Vol.2013-DPS-157 No.11
2013/10/18

# エージェント指向システムによる情報システム構築のための エージェントに対する要求仕様

藤田 茂[1,a)]

概要：ソフトウェア開発時にオブジェクト指向方法論に基づいて開発することが，一般的になってきている．例えば，要求仕様作成時や設計仕様作成時に，UML を用いる．また，多くのプログラミング言語にオブジェクト指向の枠組みが取り込まれ，ソフトウェア開発者にとってオブジェクト指向開発は，ソフトウェア開発時の基本的な考え方になっている．Shoham は，オブジェクト指向プログラミングを進めモデルとして，エージェント指向プログラミングを提唱し，Agent-0 を作成したが，オブジェクト指向ほどは普及しなかった．しかし複雑化，高度化する計算機とネットワーク環境下において，利用者の要求に対して迅速にサービスを構築するためには，エージェント指向方法論が一助となると考えた．本稿では，エージェント指向方法論確立のために，エージェントのモデルを定める．

## 1. はじめに

山本は，エージェントプログラミングによって，スケーラビリティの高い分散処理処理システム開発方法論を確立し，FX(外国為替証拠金取引) を取り扱う業者向けにフレームワークを提供している [2]．この開発方法論は，オブジェクト指向から発展した自律性，永続性を備えたエージェントモデルに基づくものである．山本によれば，「顧客であるシステムエンジアは，エージェントという考え方にすぐに馴染みます」ということであり[*1]，一見するとバズワード化していたエージェント技術が既に，高い応答性を要求される FX 取引という場において，実用に供され，研究者のみならず，開発者にも受け入れられていることが明らかになった．山本の開発方法論では，エージェントは顧客にほぼ 1 対 1 に対応する要素である．また，Java VM 上で動作している．

Shoham は，“Agent-oriented Programming”[1] によって，“mental state” を持つエージェントを基本単位とする，エージェント指向プログラミングを提案し，Agent-0 によって提案が実現可能であることを示した．Shoham の提案によるエージェント指向開発方法論でのエージェントは，山本のエージェントのモデルに対して，この “mental state” が存在することや，メッセージ形式が定められていること，メソッドの実行に際して一貫性や誠実性定められているというモデルとしての制約が存在する．この結果，エージェント指向のモデルとしては，Shoham のモデルが，より明確に定義されていると言えるが，実務プログラミングには向かなかった．

ソフトウェア開発，プログラミングの場において，関数型言語や定理証明器をベースとするプログラミング言語に注目が集まっている．例えば，Haskell や coq である．これは，大規模化，複雑化する分散処理システムの設計と開発に貢献すると期待されているからであり，通信システムの開発等に利用されている．

通信システムの形式仕様記述言語として，LOTOS, Z, CHILL が存在するが，仕様記述と検証に留まっており，プログラミング言語としては成立しなかった．

Shoham の AOP: Agent-oriented Programming と山本のエージェント指向開発法論の差，LOTOS などの形式仕様記述言語と Haskell, OCAML などの関数型言語の差は，開発者自身が行うデータ構造の定義に対する厳格な要求あると思われる．前者がより一貫性を求めているのに対して，後者は Ad hoc なデータ構造を許容するか，型推論の機構を持つ．

JADE は，メッセージ形式とプロトコルの提案によって，エージェント研究者のシステム開発フレームワークとして利用されている．しかし，データ構造に関しては標準が提案されていない．JADE の前身として，直接の関係は無いが FIPA によるエージェント間通信の標準化が行われ

[1] 千葉工業大学情報科学部
Faculty of Information and Computer Science, Chiba Institute of Technology, 2-17-1, Narashino-shi, Chiba-ken, 275–0016, Japan
[a)] fujita@cs.it-chiba.ac.jp

[*1] JAWS2013 でのポスターセッションの場での質疑応答による

ていた．

W3C の提案する Semantic Web, Linked Open Data は，これに対して，データの構造 RDF を定義している．また RDF への問い合わせ言語 SPARQL が存在し，データへの問い合わせ/更新が一定の形式で行えるようになっている．しかし，エージェントの情報を，データベースのように更新してしまうことは，エージェントの動作履歴の一貫性維持に問題があると考える．

人工知能分野では，推論結果の系の変数が全て真であることを保守する TMS: Truth Maintenance System と，系の変数間に矛盾が無いことを保守する ATMS: Assumption-based Truth Maintenance System が知られており，分散する情報の一貫性維持に用いることが可能である．しかし，RDF 更新を行う時の問題と同じく，情報が書き換えられる前と情報が書き換えられた後の操作がモデルに取り込まれていないので，動作履歴の一貫性維持に問題があると考える．

本稿では，上述の考察に基づいて，自律的，永続的，知的なエージェントを基本要素とするエージェント指向システムによる，情報システム構築のために必要な，エージェントに対する要求仕様を述べる．

## 2. エージェント指向開発方法

### 2.1 メッセージ形式，データ構造

Shoham による AOP の提案では，Object Oriented Programming との構成要素の差は，

(1) 基本要素の状態を定義する語
　信念 (beliefs)，コミットメント (commitments)，選択 (choices) など
(2) メッセージの型
　通知 (inform)，要求 (request)，提案 (offer)，拒絶 (decline) など
(3) メソッドの制約
　誠実性 (honesty)，一貫性 (consistency) など

にある [1]．本稿でも，開発法論として AOP を踏襲する．

’はじめに’で述べたように，メッセージの型については，すでに FIPA/JADE 等で提案が行われ，エージェント研究者によって複数の試作システムが作られるなど，一定の実績があると思われる．そこで本稿では，エージェント間の通信プロトコルの再定義や，新規のメッセージ形式を提案することは行わない．

また，エージェント間通信で用いられるデータ構造については，既に LOD でも用いられている RDF が事実上の標準となっていると考えられることから，新規にデータ構造を定義することは行わない．

一方で，エージェントが新たな情報を獲得した後の動作について，AOP では単に “心的状態の更新 (Update mental state)” とかかれているのみであって，個別のエージェントに対する設計指針やエージェントのアーキテクチャについて記述がなかった．

本稿では，エージェント指向開発方法論を定めるためには，エージェントのアーキテクチャを新規に定めることが必要であるとして，以下にその要求仕様を示す．

### 2.2 Shoham のエージェントモデル

まず始めに，Shoham が AOP[1] の中で示したエージェントインタプリタの動作フローを図 1 に示す．

図 1 に示した要素は，以下の 7 項目である．

(1) Initialize mental state and capabilities
　心的状態と能力の初期化を行う
(2) Define rules for making new committments
　新しいコミットメントを行うルールを定義する
(3) Clock
　エージェントの動作を駆動する
(4) representation of mental state and capablity
　現時点のエージェントの心的状態と能力の情報を保持する
(5) Update mental state
　rule に基づいて，incoming message, (4) を使って，(4) を更新する
(6) Execute commitments for current time
　現時点に実行するべきコミットメントを実行する
(7) incomming message, outgoing message
　エージェント間通信を実現するメッセージ

エージェントの基本動作は次の 2 つのステップからなる．

(1) エージェントインタプリタは，メッセージを読み込み，心的状態を更新する．
(2) 現時点のエージェントが実行するべきコミットメントを実行する．この結果，心的状態が更新される，またメッセージを他のエージェントに送信する．

Shoham のエージェントインタプリタのモデルでは，mental state の更新がエージェントの振る舞いを決定しているにも関わらず，その仕様の細部はエージェント作成者の設計に委ねられており，エージェント指向プログラミングを行うためには，より詳細なモデルを示す必要があると考える．

### 2.3 情報システム構築のためのエージェントモデル

Shoham のエージェントモデルの中の，心的状態更新を次の 5 ステップで実行するように定める．

(1) representation of mental state and capability の保存を行う．このとき，時刻 t-1 であるとすると，MSC(t-1) と表記する．
(2) 時刻 t における incomming message の取り込み．このとき，incomming message の集合を IM(t) と表記する．
(3) IM(t) と MSC(t-1) を Rule によって処理し，新しい

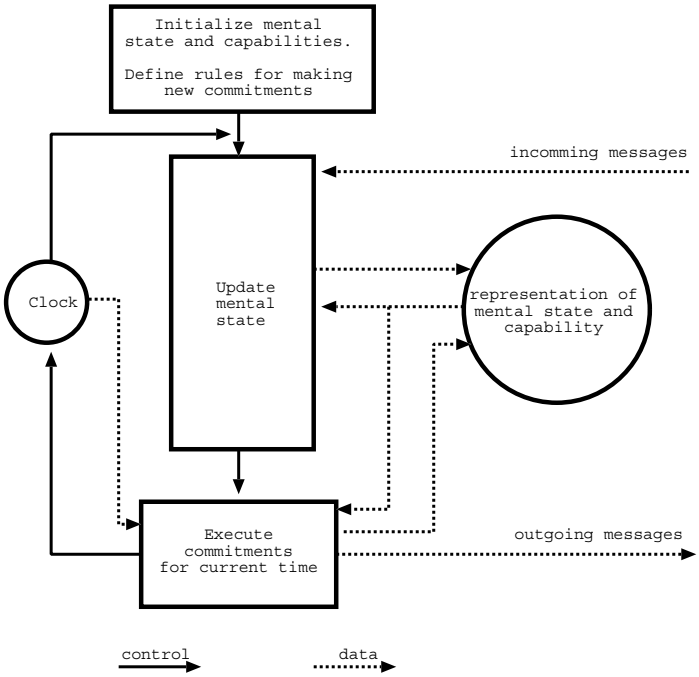


図 1　エージェントインタプリタ [1]

representation of mental state and capability, MSC(t) が更新される．

(4) MSC(t) に基づいて，コミットメントを実行する．これを C(t) と表記する．

(5) C(t) によって，新しい MSC(t+1) が更新される．

さらに，エージェントの心的状態 MSC の更新に際して，矛盾が生じた場合，これをなんらかの形で解消する必要がある．しかし，Shoham の AOP では，この矛盾に関しては言及が無く，AOP の作成者に委ねられている．

情報システムの動作中に発生する矛盾は，ソフトウェア工学的な観点からは要求分析が不十分であるとされ，システムエンジニアの対応が必要である．しかし，実システムが既に稼働している状況下においては，矛盾に気づくこと無く処理が継続され，情報システムの外部からの指摘によって，設計時の要求分析あるいは要求獲得が不十分であったことに気づく．

ここで，システムエンジニアが再設計を行ってシステムを改修するか，プログラマが稼働中のコードを改修してシステムを改修する．しかし，この工程は，利用者にとって稼働中のサービスを一旦停止し，再度の実行を伴うことから，大きな負担となる．

一方で矛盾の発生を情報システムの側が自律的に行う，あるいは，さらに推し進めて矛盾解消の手立てを情報システム側が行うことをエージェントモデルの中に取り込む．

### 2.4 矛盾発見時の処理フロー

データベースの処理において，トランザクションに失敗すれば，その処理自体をなかったものとしてロールバックする．しかし，エージェント指向モデルに基づいて，処理を行った場合，心的状態 MSC(t) が更新されてしまえば，C(t) に基づいて outgoing message は他のエージェントに配送され，そのメッセージを受け取ったエージェントの MSC(t) も更新されてしまう．エージェント指向モデルで情報システムを構築した際には，全体のロールバックという処理を組み込むことは出来ない．

エージェント単独の処理の中で矛盾が発見されるが，この矛盾対応処理について，以下のように定める．

(1) 矛盾発見
  (a) コミットメント C(t) によって変化するであろう MSC(t+1) を予めエージェントが作成する．これを MSC'(t+1) と表記する．
  (b) C(t) が実行された後に，MSC'(t+1) と実際に変更された MSC(t+1) を比較する．このとき差が大きければ，矛盾の発見である．

(2) 矛盾発生の推定
  (a) MSC'(t+1) を生成したルールを原因とする．
  (b) MSC'(t+1) の生成要因となった incomming message を原因とする．
  (c) MSC(t+1) を生成した C(t) の元となったルールを原因とする．
  (d) (2a から 2c) の組み合わせを原因とする．
    (i) (2a) と (2b) の組み合わせ
    (ii) (2a) と (2c) の組み合わせ
    (iii) (2b) と (2c) の組み合わせ
  (e) 矛盾発生の推定を放棄する

(3) 矛盾解消の試行
  (2) で推定された原因を解消するための動作をコミットメントするルールを実行する

上述のフローに類似の概念として，仮説に基づく動作を実行できるインタプリタとして，Agent-1 の提案が [1] に存在するが，実現はされていない．また，エージェントが incomming message を全て真として取り込んでしまうことは，[1] でも意味論とアルゴリズム上の課題であることが指摘されている．incomming message を取り込んだ後に発生する矛盾については，上述のフローで対応可能である．仮説に基づく動作についても，コミットメントとして，部分的な MSC(t) のコピーを作成し，その MSC(t) にエージェント自体ののルールを適応することで対応可能である．

### 2.5 フレームワーク

## 3. 例題

本提案モデルを検証するために，矛盾や誤りが発生するシナリオを取り扱う．Agent-0 においては，航空券予約システムの動作を示しているが，この例では矛盾が発生する要因が少なく，また航空券予約システムで発生する不具合は，矛盾というよりは誤りであって容易に解決可能な課題である．

そこで本稿では，本格的な情報システム構築では無く，矛盾の発生しやすいゲームをプレイする情報システムを取り扱う．このとき，プレイヤーには人間の参加者とエージェントの参加者混在するものとする．

ゲームの対象として，人狼と Barnga を検討している．それぞれのゲームの詳細ついて，ここでは述べないが，どちらも完全ゲームではない．このためゲームの進行は偶然に左右される要素もあるが，推論によってゲームが進行する．

また，人狼においては，プレイヤーが生成する仮説の中に，他のプレイヤーから自分がどのように思われているかを取り込む必要がある．例えば，自分が人狼では無いかと疑っているプレイヤーから，自分こそ人狼ではないかと疑われているという状況がある．そのような状況を仮説として表現し，他のプレイヤーに受け入れ可能な新たな仮説を示す必要がある（あるいは，あえて発言をしない，という選択をする必要がある）．

このような不完全ゲームは単にゲームとしてではなく，

例えばルータの故障/設定ミス，あるいは配線ミスによって発生するネットワークトラブルの原因推定のアルゴリズム考察のモデルとしても利用可能である．

本稿では，人狼の場合のシナリオを示す．P1〜P10 はプレイヤー 1 からプレイヤー 10 を示すとする．実際のプレイは，形式言語[*2] によって実行される予定であるが，ここでは理解を容易にするために自然言語で表記する．

### 3.1 シナリオ

序盤においては，村人は情報が無い状態で発言を行う必要がある．人狼は互いに人狼であることを知っているが，それを村人に悟られては追放されてしまう．

ゲームマスター (GM) は，全てのプレイヤーの役割を把握し，ゲームを進行する．GM が昼を宣告し，プレイヤーに発言を促す．昼の時間は制約されるのが普通であり，ここでも発言数の上限を予め決めてゲームを進行する．

(1) P1: P2 は人狼だと思う．
(2) P2: 自分は人狼ではありません．占い師が居たら自分を占って下さい．
(3) P1: まだ占い師が名乗り出るには早いでしょう．P2 は自分を人狼と指名させることで占い師を排除させようとしている．
(4) P3: P1 こそが人狼ではないでしょうか？
(5) P4: 意見はありません．
(6) ...

ゲームマスター (GM) により夜が宣告される．人狼同士はアイコンタクトによって村人の中から一人を選択して，排除する．

(1) GM: P2 が排除されました．(霊媒師のみへ P2 の素性 (人狼か村人か) を伝える)．話し合いを始めてください．

プレイヤーは新しくなった事実を取り込み，各自の仮説を検証し，さらに情報を交換する．

(1) P1: P2 を疑ってしまったが誤りだった．人狼が人狼を殺すとは考えにくい．
(2) P3: やはり P1 が怪しいです．
(3) P4: 自分は占い師です．P1 は人狼です．
(4) P5: 自分こそ占い師です．P4 は人狼です．
(5) ...

人狼において虚偽の宣告は充分にありえるし，狂人という役をもつ村人は，人狼側の勝利を誘導するので，複数の占いが宣告される場合もある．

ゲームマスターは，プレイヤーに誰を排除するかを公開投票するように促す．この結果，上位 2 名が村人の再投票によって排除される．排除の前に弁明の機会が与えられる．

(1) GM: P1 と P4 が排除候補です．弁明を行って下さい．
(2) P1: 最初に P2 を疑ったのは，誤りでした．こんなに早く占い師が名乗り出るのは，おかしいので P4 は偽者で狂人だと思います．
(3) P4: 早く名乗ってしまいましたが，人狼を一人排除できるチャンスです．

ゲームマスターは上位 2 名のどちらを排除するかを，再度公開投票によって決定する．

(1) GM: P4 が排除されました．そして夜です．

再び人狼がプレイヤーの中から一人を排除する．

このようにして，村人が減っていき，人狼と村人の数が等しくなれば村人側の負け，人狼を全て村から排除できれば村人側の勝ちである．

### 3.2 仮説の保持

人狼のプレイヤーは，他のプレイヤーの発言から，他のプレイヤーの属性 (村人か人狼か) を推定する．プレイヤーの発言は虚偽であっても構わないので，初期にプレイヤーが推定して作成する仮説は誤りである可能性が高く，また必要な情報が欠けているのが普通である．しかし，プレイヤーは，他のプレイヤーの問いかけに反応する，他のプレイヤーに問いを発する，他のプレイヤーに宣言を行うなどして，情報を増していく必要がある．

この推論のための仮説生成と仮説保持は，多重の MSC'(t) として表現できる．MSC'(t) 生成に利用したルールを，次の MSC(t+1) の際に有効と判定するか，無効と判定するか，判断を保留するかと決定することで，自らの作成する仮説もしくは，ルールの取捨選択を行い，他のプレイヤーとの協調と村人側の勝利を計る (もしくは，人狼側として村人を排除していく)．

### 3.3 仮説の保守

人狼のプレイヤーは，ほぼ他の情報が無い状態で，推論を行って発話を行い，行動を決定することで，ゲームの局面を変化させ，情報を得ることで，自らの陣営を勝利に導こうとする．

この過程で仮説をどこまで妥当なものとして保持するかは，人間のプレイヤーにとって記憶力に大きな負担をかける（故にゲームとして成立しているとも言える）．一方で，エージェントのプレイは，メモリさえ充分であれば多くの仮説を生成し，それらの中からもっとも確信度の高い仮説に基づいてメッセージを生成することになる．

MSC(t) が更新される度に，仮説が減少していくことになる．

### 3.4 仮説生成ルールの保守

エージェントのプレイを支える仮説生成ルールは，ゲームの進行状況よって適用するべきでないと判明する場合がある．このとき，人間のプレイヤーの誤った推論によって

[*2] 大澤博隆氏 (筑波大学大学院システム情報工学研究科) によって策定中，JAWS2013 での質疑応答による [3]

も，ゲームが変化してしまうために，ルールが完全であることや，変化した状況が合理的な推論よって行われると仮定することもできない．

このため，仮説生成ルールを一時的に利用しないことを選択する必要がある．

## 4. おわりに

エージェント指向システムによる情報システム構築のためには，矛盾を検出する，矛盾を解決できる知的な要素が必要であることを述べた．また，矛盾を生じやすい例として，人狼をプレイするエージェントシステムを取り扱うことを示した．

### 参考文献

[1] Y. Shoham, “Agent oriented programming”, Artificial Intelligence, Vol.60, No.1, p.51 - 92, 1993
[2] 山本学, “エージェントプログラミングモデルの業務システムへの適用からの考察”, JAWS2013, ポスター, 2013/9/19
[3] 大澤博隆, “人狼言語の設計: 対人コミュニケーションゲームにおける言語のモデル化”, JAWS2013, ポスター, 2013/9/20