Vol.2014-IS-129 No.6
2014/9/11

知識としてのマスタデータに関する一考察
― MDM エージェントを介したアクセスから見えてきたこと ―

森木洋† 児玉公信†

一般にマスタファイルは，商品マスタや取引先マスタのように，トランザクションデータから分離された被参照ファイルである．たとえば注文データの処理では，その処理プログラムは商品マスタや取引先マスタを読んで，注文データの妥当性を確認し，商品の単価を参照して金額を計算したり，取引条件を参照して納入日のルールによってその振る舞いを変えたりする．

本稿では，マスタデータを知識ととらえて，マスタデータに質問すると答えるエージェントを考える．たとえば，処理プログラムが注文の商品は 9 月 15 日に納入できると判定したのち，その取引先に納入可能日を尋ねるとする．その取引先の納入日のルールは，「月水金で，その日が休日の場合は直後の稼働日」となっていた場合，エージェントは 9 月 17 日と答えを返す．処理プログラムがルールを読んで解釈しないようにする．また，製造業における部品表や製造手順表を知識ととらえて，作るべき製品の製造手順をエージェントに尋ねるとする．エージェントは製品のオプションの組合せを解釈して一連の製造指示データを生成して返す．

このようなエージェントを MDM（マスタデータ管理）システムとして実装することで，“マスタ”の本質と実装のあり方について考察したことを述べる．

A Study on the Master Data as a Knowledge
- Inspired by accessing through the MDM agent -

HIROSHI MORIKI† KIMINOBU KODAMA†

1. はじめに

「マスタデータの統合」は企業情報システムにとって常に大きな関心事であり，IT 調査会社である ITR が毎年発表している「国内 IT 投資動向調査」1)2)3)においても，少なくともこの数年間は高い重要課題となっている．

「マスタデータの統合」を実現するための製品の総称である MDM（Master Data Management）は，文献 4)によると，「分散して 存在している複数の既存システムからマスターデータを抽出し，フォーマットを変換したり，欠損あるいは不正確なデータを修正するクレンジング（洗浄），そしてマスターの統合管理と配信」の機能からなる．

本報告では，上記のようなマスタデータの管理とは異なり，マスタ参照そのものの意味と，それが果たすべき機能をあらためて問い直し，クラウドコンピューティング環境上で実務に適用することを通して得た，新しい「マスタデータの統合」のあり方を提案する．

2. 研究の概要

2.1 研究の背景

筆者は，児玉 5)によって提案された企業情報システムアーキテクチャ（図 1）に基づいて，実際の基幹系システムの再構築を進めている．その過程で，階層化されたサブシステムを越えてどのようにマスタデータを一元管理し，データを提供するかは，ITR の調査結果と同様に，大きな課題となっていた．筆者はその解決策として，すべてのマスタデータを「知識」ドメインとして集約し，問合せメッセージに答える方式を採用することとした。

これはマスタデータを統合するという観点からは，従来の MDM に模されるが，メッセージによってマスタデータを提供するという点で大きく異なる．

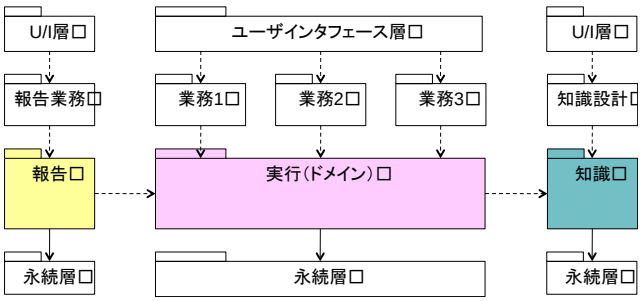

図 1 情報システムアーキテクチャ 5)

この「マスタデータを『知識』ドメインとして集約する」設計を以下では“新 MDM”と呼び，「メッセージによってマスタデータを提供する」方式を“MDM エージェント”方式と呼ぶことにする．このようなアーキテクチャが，膨大で高頻度なトランザクション処理において，実用に足る性能を発揮しうるのか，従来のマスタ検索方式に比べて，プログラム作成プロセスおよび運用プロセスにおいて，どのような利点をもつのかについて，マスタデータの再概念化，実装方式の構想，設計，実装を通して明らかにする．

†株式会社情報システム総研
Information Systems Institute, LTD.

2.2 マスタデータの扱い

まず，従来方式と MDM エージェント方式との違いを概観する．

(1) 従来の MDM におけるルールの扱い

従来のマスタデータは，端的に言うと，第三正規形の被参照表である．ユースケース実現である処理プログラム（クライアントとも言う）は，SQL などの検索機能を通して被参照表から参照したいキーに一致する組を取り出し，射影して読み出す．その中から目的の属性の値を“解釈”して，自分自身の振る舞いを変える．たとえば，取引先ごとに異なる商品の納入日を決める処理プログラムがあるとする．処理プログラムは，商品の在庫状況から 9 月 15 日に納入できると判定したのち，その取引先コードをキーにして取引先マスタのレコードを取り出し，“納入可能日”フィールドの値が“3”の場合は月，水，金曜日が納入可能であると“解釈”する．つぎに，9 月 15 日をキーにしてカレンダマスタを参照し，曜日を得たところ，月曜日でかつ休日であることが判明する．休日の場合の処置を判断するために，再び取引先マスタの“休日処置”フィールドを見て，その値が“2”で，直後の稼働日を納入日とすると“解釈”する．さらに翌対象日である 17 日が稼働日であるかどうかカレンダマスタを検索して，稼働日であることが判明したので，9 月 17 日を納入日と決める。

このように，マスタデータにはフラグや区分を持たせて，処理プログラムがその値（マジックナンバ）の意味づけを知っている．この結果，マスタ管理者は，新しい振る舞いの条件を追加するときは，プログラマと相談した上で，プログラムとデータを同時に変更しなければならない．マスタデータにはますますマジックナンバが増えていき，処理プログラムはますます煩雑になる．マスタデータだけをみても意味が分からないので，アドホックな対応によりデータの品質が低下する（たとえば，類似のフラグデータが増える）．この問題に対処するには，データとプログラムの両方を同時に正して行かなければならない．

マスタデータを MDM 機能によって集中管理しようが，処理プログラムの複雑化やマスタデータの品質低下の解決にはならない．

(2) MDM エージェントによるルールの扱い

MDM エージェント方式では，処理プログラムが 9 月 15 日以降の妥当な納入日を取引先マスタに問い合わせると，その取引先の納入日のルールは「月水金で，その日が休日の場合は直後の稼働日」となっているので，エージェントはカレンダマスタに問い合わせて，9 月 17 日が妥当な納入日であると回答する．処理プログラムがルールを読んで解釈することはしない．この意味で，マスタは処理プログラムから見て「知識」源である．

これは，製造業における部品展開や工程展開も同様である．ものづくりの知識（一般には，部品表や製造手順表，資源表など）をマスタに記録しておき，作業展開を MDM エージェントに依頼すると，エージェントは製品のオプションの組合せを解釈して適切な一連の製造指示データを生成して返す．

つまり，マスタデータを単なる被参照表として捉えるのではなく，「実行」ドメインの振る舞いを制約する「知識」として再概念化する．新しい MDM は「知識」の「容れ物」であり，MDM エージェントは「知識」問合せの窓口（facade）であり，その実装を隠蔽する役割をもつ．

2.3 研究設問

本研究のテーマは，上記のように再概念化したマスタデータの記録とアクセスの仕組み（新 MDM）において，属性値に格納されたさまざまなルールの解釈を MDM エージェントが行うことによって，処理プログラムはより良く機能するかを問うことである．これが実証されれば，多くの処理プログラムの機能は一律化されることで開発生産性と品質は向上するだけでなく，情報システムの開発中にマスタデータの構造変更などがあっても開発へのインパクトを最小化でき，運用段階に進んでもルールの変更だけで，安全に情報システムの振る舞いを変えられることになる．

2.4 研究の方法

新 MDM を設計し，クラウドコンピューティング環境上で，Java，JSON，PostgreSQL，Hibernate（O/R Mapping）を用いて実装し，負荷テストを通過することを通して，研究設問に答える．また，取引先（契約）マスタとカレンダマスタを対象に行った MDM エージェントの実装の内容と，その過程で得られた知見を述べる．

3. 新 MDM の設計

3.1 概念モデル

新 MDM の概念モデルは図 2 のとおりである．ステレオタイプ《履歴》については後述する．

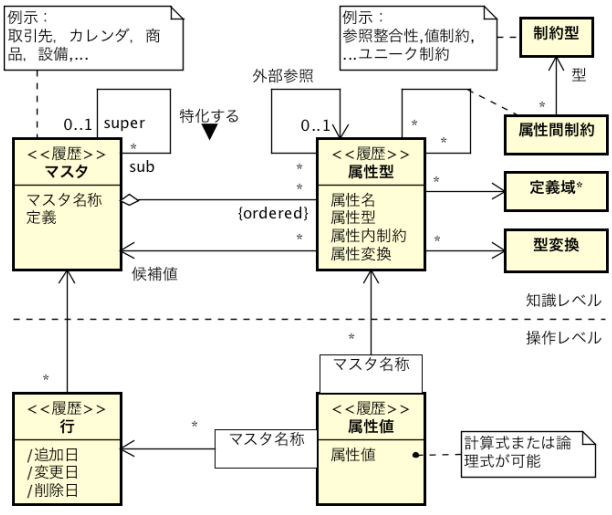

図 2　新 MDM の概念モデル

3.1.1 観測パターン

この基本構造は，マスタ名称，属性型のメタデータ（知識レベル）と，それぞれに対応する行，属性値の値データの構造（操作レベル）からなる．これは，アナリシスパターン 8)の「観測パターン」に基づいている．

ある「マスタ」は 0 個以上の「属性型」を持つ．「マスタ」と「属性型」はそれぞれ，データベースでいう表と列の定義に相当する．一方，「マスタ」は 0 個以上の「行」を持ち，それは「属性型」に対応して「属性値」を持つ．この 4 つのクラスからなる構造は，データベース管理システムにおける表定義のモデルそのものである 9)．屋上屋を重ねるようだが，マスタデータの版管理，属性型の動的追加・変更，マスタ登録時の入力補助のための候補値提示，入力値の検査を実現する機能を追加するための方便である．

このモデルのインスタンス図を図 3 に示す．これは商品マスタと商品群マスタが関連している例である．これらのマスタを参照する処理プログラムからは，右の「商品群」–「商品」クラスのように見える．

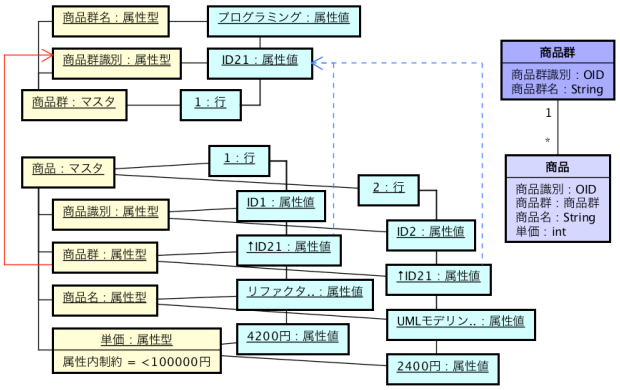

図 3　新 MDM のインスタンス図

3.1.2 属性型

(1) 定義域

「定義域*」は属性がとる型の宣言である．「文字列」，「0 以上 100 万以下の整数」など「属性値」が取りうる値を型宣言または論理式で定義する．たとえば，「0 以上 100 万以下の整数」のようにユーザが定義することもできる．

(2) 型変換

属性値はすべて Sting 型で永続化される．処理プログラムに対しては，属性クラスで定義された定義域の型に変換して渡す必要がある．型変換は default の変換先の型を定義する．この方法は，マスタ登録時に定義域の制約チェックを通ったデータが格納されていることで可能となる．

(3) 候補値

候補値は，属性がとりうる値の列挙である．マスタ登録のための入力補助機能（登録画面ではプルダウンメニューで利用する）であり，候補値の中のどれかでなければならない場合と，入力のガイドとして使う場合とがある．本来は，他マスタの属性のうち，人にとって readable な候補キーを用いるのがよいが，実装上は，候補値リストを別個に用意する．

(4) 属性内制約，属性間制約

属性内制約は「単価は 100 円以上である」など，一つの属性に関する制約記述であり，登録時の妥当性検査に用いる．属性間制約は他の属性（マスタをまたがっても良い）間の関係における制約記述であり，たとえば，「契約.値引き額は，商品.標準原価の 50%を越えない」のような制約も属性間制約となる．これも，登録時の妥当性検査に用いる．

(5) 外部参照

自マスタまたは他マスタの属性値を参照する関係を定義する．この参照関連は，マスタどうしが構造を持つ状況を記述するために用いられる．図 3 のインスタンス図では「商品群」と「商品」がこの構造をしており，「商品」から「商品群」への外部参照をもっていた．

(6) 特化関係

特化は，ある（sub）マスタが他のマスタ（super）を拡張（継承）する関係にあることを記述する．MDM エージェントは super のマスタに対する処理プログラムからの要求に対して，その拡張である sub のマスタを含めた参照機能を提供する．たとえば，「商品」マスタがあり，扱う商品の種別により格納する属性が異なるため，特化した「種別商品」マスタが作成される．MDM エージェントは MDM 処理プログラムからの「商品」マスタに対する参照要求に対して，ポリモーフィックに対応する．

3.2 新 MDM の機能

3.2.1 マスタ参照とインタフェース

従来の MDM は，サブシステムごとに必要なマスタデータを適当なタイミングで“配信”し，再現することで従来の処理プログラムのアクセス方法を変更しないようにする．

新 MDM は，すべてのマスタ参照をメッセージの受け渡しで実現する．そのために，MDM エージェントは汎用の API（Application Program Interface）を提供する．マスタ参照は Read Only なので，アクセス要求の増減に対しては，エージェントのインスタンスを動的に増減することで対応できる．アクセス API の実装は次のとおり．

(1) 値取得について

- `public Map getMasterRowValueByPK(String masterName, Collection target, int mode, Map pkVal)`

  説明：指定した PK 値に該当する指定した属性の属性値を取得する．属性値は String 型または定義域の型での取得が可能となる．新 MDM における PK 値はテーブルの PK 値の意味ではなく行を一意に識別する値となる．

  ➢　パラメータ：

masterName：マスタ名称
target：取得対象の属性名
mode：型指定．0:String，1:定義域の型
pkVal：PK 値．Map<属性名：属性値>で渡される

➢ 戻り値：
属性値

- public Map getMasterRowValue(String masterName, Collection target, int mode, Map where)
説明：指定した検索条件に該当する指定した属性の属性値を取得する．属性値は String 型または定義域の型での取得が可能となる．

➢ パラメータ：
masterName：マスタ名称
target：取得対象の属性名
mode：型指定．0:String，1:定義域の型
where:検索条件．Map<属性名:属性値>で渡され，複数の場合は AND 条件として評価される

➢ 戻り値：
属性値

(2) 式解釈について

- public Map evalXXXXX(Map target, Map param)
説明：指定したターゲットの式値（式を値に持つ属性値）の指定したパラメータにおける式解釈結果を取得する．式ごとに API を提供する．XXXXX は式名称となる．本来的には式解釈の API は 1 本になると考えている．今後の開発を通して，API の設計を見直す予定である．

➢ パラメータ：
target：式値の特定情報（2.2(2)の例であれば取引先識別情報となる）
param：式のパラメータ（2.2(2)の例であれば不要となる）

➢ 戻り値：
式解釈結果

3.2.2 履歴管理

トランザクションデータが，ある時点の事実の記録であることに比べ，マスタデータはトランザクションとは非同期に，変化（進化）するものである．そのため，変更履歴管理が必須であり，トランザクションデータの時点と版との適切な対応を保証する必要がある．

(1) ステレオタイプ《履歴》の意味

図 2 の概念モデルにおいて，履歴管理の仕組みはクラスにステレオタイプ《履歴》と記すことで暗示していた．このステレオタイプの定義は図 4 のとおりである．このステレオタイプを持つクラスのオブジェクトは，変更「Event」を持ち，処理プログラムが要求する参照日時とマスタ Object の発効日時を基に参照すべき Object を決定する。

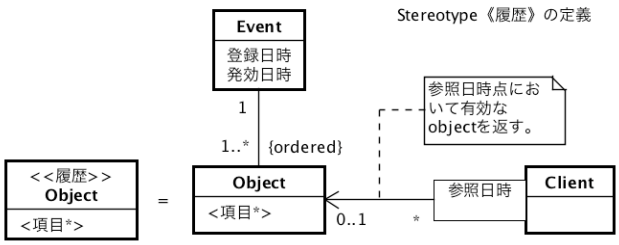

図 4 概念モデルにおけるステレオタイプの定義

(2) 変更日と参照日

履歴管理機能について，タイムセールのためにマスタ変更を行うシナリオの例で示す（図 5）．

商品名「UML モデリングの本質」の単価は，2011 年 5 月 30 日時点で 2400 円となっている．現在は 2014 年 9 月 11 日とする．20 日後の 2014 年 10 月 1 日の一日だけ，これを 1200 円で販売することにした．本日を登録日として，適用日を 2014 年 10 月 1 日から 1200 円の変更と，適用日を 2014 年 10 月 2 日から 2400 円の変更をそれぞれ登録する．

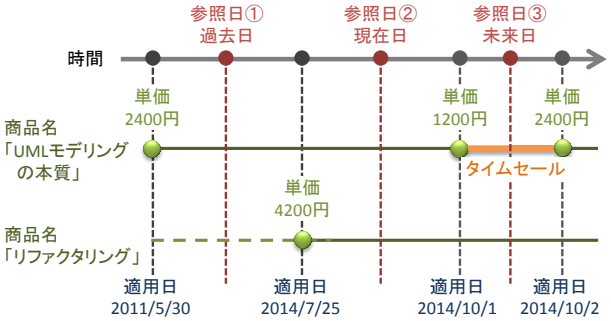

図 5 履歴管理の例

処理プログラムが商品マスタを参照する際，参照日が 2014 年 10 月 1 日の場合のみ，単価が 1200 円と返され，それ以外の参照日では 2400 円と返される．この版管理は商品ごとに行われるので，商品名「リファクタリング」は，参照日が 2014 年 7 月 24 日以前では存在しないことになっている．

この例で示すとおり，未来日で適用となるマスタを予め登録しておくことで，現在と過去だけでなく未来の時点におけるマスタデータが取得できる．

3.2.3 MDM によるルール解釈の有効性

新 MDM の概念モデル（図 2）の属性値型の注釈で記述したとおり，属性値に論理式の登録が可能である．処理プログラムの処理プログラムからのアクセス要求メッセージに対応して，MDM エージェントはこの論理式を解析し，変数を束縛して解釈し，評価してその結果を返す．この具体例については，すでに 2.2(2)で述べた．

従来の MDM ではルールが変更となる度に処理プログラムのルール解釈部分の改修が必要となるが，新 MDM では処理プログラムにてルール解釈は行わないため処理プログラムの改修は不要であり，MDM エージェントのプログラ

ムにおいてもルール表現が変更しない限り，ルールが変更されてもプログラムの改修は不要となる．実際にルール変更時にもプログラム改修が不要となることは，新 MDM の実装例で確かめる．

3.3 属性の追加および属性変換の変更

新 MDM において，あるマスタに属性型を追加することは容易である．また，属性の変換ルールを変更することも容易である．これは，従来の MDM もほぼ同様である．

4. 実装時の考慮点

MDM の骨格である観測パターンは，登録，検索ともに，性能面での配慮が必要となる．4 つのクラスごとに履歴を管理し，リンクを張り直す処理は，意外と煩雑である．現時点では，「属性値」クラスを「行」クラスと一緒にして「行値」クラスとし，マスタ登録時に行としてデータを保持する対策を施している．これにより，クラスが一つ減ることと，参照時に行の組み立て処理が不要（その代わり属性値の分離処理が必要になる）となる分，性能が向上している．性能対応した実装モデルを図 6 に示す．

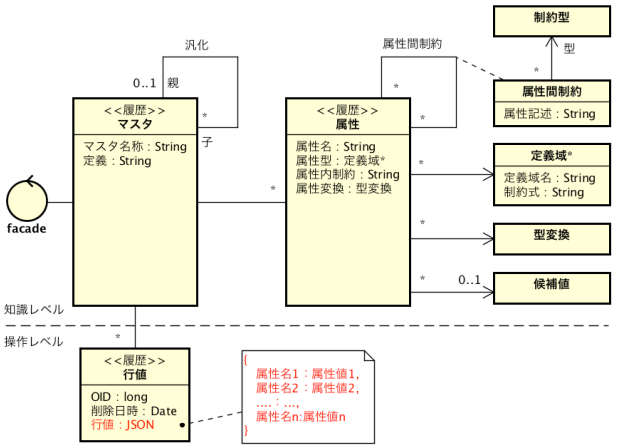

図 6　新 MDM 性能対応仕様モデル

「行値」クラスの行値は JSON 形式で持たせることで，属性値の分離が容易に実現できる．この実装におけるトレードオフポイントとしては，以下が考えられる．

(1) 属性値単位の履歴情報の欠落

行単位での履歴管理となるため，どの属性値が変更されても改版されることになる．ただし，このことが実際上の問題となることはほとんどないと思われる．

(2) 属性値の重複記録

行単位の履歴管理となるため，更新された属性値だけでなく更新されていない属性値も複製して次の世代の行として記録される．属性数が多く，ある特定の属性のみが頻繁に更新されるような状況では，複製コスト（メモリ，DB 領域の圧迫）が大きくなる．これに対しては，更新されていない属性値の複製を止め，マスタ取得時に複数世代から合成する方法が考えられるが，マスタ取得時の性能とのトレードオフとなる．

5. 新 MDM の実装例

5.1 システム要求事項

新 MDM の実装サンプルとして「カレンダ」を取り上げる．「日」には顧客の営業日，工場の稼働日，請求の締日，自社の決算日などさまざまな「日」が存在するが，「日」のルールを持たない企業はなく，その情報システムは「日」のルールに依存して構築されている．「カレンダ」は「日」を扱うマスタであり，研究設問に対する良い実装サンプルとなりうる．

カレンダマスタの要件は次のとおりである（図 7 参照）。

- カレンダの定義
  カレンダとは，グレゴリオ暦を暦法として任意の 1 年の中で複数の観点で「日」を定義したものである．
- カレンダの構成
  - 「カレンダ」は「所有者」を持つ
  - 「日」は，「稼働日」と「特定日」の観点を持つ
  - 「稼働日」にはそのタイプとして「国民の休日」と「各組織の休日，休出」がある
  - 「各組織の休日，休出」は「国民の休日」をオーバライドする
  - 「特定日」は，「稼働日」を前提として定義される（特定日の定義に「稼働日」，「不稼働日」を定義言語として利用）（同様に「国民の休日」は「国民の祝日」を前提として定義されるが説明は省略する）
- カレンダの責務
  カレンダは自身の「日の定義」を解釈し，その実行結果を提供する．提供される「日」は処理プログラムの問合せ内容に応じて変化する．

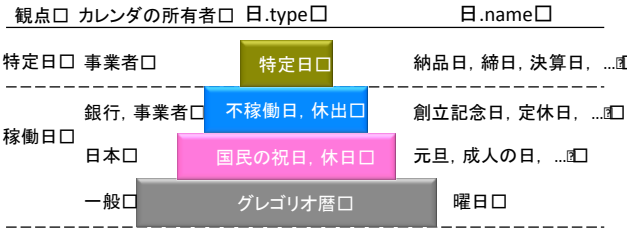

図 7　カレンダの知識構造

5.2 設計

5.2.1 知識表現

システム要求事項の分析から，カレンダの本質的なデータは「日の定義」であることを得た．マスタとして「日の定義」を登録するため，「日の定義」方法を設計した．

日は，「年（year）」「月（month）」「日（day）」「曜日（day of week, dow）」および「不稼働日（holiday）」で特定される．これらを次のように，Key-Value 形式（表 1）で表現する．

表 1　カレンダ知識表記ルール

key	value	value 例
type	日のグループ名	祝日，特定日
name	日の名前	元旦，納品日
month	月の特定（正規表現）	1-12, *
day	日の特定（正規表現）	1-31, *
dow	曜日の特定（正規表現）	0-6, *
holiday	Month & day & dow の指定が不稼働日の時の代替日特定	preWorkingDay, nextWorkingDay

5.2.2 MDM エージェント API

分析によりカレンダの責務は，「日の定義」の解釈とその実行であることを得た．MDM エージェントが提供する API は，カレンダが持つ「日の定義」とそれを利用する処理プログラムにより設計されていくが，今回の実装では「稼働日」の解釈と実行を対象に，次の API を設計した．

- `public Date getPrevWorkingDay(Owner calOwner, int rel)`
  説明：カレンダの所有者と起点を提示し，起点から一番近い（起点を含む）過去の稼働日を求める．
  - パラメータ：
    `calOwner`：カレンダの所有者
    `rel`：起点．本日を 0 とした本日からの経過日数で表記し，未来をプラス，過去をマイナスで表現する．（例：0:今日，+1:明日，-1:昨日）
  - 戻り値：
    直近の過去の稼働日
- `public Date getNextWorkingDay(Owner calOwner, int rel)`
  説明：カレンダの所有者と起点を提示し，起点から一番近い（起点を含む）未来の稼働日を求める．
  - パラメータ：
    `Owner`：カレンダの所有者
    `rel`：起点（getPrevWorkingDay と同様）
  - 戻り値：
    直近の未来の稼働日

5.3 実装

5.3.1 知識の実装

実装プログラミング言語には Java を，知識表現の言語には JSON を採用した．下に JSON で記述したマスタデータ（知識）の例を示す．

1 月 1 日

```
{
  "type":"祝日",
  "name":"元旦",
  "month":"1",
  "day":"1",
  "dow":"*"
}
```

1 月の第 2 月曜日

```
{
  "type":"祝日",
  "name":"成人の日",
  "month":"1",
  "day":"8-14",
  "dow":"月"
}
```

月水金で，その日が休日の場合は直後の稼働日

```
{
  "type":"特定日",
  "name":"納品日",
  "month":"*",
  "day":"*",
  "dow":"月,水,金",
  "holiday":"nextWorkingDay"
}
```

5.3.2 知識の展開

マスタデータとして管理するのは「日の定義」であり，その内容は前述のとおり JSON 形式の文字列とする．エージェント API が応答を返すためには，知識の展開（「JSON 文字列の Java オブジェクトへのパース」および「カレンダ知識表記ルールのパース」）が必要となる．

実装においては性能が主要な関心事となるが，本件では知識の展開タイミングが考慮点となる．知識の展開タイミングは大きく「知識登録時」と「MDM エージェント API 利用時」の 2 つに分けられる．当然であるが，MDM エージェント API の応答は「知識登録時」において知識展開した方が速くなるが，製造手順展開のような複雑な知識の扱いは，必然的に「MDM エージェント API 利用時」となり性能に対するさまざまな工夫が必要となる．

6. 結論

6.1.1 研究設問に対する結論

今回の実装を通して，新 MDM において，属性値に書かれたさまざまなルールの解釈を MDM エージェントが行えることを確認した．これによって，処理プログラムをシンプルに実装できることを確認できたが，研究設問に対しての確認は十分ではない．今後，マスタを利用している様々な処理プログラムに対して MDM エージェント方式を適用することで，引き続き MDM エージェント方式の有用性を確認していく．

6.1.2 得られた知見

(1) 成長するマスタ

図 2 で示したとおり観測パターンとしてマスタを持つことにより属性の追加が容易となる．また，MDM エージ

ェントを設けることで実行ドメインのアプリケーションが直接マスタに依存しなくなり，属性の追加がさらに容易となる．

これにより，マスタ構築作業において，既存マスタデータを消して良いかの判断がつかないから移行するのではなく，実行ドメインのユースケース実装により必要性が出てきた段階で追加するというアプローチをとることがきる．

基幹システムの再構築において，既存マスタデータの精査は非常に大変で，その必要性を明らかにするだけで多くの工数をとられることが普通であるため，本アプローチがとれることの意味は大きい．

(2) 情報システムアーキテクチャの重要性

システム開発の現場にいると「このマスタが何に利用されているかわからない．変更したらどんな影響があるかわからない」といった話をよく聞く．これはマスタデータの解釈を処理プログラムに任せてきた結果であると考える．別の言い方をすれば，そのように情報システムを設計した結果であると言える．

MDM エージェント方式により情報システムを構築することで，上記の問題を解決できると考えている．これは，情報システムアーキテクチャの重要性を示す事例にもなる．今後の研究対象としていきたい．

7. おわりに

今回の実装を通して，マスタを単なる基本データとして捉えるのではなく知識として捉え，MDM エージェントを介したマスタ（知識）アクセスを行う情報システムの構築が有用であるとの考えを強くした．

今後は，提示したモデルの実装をすすめ，実際の業務運用での使用を通して，その有効性を検証したい．

また，散在している知識を統合すること，それを論理式で記述すること，その知識の妥当性を検証することの必要性を認識している，そのためのツールとして，知識エディタ，シミュレータは必須である．知識が即時に実行ドメインに連動する仕組みであるため，知識の定義ミスは即時に情報システムの異常事態を招くことになる．新たに設計した知識による実行ドメインの挙動を事前に確認するシミュレータの機能が必要であり，この仕組みを考えていく予定である．

参考文献

1) アイ・ティー・アール：「主要な IT 動向に対する重要度指数と実施率の変化」in IT 投資動向調査 2011，入手先 <http://www.itr.co.jp/company_outline/press_release/101126PR/index.html/>（参照 2014-08-06）
2) アイ・ティー・アール：「主要な IT 動向に対する重要度指数と実施率の変化」in IT 投資動向調査 2013，入手先 <http://www.itr.co.jp/company_outline/press_release/121205PR/>（参照 2014-08-06）
3) アイ・ティー・アール：「主要な IT 動向に対する重要度指数と実施率の変化」in IT 投資動向調査 2014，入手先 <http://www.itr.co.jp/company_outline/press_release/131203PR/>（参照 2014-08-06）
4) IT Leaders：「マスターデータ統合の方法」in ERP 中心から汎用指向、SOA 基盤まで—主要 MDM 製品の特徴を見る，入手先 <http://it.impressbm.co.jp/articles/-/6154>（参照 2014-08-06）
5) 児玉公信：「計画・実行システムの一般モデル：生産管理システムと金融業務システムの共通性」，情報処理学会研究報告，IS-109, 2009-09-14.
6) 児玉公信：「情報システム設計における概念モデリング」，人工知能学会誌 25(1), 139-146, 2010-01-01.
7) 児玉公信:「UML モデリングの本質 第 2 版」, 日経 BP 社, 2011.
8) Fowler, M.，堀内 一監訳：「アナリシスパターン」，ピアソンエデュケーション，1998.
9) C.J.Date：“Database in Depth Relational Theory for Practitioners”.株式会社クイープ訳：「C.J.Date のデータベース実践講義」，オライリー・ジャパン，2006.