社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2008－DD－69（1）
2008 / 11 / 27

発展型ソーシャルウェアのためのリポジトリ型フレームワーク(III)

打矢 隆弘† 前村 貴秀‡ 木下 哲男§

†名古屋工業大学 情報基盤センター 〒466-8555 名古屋市昭和区御器所町
‡東北大学 大学院情報科学研究科 〒980-8577 仙台市青葉区片平 2-1-1
§東北大学 サイバーサイエンスセンター 〒980-8577 仙台市青葉区片平 2-1-1
E-mail: t-uchiya@nitech.ac.jp, maemura@ka.riec.tohoku.ac.jp, kino@riec.tohoku.ac.jp

あらまし これまで我々は，ユーザや環境とのインタラクションを通して獲得された情報・知識・経験を活用しソーシャルウェアの基本的な特性を維持しながら，ソーシャルウェア全体としての機能・性能を漸進的に改善してゆく「発展型ソーシャルウェア」を提案してきた．本発表では，発展型ソーシャルウェアの基盤となるリポジトリ型フレームワーク上で動作する発展型エージェントシステムの設計モデルとその監視アーキテクチャについて検討した結果を報告する．

キーワード ソーシャルウェア，リポジトリ型フレームワーク，発展型エージェントシステム，組織再構成

Repository based Agent Framework for Evolutional Socialware (III)

Takahiro UCHIYA† Takahide MAEMURA‡ Tetsuo KINOSHITA§

† Information Technology Center, Nagoya Institute of Technology. Gokiso, Syowa, Nagoya, 466-8555 Japan
‡ Graduate School of Information Sciences, Tohoku Univ. 2-1-1 Katahira, Aoba, Sendai, 980-8577 Japan
§ Cyber Science Center, Tohoku Univ. 2-1-1 Katahira, Aoba, Sendai, 980-8577 Japan
E-mail: t-uchiya@nitech.ac.jp, maemura@ka.riec.tohoku.ac.jp, kino@riec.tohoku.ac.jp

Abstract We have proposed the evolutional socialware that improve the function and performance of overall of socialware by utilizing the information, knowledge, experience which is acquired from the interaction with the environment and the user.
In this paper, we describe the overview of the design model of evolutional agent system and its monitoring architecture that performs on the repository based agent framework.

Keyword Socialware, Repository based agent framework, Evolutional Agent System, System Reorganization

1. はじめに

サイバー社会のメンバやグループの快適な活動，柔軟な協調・連携を実現するための新しいソフトウェア基盤の一つとして，これまで我々は「発展型ソーシャルウェア」を提案してきた．

ソーシャルウェア[1]の本質は，現実社会から継承した社会知や，サイバー社会で新たに創造される社会知を効果的に活用し，メンバやグループの活動を支援することであるが，特に多様なコミュニティの特質やユーザ／環境の変化に状況依存的に適応するために，ソーシャルウェア自体が自律的に発展してゆく仕組みを備えたソーシャルウェアを「発展型ソーシャルウェア」と定義している．

先行研究[2][3]において，我々はこの発展型ソーシャルウェアの実現基盤としてエージェントシステムの動的な構成／再構成を可能とするリポジトリ型マルチエージェントフレームワーク(以下リポジトリ型フレームワークと略)[4]が特に有効であることを明らかにし，リポジトリ型フレームワークに求められる機能要件の定義・諸機能の設計・実装を進めてきた．本稿では，機能要件の一つである「発展的再構成機能」に特に焦点を当て，当該フレームワーク上で動作する発展型エージェントシステムの設計モデル及びシステムの監視アーキテクチャについて検討した結果を報告する．

2. リポジトリ型フレームワークによる発展型ソーシャルウェアの実現

2.1. 発展型ソーシャルウェア

発展型ソーシャルウェアとは，ユーザや環境とのインタラクションを通して獲得された情報・知識・経験を活用して，ソーシャルウェアの構成要素となるソーシャルウェアコンポーネントの動作知識の更新，或い

は，コンポーネント相互間での連携関係の動的な変更によって，ソーシャルウェアの基本的な特性を維持しながら，ソーシャルウェア全体としての機能・性能を漸進的に改善してゆくシステムである．

2.2. 発展型ソーシャルウェアの実現法

本研究では，発展型ソーシャルウェアの設計・実装において，個々の知的主体が分散・協調して自律的に組織化・再組織化を行うマルチエージェントシステムを採用している．

この方式では，ソーシャルウェアの各機能を知識に基づき動作するエージェント実装型ソーシャルウェアコンポーネントとして実現することで，システム内部を形成しているコンポーネントの特性やそれらの論理的な構成を変更する事が容易になる．

こうしたソーシャルウェアコンポーネントの組織化や要求指向のソーシャルウェアの生成・運用を実行する主体がリポジトリ型フレームワークである．

2.3. リポジトリ型フレームワークの採用

リポジトリ型フレームワーク(図 1)の中核機構はエージェントリポジトリ機構(以下，リポジトリと略) であり，その特徴的な機能は以下の通りである．

- 設計／開発されたエージェントの一元的な集積とライフサイクルの管理
- ユーザのサービス要求に応じたリポジトリ内でのエージェント群の組織化
- 組織化の結果として，ユーザにサービスを提供するエージェントシステムを動作環境であるエージェントワークプレースに動的に生成・活性化
- 活性化され稼動しているエージェントシステムの動的な再構成

こうした特徴を有するリポジトリ型フレームワークを採用することにより，知識や支援機能のソーシャルウェアコンポーネント化，要求指向のソーシャルウェアコンポーネント組織化，支援状況の変化に対応したソーシャルウェア組織再構成などを満たすことができる．そのため，同フレームワークを強化・拡張することにより，発展性を備えたソーシャルウェアの実現が容易となる．

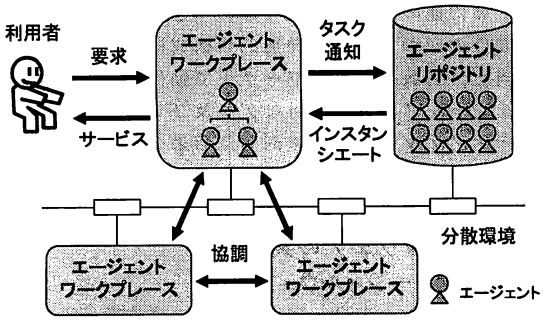

図 1：リポジトリ型フレームワーク

2.4. リポジトリ型フレームワークの機能要件

リポジトリ型フレームワークにおいて発展性を実現するための機能要件を以下に述べる．各機能の内部設計に関しては文献[5]を別途参照されたい．

(機能 F1) エージェント／エージェントシステム再利用機能：
稼働中のエージェントの休止・保存・再活性化，有用なエージェントの判定処理機能などを導入し，エージェント／エージェントシステムやシステムの動作記録を利用する機能である．

(機能 F2) 異種エージェント組織化機能：
リポジトリと協調する相互接続機構(ファシリテータ)を導入し，多様なエージェントプラットフォーム上のエージェントの相互運用を支援する機能である．

(機能 F3) 発展的再構成機能：
構成されたエージェントシステムに対して調整・再構成を行い，性能や機能を維持／改善するための機能である．

上述した機能を備えた新しいリポジトリ型フレームワークにおいて，特に(機能 F3)の発展的再構成機能に焦点を当てフレームワークの内部機構の構成を検討したものを**発展型フレームワーク(EF)**と呼び，このフレームワーク上で動作するエージェントを**発展型エージェントシステム(EAS)**と呼ぶ．

以降，3 章では発展型エージェントシステムの設計モデルについて考察し，4 章では発展型フレームワークにおけるエージェントシステムの状態監視アーキテクチャについて考察する．

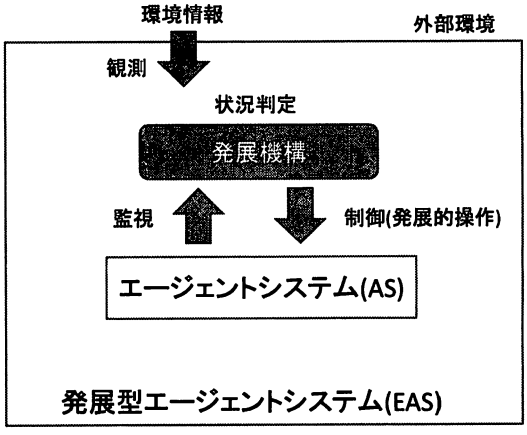

図 2：EAS の基本アーキテクチャ

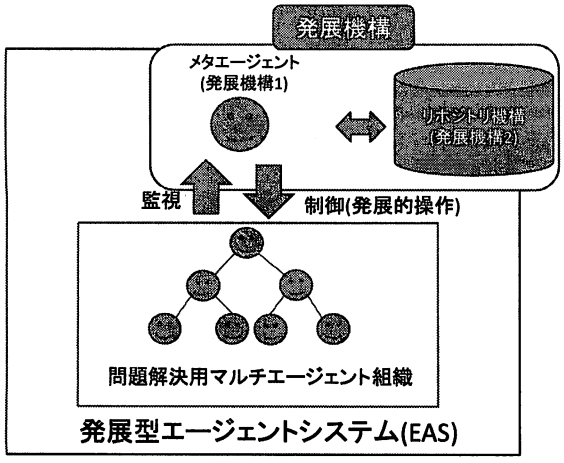

図 3：メタエージェントによる EAS

3. EAS の設計モデル

3.1 EAS の基本アーキテクチャ

発展型フレームワークで動作する発展型エージェントシステム(EAS)は，問題解決を担当するエージェントシステム(AS)，及び，AS 全体としての動作制御を担当する発展機構から構成される(図 2).

EAS の発展機構は，AS の稼働状況や外部環境等の情報をもとに AS の動作制御(発展的操作)を行い，AS の恒常性と発展性の実現を目指す.

本稿では，リポジトリ機構を基盤とした発展機構の構成法を検討するために，AS が単独のエージェント組織である場合に，当該 AS に対応して定義される単体エージェント(メタエージェントと呼ぶ)によって発展機構を構成する手法について考察する(図 3).

3.2 EAS の設計仕様

EAS の設計仕様は，以下の 6 項組みにより与えられる.

EF = < ne, R, S, F, E, P >

ここで，ne はシステム名(name of EAS)，R は要求，S は組織構造，F は機能，E は環境，P は特性である.

(1) 要求(タスク) R は，所与の要求記述の集合である.

R = { reqk | reqk = < req-f, req-f-q >k; k = 1, .., Nr },

ここで，req-f は要求サービス，req-f-q は同サービスの品質を表す.

(2) 組織構造 S は，次の 3 項組みで与えられる.

S = < AG, STR, AL-AG >

AG は AS の構成メンバとなるエージェントの集合，STR は AG のエージェント組織の情報，AL-AG は AG のメンバと交代可能な代替エージェントの集合である.

メンバ : AG = { ag i | ag i = < na i , M-ag i, Sv-ag i , K-ag i, w-ag i , CoA(i) >; i=1, ..., Na }

na i : エージェント ag(i)の識別名

M-ag i : 受理メッセージの集合

Sv-ag i : サービス集合

Sv-ag(i) = { sva(i) | sva(i) = < func, func-q >i; i = 1,..., m }

K-ag i : ag(i)の動作知識記述の集合

w-ag i : 時刻 t における ag(i) の処理負荷

[例] w-ag i = nr* (1+ m-in + m-out)/2

nr: ΔT 間の発火ルール数

m-in: ΔT 間の受信メッセージ数

m-out: ΔT 間の発信メッセージ数

ct: 推論サイクルの実行時間

CoA i : 協調エージェントセット

構造: STR = { rel_{ij} | rel_{ij} = < r_{ij}, ag_i, ag_j >; $ag_i, ag_j \in AG$; $r_{ij} \in R$ }

r_{ij}: ag_i と ag_j の関係

R: エージェント相互間での関係集合

代替メンバ: AL-AG = { ag_j | ag_j = < na_j, M-ag_j, Sv-ag_j, K-ag_j, nil, CoA_j >; j = 1, ..., **Nalt** }

(3) 機能 F はメンバの機能の和集合である.

F = ∪Sv-ag_i, $ag_i \in AG$, i = 1, ..., Na.

Nn = |F| , Nr ≦ Nn

F は，EAS の機能設計の過程で決定される. すなわち，全ての req_i =< req-f, req-f-q > $_i$; i=1,...,Nr に対して，sva_j =< func, func-q > $_j$ ∈ S が存在し，req-f = func かつ req-f-q≡func-q となるとき，EAS は実現可能と呼び，要求とサービス(機能)との対応付けが行われ，F が確定する.

(4) 環境(情報)E は，エージェントプラットフォーム Ag-Plfms，分散プラットフォーム Ds- Plfms，エージェントリポジトリ Ag-Rpgs の情報により定義される.

E = < Ag-Plfms, Ds- Plfms, Ag-Rpgs >

Ag-Plfms = { ap_i | ap_i =< nap_i, ap-typ $_i$, ap-pef $_i$, Mg-ag_i, S-AP $_i$ >, i = 1,..., **Nap** }

処理数: Mg-ag_i = { $ag_j \in AG$, j=1,..., J }

機能数: S-AP $_i$ = {ap_k, k=1,.., K, k≠j }

処理効率: ap-pef $_i$ ← ASM(ap_i)

Ds- Plfms = { dp_i | dp_i =< ndp_i, dp-typ $_i$, dp-pef $_i$, Sv-dp_i >, i = 1,..., **Nup** }

サービス数: Sv-dp_i = { svd_j | svd_j = < func, func-q > $_j$; j = 1,..., L }

処理効率: up-pef $_i$ ← USM(up_i)

Ag-Rpgs = < p-AgR, S-AgRs >

p-AgR: 主 AgR, S-AgRs: 副 AgR 集合

(5) 特性 P は，通常 EAS の発展機構によって決定される EAS の動作特性であり，AS 個性を構成する要素となる. 本稿の場合，メタエージェント mag が管理する内部情報として定義・実現される.

mag = < n-mag, M-mag, Sv-mag, K-mag, t-ag, op-mode, History, Prop >

t-ag: AS 組織階層の中核エージェント

op-mode: 動作モード

op-mode = normal / transit / evolutional

History: 動作特性(Prop)の履歴のリスト

Prop: 動作特性

動作特性は，計測可能な特性 prop(i)の集合である．

Prop = { prop i | i = 1, ..., np }

prop i の例として，以下のような指標が考えられる．

ut: 機能利用率 (例) ut = Fu(AG, F) = Nf / Nn

ds: 機能分布率 (例) ds = Fd(AG, F) = Nf / Na

pf: 潜在機能率 (例) pf = Fs(AG, AL-AG, E) = Nn / Na

al: 余裕度 (例) al = Fa(AG, AL-AG, E) = Nalt / Na

cr: 脆弱度 (例) cr = Fc(AG, AL-AG, E) = Nc / Na

Nc: 代替不可能なメンバエージェントの数

awl: エージェント稼働率

(例) awl = Faw(AG, E, t) = (1/Na) * Σ w-ag i

swl: システム稼働率

(例) swl = Fsw(AG, E, t) = (1/Nup) * Σ wl-ap i

wl-ap i : Ag-platform i の平均処理負荷

wl-ap i = < Ag-platform i CPU 利用率> /

< Ds-platform i CPU 利用率>

qlt: EA 特性品質 (例) qlt1: タスク品質(QoT)

QoT = < Rlz_QoT , Req_QoT >

実効品質: Rlz_QoT = Frlt (S, E)

要求品質: Req_QoT = Frqt (R, E)

(例) qlt2: 品質余裕度(M_QoT)

M_QoT＝Fmg (R_QoT. Rlz_QoT, E)

＝(Rlz_QoT－Req_QoT)

3.3 EAS における変動

EAS 稼動中に発生する変動は，EAS の構成要素の変化として検出される．例えば，組織構造 S の変化をΔS，これによる組織構造 S の変化を S+ΔS と表記する．以下，変化に関わる幾つかの概念を整理する．

まず，時刻 t において，EAS の観測可能な対象 obj(t) によって検出される EAS の変化を，

Ch(obj(t), ψ, t)

と表記する．また，対象の変化Δobj と変動率 rat_Δobj(t) を次のように表記する．ここで，ψは閾値である．

Δobj(t) = *chg*(obj(t-Δt), obj(t))

rat_Δobj(t) = *rat*(Δobj, obj)

変化 *Ch*(obj(t), ψ, t)の判定条件は

if Δobj(t)≠0 and rat_Δobj> ψ

then *Ch*(obj(t), ψ, t) = Δobj

である．また，EAS にとって好ましくない変化(劣化)を

Dg(obj, ε, t)(*Ch*(obj, ε, t) < 0)と表記する．以下，EAS の変動を捉える変化イベントの例を挙げる．

[例 1 : ΔS イベント *Ch*(uf, ψ, t)]

AG のメンバエージェントの変動(AG→AG')を，以下のように，

Δuf = (Nf+ΔNf) / (Na+ΔNa) - Nf / Na,

ΔNf = |F'| - |F|, ΔNa = |AG'| - |AG|

(degradation |AG|＞|AG'|)

機能利用率 uf,の変化(uf→uf+Δuf)として検出する．

[例 2 : ΔE イベント *Ch*(al, ψ, t)]

リポジトリ内の代替エージェントの変動(AL-AG→AL-AG')を，

Δal = ΔNalt / Na, ΔNalt = |AL-AG'| - |AL-AG|

潜在機能率の変化(al→al' = al+Δal)として検出する．

[例 3 : ΔP イベント *Ch*(awl, ψ, t)]

エージェント動作環境の変動を，エージェント稼働率の変動，メンバエージェントの処理負荷変動Δawl により，例えば，

Δawl = (1/Na) * Σ Δw-ag$_i$ i = 1,..., Na

として検出する．

[例 4 : ΔF イベント *Ch*(QoT, ψ, t)]

機能の変化を，タスク品質の変化により検出する．

ΔM_QoT≠0 & rat_ΔM_QoT > ψ → *Ch*(QoT, ψ, t)

また，これをもとに，次のようなイベントも定義できる．

・サービス品質劣化 *Dg*(QoT, ε, t)

Ch(QoT, ε, t) & ΔM_QoT < 0 & rat_ΔM_QoT > ε

→ *Dg*(QoT, ε, t)

・短期的劣化 *TDg*(QoT, ε, t, ξ)

TDg(QoT, ε, t, ξ)

⇔ ∃*Dg*(QoT, ε, t) , ∃ γ0, γ, ∃ξ, γ0< ξ ≦ γ,

∀ t', t < t' < t+ ξ, *Dg*(QoT, ε, t')

ここで，ξ は，*Dg*(QoT, ε, t)の継続時間である．

・長期的劣化 *PDg*(M_QoT, ε, t, ξ)

PDg(QoT, ε, t, ξ)

⇔ ∃γ, *TDg*(QoT, ε, t, γ) and ∀ξ> γ,

∀ t' > t+ ξ, *Dg*(QoT, ε, t+ ξ)

3.4 EAS の変化と対応

EAS に対する変動が発生すると，EAS は，(1)変動検出(変化要素の同定)，(2)目標設定(AS 個性に基づく制御目標の設定)，(3)対応処理(操作の選択と適用)，(4)結果判定，という手順で対処する．ここで，EAS の変化を次のように模式的に表現する．

EAS / < ne, R, S, F, E, P >

→ EAS' / < ne, R+ΔR, S+ΔS, F+ΔF, E+ΔE, P+ΔP >

すなわち，EAS の各要素に対する変動(Δ)により，EAS が EAS' に変化することを表す．例えば，メンバエージェントの故障に起因する変化(ΔS)の発生により，組織構造 S が S+ΔS に変化し，EAS の特性も P → P' と変化する．こうした変化が検出された場合の EAS の

対応(動作)と期待効果について，AS の恒常性と発展性という二つの観点から整理してみる．以下，(1)と(2)が恒常性の観点，(3)と(4)が発展性の観点に関するものであり，また，⇒は，それぞれ名称を付した対応による EAS の変化を表す．

(1) 調整

EAS' ⇒ EAS

P' → P

期待効果： 当初の EAS に復帰する．

(2) 再構成

上記(1)では対応できない場合に適用される．

EAS' ⇒ EAS^{+}

P' → P^{+} , $\| P^{+} - P \| \leq \theta$

期待効果： 再構成によって S が変化しているため，当初の EAS と類似した EAS^{+}に復帰する．新旧 EAS の変分は，動作特性 P(及び，AS 個性)の類似性を規定するパラメータ θ で制御する．なお，変動要因の解消などにより，旧 EAS に復帰しようとする場合には，EAS の復帰処理により EAS^{+} ⇒ EAS を実現する．

(3) 発展的再構成

上記(2)では対応できない場合に適用される．

EAS' ⇒ EAS^{*}

P' → P^{*} , $P^{*} \geq P$

期待効果： 発展的再構成により EAS'が新しい EAS^{*}に遷移する．このとき，動作特性(及び AS 個性)も新たな P^{*}に変化する．また，新 EAS の P^{*}は旧 EAS の P に比べて改善されている．

(4) 発展的再設計

上述した処理では対処できなかった変化については，EAS の設計者による対応を要請する．

EAS^{*} with δR ⇒ EAS^{**}

期待効果： 通常，上記(3)の適用によって EAS は EAS^{*}に変化しているが，この段階で対応不可能となった場合に，設計者に対して再設計などの非常措置を要請する．

3.5 EAS の対応処理

変化した要素に応じて EAS が選択・適用する対応処理を整理する．以下，Δは変化を，δ は対応操作を表す記号である．

(1) 要求変動への対応

ΔR ⇒ δF, δS による処理

EAS' / < ne, R+ΔR, S, F, E, P' >

⇒ EAS^{+} / < ne, R+ΔR, S, F+$\underline{\delta F}$, E, P^{+} > |

EAS^{**} / < ne, R+ΔR, S+$\underline{\delta S}$, F+$\underline{\delta F}$, E, P^{*} >

ここで，例えば， F+$\underline{\delta F}$ は，機能 F に対する対応操作が適用されたことを表す．また，EAS^{+} (特性の変化 P^{+})は従来の調整や再構成を適用して更改された EAS を表し，同様に，EAS^{**}は発展的再設計による更改，EAS^{*}は発展的再構成による更改の適用結果(これらの特性の変化は P^{*})を表す．(以下，同様)

(2) 構造変動への対応

ΔS ⇒ δF, δS による処理

EAS' / < ne, R, S+ΔS, F, E, P' >

⇒ EAS^{+} / < ne, R, S+ΔS+$\underline{\delta S}$, F+$\underline{\delta F}$, E, P^{+} > |

EAS^{*} / < ne, R, S+ΔS+$\underline{\delta S}$, F+$\underline{\delta F}$, E, P^{*} > |

EAS^{**} / < ne, R+$\underline{\delta R}$, S+ΔS+$\underline{\delta S}$, F+$\underline{\delta F}$, E, P^{*} >

(3) 機能変動への対応

ΔF ⇒ δF, δS による処理

EAS' / < ne, R, S, F+ΔF, E, P' >

⇒ EAS / < ne, R, S, F+ΔF+$\underline{\delta F}$, E, P > |

EAS^{+} / < ne, R, S+$\underline{\delta S}$, F+ΔF+$\underline{\delta F}$, E, P^{+} > |

EAS^{*} / < ne, R, S+$\underline{\delta S}$, F+ΔF+$\underline{\delta F}$, E, P^{*} > |

EAS^{**} / < ne, R+$\underline{\delta R}$, S+$\underline{\delta S}$, F+ΔF+$\underline{\delta F}$, E, P^{*} >

(4) 動作特性の変動への対応

ΔP ⇒ δF, δS による処理

EAS' / < ne, R, S, F, E, P+ΔP >

⇒ EAS / < ne, R, S, F+$\underline{\delta F}$, E, P > |

EAS^{+} / < ne, R, S+$\underline{\delta S}$, F+$\underline{\delta F}$, E, P^{+} > |

EAS^{*} / < ne, R, S+$\underline{\delta S}$, F+$\underline{\delta F}$, E, P^{*} > |

EAS^{**} / < ne, R+$\underline{\delta R}$, S+$\underline{\delta S}$, F+$\underline{\delta F}$, E, P^{*} >

(5) 環境変動への対応

ΔE ⇒ δF, δS による処理

EAS' / < ne, R, S, F, E+ΔE, P' >

⇒ EAS / < ne, R, S, F+$\underline{\delta F}$, E+ΔE, P > |

EAS^{+} / < ne, R, S+$\underline{\delta S}$, F+$\underline{\delta F}$, E+ΔE, P^{+} > |

EAS^{*} / < ne, R, S+$\underline{\delta S}$, F+$\underline{\delta F}$, E+ΔE, P^{*} > |

EAS^{**} / < ne, R+$\underline{\delta R}$, S+$\underline{\delta S}$, F+$\underline{\delta F}$, E+ΔE, P^{*} >

EAS の変化への対応処理は，基本的に，EAS の機能と構造の操作，すなわち，δF とδS により実現される．また，発展的操作の場合には，目標となる新たな特性を与える操作 δP を伴う．一方，要求の操作δR を伴う対応は，発展的再設計だけとなる．

以下，これらの操作の一般形を EAS の設計仕様と EAS の変化を引数とする関数としてまとめておく．

(EAS／EAS^{+} 導出の場合)

Tuning ［調整］

δF ← *TF* i(t, *Dg*(obj, ε, t), R, S, F, E, P, τi)

ここで，τi は対応操作の実行時間(以下，同様)．

Reorganization ［再構成］

δS ← *RO* i(t, *Dg*(obj, ε, t), R, S, F, E, P, τi)

Recover ［復帰関数 *Rcv*］

AS ← *Rcv*i(t, *Dg*(obj, ε, t), R, S, F, E, P, τi)

(EAS^{*}／EAS^{**} 導出の場合)

E-Tuning ［発展的調整］

δP ← *EG* i(t, *Ch*(obj, ε, t), R, S, F, E, P, τi)

δF ← *ET* i(t, *Ch*(obj, ε, t), R, S, F, E, P, τi)

なお, *EG* は新たな特性を定める関数である.

E-Reorganization ［発展的再構成］

δP ← *EG* i(t, *Ch*(obj, ε, t), R, S, F, E, P, τi)

δS ← *ER* i(t, *Ch*(obj, ε, t), R, S, F, E, P, τi)

E-Reorganization ［発展的再設計］

δR ← *User* : given from user

δP ← *EG* i(t, *Ch*(obj, ε, t), R, S, F, E, P, τi)

上記の操作は，メタエージェントの動作知識として設計・実装される.

以上 EAS の設計モデルについて説明したが，文献[6][7]では，環境 E の情報を収集・管理するための機構であるユビキタスサービス管理機構 (USM: Ubiquitous Service Management mechanism), エージェントサービス管理機構(ASM: Agent Service Management mechanism)についても報告している.

4．メタエージェントを用いた EAS の試作

メタエージェントを用いた EAS 実現の一例として，メタエージェントが AS の状態監視を実行する監視アーキテクチャを試作した(図 4).

監視エージェント(メタエージェント)は従来の AS 内の動作とは独立して動作し，監視対象となるマルチエージェントシステムの観測情報を集約する．また，集約された観測情報に基づき発展的再構成の起動制御を実行する.

この監視アーキテクチャは，従来の AS に監視エージェントを自動的かつ暗黙的に付加すること，及び，監視対象となるエージェントの生成時に共通の監視手順(動作知識)を自動的に付与することにより，AS 内で自己監視・調整機能を実現する従来手法の以下の課題を解決することが可能である.

(P1) 各エージェントに埋め込まれる自分自身の監視や制御に関する知識の設計では，他のエージェントやシステム全体の動作状況を考慮することが困難

(P2) 未知のエージェントに関する監視・起動制御を付与することが困難

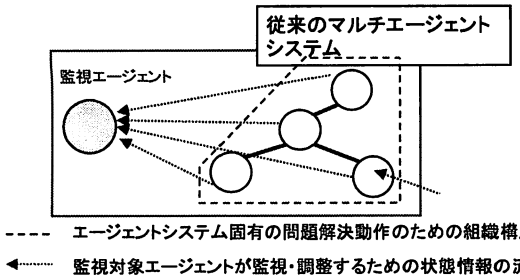

図 4：監視アーキテクチャの概要

監視アーキテクチャの設計・実装・評価実験の詳細については別途報告することとし詳細は割愛するが，本研究では監視アーキテクチャの適用例としてネットワーク管理支援システムを試作し，

・監視エージェントがシステム全体を観測し，再構成が検討される状況を適切に判定できるか

・再構成の検討が必要であると判定された状況で，監視エージェントが適切な再構成プランを選択するか

などの実証実験を行った．その結果，監視エージェントがシステム全体の挙動を包括的に監視し，適切な再構成プランを選択している事が確認できた．以上の実験により監視アーキテクチャの有用性を実証している.

5. おわりに

本稿では，発展型ソーシャルウェアの実現に向けて，発展型フレームワーク上で動作する発展型エージェントシステムの設計モデルの詳細と，発展型エージェントシステムの監視アーキテクチャの概要について説明した．今後は EAS の設計モデルとこれに基づく設計手法の精緻化を行うとともに，発展型ソーシャルウェアの構築と評価実験を通し，同フレームワークのさらなる改良を図る予定である.

謝辞: 本研究の一部は，科学研究費補助金(19200005)の援助を受けて実施した.

文　献

[1] (財)日本情報処理開発協会(編), “情報ネットワーク社会の未来 ～サイバー社会を創る知的情報技術～”,富士通ブックス, 1997.

[2] 打矢隆弘, 前村貴秀, 今野将, 木下哲男, “発展型ソーシャルウェアのためのリポジトリ型フレームワーク”, 信学技報, Vol.106, No.400, AI2006-26, pp.7-12, 2006.

[3] 打矢隆弘, 前村貴秀, 今野将, 木下哲男, “発展型ソーシャルウェアのためのリポジトリ型フレームワーク (II)”, 信学技報 AI2007-15(2007-11), pp.23-28, 2007.

[4] 打矢隆弘, 原英樹, 高垣暁, 菅原研次, 木下哲男, “リポジトリ型エージェントフレームワークの開発と評価”, 情報技術レターズ, Vol.2, pp.139-141, 2003.

[5] 打矢隆弘, 前村貴秀, 今野将, 木下哲男, “発展型エージェントシステムのための高機能エージェントリポジトリ”, 合同エージェントワークショップ＆シンポジウム JAWS2006 講演論文集, 2006

[6] 木下哲男, “発展型エージェントシステムの動作状況認識機能”, 合同エージェントワークショップ＆シンポジウム JAWS2008 講演論文集, 2008.

[7] Li Xiaolu, Takahiro Uchiya, Susumu Konno and Tetsuo Kinoshita, " Proposal for Agent Platform Dynamic Interoperation Facilitating Mechanism", Proc. 21th Int. Conf. Industrial, Engineering and Other Applications of Applied Intelligent Systems (IEA/AIE 2008), LNAI5027, pp.825-834, 2008.