企業情報統合ミドルウェアを実現する 多段購読コンポーネントモデル

非会員 石原 鑑* 非会員 大崎 雅代*
正 員 高田 秀志** 非会員 小島 泰三*
非会員 杉本 明*

A Component Model with Multilevel Subscription for Realization of Enterprise Application Integration Middleware

Akira Ishihara*, Non-member, Masayo Osaki*, Non-member, Hideyuki Takada**, Member,
Taizo Kojima*, Non-member, Akira Sugimoto*, Non-member

This paper describes a component model suitable for development of Enterprise Application Integration Middleware in the industrial field. In this model, there are two types of subscription. In component level subscription, an interaction between components is defined by subscription to changes of component's attributes or to occurrences of spontaneous events. A connector object attached to a subscribed component manages these inter-component dependencies. This results in independency of a component and enables localized modifications when systems are modified or extended. In application level subscription, adaptor components map inter-component dependencies into inter-application dependencies. In this paper, our proposed middleware, DiaSynapse, has been developed based on this model, and applied to the development of wide-area monitoring systems and network management systems.

キーワード：コンポーネント，ミドルウェア，発行購読モデル，企業アプリケーション統合

Keywords: component, middleware, publish-subscribe model, enterprise application integration

1. はじめに

企業の経営効率化や業務改善を目的に，企業アプリケーション統合 (EAI:Enterprise Application Integration) システムの開発が加速している。EAIシステムは，企業内外の複数のアプリケーションを統合し⁽¹⁾，ビジネスプロセスを実行する。ビジネスプロセスとは「企業目的を達成するための活動や作業のつながりである⁽⁴⁾」とする。アプリケーション (以下 AP) とは，活動や作業の一部を計算機システム上で実現したプログラムあるいはデータベースを指す。

EAIシステムの機能要件として，以下が挙げられる。

- APに固有の利用手順 (インタフェース，プロトコル)

に従ってデータを取得し，それを他の AP が利用できるように変換する。本研究ではこの機能要件を，AP と EAI の接続と呼ぶ。

- AP から取得したデータを適切な AP に配送する。この機能要件を AP 間連携の定義と呼ぶ。
- 既存の AP を連携させただけではビジネスプロセスが達成できないとき，新たな AP を開発し，ビジネスプロセスに組み込む。本研究では，これをカスタム AP と呼び，この機能要件をカスタム AP の定義と呼ぶ。

これらの機能要件を満たすべく EAI のミドルウェアが開発されている。メッセージブローカ⁽¹⁾では，発行購読モデル (Publish/Subscribe Model) に基づくメッセージフローとして AP 間連携の定義を行う。また AP とメッセージフローを接続するためにアダプタが配置される。Web アプリケーションサーバ⁽²⁾⁽³⁾は，EJB⁽²⁾等のコンポーネント間の相互作用にて AP 間連携を定義し，AP とコンポーネントをアダプタを介して接続する。ワークフロー⁽⁴⁾⁻⁽⁷⁾は，ビジネスプロセスを直接的に表現したプロセスフローとして AP 間連携の定義を行う。また AP とプロセスフローを接続するためにアダプタが配置される。ワークフローでは，

* 三菱電機 (株) 先端技術総合研究所

〒661-8661 兵庫県尼崎市塚口本町 8-1-1

Advanced Technology R&D Center, Mitsubishi Electric Corp.
8-1-1, Tsukaguchi-Honmachi, Amagasaki City, Hyogo 661-8661

** 京都大学情報学研究科

〒606-8501 京都府左京区吉田本町

Graduate School of Informatics, Kyoto University
Yoshida Honmachi, Sakyo, Kyoto, 606-8501

カスタム AP を開発し、その入出力データを明確化し、既存のプロセスフローに組み入れることで、カスタム AP の定義を実現している。実際に、銀行系の営業店システムで新商品や新サービスの追加のためにこの特徴が利用されている。

産業分野では、例えば広域統合監視システム (Wide-Area Monitoring Systems, 以下 WAMS) やネットワーク管理システム (Network Management Systems, 以下 NMS) は、センサ、コントローラ、ネットワーク機器等の対象設備に対する監視、制御、維持管理の業務を実現するために、複数の AP を統合する EAI システムである。本研究では、一般の EAI システム (ビジネス系 EAI システムと呼ぶ) と区別して、WAMS や NMS を産業系 EAI システムと呼ぶ。産業系 EAI システムでは、以下の独自の機能要件がある。

- ビジネス系で扱うデータは、伝票、帳票形式で表現することが容易な定型データであることが多いが、産業系で扱うデータは、対象設備を業務の様々な視点から表現した非定型データであることが多い。これらに関連付け、かつ AP 間で共有する機能が必要である。また、AP 間のデータ連携がより頻繁である。例えば、ある AP が自身のデータを変更した場合、それと関連付けられる別の AP のデータを迅速に更新し、EAI システム全体でのデータの整合性を保たなければならない場合がある。
- メッセージフローとプロセスフローが混在する。例えば、NMS の監視業務はほとんど人手を介さないメッセージフローとして表現できる。一方、維持管理業務は、維持管理作業を計画、実行、承認する典型的なプロセスフローである。これら連携させ、これらの間で警報や作業状況等のデータを受け渡す必要がある。

先に挙げた機能要件と、産業系独自の機能要件を満たすことを目的として、本研究では、発行購読モデルを応用した、多段購読コンポーネントモデル (Component Model with Multilevel Subscription, 以下 CMMS) を提案し、CMMS を内蔵したミドルウェア DiaSynapse を開発する。ここでコンポーネントとは、ある特定のアーキテクチャのもとで開発、使用される、再利用かつ交換可能なソフトウェアモジュールである^{(9)~(11)}。本研究では、DiaSynapse を WAMS と NMS の開発に適用し、CMMS とそれを内蔵したミドルウェアが産業系 EAI システムの機能要件を満たしていることを検証し、生産性向上の観点からその有効性を示す。

以下 2. 章では産業系 EAI システムの機能要件についてまとめ、3. 章では CMMS を提案し、4. 章では CMMS を内蔵したミドルウェア DiaSynapse の実現について説明し、5. 章でシステム開発への適用による評価を行う。6. 章は関連研究と考察であり、7. 章はまとめである。

2. 産業系 EAI システムの機能要件

本章では、用語の定義を行い、WAMS と NMS を例に産業系 EAI システムの機能要件を考察する。

図 1 産業系 EAI システムの構成例

Fig. 1. An example of industrial EAI systems.

2.1 用語の定義 EAI システムをシステム構成によりハブ&スポーク形態とバス形態に分類する⁽¹⁾。ハブ&スポーク形態では、AP 間にデータの流れるとき、それを管理するプログラムであるハブを配置する。バス形態では、ハブを配置せずに AP 間を直接接続する。

A という AP が EAI システムにデータを入力し、EAI システムがそれを B という AP に配送するとき、本研究では、A をソース AP、B をターゲット AP と呼ぶ。また、ソース AP からターゲット AP へデータが流れる一連の処理を、AP 間連携処理と呼ぶ。

2.2 産業系 EAI システム 図 1 に産業系 EAI システムの構成例を示す。WAMS は、地理的に分散している監視 AP (Monitoring Applications)、警報管理 AP (Alarm Management Application) を統合する。NMS ではこれに作業管理 AP (Maintenance Task Management Application) が加わる。ユーザは、表示端末 (HMI) や Web 端末 (Web HMI) を利用して EAI システムに接続する。以下では、WAMS と NMS の特徴を併せて述べるとき、WAMS/NMS と示す。

監視 AP は、プロセスデータ管理機能、警報管理機能、ヒストリー管理機能を持つ⁽¹³⁾。プロセスデータ管理機能は監視対象の物理データをセンサで取得し、それに値の妥当性や時刻印を付加してプロセスデータと呼ばれるデータを生成する。警報管理機能は、プロセスデータの異常を検出して警報を発生させる。ヒストリー管理機能は、プロセスデータからヒストリーデータを生成する。ヒストリーデータは、プロセスデータを収集時刻に応じて、時単位、日単位、月単位というようにグループ化し、統計処理を施すことにより生成される。作業管理 AP は、ネットワーク機器や設備の維持管理作業の計画、実行、承認を行う一種のワークフローである。

WAMS/NMS は以下のような主機能を実現する。

(主機能 1) 監視 AP が持つプロセスデータと警報を、表示端末と Web 端末上で閲覧可能とする。

(主機能 2) 複数の監視 AP のプロセスデータやヒストリーデータを集計し、新たなヒストリーデータを生成する。

(主機能3) 監視 AP のプロセスデータや警報を警報管理 AP に配送する。警報管理 AP はプロセスデータや警報と、設備管理データベース(図1の Equipment Database) 上の設備データを用いて、監視 AP 単独では検出できない警報を発生させる。

(主機能4) 主機能4では、監視 AP の警報を表示端末や Web 端末で監視する監視業務と、設備の維持管理業務を連携させる。EAI システム(図1の EAI System) は作業管理 AP から設備の管理作業開始通知を受け、対応する設備データを特定し、これに基づいて監視 AP の警報をフィルタリングする。この結果、維持管理作業対象に関する警報は、表示端末や Web 端末で表示されなくなる。

各監視 AP が異なる企業や組織で開発されたり、既存の監視 AP を統合することが求められるので、あらかじめデータ形式や利用手順を統一しておくことが難しい。すなわち、AP と EAI の接続が重要となる。

主機能1, 3, 4は AP 間連携処理を含む。主機能3では、ソース AP は監視 AP であり、ターゲット AP は警報管理 AP である。このように WAMS/NMS では AP 間連携処理が多くの機能の実現に利用されるため、これを容易に定義できることが望ましい。

主機能2を取り上げて、カスタム AP の具体例を説明する。WAMS の監視対象が広域雨量データであり、監視 AP は市町村単位の雨量データを計測しているとする。カスタム AP は市町村単位の雨量データを入力とし、県単位のヒストリーデータを生成する。また、Web 端末などの他の AP から、生成したヒストリーデータを利用できるようにする。

2・3 産業系独自の機能要件 対象設備の静的な構造を表す設備データを中心として、プロセスデータ、警報、ヒストリーデータ、作業管理データ等が、様々な業務の視点から定義されている。これらを統合的に扱うことで、個々の業務を効率化することが産業系 EAI システムの目的の一つである。例えば、主機能4が実現される以前は、作業データを手作業で監視 AP に入力しており、手間がかかっていた。これらデータは、伝票や帳票形式で扱うことが難しい非定型なデータであり、これらを EAI システム上で構造化し、AP 間で共有可能とすることが求められる。

産業系 EAI システムでは、実行形式の異なる処理のフローを連携させ、それらの間でデータを受け渡す。例えば、上記の県単位のヒストリーデータの生成では、監視 AP から雨量データを取得する処理は秒単位に行われるリアルタイム性の高い処理であり、ヒストリーデータの生成は分単位、時間単位、日単位に周期的に実行される処理であり、ユーザが Web 端末からヒストリーデータを検索する処理はユーザイベントで駆動される処理である。

主機能4はメッセージフローとプロセスフローが混在する例となっている。監視業務はほとんど人手を介さないメッセージフローとして表現でき、維持管理業務は典型的なワークフローである。維持管理業務の開始は、監視業務から維

持管理業務に警報イベントが入力されたときである。また、維持管理作業を始めると、設備データにフラグが立ち、監視業務では、その機器に関する警報を無視する。このように異なる形式の処理フロー間でデータを授受できる枠組みが必要である。

3. 提案モデル

3・1 提案モデルの概念 2・2 節の主機能1から3を例に、図2を用いて、CMMS の基本概念を説明する。主機能1から3を実現するには、図2(a)に示すように、Web 端末(Web HMI)、監視 AP(Monitoring Application)、警報管理 AP(Alarm Management Application)を連携させる。さらに、主機能2を実現するには、新たにヒストリーデータを生成するカスタム AP を定義する。

図2(b)を用いて CMMS の基本概念を説明する。CMMS では、AP 間連携処理が必要ときには、EAI システム上にコンポーネントを配置し、AP はコンポーネントと接続するようにする。

EAI システム上には、以下の種類のコンポーネントを配置する。

- AP のデータ形式や利用手順を理解し、AP からのデータ取得や配送を行なうコンポーネント(図2(b)の Component A, D, E)
- データ形式を AP に依存しない形式に変換するコンポーネント(図2(b)の Component B)
- データを保持するコンポーネント(図2(b)の Component C)
- カスタム AP を定義するコンポーネント。図2(b)では、Component Fm, Fh, Fd がそれに当たり、それぞれ分単位、時間単位、日単位でプロセスデータをグループ化しヒストリーデータを生成する。

このように細かな粒度でコンポーネントを構成すれば、AP のデータ形式や利用手順が変わっても、影響を受けるコンポーネントが限定されるという効果がある。その反面、コンポーネントの数が増え、コンポーネント間の接続関係が複雑化してしまう欠点がある。そこで、コンポーネント間の自由なメソッド呼び出しは禁止し、全て発行購読モデルによりコンポーネント間を接続する。

さらに、発行購読モデルにより取得したデータを処理する時期、発行購読モデルによりデータを発行する時期を定義できるようにコンポーネントを構成しておき、これにより処理の流れを管理する。例えば図2(b)の Component C から Fh へは、C のプロセスデータが更新されるたびにデータが流れ、Fh の統計処理は一時間に一度実行されるように定義できるものとする。

図2(b)では、Web 端末は Component D と接続して監視 AP のプロセスデータを取得している。新しくヒストリーデータを表示する機能が必要となったなら、Component D は図2(b)の Component Fm, Fh, Fd と接続する。

以上が CMMS の基本概念である。なお、2・2 節の主機

図 2 提案モデルの基本概念

Fig. 2. Basic concept of proposed model.

図 3 コンポーネントの内部構造

Fig. 3. Component internal structure.

能 4 については、4.4 節にて、CMMS を内蔵するミドルウェア DiaSynapse の実現の中で詳しく述べる。本章の以降の節では CMMS について詳細に説明する。

3.2 発行購読モデルを応用した階層型コンポーネントモデル CMMS は、発行購読モデルを応用して、下位コンポーネントの上に段階的に新規コンポーネントを積み上げていく階層型のコンポーネントモデルである。CMMS では、コンポーネントレベル発行購読関係 (Component Level Subscription Relation, 以下 *CLS* 関係) とアプリケーションレベル発行購読関係 (Application Level Subscription Relation, 以下 *ALS* 関係) を定義し、それらを用いて *AP* と *EAI* の接続と *AP* 間連携処理の定義を行なう。*CLS* 関係は、コンポーネント間の発行購読モデルによる接続をモデル化する。*ALS* 関係は、ネットワーク上に分散するコンポーネント間の発行購読モデルによる接続と、ネットワークを介したコンポーネントと *AP* との接続をモデル化する。

3.3 *CLS* 関係 図 3 は、CMMS のコンポーネントの内部構造を図示している。コンポーネントは、属性の集合 (attribute set)、変換処理 (conversion)、そして実行処理 (execution) を持つ。変換処理と実行処理の内部でイベントが発生する。コンポーネントは外部とのインタフェースとしてデータ要求端子 (data request terminal) とデータ提供端子 (data provide terminal) の集合を持つ。コンポーネント、コンポーネントのデータ要求端子、データ提供端子はそれぞれ一意な名前を持つ。

図 2(b) ではコンポーネント間の点線矢印で *CLS* 関係を表している。二つのコンポーネントが接続されているとき、

図 4 コンポーネントの接続モデル

Fig. 4. Model of component connection.

矢印が指すコンポーネント (データ発行者) のデータ発行処理が実行されると、矢印の根元のコンポーネント (データ消費者) のデータ消費処理が実行される。データ発行処理とは、コンポーネントが属性の集合からある属性値を取り出してデータ提供端子に置くことである。データ消費処理とは、コンポーネントのデータ要求端子に値が設定されることである。この関係を、{データ発行者 $\xrightarrow{CLS}$ データ消費者} と表す。*CLS* 関係は、データ消費者がデータ発行者の属性の変更を購読する *attribute-CLS* と、データ発行者の変換処理あるいは実行処理で発生するイベントを購読する *event-CLS* の二種類である。

図 4 は、CMMS におけるコンポーネントの配置とコンポーネント間の接続の方法を示している。CMMS では、アプリケーション層 (Application-Layer) とミドルウェア層 (Middleware-Layer) に分けてコンポーネントを管理する。コンポーネントはアプリケーション層に配置される。コンポーネントは、ミドルウェア層に、固有の接続管理者 (Connector Object) を一つ持ち、コンポーネントを特定すれば、その接続管理者が特定できるものとする。接続管理者は、対応するコンポーネントを購読先とする *attribute-CLS* を管理する。以下で *attribute-CLS* によるコンポーネントの接続手順を示す。項番は図 4 中の矢印横の数字に対応している。

- (1) データ消費者 *Cs* は、ミドルウェア層に対してデータ購読を発行する。データ購読は、{データ発行者名、データ提供端子名、データ消費者名、データ要求端子名} の四つ組みである。ミドルウェア層はデータ購読に含まれるデータ発行者名から、データ発行者 *Cp* を検索し、*Cp* に固有の接続管理者 *Mc* にデータ購読を登録する。
- (2) *Cp* は、属性の更新が起こると、データ提供端子に属性値を出力する (データ発行処理)。
- (3) *Mc* は、データ提供端子名を検索キーにデータ購読を検索し、*Cs* のデータ要求端子に属性値を配送する (データ消費処理)。

図 4 には図示されていないが、ミドルウェア層にイベント配送者を配置し、イベント配送者により *event-CLS* を

図 5 コンポーネントの遠隔接続モデル

Fig. 5. Model of component remote connection.

図 6 アプリケーションとの接続モデル

Fig. 6. Model for connecting to an application.

管理する。

3・4 ALS 関係 コンポーネントがネットワーク上に分散して配置されているとき、CLS 関係を用いてコンポーネント間を接続する方法について図 5 を参照しながら述べる。CMMS では、分散コンポーネント間の CLS 関係を管理するアダプタコンポーネント (Adaptor Component) をミドルウェア層に配置し、以下に述べる手順で分散コンポーネント間で CLS 関係を確立する。なお、以下に述べる手順では、attribute - CLS について説明し、項番は図 5 中の矢印横の数字に対応している。

- (1) データ消費者 $Cs1, Cs2$ は、ミドルウェア層に対してデータ購読を発行する。ミドルウェア層はデータ購読に含まれるデータ発行者名からデータ発行者 Cp が存在するアドレス空間に接続するアダプタコンポーネント As を検索し、 As にデータ購読を登録する。このとき、 As と $Cs1, Cs2$ 間を CLS 関係で接続する。 As はデータ購読を、データ発行者側のアダプタコンポーネント Ap に配送する。このとき、 Ap と Cp 間を CLS 関係で接続する。
- (2) Cp は、属性の更新が起こると、データ提供端子に属性値を出力する。 Ap は、データ提供端子名を検索キーにデータ購読を検索し、 As にデータ更新を配送する。
- (3) As は、データ更新に該当するデータ購読を検索し、 $Cs1, Cs2$ のデータ要求端子に属性値を配送する。この関係は、 $\{Cp \xleftarrow{CLS} Ap \leftarrow As \xleftarrow{CLS} Cs1, Cs2\}$ と表す。

CMMS で、 AP と EAI の接続を行う方法を図 6 を参照して説明する。 AP に固有のデータ形式や利用手順を処理するアプリケーション特化アダプタコンポーネント (ASAC: Application Specific Adaptor Component) を AP ごとにミドルウェア層に配置する。 AP には一意な名前付けをする。また AP が持つデータや機能のうち、EAI システムで利用するものについても名前付けする。このとき以下のように AP と EAI を接続する。項番は図 6 中の矢印横の数字に対応している。

- (1) データ消費者 Csa は、ミドルウェア層に購読要求を発行する。購読要求は $\{AP \text{ 名}, AP \text{ が提供する}$

図 7 AP 間連携処理のモデル

Fig. 7. Model for application cooperation.

データ (機能) 名, データ消費者名, データ要求端子名} の四つ組である。ミドルウェア層はこの購読要求の AP 名を検索キーに $ASAC$ を検索し、購読要求を登録する。他のコンポーネント $Cs1, Cs2$ は、 Csa と CLS 関係を結ぶ。 $ASAC$ は購読要求を処理し、 AP に固有のデータ形式や利用手順に従って購読要求を AP に送信する。

- (2) $ASAC$ が AP からデータを受信すると、購読要求に従って Csa のデータ要求端子にデータを配送する。結果として、 $Cs1, Cs2$ は AP のデータ、機能を利用できる。

3・5 AP 間連携の定義 本節では、CLS 関係と ALS 関係を用いて、 AP 間連携処理をモデル化する。図 7 を用いて、ソースアプリケーション SA とターゲットアプリケーション TA の連携を例に説明する。3・4 節で述べた手法に従い、 SA, TA に対してアプリケーション特化アダプタコンポーネント $ASAC(SA), ASAC(TA)$ を用意し、 AP のデータ、機能を EAI システム上のコンポーネント Csa, Cta として実現する。そして、CLS 関係を用いてこれらのコンポーネントを接続し、 AP 間の連携処理を表現する。連携の際にデータの型変換が必要ならば、型変換を行なうコンポーネント $Cconv$ を開発し、 Csa と Cta の間にはさみこむ。このとき、この連携は、 $\{SA \leftarrow ASAC(SA) \xleftarrow{CLS} Csa \xleftarrow{CLS} Cconv \xleftarrow{CLS} Cta \xleftarrow{CLS} ASAC(TA) \leftarrow TA\}$ というコンポーネントの接続関係としてモデル化できる。

4. ミドルウェア DiaSynapse の実現

本章では、CMMS を内蔵したミドルウェア DiaSynapse

の実現について述べる。DiaSynapse のプログラミング環境としては、Java2SE を用いた。DiaSynapse では、CMMS を内蔵することで AP と EAI の接続と、AP 間連携処理の定義を実現した。本章では、DiaSynapse の基本構造とコンポーネントの実現について述べた後、カスタム AP の定義について、DiaSynapse における実現方法を説明する。本章の最後で産業系 EAI システムの機能要件への対応について述べる。

4.1 基本構造 図 8 に DiaSynapse の構造を示す。CMMS のコンポーネントを最小粒度のコンポーネントとする。コンポーネントを組み合わせた複合コンポーネントをユニット (Unit) と呼ぶ。ユニットは、コンポーネントの機能や実行形式に基づいて、コンポーネントをグループ化する。またユニットはコンポーネントの実行条件定義を持つ。コンポーネントの実行条件定義については、4.3 節で後述する。ノード (Node) は、ユニットやコンポーネントを管理するもので、CMMS のミドルウェア層の機能を提供する。具体的にノードは、1) コンポーネントの名前管理、2) 接続管理者、イベント配送者の管理、3) CLS 関係に基づくコンポーネント間の接続、ALS 関係に基づく分散コンポーネント間の接続、コンポーネントと AP の接続、を行なう。

4.2 コンポーネントの名前管理手法 名前管理手法は、コンポーネント、コンポーネントの属性、イベント、データ要求端子、データ購読端子のそれぞれに一意な名前を与える。ノードはシステム全体で一意な名前を持つ。ユ

ニットはノード内で一意な名前を持つ。コンポーネントはユニット内で一意な名前を持つ。各コンポーネントの属性、端子、発生するイベントは、そのコンポーネント内で一意な名前を持つ。ノード、ユニット、コンポーネントという階層構造に従って、各名前を連結することでコンポーネントの正式名を定義する。DiaSynapse では正式名の実現形式に URL を用いた。例えば、図 9 のコンポーネント b1 の正式名は、synapse://n/u3/b1 である。このようなコンポーネント階層と URL に基づく命名法は、HADAS⁽¹²⁾ でも採用されており、それ自体新しいものではない。DiaSynapse では、このように定義した正式名を用いてネットワーク上のどこからでもコンポーネントや端子に接続し、CLS 関係を確立できることに特徴がある。

event – CLS のデータ購読におけるデータ発行者を指定する際にもこの正式名を使用する。ただし、URL を構成する部分名にワイルドカード指定を許す。例えば、synapse://n/u1/* という URL を指定することで、該当するユニットに含まれるコンポーネントで発生する全てのイベントを購読したことになる。

4.3 カスタム AP の定義 カスタム AP をコンポーネントの組み合わせとして開発し、CLS 関係や ALS 関係を用いて、既存のコンポーネント間接続関係に、カスタム AP を組み込む。

処理の流れをモデル化するために、DiaSynapse で以下のコンポーネントの実行形式を用意する。コンポーネントの実行形式を定義したものを、コンポーネントの実行条件定義と呼ぶ。コンポーネントを管理するユニットがコンポーネントの実行条件定義を持つ。

更新伝播による実行 データ消費処理を契機としてコンポーネントの実行を行う実行形式である。図 3 の黒矢印のフロー (data propagation flow) が示す処理の流れである。コンポーネントのデータ要求端子 (図 3 の data request terminal) がデータを受信すると、コンポーネントの変換処理 (図 3 の conversion) を実行する。変換処理の結果に応じて、コンポーネントはデータ発行処理を実行し、別のコンポーネントの変換処理が連鎖的に実行される。

条件による実行 周期的あるいは定時刻に行う処理等、ある条件が満たされたときに処理を行なう実行形式である。図 3 の白矢印 (execution flow) が示す処理の流れである。ノードは、ユニットが管理するコンポーネントの実行条件定義を検査し、満たされていればコンポーネントの実行処理 (図 3 の execution) を実行する。実行処理の結果に応じて、コンポーネントはデータ発行処理を実行する。実行条件定義として例えば、時間指定 (定周期、タイムアウト、定時)、データ受信時に常に実行、購読しているデータがそろったところで実行、等を指定できる。

メッセージングによる実行 ユーザが表示 AP (例えば、図 1 の HMI, Web HMI 等) を利用して行なう入力を契機としてコンポーネントの実行を行なう実行形式である。

図 8 ミドルウェアの構造
Fig. 8. Structure of middleware.

図 9 コンポーネントの名前管理
Fig. 9. Naming of components.

コンポーネントはメッセージを受けると、実行処理（図 3 の execution）を実行する。実行処理の結果に応じて、コンポーネントはデータ発行処理を実行する。

カスタム AP の定義について、例を用いて説明する。ソースアプリケーション SA1, SA2 のデータを利用するカスタム AP を実現し、その結果をターゲットアプリケーション TA に配送する例を考える。カスタム AP を実現したコンポーネント *Cintg* が開発されているとする。3・5 節で述べた AP 間連携処理のモデルに従い、SA1, SA2, TA1 のデータを持つコンポーネント *Csa1*, *Csa1*, *Cta* を EAI システム上に配置する。簡単のため AP 間連携処理のための型変換は必要ないものとする。このとき、データの流れの定義は、 $\{Csa(SA1), Csa(SA2) \xrightarrow{CLS} Cintg \xrightarrow{CLS} Csa(TA)\}$ というコンポーネント接続関係として表現できる。これに実行形式を指定することで処理の流れを定義する。例えば、*Cintg* の実行形式を “購読しているデータがそろったところで実行” と定義すると、SA1, SA2 からのデータがそろったところで *Cintg* の実行処理（図 3 の execution）が実行され、結果が TA に配送される。

4・4 産業系 EAI システムの機能要件への対応 AP と CMMS のコンポーネントを接続し、AP のデータをコンポーネントの属性として集約する。業務の視点から対象設備を表現する際には、コンポーネントを配置し、データの依存関係に従って、コンポーネント間を CLS 関係で接続する。この結果、対象設備のデータを関連付けし、AP 間で共有されるデータ共通空間をノード上で実現できる。AP がコンポーネントの属性を更新すると、CLS 関係に従ってデータが配送され、別のコンポーネントの属性に反映することができる。すなわち、EAI システム全体でのデータの整合性を保つことが容易である。

メッセージフローは以下のように実現される。ソース AP が発行したメッセージは、CMMS のコンポーネントの一連のデータ発行処理に一旦変換され、ターゲット AP へ配送される。このときコンポーネントは、他のメッセージフローやプロセスフローが利用できるようデータを自身の属性に保持しておく。

プロセスフローは以下のように実現される。DiaSynapse では、JEDI⁽¹⁸⁾ のワークフロー管理システム実現方式を CMMS 向けに変更してプロセスフローを実現した。JEDI では、プロセスフロー定義を持つアクティブオブジェクト（以下 AO）を設け、これが他の AO とイベントを交換することによりプロセスフローを実行する。DiaSynapse ではプロセスフロー定義を属性として持つコンポーネントを設け、他のコンポーネントとデータ、イベントを交換することでプロセスフローを実行する。

メッセージフローとプロセスフローの間のデータの連携は、CMMS コンポーネントの属性の集合で形成されているデータ共有空間を用いて行なう。この結果、DiaSynapse では、メッセージフローとプロセスフローが共に実行できるとともに、それらの間でデータを共有することが可能と

なった。

5. WAMS/NMS への適用と評価

本章では、DiaSynapse の WAMS/NMS の製品系列開発への適用事例に基づく評価を行う。

5・1 適用と評価 約 2 年間で 9 件のシステム開発を実施した。本研究では、これらのシステム開発を、その適用分野、開発チームの構成から製品系列 A, B に分類する。製品系列 A のシステム開発は 8 件、製品系列 B のシステム開発は 1 件である。

まず、DiaSynapse 適用前と適用後の WAMS/NMS のシステム開発を比較し、DiaSynapse の有効性を評価する。適用前は AP 間の連携ごとに専用の連携 AP を作成していた。適用前と適用後で連携 AP のステップ数を比較したところ、ほぼ同ステップ数であった。しかし、適用前は複数の連携 AP が存在したのに対し、適用後はそれらが EAI システム上のコンポーネントに集約され、システム構成が簡素化でき、保守性が向上した。

次に、DiaSynapse の適用による生産性の向上について、再利用性の観点から評価する。図 10 は、DiaSynapse 本体のステップ数（Middleware Code）、製品系列向けに開発された再利用コンポーネントのステップ数（Product Line Specific Code）、製品特化に開発されたコンポーネントのステップ数（System Code）が総ステップ数に占める割合を示したものである。例えば、再利用コンポーネントとしては、プロセスデータ管理コンポーネントやプロセスデータの異常を検出し警報を発生させる警報管理コンポーネントがある。製品特化に開発されたコンポーネントとしては、監視 AP とコンポーネントを接続するアプリケーション特化アダプタコンポーネントがある。図 10 の製品系列 A（Product Line A）では、DiaSynapse 本体のステップ数が約 40%、再利用コンポーネントのステップ数が約 45%で、製品適用を繰り返すことで総ステップ数の約 85%が再利用可能となり生産性が向上していると言える。製品系列 B（Product Line B）は、別の製品系列開発の初回適用時におけるデータである。DiaSynapse 本体のステップ数が約 20%、再利用コンポーネントのステップ数が約 20%であり、全体の再利用率は約 40%である。ただし、初回適用であるために、再利用性を考慮した開発が行なわれていない。しかし、開発チームにアンケートしたところでは、開発後の見直しで、DiaSynapse 本体と再利用コンポーネントを合わせた再利用率を約 60%まで高められるとの見通しである。

本節の最後に、DiaSynapse のシステム開発支援環境について述べる。システム開発を支援するツールとして図 11 の、コンポーネントを接続して連携手順を定義する機能を備えた視覚的編集ツールを用意した。システム開発者は、必要なコンポーネントを開発した後は、このツールを用いてプログラムレスでシステム開発を行うことができる。9 件のシステム開発中 4 件でこのツールを使用し、ツールを使用した場合に開発工程の工数を約 30%削減できた。

図 10 製品開発系列への適用結果

Fig. 10. Results of applying to lines of products.

図 11 コンポーネント接続の視覚的編集ツール

Fig. 11. A visual tool for connecting components.

6. 関連研究と考察

6.1 フレームワークとコンポーネント システム開発における生産性向上技術としてオブジェクト指向フレームワークがある⁽¹⁴⁾。大規模監視制御システム開発向けオブジェクト指向フレームワーク GRASS⁽¹⁵⁾は、アプリケーションレベルでの発行購読モデルを定義している。この手法では、APから取り出せる情報をあらかじめ規定するため、入出力の変更に当たっては、設計段階まで立ち戻って新たな機能を作りこむ必要がある。本研究の手法では、カスタムAPの追加等の仕様変更に対して、コンポーネントを段階的にEAIシステム上に配置していくことで対応できるという利点がある。

6.2 遠隔手続き呼び出しと発行購読モデル 分散システムをコンポーネントの組み合わせで構築する技術としては、ポイントツーポイント通信モデル（遠隔手続き呼び出し）に基づく、CORBA⁽¹⁶⁾、RMI⁽¹⁷⁾等が存在する。OPENDREAMS⁽¹³⁾は、CORBAコンポーネントで監視制御システムを構成する。

ポイントツーポイント通信モデルでは、コンポーネント間が密に結合するという問題があり⁽¹⁸⁾、コンポーネントの疎な結合や動的な結合を必要とする場合には、発行購読モデルが採用されている⁽¹⁸⁾⁽¹⁹⁾。JEDI⁽¹⁸⁾はインターネットス

ケールの分散システムの開発を目的としており、Dragon⁽¹⁹⁾は家庭内ネットワーク等の小規模ネットワークで一般利用者が分散システムを構成することを目的としている。本研究では、細かな粒度でコンポーネントを構成するため、コンポーネントの数が増え、コンポーネント間の接続関係が複雑化するという問題を回避するため発行購読モデルを採用した。

発行購読モデルでは、イベントの発行者が増加するとイベントの購読者がイベントを扱いきれなくなる問題が起こる⁽¹⁹⁾。本研究の手法では、コンポーネントは、イベント受信時に変換処理（図3のconversion）でイベントを貯めておき、実行処理（図3のexecution）でまとめて処理することでこの問題に対応できる。実行処理を行うタイミングはコンポーネントの実行条件定義で設定する。

6.3 従来ミドルウェアとの比較 本節では、メッセージブローカ、Webアプリケーションサーバ、ワークフローを、産業系EAIシステムの機能要件の観点から検討し、本研究の有用性を示す。

6.3.1 メッセージブローカ 産業系EAIシステムでは、対象設備を業務の様々な視点で表現するデータ共有空間を構成し、APはそれと密に連携することが望まれる。メッセージブローカはこのようなデータ共有空間を提供するものではないので、AP間でデータを共有し、データの整合性を保つためには、大量のメッセージ交換が発生する。本研究では、DiaSynapseノード上のCMMSコンポーネントの集合で共有データ空間を表現する。すなわち3.4節の図6を用いて説明したように、APのデータをコンポーネントの属性に集約、展開している。

産業系EAIシステムでは、メッセージフローとプロセスフローを連携させることがある。メッセージブローカでは、制御の実体は各APが持ち、システム全体を統括するものは存在しない。プロセスフローの実現には、システム全体を統括するAPを開発することになり、設計、実装にメッセージブローカとはまた別の技法が必要となる。これは生産性の観点から問題である。本研究では、DiaSynapseノードが制御の実体を持ち、ここにコンポーネントを配置することでメッセージフロー、プロセスフローのいずれも実行することができる。

6.3.2 Webアプリケーションサーバ AP間でのデータ共有は、Webアプリケーションサーバ上の分散コンポーネントの集合で表現可能である。分散コンポーネントの属性として、様々な粒度のデータを持たせることができ、非定型データを扱うこともできる。しかし、AP間連携の定義を、ポイントツーポイント通信モデルで実現する必要があり、分散コンポーネント間が密結合し、生産性の観点で問題となる。新しい分散コンポーネントが既存の分散コンポーネントの処理を呼び出して、それを不整合な状態にする可能性があるため、既存のEAIシステムの機能の内部を細部まで把握して分散コンポーネントの追加を行なう必要がある。本研究では、CMMSコンポーネントの追加の

際に、既存の CMMS コンポーネントの属性、イベントを購読する *CLS* 関係、*ALS* 関係を定義するだけであり、既存の EAI システムの機能には影響を与えないことがミドルウェアで保証されている。

6.3.3 ワークフロー まず、WfMC⁽⁸⁾ の定義を整理する。ワークフロー定義とは、ビジネスプロセスが表す業務の手順について、特に計算機により流れの自動化が見込めるものを定義したものである。ワークフローインスタンスとは、実際に実行されているワークフロー定義の事例である。本研究でプロセスフローと表現してきた処理はこれにあたる。ワークフローエンジンとはワークフローインスタンスを実行するソフトウェアである。アクティビティとは、ワークフローインスタンスを構成する個々の作業を表わす。アクティビティを実行するのに必要なものがリソースである。アクティビティの実行者は人あるいは *AP* である。

産業系 EAI システムでは、ワークフローエンジンとは独立して運用されている *AP* がある。それらの *AP* は、EAI システムのデータを随時更新する。生成されたワークフローインスタンスはそれらデータを利用しながら処理を進める。まず、ワークフローエンジンに、*AP* 間で共有されるデータを保持する仕組みが必要である。次に、共有データと *AP* 間の相互作用を定義、実行する枠組みがワークフロー定義以外に必要である。最後に、WfMC の定義ではワークフローエンジンは必要に応じて *AP* を起動するとあるが、産業系 EAI システムでは、自律的に動作する *AP* とワークフローエンジンが協調動作する仕組みが必要である。WfMC が定義する一般的なワークフローでは、産業系 EAI システムに適用するためにこれらの機能の追加開発が必要であり、これらを実現しているところに本研究の優位性がある。

近年、ワークフローは WfMC が定義する範囲を超えて EAI システムの実現に向けた拡張がなされてきているので、以下ではこれらについて考察する。

文献(5)では、ワークフロー定義とは別に、*AP* 間、ワークフローインスタンス間で共有するデータを、ビジネスオブジェクトという仕組みで表現できる。またビジネス系 EAI システムの生産性向上のために、様々なビジネスオブジェクトの雛形、*AP* を実現するためのパッケージ、および *AP* とビジネスオブジェクト間の API (Application Programming Interface) が提供されている。産業系 EAI システム実現のためには、対象設備を様々な業務の視点から表現したデータ構造を、ビジネスオブジェクトとして設計、実装するとともに、既存の *AP* を、API に合わせて改修していく作業が必要となる。

JEDI⁽¹⁸⁾ では、自律的に動作する *AO* で、*AP* とワークフローエンジンを実現している。*AO* 間のイベント交換でデータの共有を表現可能であり、かつそれとは別に *AO* 間の相互作用を定義、実行できることから、産業系 EAI システムの機能要件を満たしたワークフローと言える。しかし、JEDI ではシステム構成の最小単位が *AO* であるのに対し、本研究では *AO* に相当する DiaSynapse ノードを、ユニッ

ト、コンポーネントと詳細化しているところに特徴がある。JEDI の再利用単位が *AO* であるのに対し、本研究ではユニット、コンポーネントが再利用単位となることで、再利用性が向上し生産性の向上が図れたと思われる。

銀行系の営業店システムで実用化されている EAI システムの実現方式は、データベース、メッセージバスをワークフローで統合するハイブリッド方式である。データベースで共有データを一元管理し、*AP* とワークフロー間でメッセージ交換を行う際にはメッセージバスを用い、ワークフローが全体を統括する。本研究ではこれら三つの概念を単独のモデル、ミドルウェアで実現するアプローチを取った。このようなハイブリッド方式の産業系 EAI システムでの有用性の評価は今後の課題である。

7. ま と め

産業系 EAI システムの生産性向上を目的として、そこで求められる機能要件について考察した。機能要件を満たすために、多段購読コンポーネントモデルを提案し、それを内蔵するミドルウェア DiaSynapse を開発した。また、ミドルウェアを広域統合監視システムとネットワーク管理システムの開発に適用してその生産性向上の度合いを評価した。適用の結果、異なる二つの製品系列開発で、再利用されるソフトウェアの割合が 40%(見直し後 60%)と 85%という結果となり、開発段階における生産性向上が果たせたとと言える。

(平成 15 年 9 月 24 日受付、平成 16 年 2 月 3 日再受付)

文 献

- (1) D. Linthicum: "Enterprise Application Integration, Addison Wesley Longman", Inc. (2000) (in Japanese)
吉鋪紀子(監訳):「EAI-エンタープライズアプリケーション統合」, ビアソン・エデュケーション (2000)
- (2) Sun Microsystems Inc.: J2EE 1.4 Specification, <http://java.sun.com/j2ee/1.4/docs/> (2003)
- (3) D.S. Platt: Introducing Microsoft. Net 3rd Edition, Microsoft Pr. (2003)
- (4) 戸田保一・飯島淳一・速水治夫・堀内正博: ワークフロー ビジネスプロセスの変革に向けて, 日科技連 (1998)
- (5) A. Richayzen, J. Dart, C. Brennecke, M. Schneider: Practical Workflow for SAP, Galileo Press (2002) (in Japanese)
(株)コスモユノー(訳)SAP 実践ワークフロー, 日経 BP 社 (2003)
- (6) 電気学会ワークフロー調査専門委員会編: ワークフローの実例, 日科技連 (1999)
- (7) M. Oba, S. Onoda, and N. Komoda: "Evaluating the Quantitative Effects of Workflow Systems Based on Real Cases", *T. IEE Japan*, Vol.120-C, No.7, pp.1043-1049 (2000-7) (in Japanese)
大場みち子・小野田仙一・薦田憲久:「ワークフローシステム導入事例に基づく導入効果の定量的評価」, 電学論誌 C, **120**, 7, pp.1043-1049 (2000-7)
- (8) The Workflow Management Coalition, <http://www.wfmc.org>
- (9) J. Hopkins: Component Primer, Communication of the ACM, Vol.43, No.10 (2000)
- (10) M. Aoyama: Componentware; "Building Applications with Software Components", *IPSJ*, Vol.37, No.2, pp.71-79 (1996-2) (in Japanese)
青山幹雄:「コンポーネントウェア:部品組み立て型ソフトウェア開発技術」, 情報処理, **37**, 2, pp.71-79 (1996-2)
- (11) P. Stevens and R. Pooley: "Software Engineering with Ob-

jects and Components”, Udated Edition, Peason Education Limited (2000) (in Japanese)

児玉公信 (監訳) オブジェクト指向とコンポーネントによるソフトウェア工学 - UML を使って -, ピアソン・エデュケーション (2000)

- (12) I. Ben-Shaul, O. Holder, and B. Lavva: “Dynamic Adaptation and Deployment of Distributed Components in Hadas”, *IEEE Trans. Software Engineering*, Vol.27, No.9, pp.769–787 (2001)
- (13) M. Fayad and R. Johnson: “Domain-Specific Application Frameworks, Frameworks Experience by Industry”, Wiley, pp.231–249 (2000)
- (14) M. Fayad and D. Schmidt: “Object-Oriented Application Frameworks”, *Commun ACM*, Vol.40, No.10 (1997)
- (15) T. Kojima, H. Nakata, M. Tange, and S. Nishida: GRASS: An Object-Oriented Framework for SCADA Systems, *T. IEE Japan*, Vol.119-C, No.10, pp.1274–1282 (1999-10) (in Japanese)
小島泰三・中田秀男・丹下正純・西田正吾:「監視制御システム向けアプリケーションフレームワーク」, 電学論誌 C, 119, 10, pp.1274–1282 (1999-10)
- (16) Object Management Group: CORBA/IIOP 2.2 Specification, <ftp://ftp.omg.org/pub/docs/formal/98-07-01.pdf>, (1998)
- (17) Sun Microsystems Inc.: Java Remote Method Invocation Specification, technical report (1998)
<ftp://ftp.javasoft.com/docs/jdk1.2/serial-spec-JDK1.2.pdf>,
- (18) G. Cugola, E.D. Nitto, and A. Fuggeta: “The JEDI Event-Based Infrastructure and Its Application to the Development of the OPSS WFMS”, *IEEE Trans. Software Engineering*, Vol.27, No.9, pp.827–850 (2001)
- (19) M. Iwai, J. Nakazawa, N. Nishio, and H. Tokuda: “A System for Improved Composition of Component-based Applications”, *IPSIJ*, Vol.43, No.6, pp.1664–1675 (2002-6) (in Japanese)
岩井将行・中澤 仁・西尾信彦・徳田英幸:「分散コンポーネントによる即興的アプリケーション構成機構の実現」, 情報処理学論誌, 43, 6, pp.1664–1675 (2002-6)

石 原 鑑 (非会員) 1967 年 10 月 30 日生。1991 年 3 月大

阪大学基礎工学部情報工学科卒業。1993 年 3 月同大学大学院基礎工学研究科情報工学専攻修士課程修了。同年 4 月三菱電機 (株) 入社, 現在, 先端技術総合研究所にて, 分散システム, コンポーネント技術, イントラネット産業応用システムなどの研究開発に従事。情報処理学会会員。

大 崎 雅 代 (非会員) 1968 年生 11 月 22 日生。1991 年 3

月神戸大学理学部数学科卒業。同年 4 月三菱電機 (株) 入社。現在, 先端技術総合研究所にて, ヒューマンインタフェース, インタネット関連の研究開発に従事。

高 田 秀 志 (正員) 1968 年 6 月 26 日生。1991 年 3 月京都

大学工学部情報工学科卒業。1993 年同大学大学院工学研究科情報工学専攻修士過程修了。同年 4 月三菱電機 (株) 入社。2001 年京都大学大学院情報学研究科社会情報学専攻博士後期課程修了。2003 年 12 月に三菱電機 (株) 退職し, 2004 年 1 月より, 京都大学情報学研究科 COE 研究員。産業用データベース, 履歴データベース, 教育システムなどの研究に従事。博士 (情報学) 情報処理学会, システム制御情報学会, ACM 各会員。

小 島 泰 三 (非会員) 1960 年 2 月 9 日生。1982 年 3 月東

京大学工学部船用機械工学科卒業。同年 4 月三菱電機 (株) 入社。同社中央研究所にて精密機械の計測と制御, 設計支援システム, オブジェクト指向言語などの研究に従事。現在, 同社先端技術総合研究所にて, 監視制御システム, 組み込みシステム等の SW 構築技術の研究開発に従事。工学博士。情報処理学会, IEEE 各会員。

杉 本 明 (非会員) 1953 年 5 月 14 日生。1977 年 3 月京

都大学理学部数学科卒業。1979 年 3 月同大学大学院工学研究科数理工学専攻修士課程修了。同年 4 月三菱電機 (株) 入社。現在, 先端技術総合研究所にて, 設計支援システム, オブジェクト指向言語, 分散システムなどの研究に従事。工学博士。情報処理学会, 電子情報通信学会, ACM 各会員。