車載ソフトウェアのための
イベント駆動サービス指向アーキテクチャの提案と評価

M2008MM021　宮脇　聡

指導教員:青山　幹雄

1. はじめに

近年，車載ソフトウェアの再利用や拡張，ネットワークを介した連携を柔軟に行うためにサービス指向アーキテクチャ(SOA: Service-Oriented Architecture)の適用が研究されている[1]. しかし，従来の SOA では，同時，並行して発生するイベントにリアクティブに処理することに対応していない．本稿では，SOA を車載ソフトウェアへ適用するためにイベント処理を可能にするイベント駆動 SOA を提案する.

2. 車載ソフトウェアへの SOA 適用の課題

イベント処理を可能にした SOA をイベント駆動 SOA と定義する．本研究では，パブリッシュ/サブスクライブ型の処理を行う分散イベントシステム[2]を SOA に統合することによりイベント処理を可能にする．車載ソフトウェアにイベント駆動 SOA を適用するためには，以下の 2 つの課題がある.

(1)　直接サービス起動と間接サービス起動の混在

車載ソフトウェアでは，スイッチ操作などユーザからの直接サービス起動と車の速度変化などのイベントの発生による間接サービス起動の 2 つの異なる相互作用を統一的に制御しなければならない.

(2)　車載システムの多様なイベントへ対応

車載システムでは，多くのセンサからの複数のイベントが存在する．これらのイベントは，それぞれ異なるタイミングで発生する．このようなイベントを処理するためには，システムがイベントの発生元(イベントソース)を知るとともに，どのようなタイミングで発生するかを知る必要がある.

3. 関連研究

本稿に関連する研究として，イベント通知に関する研究と型と属性によるイベントの利用の研究を示す.

(1)　イベント通知に関連する研究

分散サービス起動[6]と WS-Notification[3]では，イベントを通知する方法が提案されている．分散サービス起動では，複数のイベント通知エンジンを用いてイベントを配信し，WS-Notification では，Web サービスにおけるイベント通知に関する標準を定めている.

(2)　型と属性によるイベントの利用

複数の分散イベントシステムで共通のイベントを利用し，イベントに型と属性を付加することで異なるイベントベースシステム間の連携を柔軟に行う方法が提案されている[4].

4. 問題解決へのアプローチ

イベント駆動 SOA を車載ソフトウェアに適用するためにサービスブローカの拡張を行い，多様なイベントに対応するためにイベントをモデル化する.

(1)　サービスブローカの拡張

従来の SOA が用いる要求/応答型によるユーザからの処理と，分散イベントシステムで用いられるイベントを処理するパブリッシュ/サブスクライブ型の処理を統一的に制御する必要がある．また，車載システムでは，オプション等によるセンサやサービスの追加，変更に柔軟に対応し，サービスの状態変化に伴う処理が必要である．これらの処理に対応するためにサービスブローカを拡張する.

(2)　イベントのモデル化

車載システムで発生する多様なイベントを処理するために車載システムでどのようなイベントが存在するかを把握する必要がある．そのためにイベントについての情報をイベントモデルとして定義する．このイベントモデルを用いることで，サービスブローカは，イベントを処理するための情報を知ることができる.

5. イベント駆動 SOA の提案

提案イベント駆動 SOA は，イベントの情報を定義したイベントモデルとイベントを処理するサービスブローカから構成される．イベントモデルは，イベント登録とサブスクリプションで利用する.

5.1. アーキテクチャの構成要素

図 1 にアーキテクチャの構成要素を示し，処理の流れを説明する．センサは，検出情報をイベントとしてサービスブローカへ送信する．サービスブローカは，イベントをフィルタリングし，イベント通知をサービスへ配信する．サービスは，受信したイベントに基づきアクチュエータを制御する.

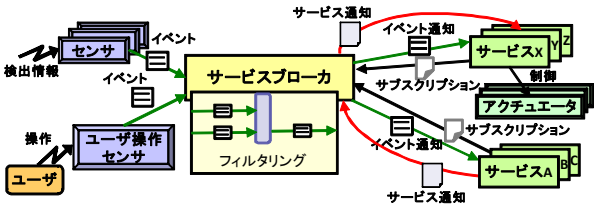

図 1:アーキテクチャの構成要素

5.2. イベントモデル

イベントモデルは，次の 2 つのモデルから成る(図 2).

(1)　イベント構造モデル

ベントを構成するイベントソースやイベントの付加情報などの構造の情報を示す.

(2)　イベント発生モデル

周期，非周期など，イベントの発生するタイミングを示す.

イベントモデルに基づいてイベントの登録やサブスクリプションを行う.

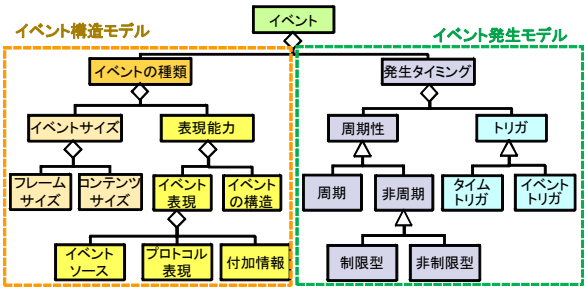

図 2:イベントモデル

5.3. サービスブローカ

通知の情報管理, イベント処理, サービスへの通知処理を行うためにサービスブローカを拡張する. これらの処理を PublisherRegistration マネージャ, Subscription マネージャ, Service マネージャの 3 つのコンポーネントで行う.

5.3.1. サービスブローカの構造

図 3 でサービス, センサ, サービスブローカの 3 つのコンポーネント間の関係を示す.

(1) PublisherRegistration マネージャ

PublisherRegistration マネージャは, センサが送信するイベントとサービスが送信するサービス通知の情報を管理する. センサからのイベントでは, イベントモデルに基づいた情報が登録される. サービスからのサービス通知では, サービス名, 送信されるデータの種類, データ内容, サービスのエンドポイントが登録される. これらの情報は, Subscription マネージャや Service マネージャからそれらが受信したサブスクリプションが利用可能か照合する時に利用される. また, センサやサービスが追加, 変更, 削除がされた場合, この登録情報を変更する.

(2) Subscription マネージャ

サービスからのイベントに関するサブスクリプションの管理とセンサからのイベントを処理する. イベント処理では, センサからのイベントを受信し, 登録されたサブスクリプションに基づきフィルタリングを行い, Service マネージャへイベント通知を配信する. このイベント通知には, 受信したイベントの情報に送信先のサービスのアドレスが付加されている. サブスクリプション受信時に, PublisherRegistration マネージャに対してそのサブスクリプションを扱うことが可能か問い合わせる. また, Service マネージャからサブスクリプション停止や再開要求を受信し, 利用されるサブスクリプションを変更する.

(3) Service マネージャ

Service マネージャは, Subscription マネージャからのイベント通知やサービスからのサービス通知を受信し, サービスを起動する. サービス通知に関するサブスクリプションを管理する. サービスからの通知を受信し, 登録されたサブスクリプションに基づきフィルタリングを行い, そのサブスクリプションと一致した場合にその通知をサービスへ配信する. サブスクリプション受信時には, PublisherRegistration マネージャに対してそのサブスクリプションを扱うことが可能か問い合わせる. また, サービスの状態変化により利用するサブスクリプションが変化するとき, サービスからサブスクリプションの停止, 再開要求を受信し, そのサブスクリプションを管理している Subscription マネージャか Service マネージャにサブスクリプション停止, 再開を通知する.

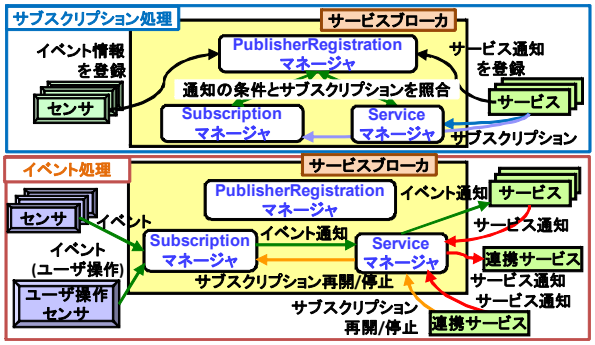

図 3:サービスブローカの構造

5.3.2. サービスブローカの振舞い

サービスブローカの PublisherRegistration マネージャ, Subscription マネージャ, Service マネージャを用いて, サブスクリプション処理, ユーザ操作処理/イベント処理, サービス連携, サービスの状態変化の振舞いを示す.

(1) サブスクリプション処理

通知の登録からサブスクリプションまでを図 4 に示す. センサとサービスが PublisherRegistration マネージャに通知の情報を登録する. サービスは, イベント情報を取得し, サブスクリプションする. Subscription マネージャ, Service マネージャは, サブスクリプションを受信し, そのサブスクリプションの条件の通知を扱うか PublisherRegistration マネージャに問い合わせ, その結果を得る. 各サービスは, サービスが停止状態のときに不必要なサブスクリプションを停止する.

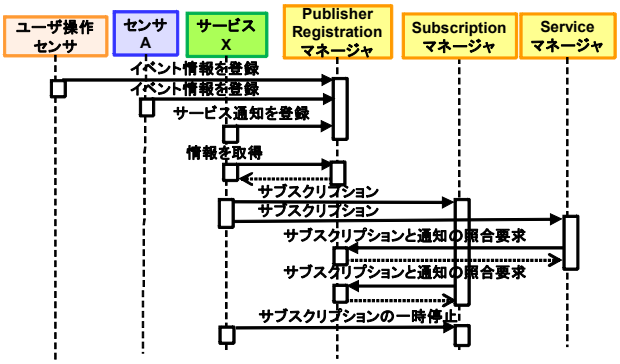

図 4:サブスクリプション処理の流れ

(2) ユーザ操作処理/イベント処理

ユーザ操作処理とイベント処理の相互作用を示す(図 5(a)). これらの処理は, イベントを Subscription マネージャにイベントを送信し, 登録されたサブスクリプションに基づきフィルタリングを行い, 一致した場合 Service マネージャへイベント通知を送り, サービスへイベントを配信する. ただし, ユーザ操作は, 発生毎にすべてサービスへ通知される.

(3) サービスの状態変化

イベント通知などによりサービスが起動し, その状態が変化した場合, サービスは, サブスクリプション停止/再開要求を Service マネージャに送信する. Service マネージャは, そ

の要求のサブスクリプションを管理するSubscriptionマネージャに配信し，サブスクリプションを停止/再開する(図 5(b)). Serviceマネージャがサブスクリプションを管理している場合，Service マネージャがその処理を行う.

(4) サービス連携

サービスからの通知により他サービスと連携する場合，Service マネージャにサービスが通知を行い，フィルタリングの結果，対象サービスに通知を配信することでサービス連携を行う(図 5(c)).

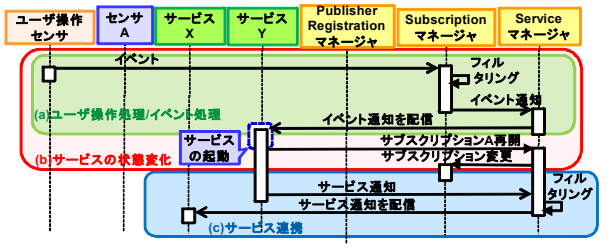

図 5:イベント発生に基づく振舞い

5.3.3. サービスブローカのイベント処理モデル

センサから発生する複数のイベントを優先度に基づきサービスブローカで処理することやイベントの処理中に送信されたイベントを一時的に保持する必要がある. これらの処理の優先度を判断するイベントディスパッチャと優先度キューを用いたイベント処理モデルで実現する(図 6). このモデルでは，センサからのイベントをイベントドライバが受信し，イベントディスパッチャへ送信する. イベントディスパッチャは，イベントの優先度を保持するイベントテーブルを用いて優先度を検索し，その優先度キューにイベントを挿入する. Subscription マネージャは，優先度の高いキューからイベント取得し，処理する. このモデルによりイベント受信とイベント処理を分離して行うことを可能にした.

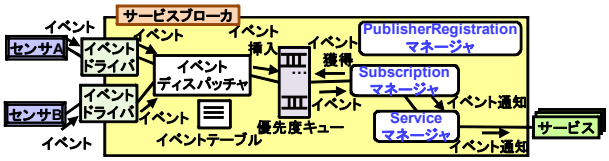

図 6:イベント処理モデル

6. プロトタイプによるイベント処理確認

図 6 に示したサービスブローカのイベント処理モデルのプロトタイプを Apache Axis2 を用いて実装した.

(1) 実行環境

プロトタイプの実行環境を表 1 に示す. サービスブローカ，サービスは，要求受信後に処理を行うために Web サーバとして Apache Tomcat を用いて実装した.

表 1:プロトタイプの実行環境

構成要素 / 実行環境	センサ	サービスブローカ，サービス
Java 実行環境	JRE1.6.0_17	JRE1.6.0_17
OS	Windows Vista Business	Windows XP Professional
SOAP エンジン	Apache Axis2 1.4.1	Apache Axis2 1.4.1
Web サーバ		Apache Tomcat 6.0.20

(2) プロトタイプの構成

図 7 にイベント生成からサービスへ通知されるまでのプロトタイプの構成を示す. センサ，サービスブローカ，サービス間は，Axis2 の SOAP エンジンを用いて通信する. センサは 3 つあり，それぞれ異なるタイミングでサービスブローカへイベントを送信する. また，Subscription マネージャは，あらかじめ起動されており，定期的に優先度キューからイベントを取得し，処理を行う.

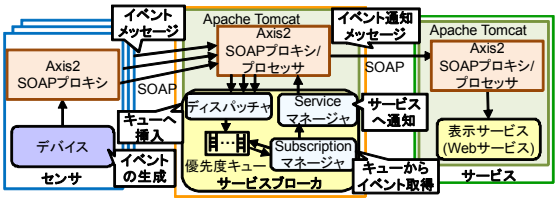

図 7:プロトタイプの構成

(3) イベント処理の確認

ユーザ操作やセンサからの異なるタイミングでイベントを発生させ，サービスへ通知した. キューへのイベントの挿入とキューからイベントを取得する処理を分離することで異なるタイミングで発生したイベントを優先度に基づいて処理することが可能であることを確認した.

7. 提案アーキテクチャの評価

イベント駆動 SOA を車載システムの CC(Cruise Control)，ACC (Adaptive Cruise Control)[5]に適用し，有効性を評価した.

7.1. 適用するシステム

適用するシステムの機能を説明する. CC の機能は，設定された速度で定速走行することである. ACC では，CC の機能に加え先行車に追従する機能を持つ.

ACC は，CC を拡張した機能を持つ. そのため，本稿で提案するアーキテクチャにおいてシステムの機能拡張における対応を確認するために CC，ACC を例題として選択した.

7.2. 提案アーキテクチャの例題への適用

図 8 に例題の分析プロセスを示す. ユースケース分析では，センサ，サービスを抽出する. イベントモデルを用いてセンサが生成するイベントのサブスクリプションに必要な情報を抽出する. そして，イベントとサービスの対応付けを行い，サブスクリプションを決定する. 以上の分析を基にセンサ，サービスをサービスブローカへ適用し，構造と振舞いの分析を行い，サービスブローカの構造とシーケンス図を作成する.

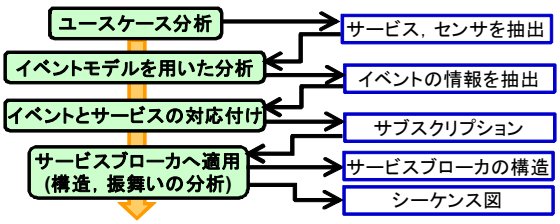

図 8:例題の分析プロセス

この分析によりサービスは，速度制御サービス，ブレーキ制御サービス，表示サービス，車間制御サービスを抽出した.

イベントモデルによる分析と仕様[5]から，サブスクリプションを用いてセンサとサービスの関係を決定した．図 9 では，それらの分析に基づいて作成した CC と ACC 適用におけるサービスブローカの構造を示す．

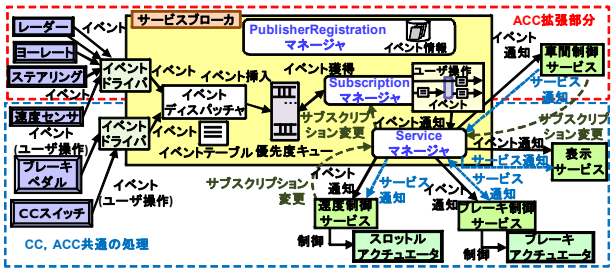

図 9:CC，ACC 適用におけるサービスブローカの構造

7.3. 評価

センサの追加や変更への対応，サービスの再利用性，イベントの統一制御，多様なイベントへの対応，サービスの状態変化への対応について評価し，提案するアーキテクチャの有効性を示す．

(1) センサの追加や変更への対応

CC から ACC へ拡張した場合に新たにレーダーセンサなどのセンサが 3 つ追加される．これにより新たなイベントが追加される．追加されたイベントは，PublisherRegistration マネージャにイベント情報が登録される．この登録された情報に基づき追加されたイベントのサブスクリプションを行う．また，イベントテーブルのイベントの優先度を変更する．これによりセンサの追加や変更によるイベントの追加や変更に対応可能である．それに伴いサービスが利用するデータを変更する必要がない．

(2) サービスの再利用性

CC から ACC へ拡張した場合，追加されたセンサとサービスは，サービスブローカにおいてイベント登録とサブスクリプションのみ変更する．CC のサービスを変更することなく ACC で再利用可能である．

(3) イベントの統一制御

サービスに対して直接作用するユーザ操作イベントと間接的に作用するセンサからのイベントがある．ユーザ操作イベントは発生毎にサービスへ通知され，センサのイベントはフィルタリングが行われる．これらの異なる処理を Subscription マネージャで行うことで 2 つの処理を統一的に制御が可能となる．

(4) 多様なイベントへの対応

イベントモデルを用いることで，イベントソースとその付加情報により 1 つのセンサが複数の異なるイベントを発生した場合への対応を可能にする．例えば，レーダーセンサの場合，先行車検知，先行車との距離，相対速度のイベントを分類可能にする．またイベント処理モデルを用いることで，イベントディスパッチャがイベントを優先度キューに挿入し，Subscription マネージャがイベントを優先度キューから取得し処理をすることで，異なるタイミングで発生するイベントを統一的に処理可能となる．

(5) サービスの状態変化への対応

CC，ACC 共にサービスが起動前に使用されないイベントのサブスクリプションを停止し，サービスが起動した場合にそのサブスクリプションを再開し有効にする．サービスの起動，停止などの状態によりサブスクリプションの停止，再開を行うことでその状態で必要なイベントのみを受信可能にする．

8. 今後の課題

(1) イベントの処理タイミング

提案したサービスブローカでは，キューからイベントを取得するタイミングによりサービスへイベントが配信されるまでの時間が決定する．このため，イベントを取り出すタイミングを考慮する必要がある．

(2) イベントモデルによるフィルタリング

イベントモデルを用いてフィルタリングを行うためには，イベントの付加情報を詳細に示し，イベントの付加情報を型などにより分類し，他の要素との関連を示す必要がある．

(3) イベントの一貫性の保証

サービスブローカがイベントを複数のサービスに送信するとき，送信の失敗などによりイベントに一貫性が失われる．送信に失敗した場合，再送するなどの処理が必要である．

9. まとめ

本稿では，分散イベントシステムを SOA に統合したイベント駆動 SOA を提案した．車載ソフトウェアへの適用を実現するために，イベント処理とサービスへの通知処理を行うサービスブローカを提案した．車載システムの多様なイベントへ対応するためイベントモデルを提案した．プロトタイプを実装し，イベント処理を確認した．さらに提案アーキテクチャを例題へ適用し，有効性を評価した．

謝辞

本研究のご支援を頂いた，(株)デンソーの岩井明史氏と佐藤洋介氏に感謝する．

参考文献

[1] 青山 幹雄，ほか，車載ソフトウェアのサービスプラットフォームのモデルとアーキテクチャ，自動車技術会 2008 年秋季大会, Oct. 2008, No. 97-08, pp. 21-26.

[2] G. Mühl, et al., Distributed Event-Based Systems, Springer, 2006.

[3] OASIS, Web Service Notification (WSN), ver. 1.3, 2006, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsn.

[4] S. Rozsnyai, et al., Concepts and Models for Typing Events for Event-Based Systems, Proc. DEBS '07, Jun. 2007, pp. 62-70.

[5] トヨタ自動車，CROWN MAJESTA 新型車解説書，2004.

[6] K. Walzer, et al., A Concept for Flexible Event-Driven Invocation of Distributed Service Compositions, Proc. ICDCSW '07, Jun. 2007, pp. 22-29.