OE2-5 エージェントサーバのWebサービス統合基盤への応用

Applying an agent server to a Web Services integration platform

小山 和也† 富沢 伸行† 藤田 悟†
Kazuya Koyama Nobuyuki Tomizawa Satoru Fujita

1．まえがき

従来より、人手によるネットワークサービス利用を支援、自動化など代行する技術としてエージェントが注目されてきた。加えて近年の Web サービスの登場により、エージェントを実際に広範囲のサービスで利用出来る環境が整いつつある。そこで筆者らは、エージェント自身も実際のビジネス領域などで利用可能とするべく、大規模・高信頼なエージェント実行基盤であるエージェントサーバ MobidgetLite の研究を行ってきた。

本稿では、この MobidgetLite の Web サービス統合基盤 BizEngine への適用について述べる。Web サービス統合基盤はビジネス領域で企業間連携を実現するものであり、本適用により、MobidgetLite エージェントの実用可能性を実証すると共に、Web サービス用の機能をエージェントから利用可能にした。

2．エージェントの実用に向けて

2.1 エージェントと Web サービス

従来より、電子市場の入札状況の監視、自動入札、複数予約処理をまとめて行う旅行予約サービスなど、人手によるネットワークサービス利用を支援、自動化など代行する技術としてエージェントが注目されてきた。しかしこうしたエージェントは、実際に利用出来るのは特定の領域やサービスに限定したものが大半であり、広範囲なサービスに適用可能な実利用できるエージェントシステムはなかなか現われなかった。理由は様々考えられるが、多くの既存サービスは人間がブラウザ経由で操作する事を対象としておりエージェントが一般的に利用できるサービスがほとんど存在しなかった事と、エージェントが研究を主目的としていたものが多く、実利用、特にビジネス領域での利用に必要な信頼性などの要件を考慮していないものが多かった事も、大きな理由であると思われる。

しかしながら、近年 XML や HTTP、SOAP などをベースとした Web サービスを巡る開発が活発になってきた。Web サービスでは、従来 WWW ブラウザを介してしか利用出来なかった各種インターネット上のサービスが XML により容易にプログラム処理可能になる事から、これによりエージェントが操作可能なネットワークサービスが多数出現し、エージェントを実用で利用できる環境が整うと期待される。

2.2 エージェントサーバ MobidgetLite

我々はこの様な観点から、エージェント自身もビジネス領域などで利用可能とするために、大量のエージェントを安全・確実に実行するための基盤として、エージェントサーバ MobidgetLite を開発してきた[1]。

MobidgetLite はサーバ上で動作するエージェントプログラムの実行基盤であり、ユーザにより記述されたエージェントの実行、障害時復旧のためのエージェント状態の DB への定期的保存、スワップによるメモリ管理などを行う事で、数万といった単位のエージェントを生成し安全に実行管理する事が可能である。

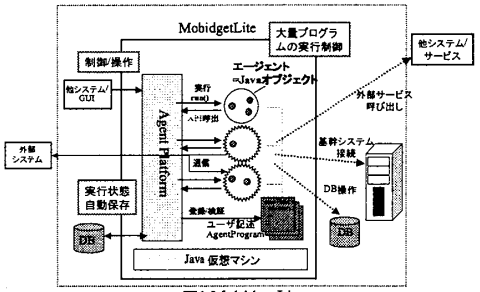

図1 MobidgetLite

MobidgetLite は Java 上のミドルウェアであり、エージェントは一つの Java インスタンスとして表現され、プログラムはそのメソッドとして記述する(図 1)。AgentPlatform と呼ぶ実行管理機構が多数のエージェントインスンタンスを管理し、そのメソッドを繰り返し呼び出す事でエージェントの処理を実行する。さらに、1 回のエージェントのメソッド呼出しを 1 トランザクションとしてエージェントと AgentPlatform の状態を DB に保存し、サーバ障害時の保存状態からのシステム復旧を可能にする。さらにメモリ管理として休止中のエージェントを DB に保存後にメモリ上から削除し、必要時に DB から復旧する事で、少ないメモリでの大量エージェントの処理を可能にする。

3. Web サービス統合基盤への応用

3.1 Web サービス統合基盤 BizEngine

MobidgetLite により大量のエージェントを安全確実に動作する事は可能になった。しかしながら MobidgetLite の提供する機能は単純なものであり、複雑な Web サービスの通信プロトコルへの適応や、容易なエージェントの構築と言った面で、単体で使用するには十分とは言えなかった。

一方 Web サービスの領域では、複数の Web サービスの実行順序や関連する通信の手順、システム内部動作等をワークフローとして記述、実行するシステムが検討されている[2]。このようなシステムでは、予め処理手順が定められている場合には、容易に処理の自動化が可能である。当社でも以前からこのようなワークフロー処理を行うための Web サービス統合エンジン「BizEngine」を開発していた。BizEngine はワークフロー処理を実行するプロセスエンジンと、RosettaNet や SOAP、ebXML などの通信プロトコルを実現するプロトコルハンドラからなり、これらの通信プロトコルを介したサービスの利用や提供とそれらのワークフローによる連携を可能としている。

そこで我々は MobidgetLite の Web サービス領域での利用の容易化と、実用性の実証を目的として、BizEngine を MobidgetLite 上に移植した。以下その詳細について述べる。

† 日本電気株式会社

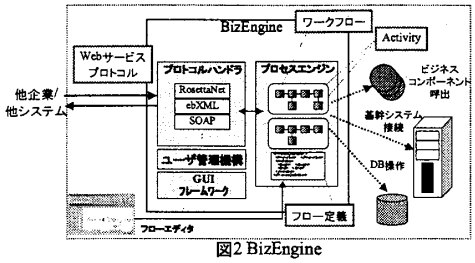

図2 BizEngine

3.2 BizEngine システム構成

BizEngine は通信機構であるプロトコルハンドラとワークフローエンジンであるプロセスエンジンからなる(図 2)。

プロトコルハンドラは複雑な Web サービスのプロトコルを内部でハンドリングし、ワークフローに単純なアプリケーションレベルでのメッセージ送受信インタフェースを提供するものである。

プロセスエンジンは、ワークフローとして定義されたプロセスを高信頼に実行する。ワークフローは基本的な処理単位である Activity の集合とその順序関係として定義される。Activity は条件分岐や並列分岐などの一般的なフロー処理の他に、ユーザが定義した任意のプログラムやコンポーネントを呼び出す事が出来る。プロセスエンジンは個々の Activity を atomic な処理単位として実行する。フロー定義は GUI ツールを使用して作成し、専用の XML 形式で出力され、プロセスエンジンに登録、実行される。

3.3 MobidgetLite の BizEngine への適用

MobidgetLite の BizEngine への適用はプロセスエンジンに対して行った(図 3)。

プロセスエンジンは MobidgetLite のエージェントとほぼ同等の機能を有しており、大きな違いは動作の記述方式が Java かフロー記述であるかのみである。よって本適用ではプロセスエンジン のワークフロー記述方式に依存した機能をワークフローインタプリタとしてプロセスエンジンより切り出し、これを一つのエージェントプログラムとして MobidgetLite 上で動作させた。ワークフローインスタンスは 1 種のエージェントとして実現され、その実行状態はエージェントの内部状態として保存される。ワークフローインタプリタはエージェントプログラムの 1 種となり、AgentPlatform から呼び出されると、1 トランザクションで実行すべき Activity のみを実行し、実行結果をエージェントの内部状態に反映する。この結果、大規模・高信頼の実現に共通に必要な実行スケジューリングや永続化、メモリ管理などの処理は MobidgetLite で実現され、動作の記述方式に依存した処理がエージェントプログラムとして実現される事になった。

一方プロトコルハンドラは MobidgetLite の内蔵通信機構と類似するが、プロトコルハンドラの機能は複雑で内蔵通信機構上に実現するのは困難である事から、内蔵通信機構は利用せず、プロトコルハンドラを外部システムとしてそのまま利用し、内蔵通信機構を用いて接続している。

3.4 実行速度高速化

ワークフローの実行状態情報はエージェントとしてにより DB に保存されるが、この情報は MobidgetLite が当初想定していたエージェントサイズより大きいため、フローの規模によっては DB 負荷が高くなり、上記の単純な実装だけでは実行速度が遅くなる場合があった。そこで高速化の改造を行った。

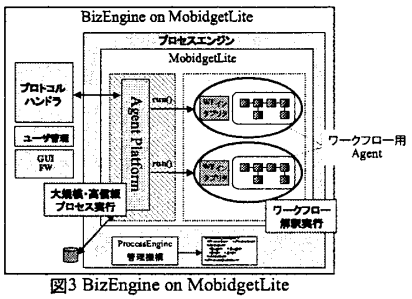

図3 BizEngine on MobidgetLite

MobidgetLite では、エージェントの状態を直列化した一つのデータとして DB に格納している。エージェントが比較的小さい場合はこの方式が高速に動作できるが、エージェントが大きくなると逆に細かく分割して変更部分のみ DB を更新するようにした方が高速である。そこで、1 つのワークフローの状態を複数エージェントに分割保持可能にし、かつ、フロー規模に応じて状態を分割するか否かが自動的に切り替わるようにした。これによりフローの規模によらず高速動作する事が可能となっている。

4. 考察

MobidgetLite を適用した BizEngine は既存システムの移植であるため、本適用によって機能的利点が生まれたわけではない。しかしながら本適用により、大規模・高信頼なエージェント実現のための機能と、特定の動作の記述方式に依存する部分が明確に分離されたため、今後、例えば AOP など何らかのエージェントモデルに基づく記述方式に変更すれば、従来のエージェント研究の成果を活かした実用的なエージェントシステムの実現が期待できる。

一方、MobidgetLite の永続化、トランザクション管理、メモリ管理等は Enterprise Java Beans(EJB) でも提供されている機能であり、EJB でも類似のシステム構築は可能である。事実、移植前のプロセスエンジンは EJB 上に実装されていた。しかし EJB は本来ビジネスコンポーネント実装のためのフレームワークであり、本稿のようなエージェント実行基盤の実装には必ずしも適さない。一番の理由は性能であり、今回の適用の場合、ワークフローの実行速度は MobidgetLite 化によりおよそ 3 倍に向上した。

5. まとめ

本稿では、エージェントの実用化を目指した実行基盤エージェントサーバ MobidgetLite と、その Web サービス統合基盤 BizEngine への適用について述べた。MobidgetLite で実現された BizEngine は、数千～数万といったプロセス数での動作や、実運用に耐える安全性が確認され、既に企業間システム連携などの領域で使用されている。

現状では実現されたシステムは従来のものと大きく変わるものではないが、本研究により実利用可能なエージェントシステムの構築へのハードルが一段低くなったものと期待している。

参考文献

[1] 小山,富沢,藤田,山之内,"大規模エージェントサーバ MobidgetLite",情報処理学会第 64 回全国大会.

[2] F. Leymann, "Web Services Flow Language (WSFL 1.0)",IBM,2001.