エージェントシステム開発における インタラクティブ設計法とその支援環境

打矢 隆弘† 前村 貴秀‡ 阿部 亨‡ 菅原 研次§ 木下 哲男‡

† 東北大学電気通信研究所 〒980-8577 仙台市青葉区片平 2-1-1
‡ 東北大学情報シナジーセンター／大学院情報科学研究科 〒980-8577 仙台市青葉区片平 2-1-1
§ 千葉工業大学情報科学部 〒275-0016 千葉県習志野市津田沼 2-17-1

あらまし 近年，エージェント技術を利用して種々のエージェントシステムの開発が試みられるようになってきている．しかし，状況依存的・非決定的に振舞うエージェントシステムの設計やデバッグは難しく，その基盤となる設計法や支援環境もほとんど提案されていない．本稿では，エージェントシステムの開発効率の向上を目指して，リポジトリ型エージェントフレームワークを基盤としたエージェントシステムのインタラクティブ設計法とその支援環境を提案する．

Interactive Design Method and Design Support Environment for Building Agent-based System

Takahiro UCHIYA† Takahide MAEMURA‡ Toru ABE‡ Kenji SUGAWARA§ and Tetsuo KINOSHITA‡

† Research Institute of Electrical Communication, Tohoku University, Japan
‡ Information Synergy Center/Graduate School of Information Science, Tohoku University, Japan
§ Department of Information and Network Science, Faculty of Computer and Information Network Science, Chiba Institute of Technology, Japan

Abstract The agent-based computing systems have been developed using recent agent technologies. However, the design/debugging of these system is difficult due to the situational and nondeterministic behavior of the agents, and the effective design method and design support environment have not been proposed. In this paper, to realize the efficiency of the development of agent system, we propose an interactive design method and design support environment based on repository-based agent framework.

1. はじめに

次世代情報処理システムの基盤技術の一つとしてエージェント技術がある．エージェントそのものの捉え方は様々であるが，本稿では，自律性や社会性などの新しい特性を備えたソフトウェアをエージェントと呼び，こうしたエージェントを構成要素とする新しい知識情報システムをエージェントシステムと呼ぶことにする．現在，エージェントやエージェントシステムの実現に関わる様々な研究開発が進められ，実問題への適用も試みられるようになってきている．しかしながら，エージェント技術に基づく新しいソフトウェアシステムを系統的に開発するための設計法や，それに基づく支援環境などは充分に整備されていない．

本稿では，エージェントシステムの開発効率の向上を目指して，リポジトリ型エージェントフレームワークを基盤としたエージェントシステムのインタラクティブ設計法とその支援環境を提案する．

2. エージェントシステム開発の現在

2.1 エージェントの開発

エージェントシステムは，種々のエージェントによって構成され，要求された情報処理（以下問題解決と呼ぶ）を実行するソフトウェアシステムと捉えることができる．こうしたエージェントシステムの構築において，開発目標となるのは，様々な機能や用途を持ったエージェントと，それらが構成する問題解決のためのエージェント組織である．

エージェントシステムの開発過程では，従来のソフトウェアシステムの場合と同様に，設計目標を規定す

る仕様，例えば，設計目標の全体的構造(アーキテクチャ)，エージェントとして実現される機能，エージェントのタイプ／粒度／制御構造／知識，エージェント協調方式，エージェント間通信方式などが決定され，目標とする問題解決を遂行するエージェントシステムが実現される．

エージェントシステムの構成要素となるエージェントの開発においては，エージェントの類型化や実現法が工夫されつつある．例えば，類型化に関しては，エージェントの性質や機能に基づいて，(a)基本型，(b)リアクティブ型，(c)熟考型，(d)複合型の4つに分類することができる．

一方，実現法に関しては，大別して，**(i)プログラミングアプローチ**，**(ii)フレームワークアプローチ**の2つに分類できる．

(i)プログラミングアプローチ： Java等の既存のプログラミング言語系や専用のエージェント記述言語系などを用いて，開発目標のエージェントの動作や知識を直接記述し，エージェントを実現する手法である．設計の自由度が高く，基本型やリアクティブ型のエージェント開発には適しているが，高度な機能や知識が要求される熟考型や複合型の場合には開発工数が増大する．

(ii)フレームワークアプローチ： 特定のエージェントアーキテクチャに基づくエージェント開発支援環境(フレームワーク)を利用するもので，プログラミングアプローチを強化・拡張するアプローチといえる．そこでは，エージェントの基本メカニズム(通信機構，協調機構，推論機構など)が予め与えられるので，問題解決や協調動作のための知識やモデルを記述するだけで，種々のエージェントを比較的少ない工数で実現することができる．

近年，フレームワークに関する研究開発事例としては，ADIPS[1]，AgentBuilder[2]，JADE[3]，JATLite[4]，ZEUS[5]等がある．

このように，エージェントの実現法は徐々に整理されつつあるが，多数のエージェントが互いに協調し，動的に組織を形成して問題解決を実行するエージェントシステムの場合には，開発上の課題がいくつか残されている．

2.2 エージェントシステムの開発

エージェントシステムの開発は大きくトップダウン型とボトムアップ型に大別できる．各々の性質と開発上の課題について考察する．

(1)トップダウン型開発

a.問題特定：解決すべき問題やその前提条件などを特定する．

b.要求仕様定義：特定された問題を解決するためにエージェントシステムに必要とされる機能と構造に関する要求仕様(問題分割・部分問題割り当て・問題解決制御構造)を定義する．

c.エージェントシステム設計：要求仕様を満たすエージェントシステムとそこで必要とされるエージェントを設計する．

d.エージェント実装：部分問題を解決する個々のエージェントの実装，エージェント組織構成，及びエージェントシステム全体としての動作のテスト・デバッグを行う．

e.運用試験：エージェントシステムの運用試験を行い，問題解決に有効であるかどうかを検証する．

上記の手順に基づくトップダウン型開発は，近年のエージェントシステム開発の主流であり，その基盤となる設計法も徐々に策定されてきている(ZEUS[5]，MaSE[6]，Tropos[7])．これらの手法の特性として，エージェント開発者により一連の工程が行われるため，開発の自由度が高い反面，開発工程数の多さから，エージェント開発者の作業負担が大きいという課題も内在する．

(2)ボトムアップ型開発

多数のエージェントが利用可能な状態で存在していることを前提としてエージェントを開発する手法である．問題解決の状況に即して，開発済エージェントが問題分割や部分問題の割り当てを自律的に実行するため，エージェント開発者は部分問題のうちの必要なエージェントのみを実装し，エージェント組織構成，結合テスト・デバッグを行えばよい．

上記に基づくボトムアップ型開発は，エージェントシステムの再利用を前提としているために，開発効率を向上させる可能性を含んでいるものの，分散環境におけるエージェントシステムの保持機構や，問題解決の過程でエージェントが動的に組織を形成するためのメカニズムが必要であるため，あまり実現されていない．しかしながら，トップダウン型開発と比較して，開発者の作業負担が大きく削減されることから，効果的な設計法の確立が求められている．

3. エージェントシステムのインタラクティブ設計法の提案

3.1 基本コンセプト

本稿では，フレームワークアプローチの観点から，特にリポジトリ型エージェントフレームワークADIPS[1]，DASHをベースとした，エージェントシステムのインタラクティブな設計法とその支援機構を提案する．

リポジトリ型エージェントフレームワーク（図1）は，エージェントリポジトリ（以下，リポジトリと略

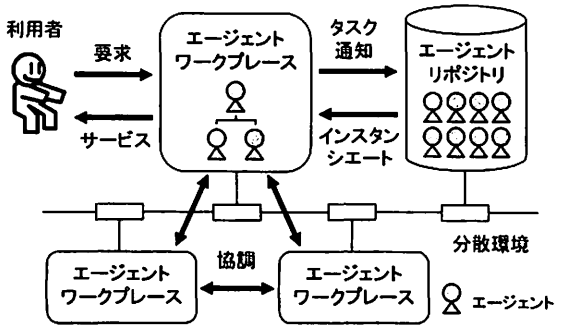

図1　リポジトリ型エージェントフレームワーク

記）と呼ばれるエージェントシステムの蓄積・管理機構を備えることが大きな特徴である．

現状では，リポジトリを中核とするフレームワークは少ないが，エージェントシステムの開発過程では，リポジトリに集積された資産の効果的な活用を支援・促進することにより，開発作業の高度化が期待できる．

本稿で提案する設計法は，上述した特徴を活かした**ボトムアップ型開発を支援する**設計法であり，既存のトップダウン型のエージェントシステム設計法と比較し，以下の特徴を持つ．

・再利用性

リポジトリに集積された開発済エージェント等の資産の利用／再利用による開発効率向上

・インタラクティブ性

設計者とエージェントの相互補完的な役割分担と協働による開発の柔軟性向上や円滑化

3.2 エージェントシステムのインタラクティブ設計法の提案

本節では，リポジトリ型エージェントフレームワークをベースとするエージェントシステムのインタラクティブ設計法を提案する．

リポジトリ活用に立脚したエージェントシステムの開発は，以下のA～Dのプロセスで進行する．

A.問題特定：

解決すべき問題やその前提条件などを特定する．

B.要求仕様定義：

特定された問題を解決するためにエージェントシステムに必要とされる機能と構造に関する要求仕様を定義する．

C.エージェントシステム設計・実装：

要求仕様を満たすエージェントシステムとそこで必要とされるエージェント群を設計・実装する．以下の工程順に進められる．

C-1 開発済エージェントシステムの再利用

要求仕様から導出された部分機能の要求仕様それぞれを，開発済エージェントシステムが保持されているリポジトリに投入し，開発済エージェントシステムの再利用を図る．リポジトリに保持されているエージェントは，自身の知識ベースに格納されている組織構成知識を利用して，自律的に与えられた仕様を満たすかどうかを判断・応答する．再利用可能なエージェントシステムが見つかった場合には，適宜必要なエージェントを開発者の環境に取込み，部分機能を満たすエージェントとして再利用する．

C-2 エージェント知識／機能のプログラミング

リポジトリ型エージェントフレームワークは，協調機構，ルール型知識処理機構，タスク処理機構からなるエージェントアーキテクチャと，ルール型エージェント知識記述言語をエージェント開発者に提供する．即ち，エージェント開発者の作業は，ルール型知識記述言語を用いたエージェントの知識記述，及び，エージェントの機能（ベースプロセス）記述に帰着される．エージェントの知識記述においては，ルールの状態遷移やルール間の繋がりを考慮し知識記述を行う必要があり，ある程度のノウハウを必要とするために，初期の開発者にとってプログラミングの負担は大きい．この課題を解消するために，リポジトリ内の開発済エージェントの利用を図る．即ち，必要に応じて，リポジトリ内の開発済エージェントの種々の知識，例えば，問題解決を実行する知識，エージェント組織を形成するための知識，他のエージェントや外界の状況を認識する知識等を参照してプログラミングを進める．また，リポジトリに格納されている知識記述テンプレートを参照して，必要な知識を効率よく記述する．

C-3 インタラクティブな動作シミュレーション

プログラミングが一段落した際には，エージェントの協調動作の観測や分析を通して，開発されたエージェントの知識／機能の検証・評価を実行する．開発途上のエージェントシステムを対象として，エージェント単体もしくはエージェント組織のレベルで，それらを試験的に動作させ，その動作結果を参考にし，設計の変更や修正を迅速に進める．

例えば，エージェント組織構成・メッセージ通信や同時並行的な動作を確認したり，単一エージェントの知識の内部状態を確認する．上記の作業において，エージェント相互間でのみ送受信される協調メッセージを，エージェントシステムに対し外界の開発者も適宜送受信／補完することにより，エージェントの動作知識が一部未完成な場合であっても，その後のシステムの挙動／性質の確認や，操作／制御を実行することができる．このように，エージェント開発者とエージェントシステムが相互補完的に協調しながら，動作シミュレートを実行してゆく．シミュレートの際に，不具合やエラーが発見された

場合には，工程 C-2 に戻り，知識記述等の修正／デバッグを実行する．

C-4 開発済エージェントシステムの登録／更新

開発したエージェントシステムの分散環境上での利用と，次回以降のエージェントシステム開発の再利用を想定して，エージェントシステムを分散環境上のリポジトリに登録する．既にリポジトリ内に同一のエージェントが保持されている場合には，エージェントの更新をリポジトリに通知する．

D.運用試験：

実環境で，エージェントシステムの運用試験を行い，問題解決における有効性やエージェント知識の妥当性などを検証し，システムを洗練してゆく．

4. インタラクティブ設計法に基づくエージェントシステム開発支援環境 IDEA の設計・実装

4.1 設計

前節で提案した設計法に基づいて，開発者によるエージェントシステム開発作業を効果的に支援するために，インタラクティブエージェント開発支援環境(**IDEA : Interactive Design Environment for Agent system**)を提案する．

IDEA の設計方針として，前節で述べた工程 C-1～C-4 を支援する機構を導入することが挙げられる．具体的には，以下の 4 つに帰着される．

(R1)リポジトリと連携し，新たに開発者が定義した要求仕様を充足する開発済エージェント／エージェントシステムをリポジトリから検索／取込み，再利用／流用設計の実行を支援すること

(R2)エージェント知識記述に適した専用プログラミングエディタを提供し，また，開発済エージェントの知識参照やテンプレート提供により，知識記述の円滑な実行を支援すること

(R3)開発者とエージェントシステムの相互補完的な役割分担と協働によるインタラクティブな動作シミュレーション（エージェント組織構成・再構成・協調・移動テスト等）を支援すること

(R4)テストを完了したエージェントシステムの登録・集積を実行することにより,それらの効果的な活用や再利用を支援すること

上記 4 つの設計方針に基づき，IDEA の支援項目を以下のように決定した．

(F1)エージェント検索・取込み支援・・・(R1)の実現
(F2)エージェントプログラム編集支援・・(R2)の実現
(F3)動作シミュレート・デバッグ支援・・(R3)の実現
(F4)エージェント登録・更新支援・・・・(R4)の実現

上記項目に対応して，IDEA は，**(M1)エージェント検索・取込み支援機構，(M2)エージェントプログラム編集支援機構，(M3)動作シミュレート・デバッグ支援機構，(M4)エージェント登録・更新支援機構**，の 4 つの機構から構成される（図 2）．

以下に各機構の設計について述べる．

(M1)エージェント検索・取込み支援機構

主に，エージェントの再利用を前提として，分散環境上のリポジトリからエージェントを検索する機能と，検索されたエージェントに対し，必要なエージェントを開発支援環境内に取込む機能を有する．前者は，エージェント名／機能名等の付加的情報を用いたキーワード入力による検索機能やリポジトリ内のカテゴリ指定による検索機能であり，後者は，検索結果表示機能，エージェント動作知識プレビュー機能，エージェント取込み機能，取込みエージェントのツリー表示・プロジェクト管理機能である．

本機構を用いて，リポジトリに集積された開発済エ

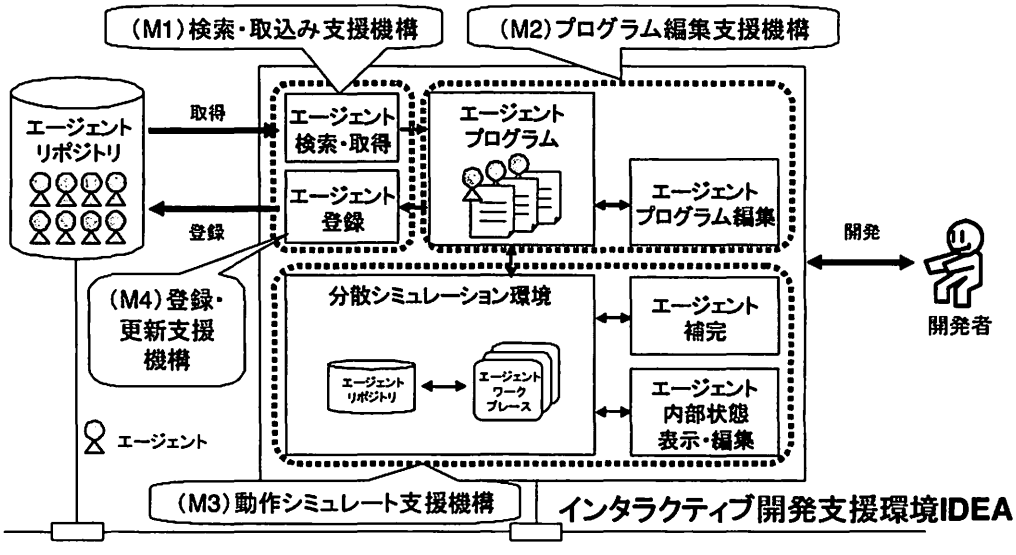

図 2　インタラクティブエージェント開発支援環境 IDEA の構成

ージェント等の資産の利用／再利用を図ることにより，トップダウン型の開発と比較して，その開発効率を向上できる．

(M2)エージェントプログラム編集支援機構

主に，エージェント動作知識を円滑にプログラミングするための各機能を有する．動作知識の記述支援を行うエディタ機能が主な機能であり，それを補足する形でコードインサイト機能，コードチェック機能等の働きにより，エージェント動作知識記述の効率化を図る．また，(M1)と連動した流用エージェントの動作知識参照機能や，テンプレート提供機能により，初心者のエージェント開発者に対し，エージェント動作知識記述のためのノウハウや方策を与える．

(M3)動作シミュレート・デバッグ支援機構

主に，インタラクティブなエージェント動作シミュレーション・デバッグを実行するための機能，及び，開発支援環境内で分散環境上のエージェントシステム挙動をエミュレートする機能を有する．前者は，開発者がエージェントシステムに対し，メッセージを送受信するためのACLエディタ機能，エージェントシステム全体の動作を確認するエージェントビューア機能，個々のエージェントの動作状態を確認するエージェントインスペクタ機能，エージェント間のメッセージを監視するメッセージ監視・検出機能等である．これらの機能を用いることで，エージェントの動作知識が一部未完成な場合であっても，開発者の介在によって，柔軟かつ円滑に，システムの挙動／性質の確認や，操作／制御を実行することができる．

一方，後者については，仮想的なリポジトリと動作環境(ワークプレース) を導入した分散エミュレート機能，マルチワークプレース機能等がある．これらの機能を利用することで，従来では実環境を用いて行っていたエージェントシステムの挙動テスト等が，開発の途中段階においても，比較的容易に行えるようになり，その結果，煩雑になりがちなテスト・デバッグ作業をより効率的に進めることができる．

(M4)エージェント登録・更新支援機構

開発済エージェントシステムのリポジトリへの登録・更新機能を有する．更新機能では，エージェント名重複チェックや，名前管理・世代管理等を実施し，リポジトリ内エージェントの管理支援を実行する．

4.2 実装

本節では，各機構の実装の概要を述べる．

(M1)検索・取込み支援機構(図3)

リポジトリに保持されているエージェントを検索するための検索条件設定部と，その検索結果表示部，エージェントプログラムのプレビュー部の各インタフェースを提供する．

検索条件設定部に，エージェント開発者が部分機能の候補となるエージェント名や機能名等の仕様を投入すると，その要求がリポジトリに通知され，合致したエージェント／エージェントシステムが検索結果欄に表示される．そこで，検索結果から必要なエージェントを選択することにより，当該エージェントプログラムの参照や取込みが行える．

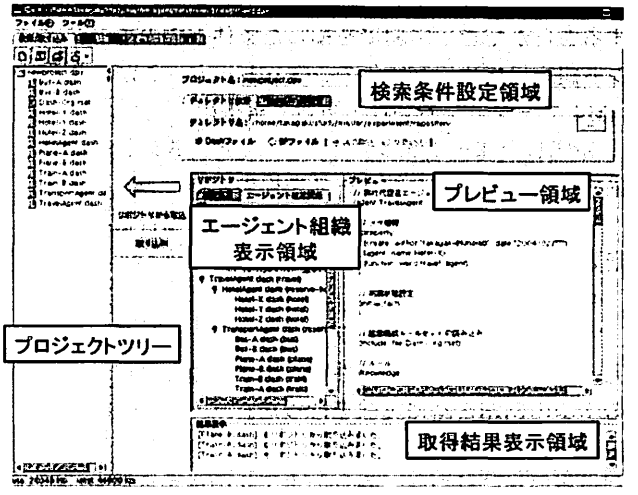

図3　エージェント検索・取込み支援

(M2)エージェントプログラム編集支援機構

ルール型の知識記述に特化した専用プログラミングエディタや，機能プロセス（ベースプロセス）のプログラミングに特化したエディタを提供する．MDI化が可能なため，複数エージェントの平行開発も円滑に実行することができ，またコードチェック・コードインサイト等の機能により，記述形式や記述内容の整合性や統一性を保証しつつ，プログラミング効率を向上して開発を促進する．

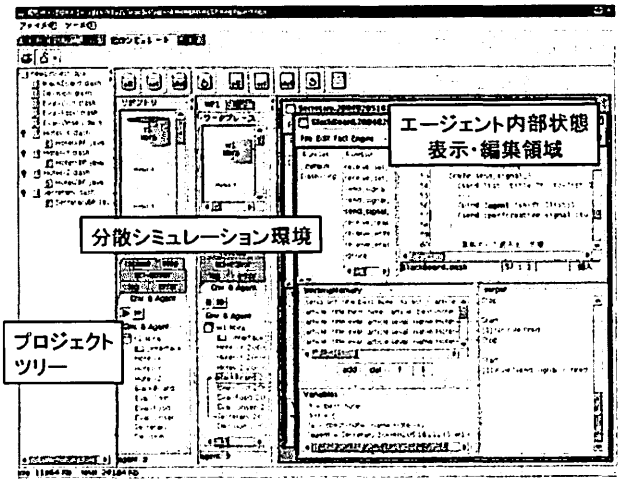

図4　動作シミュレート・デバッグ支援

(M3)動作シミュレート・デバッグ支援機構(図 4)

開発目標のエージェントやエージェント組織の動作を観測するためのエージェントビューアや，エージェントインスペクタを備え，また，エージェントに対するメッセージをインタラクティブに開発者が送受信する機能として ACL エディタを提供する．

IDEA の特徴となっているシミュレーション環境には，仮想的なリポジトリと複数ワークプレースを配置することができ，実環境を模した条件下でのエージェント組織構成や協調動作のシミュレーションが行える．

(M4)エージェント登録・更新支援機構

完成したエージェントシステムをリポジトリに登録するためのインタフェースを提供する．

登録時にエージェントの重複が生じた場合には，更新処理を実行する．

4.3 実験と評価

本稿で提案したインタラクティブ設計法に基づく開発支援環境 IDEA の有効性の評価を行うために，以下の 3 つの例題を基に実験を行った．

[例題 1]ホテル選定システム

利用者要求に基づき，複数のホテルの中から最も評価の高いホテルを選択する問題を処理するエージェントシステム

{構成：ホテルエージェント×3，個別評価エージェント×3，総合評価エージェント，秘書エージェント，黒板エージェント}

[例題 2]会議日程調整システム

複数ユーザ間の会議日程を調整するエージェントシステム

{構成：秘書エージェント×3，会議室エージェント}

[例題 3]ホテル予約システム

契約ネットプロトコルを利用して，旅行代理店がホテルを予約するエージェントシステム

{構成：ホテルエージェント×3，旅行代理店エージェント，秘書エージェント}

4.3.1 エージェントプログラム記述支援の評価

本実験について既存エージェントや契約ネットプロトコルの実装部のテンプレートを利用することにより，削減できた記述量を表 1 に示す．

表 1　実験結果

	エージェント総数	流用エージェント数	記述行数[r]	総行数[n]	記述削減率 [(n-r)/n]
例題 1	9	3	339	655	0.48
例題 2	4	1	278	385	0.28
例題 3	5	4	159	266	0.40

(M1)エージェント検索・取込み支援機構及び(M2)エージェントプログラム編集支援機構を用いた，開発済エージェントシステムの再利用に伴うコード削減率は平均して 0.39 であった．この結果から，本開発支援環境を用いた利用／再利用可能なエージェントプログラムの効果的な再利用によって，エージェント開発者の記述作業の負担を軽減し，効率的にエージェントを開発できることを確認した．

4.3.2 動作シミュレート・デバッグ支援の評価

(M3)動作シミュレート・デバッグ支援機構を用いて，ルール状態遷移等のエージェント知識処理機構の内部状態や，システム全体の挙動を見ながら動作シミュレーションを行うことができ，またエージェントの協調動作時の知識が一部未完成な場合であっても，エージェント開発者の補完的介入（ACL メッセージ投入等）を許すことにより，システムを停止・再起動することなくシステムの動作検証を継続できることを確認した．これらの実験から，本開発支援環境を用いなかった場合に比べて，開発者の動作テスト・デバッグ作業が柔軟かつ円滑に行われることを確認した．

5. おわりに

本稿では，リポジトリ型エージェントフレームワークを基盤とする，インタラクティブにエージェントシステム開発を行うための設計法，及び開発支援環境 IDEA を提案した．本開発支援環境の試作と実験を通して，エージェント利用／再利用の支援やインタラクティブ動作シミュレート・デバッグ支援の強化により，エージェント開発効率が向上することを確認した．今後，エージェントの組織設計に対する機能分割支援機能，テンプレート機能の充実，デバッグ機能の拡張等について更に検討を進める予定である．

文　献

[1] 藤田茂，菅原研次，木下哲男，白鳥則郎，"分散処理システムのエージェント指向アーキテクチャ"，情報処理学会論文誌，Vol.37, No.5, pp.840-851, (1996).

[2] AgentBuilder, http://www.agentbuilder.com/

[3] JADE, http://sharon.cselt.it/projects/jade/home.htm

[4] JATLite, http://java.stanford.edu/

[5] Hyacinth S. Nwana, Divine T. Ndumu, Lyndon C. Lee and Jaron C. Collis, "ZEUS: A Toolkit for Building Distributed Multi-Agent Systems", Applied Artificial Intelligence Journal, Vol.13 (1), pp.129-186, (1999).

[6] Scott A. DeLoach, Mark F. Wood and Clint H. Sparkman, "Multiagent Systems Engineering", The International Journal of Software Engineering and Knowledge Engineering, Vol.11 No.3, (2001).

[7] J. Mylopoulos, M. Kolp and P. Giorgini, "Agent Oriented Software Development", Proceedings of the 2nd Hellenic Conference on Artificial Intelligence (SETN-02), (2002).