Vol.2016-DC-102 No.8
2016/7/16

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

大野 邦夫† 広浦 雅敏††

地域ビジネス活性化の戦略として伝統産業を最新の情報技術で支援する手法が検討されている。本報告では、伝統産業としての日本酒製造プロセスに、M2M・IoTの要素技術であるセンサーを導入したKojimoriの高機能化を目指して、酒造プロセスをUMLでモデル化し、酒造りの専門家としての杜氏を支援するためのシステムの基礎的な検討内容を紹介する。

A Study on Modeling and Implementing Japanese Sake Production Process through M2M and IoT Technologies

Kunio OHNO†, Masatoshi Hiroura††

Strategy to activate regional business which supports traditional industries through information technologies has been studied these days. This paper describes how to model Japanese sake production process through UML, which would emphasize the sensor based M2M/IoT system of Kojimori. The goal of the system is to support tohji people who are the professionals to produce Japanese sake.

1. はじめに

地域ビジネスの活性化としては、観光と農業が期待されているが、地域の産業を支えてきた特産品や名物などの伝統産業も興味ある注目すべき業界である。その中でも日本酒製造は、日本の大衆文化の一翼を担う産業と言っても過言ではない。米を原料とする酒は、古くは奈良時代の風土記に記載されている。平安時代以降は荘園からの年貢米で寺院が酒造りを行った。これらを僧坊酒と呼ぶ。その後、室町時代に諸白（南都諸白）と呼ばれるにごり酒ではない今日の清酒に近い日本酒が製造されるようになった。その後、江戸時代には、幕府が指導して地域の産業として日本酒造りを奨励するようになった。特に寒造り３段仕込みという標準化された製法を各藩に伝授し、日本全国で地域の名産となる日本酒が誕生した。さらにそのような背景のもとで、酒造りの職人である杜氏という職業が誕生し、南部杜氏のように、地域に根ざした職人文化が生まれた[1]。

明治維新後も日本の伝統的な酒造りは江戸時代の文化を継承して発展した。戦後も日本酒文化は継続されているが、昭和20年代をピークに消費量は減少している。特に最近は、飲酒の嗜好が多様化し、気軽に飲めるワインやビールを中心とする洋酒に市場が移行している。日本酒の市場は吟醸酒のような高級品に移行しつつあり、そのための製造プロセスの管理が期待されている。

そのような状況にありながら、酒造分野における情報技術の活用は十分とは言えない。大手の酒造メーカーは、工場における大規模な日本酒生産を行っており、IT活用の恩恵に浴しているが、地域の中小企業や個人経営企業は未だその恩恵に浴していない。他方、地域の中小企業や個人経営企業が優れた個性的な名酒を造り出し、日本社会を豊かにしていることも事実である。その品質は、杜氏を始めとする関係者の血のにじむ努力によって支えられている。地方自治体もそのような企業の支援を行っており、業界支援のための情報提供、定期的な講習会の開催などを行なうと共に、技能伝承や人材育成に務めている。

†(株) モナビITコンサルティング
Monavis IT Consulting Co. LTD.
††FBトライアングル(株)
FB Triangle Co.

そのような努力を軽減すべく、ITを活用する支援技術が使用され始めている。KojimoriはそのようなITツールの事例である[2]。現在、Kojimoriは、温度センサーを用い、麹室の温度などを遠隔から監視する機能を持っているが、その機能を拡張し杜氏さんや関係者をより広範な視点で支援するシステムとしたいと考えている。そのためには、酒造プロセス全般を支援する枠組みを想定し、その枠組みに基づいて関係者を支援するシステムを構築することが望まれる。現在IT分野で注目されているM2M（Machine to Machine），IoT（Internet of Things）技術により、酒造プロセスにおける種々のフェーズの属性情報を把握し、個々の状況に基づき、関係者を的確に支援するシステムの開発が実現可能であると思われる。本検討は、そのための端緒となることを目指している。

なおこの手法は、酒造プロセスのみならず、地域の多様な産業に適用可能と考えている。Kojimori自体、農業、観光、介護などの地域のニーズに対応して使用されているが、このような分野に適用される手法となるように検討をすすめている。

2. 酒造プロセスの概要とモデル化

2.1 プロセス分析の手法

2.1.1 基本手法としてのUML

M2Mのセンサーからのデータを、関連属性と共に収集するためには、そのデータの役割を明確化する必要がある。そのためには、業務プロセスの分析、把握を必要とする。

業務プロセスの把握のための一般的な手法としては、ワークフローの管理や付随するドキュメント管理が必要とされる。そのために以前からオブジェクト分析手法が活用されており、最近はUML（Univaersal Modeling Language）として体系付けられている。UMLは、普遍的なモデル化のための言語という意味であるが、それ以前の多様な手法を統合した様々な図を活用する方法論であり、言語という言い方は適当ではない。UML自体は多様な図を用いてシステムやプロセスを図で示し理解を促す膨大な体系

であるが、その要はクラス図にある。要するにクラス図を作成し、その図に基づいて、オブジェクト指向プログラミング言語のクラスを定義し、実装することを通じてシステムを構築するのが基本である。ということから、プログラムのクラスを定義するということが、情報システムを実装する上で本質的に重要である。

2.1.2 ユースケース図とアクティビティー図

一般にクラス図を作成するためには、実際の業務のユースケース図、アクティビティー図を必要とする。ユースケース図は、アクターと呼ばれるシステムの関係者とシステムとのインタラクションを記述するもので、外界を含むシステムの空間的な概要特性を示す。他方アクティビティ図は、業務内容のフローを記述し、システムの時間的な推移を記述する。

2.1.3 酒造プロセスへの適用

酒造プロセスは、単純に述べると米を発酵させてデンプンを糖に変え、糖をさらにアルコールに変換することある。それを精米、洗米、蒸米、麹作成、酒母作成、仕込み、もろみ作成、上槽、火入れといった複合的な過程を経て清酒に仕上げ、商品化する。とりあえず、その全体を、把握するために

1) 準備フェーズ
2) 醸造フェーズ
3) 仕込みフェーズ
4) 仕上げフェーズ

の4段階に大別してみた。以上のフェーズをアクティビティー図を作成し分析し、プログラムへの実装のためのクラスを抽出する。

2.2 準備フェーズ

2.2.1 アクティビティー図

図1は、玄米を精米し、さらに洗米して蒸米を生成する準備フェーズのアクティビティー図を示す。UMLの正式

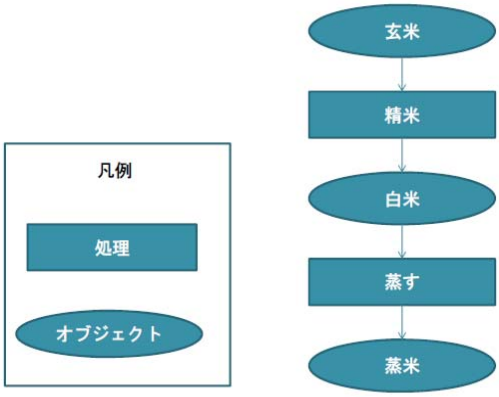

図1 準備フェーズのアクティビティー図

なアクティビティー図は、種々の規約があるがここでは単純に処理の流れが分かれば良いので簡略化した記法を用いている。図の矩形部分は処理を示し、楕円部分はオブジェクトを示す。オブジェクトは事物の概念であり、抽象的な概念と個々の具体的な事物に大別される。オブジェクト指向プログラミングの世界では、前者をクラス、後者をインスタンスと呼ぶ。

2.2.2 ユースケース図

図2は、準備フェーズのユースケース図を示す。図の四角の囲みは、システムを示し、その外側の人間を模した形状のものはアクターと呼ばれるシステム関係者である。こ

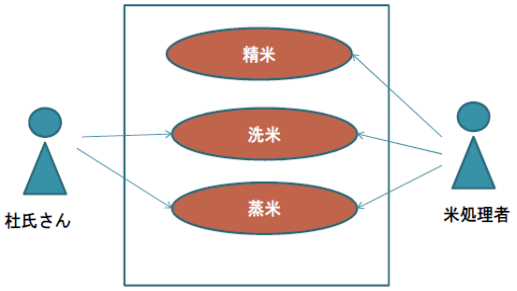

図2 準備フェーズのユースケース図

の図の場合は、杜氏さんと米処理者である。なお米処理者は杜氏さんの指示に従って、米を調達し、精米し、洗米し、蒸すという作業を行うのであれば、必ずしも独立したアクターとして扱う必要はない。要するにプログラム的に独立した変数となる要因が存在する場合にアクターは必要となる。ユースケース図の四角の囲みの中は、サブシステムとしての処理が項目として記述される。この図では楕円で精米、洗米、蒸米が示されている。ある程度、アクティビティー図の矩形の処理に対応するが、厳密なものではない。

なお、今回の検討においては、杜氏さんを中心とした分析になるため、他のアクターは特に考慮する必要が無いと思われ、以後ユースケース図は省略し、アクティビティー図のみを対象に検討する。

2.3 醸造フェーズ

2.3.1 アクティビティー図

図3は、蒸米から麹作成、酒母生成までのアクティビティー図である。この間の麹や酒母の生成プロセスの管理

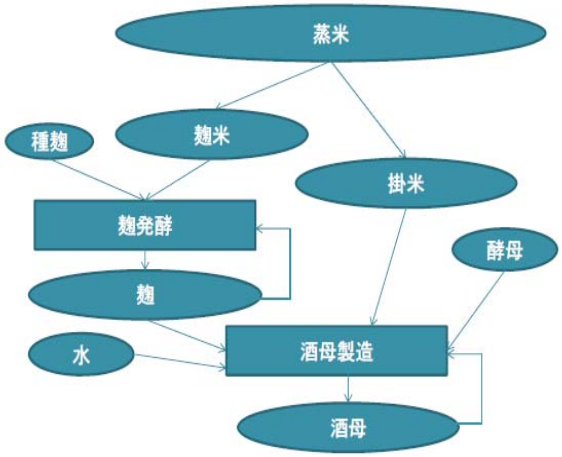

図3 醸造フェースのアクティビティー図

が、良い酒を造るための重要な要因となる。準備フェーズで生成された蒸米の一部は、麹を生成するために使用される。蒸米は、麹室に保管され、種麹が振りかけられて発酵が行われる。優れた麹を得るにはその温度管理は極めて重要である。

2.3.2 麹発酵の温度管理

図4は、麹の温度管理の一例を示す[3]。30度余りの蒸米が発酵し、切り返し、盛り、仲仕事といった操作を経て

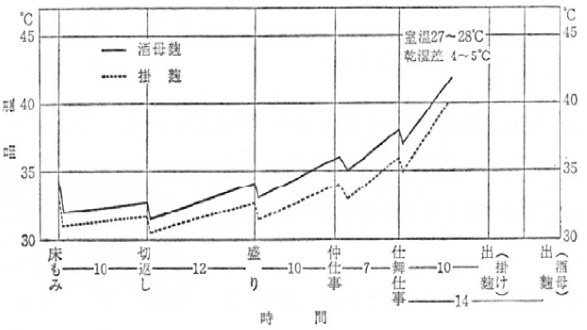

図4　麹の生成過程における温度管理

徐々に温度が上がって最後は40度余りになる。なお、この図から分かるとおり、2～3日で麹が生成される。出来上がった麹は酒母造りに使用されるが、その後の仕込みに使われるものもある。図における酒母麹（実線）が前者であり、掛麹（破線）が後者である。酒母は文字通り酒の母体となる原資で、蒸米と水と酵母がさらに加えられ醸成される。

2.3.3　酒母の温度管理

図5は、酒母の醸成における温度変化を示す[4]。この図の縦軸は温度で横軸は日数である。水麹、仕込み（麹の仕込で後のもろみの仕込みではない）といった初期の投入の後、冷却し、荒櫂・打瀬の後は10度以下まで冷却される。

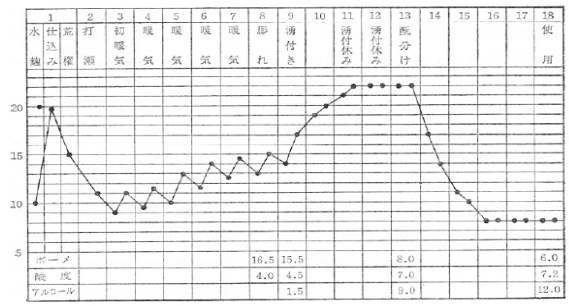

図5　酒母の醸成における温度変化

10度以下だと酵母の増殖が進まないが麹の糖化は進展する。その後、暖気プロセスで徐々に温度を上げて酵母を増やし、13日目のもと分けの後は発酵を押さえるために温度を下げ、最後は7度程度にまで下げてもろみに使用する。

2.4 仕込フェーズ

2.4.1　アクティビティー図

仕込フェーズのアクティビティー図を図6に示す。仕込は、三段仕込という古典的な伝統に基づくプロセスであるが、最近はもう一段追加されて、4段になっている。

三段仕込は、麹や蒸米、水を追加していくプロセスであり、追加することを添えるとも言うことからその名がある。三段仕込は、初添、仲添、留添という個別の処理から構成される。各々の添えプロセスで、麹や蒸米、水が追加される。なお、4段では麹は追加されない。

2.4.2　もろみの温度経過

図7は、もろみの温度経過を示す[5]。初添では15度であったのが、仲添では10度になり、留添では7～8度まで温度を下げた後に、徐々に温度を上げていく様子が分かる。

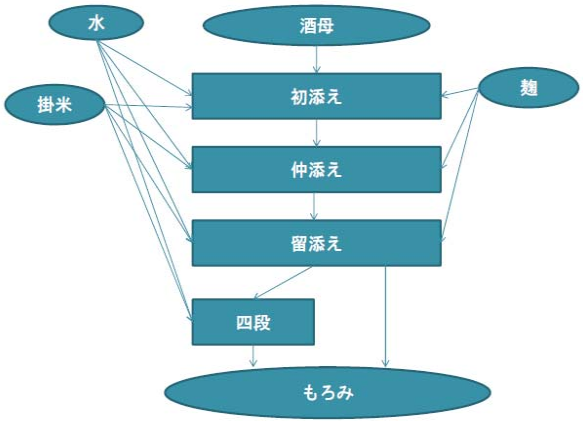

図6　仕込フェースのアクティビティー図

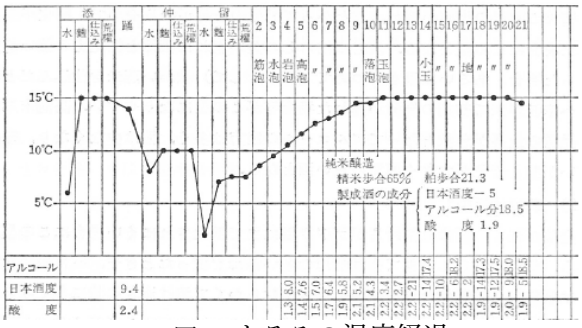

図7　もろみの温度経過

仕込は、この温度が下がる部分であり、それ以降は次の仕上げフェーズに含まれる。なお、4段の仕込プロセスは、清酒を濾過する（上槽）直前であり時間的な尺度では、事項の仕上げフェーズに含まれるが、実際のシステム上の振る舞いは、仕込なので、このフェーズに含めている。

2.5 仕上げフェーズ

2.5.1　アクティビティー図

図8は、もろみを発酵させ、最後に濾過のための搾り（上槽）、清酒が出来上がるまでの仕上げフェーズのアクティビティー図を示す。このプロセスは、図7における仕

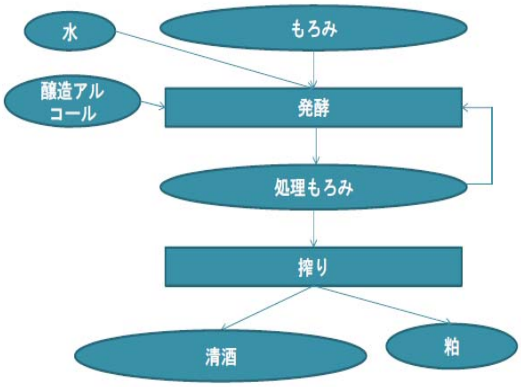

図8　仕上げフェーズのアクティビティー図

込以降の経過に対応するが、発酵に伴う泡の状況がもろみの管理上重要である。

2.5.2 泡の観察

留添の後に、筋泡（すじあわ）という現象が観察される。これはもろみ表面に泡が筋となって出現することで、酵母が増殖し発酵が開始されたことを意味する。その後白くて軽い水泡となる。その後もろみの粘度が増してくるに従い別のタイプの泡が出現する。まず岩泡という盛り上がった泡が発生するようになる。さらにそれが黄色味を帯びさらに盛り上がってくる状況を高泡と言い5～7日間継続する。高泡の後期には、泡は大きく低くなっていく。撹拌すると落ち込んで消えていく状況になるが、この状態の泡を落泡と言う。その後、泡は小粒になり玉泡と呼ばれるようになると、もろみは清酒に近い状態になっていることを示す。

2.5.3 上槽

その後、4段仕込のための蒸米の追加や醸造アルコールを加える最後の調整が行われた後に、最終的なもろみは搾られて濾過され、液体の清酒と固形分の酒粕に分離される。このプロセスを上槽と呼ぶ。

2.5.4 火入れ

搾られた清酒には酵母や雑菌が残存するので、そのまま放置すると酸化・腐敗する。そのために60～65度に加熱して残存酵母や雑菌を殺すプロセスを火入れと呼ぶ。一般の清酒は、火入れした後に貯蔵され、さらにびんに詰める直前にも火入れされる。

3. 検討対象の具体化と明確化

3.1 背景と方向性

3.1.1 酒造業界の現状と消費者ニーズ

貿易が自由化され、海外からワインをはじめとする多様な洋酒が輸入されるようになり、日本人のお酒への嗜好は変化しつつある。その影響を受けて日本酒の消費量は低下しつつあり、消費者ニーズに適合する優れた品質の製品を提供すると共に、海外への輸出を狙う新しい製品の開発が求められている。そのような状況に今日の酒造業界は置かれている。

3.1.2 ITによる支援とモデルの詳細化

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。そのための大雑把な枠組みを前章で述べたわけであるが、前章で明らかにした枠組みをさらに具体化する必要がある。そこで、公益財団法人・日本醸造協会が出版している「最新酒造講本」のデータを参照してモデルの詳細化を試みた。

3.1.3 製造計画の必要性

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必要がある。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。その具体的内容は、図3と図6のアクティビティー図に示した麹、酒母の育成、初添、仲添、留添および4段の仕込み、およびもろみの処理である。

3.1.4 日仕舞いもろみ製造の事例

「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）。それに基づいて具体化的なモデルについて検討することとする。表1と表2に仕込の際の蒸米、麹米（麹）、汲み水、醸造アルコールの配合を示す。

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%)

表1が普通酒で表2が醸造酒である。普通種は4段仕込を行うのに対し、醸造酒はそれを行わない。醸造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。

3.2 アクティビティ図の詳細化

3.2.1 蒸米からもろみまでの詳細アクティビティ図

図9に日本酒製造における蒸米からもろみ生成までのプロセスのフローを詳細化したアクティビティ図の形式で示す。これは、図3と図6のアクティビティー図をその関連も含めて統合し、さらに詳細化したものである。図3と図6と同様に、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。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説明する。

3.2.2 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でんぷんが糖化発酵する状況に置かれる。

3.2.3 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。

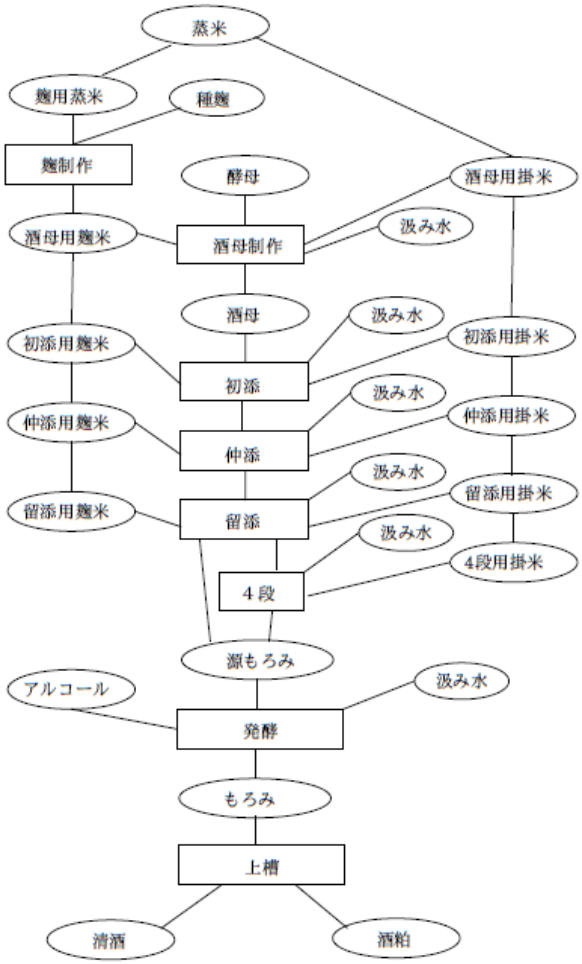

図9　統合・詳細化されたアクティビティー図

3.2.4 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。

3.2.5 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酒の量が6000リッターとなっている。

4. 日本酒製造モデルのプログラム化

4.1 クラス図

図10に、図9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。

クラスの詳細を記述する場合は、名称の下に図11に示すようにさらに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。

4.2 クラス実装の考え方とその機能

4.2.1 CLOSによる実装

前章で述べたクラス図に基づき、Common Lisp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11に示す記法を用いてCLOSプログラムを実装する。

4.2.2 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。

4.2.3 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のみを想定している。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。

4.2.4 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。

4.3 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12に示す。クラス定義は、下記のとおりである。

```
(defclass 麹米 (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?)))
```

4.4 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))
```

表3　表1・表2による製造の実施計画例

	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	ロ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	ロ5	100															140				
9月9日	イ9	40	イ6	100															140				
9月10日	ロ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	ロ10	100															140				
9月14日	イ14	40	イ11	100															140				
9月15日	ロ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	ロ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	ロ5	100	イ3	120	イ2	180	イ2	200							740				
9月20日	ロ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	ロ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	ロ5	180	ロ5	200	イ3	480	イ2	780			2000				
9月23日	イ23	40	ロ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	ロ10	100	イ8	120	イ7	180	イ7	200	ロ5	480	イ4	780			2000				
9月25日	ロ25	40	イ22	100	イ11	100	イ9	120	イ8	180	イ8	200	イ6	480	ロ5	780			2000				
9月26日	イ26	40	イ23	100	イ12	100	ロ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	ロ10	180	ロ10	200	イ8	480	イ7	780			2000				
9月28日	イ28	40	ロ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	ロ15	100	イ13	120	イ12	180	イ12	200	ロ10	480	イ9	780			2000				
9月30日	ロ30	40	イ27	100	イ16	100	イ14	120	イ13	180	イ13	200	イ11	480	ロ10	780			2000				
10月1日	イ31	40	イ28	100	イ17	100	ロ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	ロ15	180	ロ15	200	イ13	480	イ12	780			2000				
10月3日	イ33	40	ロ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	ロ20	100	イ18	120	イ17	180	イ17	200	ロ15	480	イ14	780			2000				
10月5日	ロ35	40	イ32	100	イ21	100	イ19	120	イ18	180	イ18	200	イ16	480	ロ15	780			2000				
10月6日	イ36	40	イ33	100	イ22	100	ロ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	ロ20	180	ロ20	200	イ18	480	イ17	780			2000				
10月8日	イ38	40	ロ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	ロ25	100	イ23	120	イ22	180	イ22	200	ロ20	480	イ19	780	イ2	200	2200				
10月10日	ロ40	40	イ37	100	イ26	100	イ24	120	イ23	180	イ23	200	イ21	480	ロ20	780	イ3	200	2200	イ1	1100		
10月11日	イ41	40	イ38	100	イ27	100	ロ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	ロ25	180	ロ25	200	イ23	480	イ22	780			2000	イ3	1100	イ2	
10月13日	イ43	40	ロ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	ロ30	100	イ28	120	イ27	180	イ27	200	ロ25	480	イ24	780	イ7	200	2200	ロ5	6000	イ4	イ2

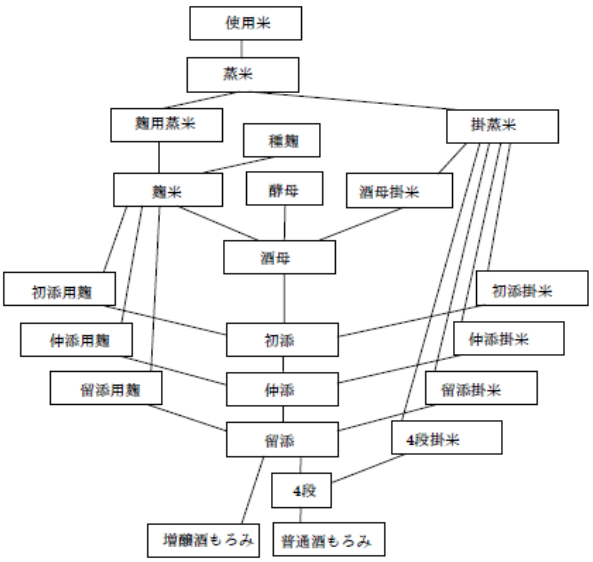

図10　詳細アクティビティー図に基づくクラス図

```
(progn
 (format t "酒母麹日時を入力してください？：")
```

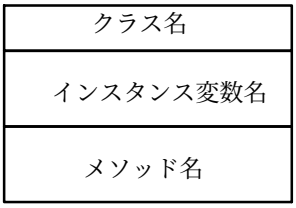

図11　クラス図におけるクラスの詳細記法

```
 (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))))
```

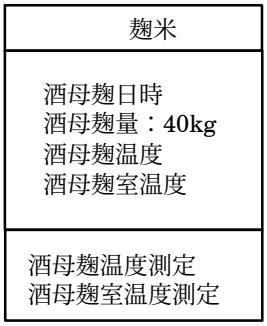

図12　麹米のクラス図

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こととする。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方針である。

4.5 シミュレーションの実行

4.5.1 基本的な考え方

このシステムで実際のシミュレーションを行うのは無理があるが、仮想シミュレーションを行うことはできる。仮想シミュレーションのためのシナリオを用意し、その実行を試みる。

4.5.2 インスタンスの生成

まずインスタンスを生成する必要がある。リスナー上で下記の処理を行うことにより、普通酒もろみのインスタンスが生成される。ここでは、イ1というインスタンスを生成する。

```
CG−USER(11): (setf イ1 (make−instance '普通酒もろみ))
#<普通酒もろみ @ #x2146f052>
```

4.5.3 データの入力

生成されたインスタンス"イ1"に対して、4.4節で定義したメソッドを用いてインスタンス変数を対話的に入力する。

```
CG−USER(12): (酒母麹日時入力 イ1)
酒母麹日時を入力してください？ : 2015−9−1
"2015−9−1"
CG−USER(13): (酒母麹温度入力 イ1)
酒母麹温度を入力してください。: 33
(("2015−9−22_22:45" 33))
CG−USER(14): (酒母麹温度入力 イ1)
酒母麹温度を入力してください。: 33.1
(("2015−9−22_22:47" 33.1) ("2015−9−22_22:45" 33))
CG−USER(15): (酒母麹温度入力 イ1)
酒母麹温度を入力してください。: 33.3
(("2015−9−22_22:47" 33.3) ("2015−9−22_22:47" 33.1)
("2015−9−22_22:45" 33))
```

変化しないデータとしての酒母麹日時、変化するデータとしての酒母麹温度を入力した。後者は、逐次入力されるデータが、入力時刻と対になったリストデータで管理される。

4.5.4 データの参照

入力されたデータを参照するためには、クラス定義時に:accessorスロットで定義されたアクセス関数を用いて参照することが可能である。

```
CG−USER(16): (酒母麹日時? イ1)
"2015−9−1"
CG−USER(17): (酒母麹温度? イ1)
(("2015−9−22_22:47" 33.3) ("2015−9−22_22:47" 33.1)
("2015−9−22_22:45" 33))
```

このようにして、生成されたインスタンスに対して、データを入力することにより、それらのデータは保管されると共に、随時参照することが可能となる。

4.5.5 総合的シナリオ

前項から、イ1のインスタンスに対して、図9のアクティビティ図の経緯の結果についてのデータを全て一元的に管理可能となる。

インスタンスのイ1は、生産ロットに相当する。米を蒸し、麹を生成して酒母を育成し、3段仕込みを通じてもろみとし、発酵させて清酒とするプロセスに関する全てのデータは、ロット単位のインスタンスとして管理される。従ってこのインスタンスを通じて、種々の分析を行うことが可能となる。

さらに複数のロットを生成し、種々のシナリオを盛り込んで関連付けると、日本酒製造プロセスの合理化や生産計画などに反映させることが可能となるであろう。

5. 考察

5.1 概念レベルにおける柔軟なモデル化

以上、2章で酒造プロセスの概要的なモデル図化、3章で図示されたモデルの具体的なクラス定義とプログラム化手法、4章で日本酒製造モデルのCLOSによるプログラム化の実践について説明した。

UMLのアクティビティ図とクラス図を用いているが、一般のオブジェクト分析設計手法としてのUMLの活用法に比べるとより概念レベルでの活用である。本来であれば、日本酒製造過程の詳細な分析に基づき、利用者のニーズを明確化して要求仕様をまとめ、クラス図を作成するところであろう。そのような手法を取らない理由は、日本酒製造過程そのものが、必ずしも明確化されておらず、現場の杜氏さんの経験や勘に頼っている多様な側面が強いからである。

別の見方をすると、そのような経験や勘を試行錯誤で取り込める枠組みでプログラムを柔軟にしておくことが必要とされる。そのようなプログラムの枠組みとしてはCLOSが適合すると思われる。

5.2 オブジェクトの粒度と実用システムへの課題

4.5節で記述したシミュレーション手法は、特定の製造ロットの全情報を保有するオブジェクト（インスタンス）に対してメッセージのやり取りをすることにより、システムの状態を把握し、必要とする処理を行うシナリオである。従ってオブジェクトの粒度はかなり大きいものである。この情報を実行環境に常時生かして走行させるのはCLOSによるシミュレーションであれば良いが、C++やJavaのようなシステムの場合は必ずしも好ましいとは言えない。そのためには、インスタンス変数に相当するデータを永続的に管理するメカニズムが必要となる。そのためには、ファイルシステムかデータベースを必要とする。さらにWeb上のクラウドとして稼働させる場合は、サーバーアプリケーションとしてのミドルウエアの枠組みでシステム化する必要がある。

5.3 クラス継承機能の活用とオントロジへの期待

CLOSという非主流のプログラムが、概念レベルのモデル化に有効であり、クラス継承という大まかな枠組みを通

じてC++やJavaのような実用システム構築用の言語と関連付けられるのは大きなメリットである。その背景には、Simula、Smalltalk、Zetalispが導入した集合的概念のプログラム言語化を通じて1980年代に進展したオブジェクト指向プログラミングのクラス定義、クラス継承機能の実現がある。今回の検討はその機能の活用を通じてモデル化を実現した。プログラム言語ではないXMLにおいても、オントロジ言語のOWLはクラス継承の枠組みを提供する。将来的には、OWLを用いてCLOSの枠組みと連携させる技術的な手法の可能性を期待したいと考える。

5.4 IoTを通じた地域ビジネス開拓への貢献

本検討は、M2M・IoTシステムとしてのKojimoriの高機能化が出発点になっている。Kojimoriの語源はは麹守であり、まさに麹室における麹米の温度管理から命名されたシステムである。Kojimoriは現在、農業分野におけるビニールハウス、観光分野における温泉やスキー場などでも使用され、地域のビジネスに貢献しつつある。今回の手法を地域でのビジネス開拓に活用する可能性は十分にあり得ると考えており、今後は横展開を考えると共に、この手法の一般化を図りたいと考える。

6. 今後の具体的課題

6.1 オントロジ化・データベース化と分散システム化

実用システムは、C++やJavaを用いることになると思われるが、管理する情報のデータ構造はCLOSで検討されたクラスに基づいて設計すれば良いと考える。そのためには、図10のクラス図を検証した上で、ローカルなセンサー側とクラウド上のオントロジ的なクラス構成との機能分担を検討することが必要となる。その検討はCLOSのデータに基づきCommon Lispの環境でソミュレーションを行えば良い。その結果としてのシステム分割、クラウド上のオントロジ設計、オントロジ実装のための処理系の実装が課題になるであろう。なおオントロジ実装に関しては、openEHRによる電子カルテの仕様[6]を包含できるものとしておくことが望まれる。今後マイナンバーの普及に伴い、電子カルテの標準化が行われることは目に見えており、さらにその情報がPIMオントロジに関連付けられる可能性もある。IoTの本質は、存在するモノのクラウド上での管理と考えることができるので、オントロジとの連携は有効な技術となると思われる。

6.2 ACLプロトコルの活用

ローカルなセンサ側とクラウド上のオントロジ・データベースとの間の通信は、以前検討されたKQML（Knowledge Query Manipulation Language）やFIPA（Foundation for Intelligent Physical Agents）によるACL（Agent Communication Language）が参考になると思われる[7]。これらのプロトコルは、通信行為（Communicative Act）という概念でデータの送受信をカテゴライズし、その枠組みで通信するので、非常に柔軟で信頼性の高い通信を行うことが可能である。

ACL自体は、WebサービスのSOAP（Simple Object Application Protocol）、UDDI（Universal Description, Discovery and Integration）、WSDL（Web Services DescriptionLanguage）の局所的なアプリケーション依存版のようなものであるが、センサーデバイスのインテリジェント化のような用途には適合すると考えられる。

6.3 IoTと連携するロボット技術への適用

クラウド上のオントロジとセンサーデバイスとの連携の次に考えられるのは、アクチュエータとの連携であろう。要するに知能ロボットのアクションにクラウド上のオントロジを活用するということである。このような用途は、本来エージェント技術のコンソーシアムであったFIPA[8]が想定したACLの適用領域である。Physical Agentsという言葉からもロボットを想定していたことは明白である。

openEHRやPIMオントロジの活用を想定したロボットとなると、さまざまな用途が想定される。医療、福祉、介護はもちろん、家事、外出時の支援など幅広い用途が考えられるが、風呂敷を広げすぎるのも問題であろう。センサーや制御技術以上に、オントロジの構成や検証がキー技術となると考えられる。とりあえずは地域ビジネスを想定した個別のクラス階層の範囲で検討したいと考える。

7. おわりに

本検討を進めるにあたり、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感謝します。

参照情報・文献

[1] 鈴木芳行; “日本酒の近現代史: 酒造地の誕生 (歴史文化ライブラリー)”, 吉川弘文館（2015.4）

[2] 大野邦夫, Biro Attila, Hajdu Csilla, 広浦雅敏; ”異文化交流によるM2M・IoT製品Kojimoriの開発と市場開拓”,情報処理学会研究報告, DC98−17（2015.7）

[3] 日本醸造協会編; “最新酒造講本”, 公益財団法人 日本醸造協会 , p.71, (1996)

[4] 日本醸造協会編; “最新酒造講本”, 公益財団法人 日本醸造協会 , p.98, (1996)

[5] 日本醸造協会編; “最新酒造講本”, 公益財団法人 日本醸造協会 , p.130, (1996)

[6] 和田康, 大野邦夫；”オントロジモデルに基づく電子カルテとアーキタイプ”,画像電子学会第3回VMAワークショップ(2012.11)

[7] 大野邦夫；”エージェント通信と異文化コミュニケーションの類似性に関する検討”, 情報処理学会研究報告, DC99−1（2015.10）

[8] 大野邦夫；”FIPAエージェントにおけるXMLの適用動向”, 情報処理学会研究報告, DD23−3（2000.5）