにはJava、Ruby、Pythonといった実装言語と関連付けるにはどうすれば良いかという問題に直面する。現在

Vol.2020-IFAT-138 No.5
Vol.2020-DC-116 No.5
2020/3/27

表6　掛米関係のクラス群のテンプレート

Class-name	酵母	酒母	初添掛米	仲添掛米	留添掛米	4段掛米
Superclass		麹米 酒母掛米 酵母	掛蒸米	掛蒸米	掛蒸米	掛蒸米
instance variable	酵母品名 酵母メーカー 酵母製造年月日 酵母購入年月日	酒母設定日時 酒母汲み水 酒母温度 酒母室温 酒母ボーメ 酒母糖分 酒母酸度 酒母アミノ酸度	初添掛米日時 初添掛米量	仲添掛米日時 仲添掛米量	留添掛米日時 留添掛米量	4段掛米日時 4段掛米量
method	酵母品名入力 酵母メーカー入力 酵母製造年月日入力 酵母購入年月日入力	酒母設定日時入力 酒母汲み水入力 酒母温度入力 酒母室温入力 酒母ボーメ入力 酒母糖分入力 酒母酸度入力 酒母アミノ酸度入力	初添掛米日時入力 初添掛米量入力	仲添掛米日時入力 仲添掛米量入力	留添掛米日時入力 留添掛米量入力	4段掛米日時入力 4段掛米量入力

表7　仕込み工程関係のクラス群のテンプレート

Class-name	初添	仲添	留添	4段	普通酒もろみ	増醸酒もろみ
Superclass	酒母 初添用麹 初添掛米	初添 仲添用麹 仲添掛米	仲添 留添用麹 留添掛米	留添 4段掛米	4段	留添
instance variable	初添設定日時 初添汲み水 初添温度 初添室温 初添ボーメ 初添糖分 初添酸度 初添アミノ酸度	仲添設定日時 仲添汲み水 仲添温度 仲添室温 仲添ボーメ 仲添糖分 仲添酸度 仲添アミノ酸度	留添設定日時 留添汲み水 留添温度 留添室温 留添ボーメ 留添糖分 留添酸度 留添アミノ酸度	4段設定日時 4段汲み水 4段温度 4段室温 4段ボーメ 4段糖分 4段酸度 4段アミノ酸度	普通酒設定日時 普通酒汲み水 普通酒温度 普通酒室温 普通酒ボーメ 普通酒糖分 普通酒酸度 普通酒アミノ酸度	増醸酒設定日時 増醸酒汲み水 増醸酒温度 増醸酒室温 増醸酒ボーメ 増醸酒糖分 増醸酒酸度 増醸酒アミノ酸度
method	初添設定日時入力 初添汲み水入力 初添温度入力 初添室温入力 初添ボーメ入力 初添糖分入力 初添酸度入力 初添アミノ酸度入力	仲添設定日時入力 仲添汲み水入力 仲添温度入力 仲添室温入力 仲添ボーメ入力 仲添糖分入力 仲添酸度入力 仲添アミノ酸度入力	留添設定日時入力 留添汲み水入力 留添温度入力 留添室温入力 留添ボーメ入力 留添糖分入力 留添酸度入力 留添アミノ酸度入力	4段設定日時入力 4段汲み水入力 4段温度入力 4段室温入力 4段ボーメ入力 4段糖分入力 4段酸度入力 4段アミノ酸度入力	普通酒設定日時入力 普通酒汲み水入力 普通酒温度入力 普通酒室温入力 普通酒ボーメ入力 普通酒糖分入力 普通酒酸度入力 普通酒アミノ酸度入力	増醸酒設定日時入力 増醸酒汲み水入力 増醸酒温度入力 増醸酒室温入力 増醸酒ボーメ入力 増醸酒糖分入力 増醸酒酸度入力 増醸酒アミノ酸度入力

のKijimoriは、センサー系、サーバ系はC言語とC++で実装され、その実装はC・C++に長けたハイガリーのITware社が担当している。

CommonLispベースでシステムを構築する場合、問題になるのはオブジェクトインスタンスの永続的な管理である。現状のCommonLispの処理系では、インスタンスが主記憶上のデータであり、システムがダウンしリセットされると消滅してしまう。ロットのインスタンスは、表3から分かるとおり、生成までに一月半、40日以上の期間を要し、さらにその後に検索などの用途に用いるとすると数ヶ月から数年の寿命を要求される。これではCommonLispの処理系での実用サービスは無理である。

しかし、Common Lispベースのオブジェクト指向データベースが使用可能であれば、サーバ系をCommonLispで実装する可能性は存在する。四半世紀前の1990年代前半に、Itasaという製品が存在し、それはCLOSによるオブジェクトのインスタンスをそのまま格納可能であった。従ってそのようなツールが商品として存在すれば可能性はあるのだが、現状では無理である。そうなると、CommonLispからC、C++への移行を考えざるを得なくなる。

6. 考察及び追加検討

6.1 分類型継承と統合型継承

CLOSからC、C++への移行を考察していて、階層の多いクラス継承は、複雑過ぎて対処が必要なことを痛感した。クラステンプレートを表形式にするのは、一つのアイデアであるが、それでも深い階層を一元的な継承関係で定義するのは、柔軟性を欠くことになる。

例えばダイアモンド継承が少なからず出現する。一般に多重継承におけるダイアモンド継承は避けるべきことが常識となっているが、意味的な関係を保持したければ変数やメソッド名の衝突を避けて実装することはできる。それでも複雑感は拭うことができない。この状況は、厳密に定義を加えるほど適用範囲は狭められるというゲーデルの不完全性定理を彷彿させるものである。

そこで思いついたのは、クラス継承を分類型と、統合型に分ける手法である。オブジェクト分析設計という分野があるが、将に分析と設計についての相違の問題である。分析は、ある事項を要素還元的に分類分析することで、設計は異質の要素を統合しつつ機能を取捨選択し、本来の目的を実現することである。分析は、直感的に把握し実現可能であるが、設計は矛盾を包含した要素を整合させるために、種々の工夫を行うのが一般的であろう。

例えば、図2（図3も同様）のクラス図を分類的な過程と統合的な過程に分けることを試みる。図6は、分類的な過程で使用米からもろみの原料となる要素までのクラス群を示している。

図7は統合的な過程で、もろみの原料からもろみが仕込まれて清酒の元に至るプロセスのクラス群を示している。このように分類過程と統合過程に分けて考えることにより、CLOSの

Vol.2020-IFAT-138 No.5
Vol.2020-DC-116 No.5
2020/3/27

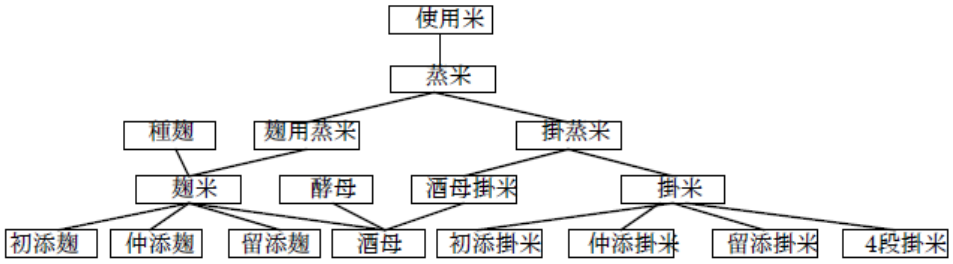

図6　分析・分類型継承

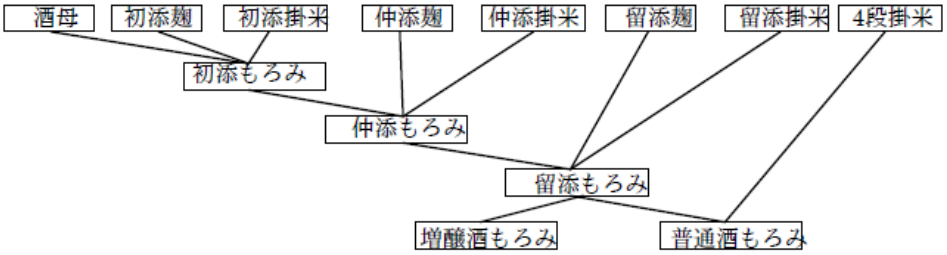

図7　設計・統合型継承

大きなインスタンスとは異なる手法により、さらに実用的・実践的な管理を目指せるのではないかと考えた。

6.2 ロット単位の管理から、要素部材の管理へ

図6と図7を統合すると、図2（図3）と同様になるのだが、このように2段階に分けることにより、異なる解釈が生まれる。先ず図6では、蒸米、種麹、酵母の3種類の原料を発酵させて、初添麹、仲添麹、留添麹、酒母、初添掛米、仲添掛米、留添掛米、4段掛米という8種類の要素部材を生成させる。次に図7では、以上の8種類の要素部材を仕込みプロセスを通じて逐次混合して、最終的なロットの管理に結び付けるのである。8種類の要素部材は、表5の右側の3種類のクラスと表6の右側の5種類のクラスが該当する。

この思想は、要素部材が明示されないCLOSインスタンスに比べると、中間段階の状況が提示されるので、ブラックボックス化を避けられる点で一見すると実用的かもしれないが、ロット毎の情報管理を目的にするならば、ブラックボックス化しても問題は無いので、必ずしも利害得失は云々できない。

6.3 C、C++による実装への方向性

C、C++による実装は、図6と図7に分ける手法が現実的であろう。初添麹、仲添麹、留添麹、酒母、初添掛米、仲添掛米、留添掛米、4段掛米という8種類の要素をRDBで定義・管理することとし、図6の分類・分析継承関係を通じて取得データをRDBに登録する。その後、図7の設計・統合型継承で、ロット毎のデータをRDBで管理するアプローチである。

図6のクラス階層の実装は、C++のクラスで行い個々のインスタンスレベルでRDBに格納するのが妥当と思われるが、図7の実装に関しては、必ずしもクラス継承は用いないで直接RDBとのインタラクションで別途実装する方が実用的と思われる。このあたりは、C++による開発体制や、プログラマの好み、習熟度に依存することになるであろう。

6.4 クラス図と実装の分離

以上から、クラス図の概念を捉え直すことを考えた。これまでは、クラス図におけるクラス名、インスタンス変数名、メソッド名に基づき、CLOSのクラスを定義して、CLOSによるシミュレーションを実現してきたが、このレベルのクラス図は、あくまでもプロトタイプの実装用に過ぎない。本格的なシステムのためには、実装のための意味的な参照のためのクラス階層と考える方法があるのではないかということである。要するにクラス図と実装は分離して構築し、前者は後者を分かりやすく理解するための参照モデルとすべきなのである。なお、この発想でオントロジを参照手段として実用化したシステムとしては、ISO標準となっている電子カルテのopenEHRが挙げられる。openEHRのオントロジ構築言語は、ADL（Architype Definition Language）と呼ばれているが、属性情報を用語と値で管理する抽象データ型の操作言語で、オントロジの階層を参照タイプとして構築しており、将に上記概念を実現している[5]。

先に、図2（図3）のクラス階層を造り酒屋オントロジと名付けて、意味的なデータ構造を想定したが、このクラス図は、必ずしも分かりやすいものではなかった。それに対して

中間に要素部材を配置する図6と図7の方が感覚的に分かりやすいと思われる。しかしながら、図6と図7のを総合したクラス階層は、図2（図3）と同一である。従って感覚的な意味は人間の視覚的な認知性の要因が大きい。

この分かり易さの問題は、人間の思考パターンを反映していると思われる。要するに、ある目標を実現するためには、要因をブレークダウンする形式で分析し、個別の単純な事項に要素還元し、その後その要素を組み合わせて設計・統合して目標を実現する。

本来オントロジは、体系を原始的な最小語彙に還元して統合することであるから[6]、中間に要素部材を配置する構成の方が本来のオントロジに近いのであろう。

7. まとめ

以上の検討の結果を要約すると下記の通りである。

(1) 日本酒製造のロット毎の意味的な情報をCLOSによるインスタンスの情報で管理する手法を確立したが、それをC、C++で管理する手法を検討した。

(2) クラス図におけるクラス名、インスタンス変数名、メソッド名を表現するクラステンプレートにスーパークラス記述を追加することにより、深い多重継承のクラス図を表形式で管理する手法を考案した。

(3) Common LispとC、C++の構文・型を分析し、入出力、制御構造、配列定義、関数定義などはほぼ対応するが、ポインタ、文字列、構造体などは無理があり、C++による多重継承のクラス定義は対応可能だが煩雑になる。

(4) 深い階層の継承関係を、分かりやすくする手法として、分類型継承と統合型継承に分割し、分類型継承により要素還元されたクラス群を要素部材的に扱い、その要素部材を組み合わせて統合型継承を実現する手法を考案した。

(5) クラス図を実装手段としてではなく、意味的な参照手段の枠組みとして応用する手法が考えられる。CLOSで実装可能なクラス図がC、C++では難航するために考えた手法であるが、その具体的な実現は今後の課題である。

8. 教訓と課題

1981年8月号のバイトマガジンを参照して以来、Smalltalk80でオブジェクト指向プログラミングのクラス継承に興味を抱き Zetalispで多重継承の面白みを知り、その後TAO, CLOS, InterleafLisp, C++, Java, Rubyでオブジェクト指向プログラミングによるクラス継承のシステムを構築してきた。文書系に関しては、DTD, XML Schema, RDF Schema, OWLを用いて、静的な意味構造を検討してきた。

私が構築・開発した動的な実用システムとしては、TAOを用いた非線形振動モデル[7]、同じく接点消耗予測システム[8]、CLOSによる接点消耗予測システム[9]、C++による公衆電話機保守支援システム[10]、CLOSによる拡張可能な電子履歴書システム[11]が挙げられる。これらのシステムでは意味的な継承を指向したが、ダイアモンド継承にはらない3〜5階層程度の浅い継承のクラスとしてシステムを継承し、現実のモノに対応するモデルを構築した。

今回の検討は、CLOSによる系統的な一元的なオントロジ構築を目指して、10階層に渡るモデルを構築したが、これだけの階層になると感覚的な管理は困難で、テーブルによる管理を行った。従ってテーブルによるクラス階層管理はアイデアと言うよりは苦肉の策である。さらに分析・設計の観点で、分析による分類段階と設計による統合段階を分けることにより、各々5階層程度のクラス群にまとめ、C++実装へのヒントを得た。

大規模システムのオントロジ的な構築は、UMLによりCLOSのような多重継承の枠組みでマクロなクラス構築を行い、分類的なクラス階層を部品として構築した後に、その部品クラスを設計の観点で統合するクラスとして実装する手法が可能な場合が多いように思われ、それをさらに試みたいと考える。多くのビジネスプロセスは、日本酒醸造のように、上流行程において分類的にクラスが展開し、下流行程において総合的に結論付けられド製品化・キュメント化される場合が多いと思われる。

CLOSのクラスによりプロトタイプシステムを構築して大まかな挙動を把握し、正規の実装はC++の実用的なクラスで行うのが妥当であるが、統合部分の実装は、多重継承のクラスで行うよりは、データベースに実装する方が実用的である。従ってCLOSのクラスは、あくまでもインスタンス変数に相当する値の参照モデルとし、実装はそのインスタンス変数に対応させたデータベースの値として実装する手法が実用的であると考えられる。

9. おわりに

本検討を進めるにあたり、これまでの検討と実装にご協力頂いたFBトライアングルの広浦 雅敏様、ITwareのBiro Attila様、Hajdu Csilla様に感謝します。さらに日本酒製造の設備を見学させていただき、専門家の杜氏として説明いただいた鶴乃江酒造の林恵子さま、名倉山酒造の中島伸一郎さまに御礼申し上げます。

参照情報・文献

[1] 大野邦夫；”日本酒製造プロセスをモデル化する手法に関する一検討”, 第2回画像関連学会連合会大会講演論文（2015.11）

[2] 大野邦夫, 広浦雅敏; ”日本酒製造プロセスのモデル化とM2M・IoT技術による実装検討”, 情報処理学会研究報告, DC102−3（2016.7）

[3] Kunio Ohno, Masatoshi Hiroura, Biro Attila; ”DEVELOPMENT OF SAKE BREWING ENTREPRENEURS SUPPORT SYSTEM IN FUKUSHIMA”, Proc. 5th International Workshop on Image Electronics and Visual Computing (2017.3)

[4] 大野邦夫；”オブジェクト分析設計におけるクラス継承とシステムの意味的概念の形成に関する考察 ”, 第5回画像関連学会連合会大会講演論文（2018.11）

[5] 大野邦夫；”個人の情報環境へのオントロジ適用の検討”, 情報処理学会研究報告, DD88−1（2013.1）

[6] 大野邦夫；”オントロジ技術の応用に関する一考察”, 情報処理学会研究報告, DD41−1（2003.9）

[7] 大野邦夫；”オブジェクト指向による非線形振動のモデル化”, 電子情報通信学会論文誌 D−II, Vol.J77−D−II, No.9 (1994.9)

[8] 大野邦夫；”オブジェクト指向プログラミングによる接点アーク現象のモデル化”, 信学技報, NLP90−42, (1990.)

[9] Kunio Ohno; ”Modeling Contact Erosion Using Object−Oriented Technology”, IEICE Trans. Electron, Vol.E77−C, No.10,(1994.10)

[10] 大野邦夫ほか; “公衆電話機保守支援システムの改良−−RESPONSE−2とAdaptiveRESPONSE”, NTT技術ジャーナル Nol.6, No.8, pp.75−79（1994.8）

[11] 大野邦夫, 角山正樹；”拡張可能な履歴書管理システムの 実装に関する検討”, 平成22年度職業能力開発総合大学校紀要（2011.3）