F-002

Web ブラウザを用いたエージェントリポジトリのモニタリング機構
Monitoring System of Agent Repository Using Web Browser

立松 直也† 打矢 隆弘 † 内匠 逸 †
Naoya Tatematsu Takahiro Uchiya Ichi Takumi

1． はじめに

近年の急速な情報化に伴い，多くの分野において情報技術を活用したサービスの提供や，研究が行われている．またコンピュータの普及が進み，コンピュータを所有している人が増加している．このような理由から，システムの動作環境や，利用者からの要求の多様化が進んでいる．そのため，システム開発者には動作環境に合わせた条件や，利用者の要求に合わせたシステムの設計が必要とされている．そこで，このように多様で複雑化した要求に対し，柔軟なシステムの設計や問題の解決を可能とする手段として，エージェントを用いたシステムの設計が提案されている．

エージェントとは，自身が動作する環境を認識し，入手した情報をもとに知的な動作をおこなうプログラムである．また複数のエージェントを協調させ，問題の解決を図る，エージェントシステムとして利用することが可能である．エージェントシステムでは，環境に合わせたシステムの構成や，複数環境へエージェントを配置することによる処理の分散が可能である．これらの特徴は利用者の要求や環境に合わせた動作を実現するための手段として有効である．

利用者がエージェントを活用するためには，エージェントフレームワークという動作環境が必要である．一般的なエージェントフレームワークには，エージェントが知的な判断を行うための推論機構や，エージェント同士が協調動作を行うための通信機構が備えられており，エージェントの基本的な動作を支えている．本研究では多数存在するエージェントフレームワークの中でも，特に DASH[1]に焦点を当てて行う．DASH には先に示した機構の他に，エージェントリポジトリ(以降リポジトリ)というエージェントを即座に応答できる状態で保持するための機構が備わっている．DASH で作成されたエージェントはまずリポジトリに格納され，そこから動作環境に生成されサービスの提供を行う(図 1)．さらにそこから動作環境に合わせエージェントシステムの再構成も可能である．このようにリポジトリを用いることで，動的なエージェントシステムの構成が可能である．

しかし，現行の DASH には問題点が存在する．現行 DASH は，リポジトリの中に格納されているエージェントの情報を，他のサブネットワークに提供する機構を備えていない．このため，利用者はネットワーク上に存在する複数リポジトリのエージェント情報を把握することができず，適切な運用が困難である．そこで本研究では，複数のリポジトリ内のエージェント情報を統一的に管理し，利用者へ提供するためのモニタリング機構を提案する．この機構を導入することで，利用者は有益なエージェントの発見が容易になる．また開発者に対してエージェントの利用状況など有益なフィードバックが可能である．

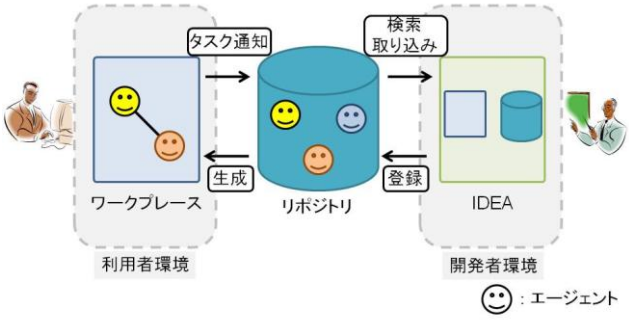

図 1：DASH の構成

2． 提案手法

本研究ではネットワーク上に複数存在するリポジトリの情報を一括で管理し，利用者に提供する機構を提案する．提案機構を用いることで，利用者は多くのエージェントの情報を取得することが可能となり，目的に適したエージェントの発見が容易になる．

2.1 概要

本研究では，エージェントの情報を管理するために，新たにアプリケーションサーバを設置する．そして，各リポジトリからエージェントの情報を送信する(図 2)．次に利用者へ情報を提供するまでの手順を示す．

step1 アプリケーションサーバへ情報送信

各リポジトリからアプリケーションサーバに対しエージェントの情報を送信する．送信する情報は，エージェントの追加，消滅など内容によって異なるが，エージェントの名前，作成者名，所有するルー

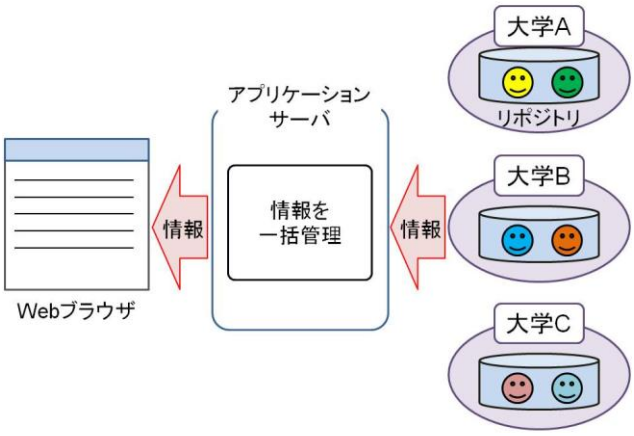

図 2：提案手法の概要

† 名古屋工業大学 大学院 工学研究科 情報工学専攻
Nagoya Institute of Technology, Graduate School of Engineering.

ル名，登録されている環境名など協調動作を行う際に必要なものである．

step2 アプリケーションサーバでの情報保存

リポジトリから送信されてきた情報をアプリケーションサーバに保存する．この際，アプリケーションサーバでの情報管理を効率的に行うために，送信された情報はリポジトリごとに異なるファイルに保存する．

step3 Web ブラウザからのリクエスト

利用者が各自の Web ブラウザを使い，設置したアプリケーションサーバへ HTTP リクエストを送信する．この際，アプリケーションサーバが管理しているエージェントの数が膨大である可能性があるため，利用者は目的のリポジトリを選択し閲覧する．

step4 情報の表示

選択されたリポジトリ内に存在するエージェントの情報を利用者の Web ブラウザ上に表示する．表示する情報は利用者にわかりやすいように，表形式にまとめる．

2.2 アプリケーションサーバ

提案手法では複数サブネットワークに存在するリポジトリの情報を管理するために，アプリケーションサーバを用意する．そこで，DASH との親和性を考慮し，アプリケーションサーバ上で動作するプログラムはサーブレット/JSP[2][3]を用いて実装する．さらにサーブレット/JSP を動作させるために必要となるサーブレットコンテナには Tomcat[4]を用いる．

2.3 サーブレット/JSP

サーブレット/JSP とは，Java EE のサブセットでサーバ上で動作する Java プログラムである．DASH 自身は Java を使い設計されているので，エージェントフレームワークと提案機構との親和性は非常に高くなる．

サーブレット/JSP を使いシステムを構成する場合，MVC モデル(図 3)と呼ばれる構造を多く用いる．主にサーブレットが論理的処理やデータベースアクセスを，JSP がレイアウトを担当する．これはサーブレットが Java ソースコード中に HTML タグを記述するという構造に対し，JSP は HTML 中に Java プログラムを記述する構造をとっているためである．提案機構でもこの構造に則り，エージェントフレームワークとの情報の授受はサーブレットで行い，利用者への処理部分は JSP で行う．

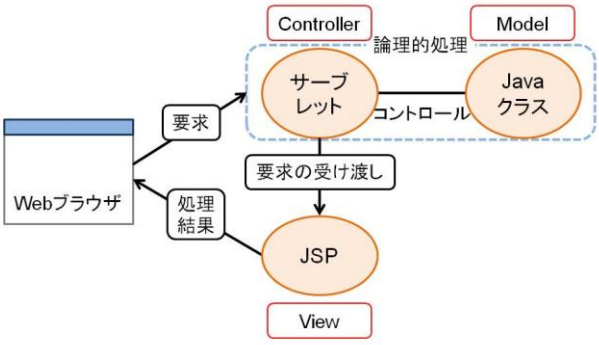

図 3：MVC モデル

2.4 サーブレットコンテナ

サーブレットコンテナとは，HTTP サーバが受信したリクエストを受け取り，サーブレット/JSP のインスタンスを作成する実行環境である．サーブレット/JSP を動作させるためには，サーブレットコンテナが必須となる．

提案手法ではサーブレットコンテナとして Tomcat を利用する．Tomcat は Apache Software Foundation から公開されているオープンソースのサーブレットコンテナである．また HTTP サーバとしての機能も備えているため，別途 HTTP サーバを用意する必要がない．

2.5 提供内容

利用者に対し提供する情報を選定する．本研究の目的は，利用者が自身の用途や目的に適したエージェントシステムの構築に必要な情報を，効率的に提供することである．そのため，エージェントの特徴や組織内での振る舞いを的確に提供する必要がある．そこで提案機構で管理／提供する情報は次のものとする．

- **エージェント名**

  エージェントの判別に用いる．一般的にエージェント名は，そのエージェントが提供するサービス内容を表しているため，これを把握することは重要である．
- **作成者名**

  作成者の名前を提供することにより，エージェントの有用性や信頼性を示すことが可能である．
- **ルール名**

  DASH のエージェントはルールが発火することにより動作する．そのためルール名からエージェントのおおよその振る舞いを把握することが可能である．
- **生成時刻**

エージェントの新旧を把握するために用いる．
- **利用回数**

  エージェントの有用性や信頼性を判断する際の参考値として用いることが可能である．また，開発者は利用者がどのような特徴を持つエージェントを必要としているか知ることができる．
- **環境名**

  現在エージェントが管理されているリポジトリを把握するために用いる．また，作成者名と同様にエージェントの有用性や信頼性を示すことが可能である．

2.6 エージェント情報の記録

アプリケーションサーバに送信される情報はリポジトリ一つに対し，一つのテキストファイルを作成し保存する．この際作成されるファイルは＜リポジトリ名＞.txt とする．ファイル内の構成は，テキストファイルの一行の情報がエージェント一つの情報と対応している．

2.7 アプリケーションサーバとの通信

利用者が常に複数リポジトリに存在するエージェントの情報を取得可能な状態にするためには，アプリケーションサーバ内で記録する情報と，各リポジトリの情報を同期させる必要がある．このためには，リポジトリ内の

状態が変化するたびに，アプリケーションサーバに変化の内容を通知しなければならない．また，提案機構ではエージェントの利用回数も提供するので，アプリケーションサーバに通知を行うタイミングとしては次のものが考えられる．

- リポジトリにエージェントを追加
- リポジトリのエージェントの消滅
- エージェントを動作環境に生成

以上のタイミングでリポジトリからの情報の通知を行う．これらの通知はサーブレットを用いた HTTP 通信により行う．リポジトリは通知を行う際，アプリケーションサーバに対し POST メソッドを送信し，その後エージェント情報を送信する(図 4).

2.7.1 エージェント追加の通知

各リポジトリからアプリケーションサーバに対し，エージェントが追加されたことを通知する．追加の際にリポジトリに送信する必要のある情報はエージェント名，作成者名，ルール名，環境名，通知内容である．この際，エージェント名と環境名は DASH 内部で利用されている値，作成者名とルール名はエージェントのソースコードから読み取った値を使い，通知内容は“add”として通知する．

アプリケーションサーバ側では，“add”の通知を受け取るとリポジトリごとに記録用のファイルを作成し，そこに送信された内容を，現在時刻，利用回数 0 回と書き込む．

2.7.2 エージェント削除の通知

各リポジトリからアプリケーションサーバに対し，エージェントが削除されたことを通知する．削除を行う場合はエージェントの特定が行えればよいので，エージェント名，環境名，通知内容をアプリケーションサーバに送信する．エージェント名と環境名は，追加の場合と同様に DASH 内の値を使い，通知内容は“delete”とする．

アプリケーションサーバ側では，“delete”の通知を受け取ると環境名から記録されているファイルを特定し，ファイル内の同一名のエージェントの情報を削除する．

2.7.3 エージェント利用の通知

アプリケーションサーバに対し，エージェントが利用されたことを通知する．これはリポジトリから利用者の動作環境に生成された場合に，エージェントが 1 回利用されたと判断し，通知をする．この際，アプリケーションサーバに対してはエージェント名と環境名，通知内容を送信する．アプリケーションサーバへ送る通知内容は“used”とする．

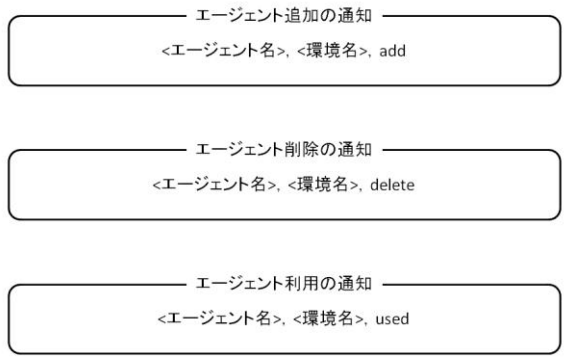

図 4：アプリケーションサーバへの通知内容

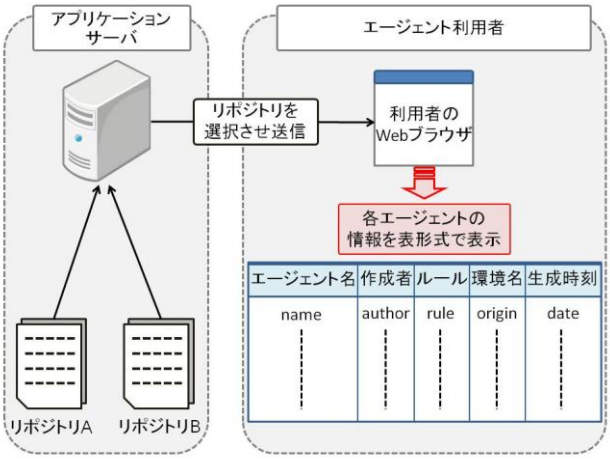

図 5：表示機能

アプリケーションサーバ側では，“used”の通知を受け取ると削除の場合と同様に，エージェントを特定し利用回数を 1 回追加する．

2.8 利用者への表示

アプリケーションサーバに保存したエージェントの情報を，利用者の Web ブラウザ上に表示する(図 5)．利用者にエージェントの情報を表示する際，エージェント数が多すぎると利便性の低下を招く恐れがある．そこで，エージェントの情報を表示する前に，利用者が閲覧するリポジトリを選択し，そのリポジトリの情報のみを表示する．

はじめに，利用者には現在アプリケーションサーバ内に情報を保存しているリポジトリの一覧を表示する．一覧の中から選択をすると，アプリケーションサーバのプログラムが情報を保存したファイルを読み込み，利用者の Web ブラウザに表示する．表示する情報は表形式とし，さらに JavaScript を使い各項目ごとにソート可能なものにする．

3． 動作確認

作成した機構が適切に動作し，利用者に対し複数リポジトリ内のエージェントの情報が提供できるかの検証を行った．作成した機能が正常に動作するかの検証するために，以下の実験を行う．

- リポジトリへのエージェントの追加
- リポジトリのエージェントの消滅
- 動作環境へのエージェントの生成

3.1 検証に用いるエージェント

検証には DASH でサンプルとして提供されているエージェントを用いる．サンプルエージェントには複雑な機能は設計されていないが，今回の実験を行う上では十分であると考えられる．実験に用いるサンプルエージェントの情報を表 1 に示す．

表 1：検証に用いるエージェント

エージェント名	作成者	ルール名		
Sample010	Har@CIT	hello		
Sample020	Har@CIT	inspector	printmsg	
Smaple030	Har@CIT	author	worker	inspector
		sorry		
Sample040	Har@CIT	startup	worker	worker2
Sample050	Har@CIT	startup	send-message1	print
		send-message2	receive-message	broadcast
		lookup-test		
Sample060	Har@CIT	startup	success	mobile
Sample070	Har@CIT	startup	plus	multi
		divide	mod	
Sample080	Yoshihiro Hashimoto	startup	instantiate 1	instantiate 2
		settext1	settext2	terminate

3.2 追記機能

● 実験内容

リポジトリ内に 8 つのサンプルエージェントを追加し，その際にアプリケーションサーバに保存されている情報や，利用者へ表示される内容を確認する．

● 結果

リポジトリにエージェントを追加すると，アプリケーションサーバに通知され，情報の保存が行われた．保存された情報は，項目ごとにカンマ区切りで正しく保存されている．さらにこの状態で Web ブラウザを使い，利用者への表示画面を確認した．図 3.1 に示したようにエージェントの情報が正しく表示されることを確認した．

3.3 削除機能

● 実験内容

先の実験でリポジトリに追加されたエージェント中から特定のものを消滅させる．

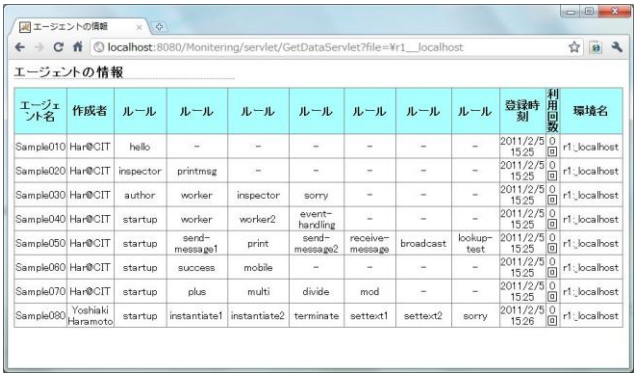

図 3.1：利用者への表示例

● 結果

リポジトリからエージェントを消滅させると，アプリケーションサーバに通知され，保存されていた情報が削除された．削除機能が正しく動作していることを確認した．

3.4 利用回数提供機能

● 実験内容

リポジトリ内のエージェントを利用者の動作環境に生成した際の，提案機構の動作を確認する．

● 結果

エージェントの生成を行うと，動作環境からアプリケーションサーバに通知が行われた．そしてこの通知を受け，アプリケーションサーバは対象のエージェントの利用回数を追加した．また，万が一エージェントの情報が保存されていない場合は追加失敗が報告される．

3.5 考察

実験の結果提案機構はリポジトリへのエージェントの追加や削除を反映させ，利用者に情報を提供することが確認できた．しかし，実験の過程で今後改良すべき課題も見られた．

現在の機構では，エージェントの機能についての情報はルール名のみを提供している．DASH の特徴を活用するためには，エージェントシステムにおける役割など，より多くの情報を加えることが必要となる．また利用者への表示方法も改良が必要である．今回の実験では，所有するルール数が少ないものを用いた．しかし，実際にはより多くのルールを持つエージェントも考えられる．現在の機構でルール数の多いエージェントの情報を提供すると，利用者に対し非常に大きい表を表示することになり，利便性が低下してしまう．今後は利用者に情報を提供する方法を工夫し，機構の利便性向上を目指す必要がある．

4. まとめ

本研究はエージェントフレームワーク DASH における，利用者へのエージェントの情報提供を目的とし，Web ブラウザを用いたリポジトリのモニタリング機構を提案した．今後は提案手法において取り扱う情報量を増加させ，さらなる利便性向上を目指す．

参考文献

[1]“DASH ユーザマニュアル”
http://uchiya.web.nitech.ac.jp/idea/html/
[2]宮本信二，“基礎からのサーブレット／JSP”，
Soft-Bank Creative，2010.
[3]山田祥寛，“JSP サーブレットサンプル集”，
秀和システム，2008.
[4]“Tomcat”，
http://tomcat.apache.org/