車載データ管理プラットフォームの運転支援アプリケーションへの適用

勝沼　聡†*　熊谷　康太†　山口　晃広†　本田　晋也†　佐藤　健哉††
高田　広章†

† 名古屋大学大学院情報科学研究科　〒 464–8601 名古屋市千種区不老町
†† 同志社大学モビリティ研究センター　〒 610–0321 京都府京田辺市多々羅都谷 1–3
∗ 現在，日立製作所 中央研究所
E-mail: †{katsunuma,yamagut}@nces.is.nagoya-u.ac.jp, ††{kumagai,honda,hiro}@ertl.jp,
†††ksato@mail.doshisha.ac.jp

あらまし　車載システムではセンサなどのデータの増加に伴うアプリケーションの開発コスト増加が課題となっている．そのため我々は組込み機器向けのデータストリーム管理システム（eDSMS）を開発し，eDSMS を用いて車載データを管理するプラットフォーム（車載データ管理 PF）を実現する．本研究では車載データ管理 PF においてアプリケーションがデータを取得するためのデータアクセスライブラリ（以下，ライブラリ）の設計方式を提案する．提案方式では特定の種類のセンサやアプリケーションに依存する処理を分けてライブラリを定義することにより，ライブラリの構成変更を容易化する．そして 3 種類の運転支援アプリケーションを適用事例としライブラリを開発することで，提案方式の有効性を示す．
キーワード　車載データ管理，データストリーム管理，組込みシステム，車載システム

Application of the Automotive Data Management Platform to Driving Support Applications

Satoshi KATSUNUMA†*, Kouta KUMAGAI†, Akihiro YAMAGUCHI†, Shinya HONDA†,
Kenya SATO††, and Hiroaki TAKADA†

† Graduate School of Information Science, Nagoya University Furo-cho, Chikusa-ku, Nagoya, 464–8601 Japan
†† Mobility Research Center, Doshisha University, 1–3 Tatara Miyakodani, Kyotanabe City, 610–0394 Japan
∗ Presently with Central Research Lab., Hitachi Ltd.
E-mail: †{katsunuma,yamagut}@nces.is.nagoya-u.ac.jp, ††{kumagai,honda,hiro}@ertl.jp,
†††ksato@mail.doshisha.ac.jp

Abstract　In the automotive systems, the amount of vehicle data has increased and then application development costs have increased. To approach the problems, we have developed the data stream management system for the embedded devices (eDSMS), and realize the automotive data management platform based on the eDSMS. In this paper, we propose the design method of the data access libraries, with which the applications obtain the automotive data in this platform. In this proposed technique, these libraries are defined by separating sensor-dependent and application-dependent processings. This helps the changes of these libraries for the modifications of the sensors and applications. We develop these libraries applying to more than one driving support applications and show the effectivity of this proposed technique.
Key words　automotive data management, data stream management, embedded system, automotive system

1. はじめに

近年，プリクラッシュセーフティ技術など，車両の状態や周辺状況を判断し，ドライバへの警告や自動制御により運転の支援を行うシステム（以下，運転支援システム）が登場している [1]．例えば車両に搭載された複数のセンサからの情報に基づ

き，操舵回避の支援を行い，衝突が避けられない状況では介入ブレーキを作動させることで衝突衝撃を緩和し被害を軽減するシステムがある [2] [3]．また車両追従，レーン逸脱警告，自動駐車などもある [4]．このような運転支援システムにおいて，周辺の物体を検知するためにミリ波レーダやレーザレーダ，カメラを始め車輪速センサ，加速度センサ，位置検出センサなど多様なセンサを複数搭載し，車々間通信や路車間通信の利用も加わって，相互に情報交換を行う必要がある．

車載組込みシステムでは，電子制御ユニット（Electronic Control Unit，以下，ECU）が多数搭載されており，ECU 毎に異なるサプライヤがアプリケーションを開発するため，仕様が公開されず，データフォーマットも統一されていない．したがって各 ECU のアプリケーションで同一センサデータに対する前処理などを行っており，運転支援システムでは，センサやアプリケーションの増加に伴うシステム開発コストの増大が課題となっている．車載アプリケーションを共通化するアプローチとしては AUTOSAR [5] が挙げられるが，AUTOSAR はソフトウェアコンポーネントのインタフェースの共通化を目指しており，データの管理は対象としない．そこで我々はこの問題を解決するため，様々な ECU 上のアプリケーションから共通で利用可能なデータを車載データとしてアプリケーションから切り離し，プラットホームで統合的に管理する車載データ管理プラットフォーム [6] [7] [8]（以下，車載データ管理 PF）を開発している．

車載データ管理 PF では，アプリケーションに配信するデータをデータアクセスライブラリ（以下，ライブラリ）により定義する．ライブラリは，データストリーム管理システム（Data Stream Management System，DSMS）[9] [11] [10] のデータ処理の記述方法である SPD（Stream Processing Description）により記述する．SPD では，データをストリームとして扱い，そのストリームに対する処理を，複数のオペレータに対してデータフローで接続した記述を取る．ライブラリの設計方法としては，先行研究 [12] でデータの抽象度に応じて階層的に定義する方法が提案されている．しかし先行研究ではセンサやアプリケーションの追加に対して柔軟性がなく，多くのライブラリの再構成が必要となるため開発コストが小さくならない．

そこで本研究では，センサやアプリケーション追加時のライブラリの再利用性を高めライブラリの変更を少なくすることを目的とした，ライブラリの設計方式 SASLD(Sensor/Application Separation Library Design) を実現する．SASLD では車間距離センサなど特定の種類のセンサに依存する処理を分けてライブラリを定義することで，センサ追加時のライブラリの変更箇所を少なくする．また衝突防止など特定の種類のアプリケーションに依存する処理を分けてライブラリを定義することで，アプリケーション追加時に多くのライブラリを再利用可能とする．

本稿の構成は以下の通りである．まず 2 節で車載データ管理 PF について述べる．そして 3 節で本稿で提案するライブラリの設計方式を述べ，4 節でその評価について述べる．そして最後に 5 節でまとめを述べる．

2. 車載データ管理プラットフォーム

本節では車載データ管理 PF のコンセプトと構成及び，ライブラリ開発について述べる．

2.1 コンセプト

車載データ管理 PF では，DSMS のクエリ記述方法である SPD により車載データ処理を定義する．そして従来の一般的な DSMS とは異なり，プラットフォーム開発者がライブラリとしてクエリの構成要素を定義し，アプリケーション開発者はアプリケーション固有の処理のみを定義することでクエリを生成する．これによりプラットフォーム開発者とアプリケーション開発者が協調した開発を実現する．また従来のサーバ向けの DSMS では実行時にクエリを解釈することにより処理を実行するが，車載データ管理 PF では実行前にクエリからバイナリコードを生成する．これにより必要最小限の機能のみ実行環境に組み込まれるため，車載システムの小容量のメモリに搭載可能とし，また車載システムの様々なハードウェアやソフトウェアプラットフォームへの適用が容易になる．

2.2 構　　成

車載データ管理 PF の構成を図 1 に示す．車載データ管理 PF は，ライブラリ（図 1（2））及び，SPD のクエリ（図 1（4））を構築するコンフィグレータ（図 1（3）），クエリからバイナリコード（図 1（6））を生成する eDSMS（embedded Data Stream Management System）（図 1（5））に分かれる．以下でそれぞれについて説明する．

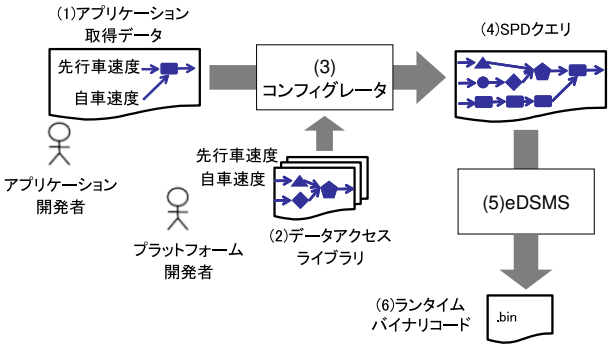

図 1　車載データ管理プラットフォーム

2.2.1 データアクセスライブラリ

ライブラリは一般的な DSMS のクエリと同様に，ストリームを入出力としそのストリームに対する処理を複数のオペレータをデータフローで接続した記述を取る．図 2 にライブラリの定義例を示す．ライブラリはプラットフォーム開発者が定義し，ライブラリの入出力となるストリームと各ストリームのスキーマ，そしてオペレータを定義する．またライブラリの入力ストリームはセンサなど外部や他ライブラリの出力ストリームから取得し，出力ストリームは他ライブラリやアプリケーションに配信する．

例えば図 2 に示すライブラリではレーダを入力ストリーム，走行車情報を出力ストリームとする．そしてレーダストリームのスキーマはカラムとしてレーダ値，時刻を持ち，走行車情報

ストリームのスキーマはカラムとして走行車有無，車間距離，時刻を持つ．さらにライブラリではオペレータ Filter，Map，Join を定義し，そのオペレータによりレーダストリームから走行車情報ストリームを算出する．各オペレータは，Borealis [13] のオペレータと同様の形式を取る．

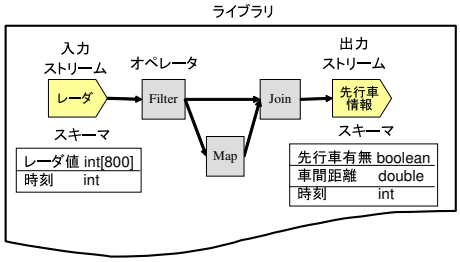

図 2　データアクセスライブラリの定義例

2.2.2　コンフィグレータ

図 3 はコンフィグレータの動作例を示す．コンフィグレータは，アプリケーションの取得データと，システムで利用可能な情報源や QoS 等に従がってライブラリを選択する．そして選択されたライブラリとアプリケーションで定義したオペレータから SPD のクエリを構築する．

図 3 に示す動作例ではアプリケーションは「ACC（アダプティブクルーズコントロール）向け先行車有無」を取得する．この場合，定義済みのライブラリ C の出力ストリームが「ACC 向け先行車有無」であるためライブラリ C を選択する．またライブラリ C の入力ストリーム「先行車有無」を出力ストリームとするライブラリ B 及び，そのライブラリ B の入力ストリーム「先行車情報」を出力ストリームとするライブラリ A を選択する．そして先行車情報ストリームによりライブラリ A，B を接続し，また先行車有無ストリームによりライブラリ B，C を接続することにより，ACC 向け先行車有無の情報を配信する SPD のクエリを構築する．コンフィグレータの詳細については先行研究の文献 [8] を参照されたい．

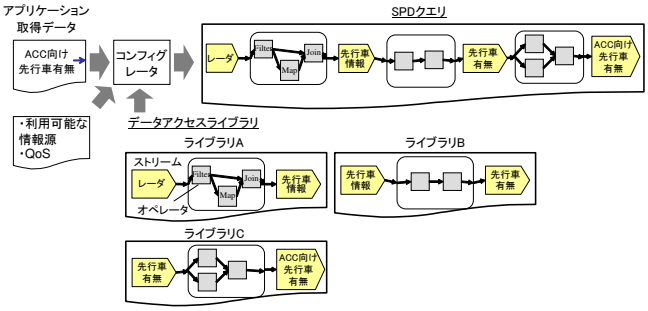

図 3　コンフィグレータの動作例

2.2.3　eDSMS

eDSMS の構成を図 4 に示す．eDSMS では汎用システム向けの DSMS とは異なり静的に決定したクエリ（図 4(a1)）を，開発環境で動作するランタイムジェネレータ（図 4(b)）に登録する．そしてランタイムジェネレータにより以下に示す構成 A〜C を持つランタイム（図 4(c)）を実現する．

- オペレータの動的接続の除去（構成 A）

登録されるクエリに対し接続切替えが必要でない箇所に対して，オペレータ間のストリームキューや，スケジューラによる接続切替えを除去し，各オペレータが次に実行するオペレータを直接，呼び出す．これにより RAM 使用量及び処理レイテンシを削減する．

- モジュール選択（構成 B）

クエリに従って必要なオペレータのみを含むランタイムを生成することにより，ROM の使用量を削減する．またストリームキューやスケジューラなどのモジュールをランタイムジェネレータに保持し，ランタイムの構成が指定されたコンフィギュレーションに従って，ランタイムに搭載するモジュールを選択する．これにより様々なハードウェア/OS に適用する可能とする．

- スレッド数削減（構成 C）

DSMS とデータを送受信するアプリケーションを静的に決定し，DSMS のコードとリンクする．そしてランタイムにおいて，クエリにより定義したデータ処理とアプリケーションを同一スレッドで動作することによりスレッド数を削減する．

- eDSMS の全体構成

構成 A〜C を含む eDSMS の全体構成について述べる．eDSMS では，ターゲットとする車載組込みシステム上で実行する全てのクエリ（図 4(a1)）をランタイムジェネレータに登録する．そして登録されたクエリに対しオペレータ間の動的な接続が不要な箇所を検出し（図 4(b1)），その検出結果に従ってストリームを介さずにオペレータを接続する最適化クエリ（図 4(b2)）を生成する（構成 A の実現）．次に最適化クエリと，メモリ領域の確保方法などがユーザにより指定されたコンフィギュレーション（図 4(a2)）に従って必要なモジュールを選択する（図 4(b3)）．そして選択したモジュールのみをプリコンパイルしランタイムのソースコード（図 4(b4)）を生成する（構成 B の実現）．最後にランタイムのソースコードをコンパイルしアプリケーション（図 4(a3)）とリンクすることにより，アプリケーションとランタイムのモジュールが関数呼出しにより接続するランタイムを生成する（構成 C の実現）．詳細については先行研究の文献 [7] を参照されたい．

2.3　データアクセスライブラリの開発

本節では車載データ管理 PF におけるライブラリの開発の要件と従来方式の問題点を述べる．

2.3.1　ライブラリ開発の要件

車載データ管理 PF のライブラリ開発の要件として以下を挙げる．

- 要件 1. センサ追加時のライブラリ開発コスト削減

車載システムでは多種多様なセンサがあり，また車種毎に搭載されるセンサは異なる．そこでセンサやカメラなど新たなセンサを追加する度に多くのライブラリの再定義が必要になる場合には，ライブラリの開発コストが大きくなりシステム全体の開発コストが削減されない．したがってセンサ追加時のライブラリの変更を少なくする必要がある．

- 要件 2. アプリケーション追加時のライブラリ再利用性

表 1　運転支援アプリケーション

ID	アプリ	説明	必要データ
1	ACC	アダプティブクルーズコントロール（ACC）は高速道路などで運転によるドライバの負担を減らすために，ブレーキやアクセルを制御し先行車と適切な車間距離を保つアプリケーションである．ACC の機能の実現には，車両前方に設けられたレーダにより取得した車車間の距離や先行車の検知情報と，車輪速センサやステアリングセンサにより取得した自車の速度や走行状況の情報が必要となる．一時的にレーダが対象車を見失っても，一定時間は対象車が存在するものとして過去の距離や速度情報から車両状況を推測し，制御を行うという特徴がある．	補完車間距離，先行車相対速度，先行車速度，ACC 対象車認識判定
2	DCA	ディスタンスコントロールアシスト（DCA）は，安全運転を支援するために，先行車へ接近した時に緩やかなブレーキをかけたり，ドライバへの警告を行ったりするアプリケーションである．DCA の機能の実現にはレーダから得られる対象車の速度や車間距離の情報と，自車速センサ，ステアリングセンサから得られる自車の速度や走行状況の情報を必要とする．ACC 同様に一時的に対象車を見失っても一定時間は車両状況を推測し，制御を行うという特徴がある．	補完車間距離，先行車相対速度，先行車速度，DCA 対象車認識判定
3	PCB	プリクラッシュブレーキ（PCB）は一般的に，危険回避や，衝突被害を軽減するアプリケーションである．前方の対象車と追突の恐れがある場合にる警報にてドライバに注意を促し，追突が避けられないと判断した場合には強いブレーキ制御をかけ，衝突速度の低減をはかるシステムである．PCB の機能の実現にはレーダによる対象車との車間距離や先行車の検知情報が必要となる．ACC，DCA と異なり，誤動作によってドライバへ不快感を与えるのを避けるために，レーダやカメラのセンサが対象車を見失った場合は，すぐに危険が回避されたと判断を行うという特徴がある．	車間距離，PCB 対象車認識判定

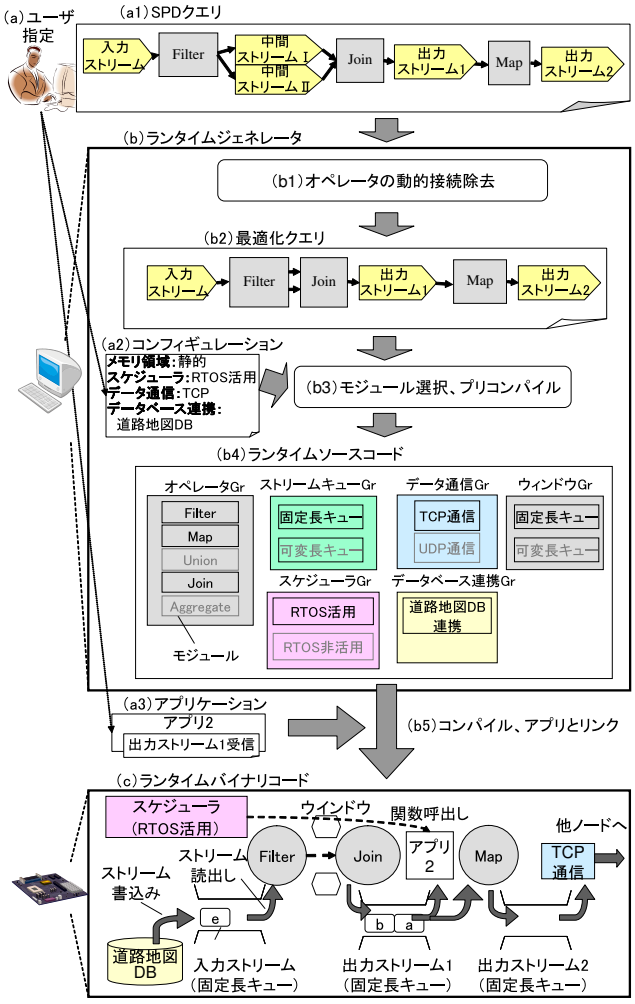

図 4　eDSMS の構成

向上

車載システムでは衝突検知や追随走行など様々なアプリケーションが搭載されており，また車種によりアプリケーションも異なる．そこで今までの車種には搭載されていない新たな機能を持つアプリケーションの追加時に，既存のアプリケーションが活用したライブラリを再利用できない場合，アプリケーションの開発コストを削減できない．そこでアプリケーション追加時に多くの既存のライブラリを活用可能にする必要がある．

2.3.2　既存方式の問題点

既存のライブラリの設計方法としては階層的データ管理方式 [12] が挙げられる．階層的データ管理方式では図 5(a) に示すようにデータの抽象度に応じて階層的にライブラリを定義する．すなわちセンサから取得した生データを第一層に定義し，そのセンサ取得値を組み合わせて自車情報または他車情報として統合したデータを第二層に定義する．そして第三層で自車情報と他車情報を組み合わせた情報を生成する．階層的データ管理方式はこのように階層的にライブラリを定義することにより，新たにライブラリを定義する際に既存のライブラリの計算結果を活用することができる．例えば図 5(a) では第一層で各センサの生データを定義する．そして第二層で生データである車速と相対的進行速度から算出する自車情報（図 5(a5)）を定義し，また前方車距離と前方車画像から算出する前方車の相対座標や検出画像（図 5(a6)）を定義する．そして第二層の自車と前方車の情報から算出する前方情報を第三層で定義する．

階層的データ管理方式ではセンサ追加時に，センサの生データを格納する第一層のストリームの再定義が必要になり，それにより第二層以降のライブラリの変更も必要になるためライブラリの変更箇所が大きい．例えば図 5(a6) では車間距離センサと車載カメラから前方車を検出しているが，同じ車間距離センサの場合にも，データの形式は様々である．したがって車間距離センサの種類が異なる場合には新たな車間距離センサと車載カメラから前方車を検出し相対座標等を算出する処理を新たな第二層のライブラリとして定義する必要がある．したがってセンサ追加時のライブラリの開発コストは大きく要件 1 を満たさない．

また階層的データ管理方式ではアプリケーション追加時に第二層のライブラリを再利用出来ない場合がある．その場合，第二層のライブラリの計算結果を活用する第三層のライブラリも再利用できずライブラリの再利用性が低い．例えば図 5(a6) では第二層で前方車との距離と前方車画像から前方車を検出する．そして前方車を検出した場合にカメラ画像表示アプリケーショ

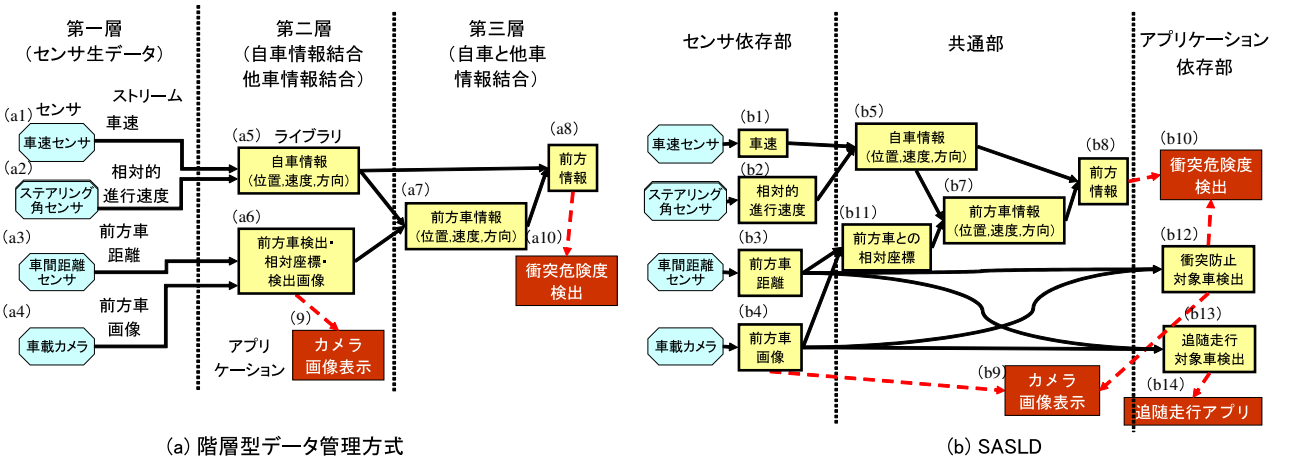

図 5　階層的データ管理方式と SASLD のライブラリ設計

ン（図 5(a9)）にその前方車の検出画像（図 5(a6)）を配信し，衝突危険度検出アプリケーション（図 5(a10)）に前方情報（図 5(a8)）を配信する．しかし各アプリケーションが対象とする前方車はその前方車との距離などの条件が異なるため，新たなアプリケーション追加時に前方車検出結果（図 5(a6)）を再利用できない．その場合，前方車の検出画像（図 5(a6)）や前方情報（図 5(a8)）も再利用できない．したがってアプリケーション追加時のライブラリの再利用性が低く要件 2 を満たさない．

3.　データアクセスライブラリの設計方式の提案

本節では本研究で提案するライブラリの設計方式 SASLD について述べる．

3.1　ライブラリ設計方針

SASLD では階層的データ管理方式と同様に，階層的にライブラリを定義することで新たにライブラリを定義する際に既存のライブラリの計算結果を活用可能とする．そして階層的データ管理方式とは異なり方針 1，2 に示すように，特定のセンサ及びアプリケーションに依存するライブラリを分離することによりセンサ，アプリケーション追加時の開発コストを削減する．以下で方針 1，2 について説明する．

- 方針 1. センサ依存部のライブラリを分離

車間距離センサなど特定の種類のセンサに依存する処理をセンサ依存部のライブラリとして，共通部のライブラリと分ける．これによりセンサ追加時のライブラリ変更箇所をセンサ依存部に留め，共通部のライブラリの変更を不要にする（要件 1 の満足）．先ほどの例では図 5(b1)〜(b4) に示すように，各センサの生データを入力とし特定センサに依存しない形式に変換するライブラリをセンサ依存部に定義する．これにより例えば新たな車種で車間距離センサの種類が変更になった場合にも，新たな車間距離センサのデータ形式変換ライブラリをセンサ依存部に定義することで，前方車との相対座標を算出するライブラリ (図 5(b11)) などの変更を不要にする．

- 方針 2. アプリケーション依存部のライブラリを分離

衝突防止など特定の種類のアプリケーションに依存する処理をアプリ依存部のライブラリとして分離し，アプリケーション依存部を除いた処理を共通部のライブラリとして定義する．これによりアプリケーション追加時に多くの共通部のライブラリを再利用可能とする（要件 2 の満足）．例えば図 5(b) に示すように階層型データ管理方式とは異なり，前方車画像（図 5(b4)）や前方情報（図 5(b8)）の算出処理を前方車検出判定処理と分離する．そして前方車検出判定処理を衝突防止対象車検出ライブラリ（図 5(b12)）としてアプリケーション依存部に定義する．これにより対象とする前方車が異なる追随走行アプリケーション（図 5(b14)）の追加時にも，アプリケーション依存部に追随走行対象車検出ライブラリ（図 5(b13)）を定義することで，前方車画像（図 5(b4)）や前方情報（図 5(b8)）は再利用可能となる．

3.2　運転支援アプリケーション

本研究では 3 種類の運転支援アプリケーションを対象とし，ライブラリを定義する．対象アプリケーションは表 1 に示すアダプティブクルーズコントロール（ACC，表 1(1)），ディスタンスコントロールアシスト（DCA，表 1(2)），プリクラッシュブレーキ（PCB，表 1(3)）である．各アプリケーションは ZMP 社の RoboCar [14] に搭載されたレーダ，車速，ステアリングのセンサを入力とし動作する．本節ではライブラリを定義するために対象アプリケーションの要件から必要データを導く．

対象アプリケーションが必要とするデータを表 1 の必要データに示し，各データの説明を表 2 に示す．各アプリケーションの基本的な動作としては，まず対象とする先行車を認識し，認識できた場合にはその先行車の（補完）車間距離，先行車相対速度，先行車速度などに基づきアクセスやブレーキ等の制御方法を決定する．そこで必要データとしてはまず対象とする先行車の有無の情報が挙げられる．対象とする先行車はアプリケーション毎に異なるため，ACC，DCA，PCB の対象車の有無（表 2(5)〜(7)）をデータとして定義する．またアクセスやブレーキの制御のために，ACC，DCA は補完車間距離（表 2(2)），先行車相対速度（表 2(3)），先行車速度（表 2(4)）を必要とし，PCB は車間距離（表 2(1)）を必要としているため，これらのデータを定義する．

表 2　必要データ

ID	データ	説明
1	補完車間距離	補完車間距離はレーダが先行車を認識できなかった時間を補完した車間距離である．ACC や DCA はカーブなどで一時的に先行車を見失ってもレーダの最新の計測値から算出された近似値をもとに制御を行うため，補完された車間距離を必要とする．
2	車間距離	車間距離はレーダセンサから得られた最新の車間距離である．PCB は必要時以外に動作すると運転者にストレスを与えると考えられているため，必要時以外はできる限り動作を起こさないよう設計されている．先行車をレーダが見失えばすぐに PCB の対象から外すために補完された車間距離ではなく，レーダから取得した最新の車間距離を必要とする．
3	先行車相対速度	先行車相対速度は，補完車間距離の推移から算出される先行車の相対速度である．ACC や DCA で適切な加速の制御量を決定するために必要となる．
4	先行車速度	先行車速度は先行車相対速度に車輪速センサから得られた自車速度を足し合わせて算出される先行車の速度である．ACC や DCA で適切な加速の制御量を決定するために必要となる．
5	ACC 対象車の有無	ACC 対象車の有無は入力情報から判定された ACC 対象車の有無を保持するデータである．補完車間距離が一定値以内であること，ステアリングの値が有効範囲内であること，一時的な誤検出を防ぐため検知確度 (レーダの連続検知時間) が一定値以上であること，ACC は停止車両を対象としないため先行車速度が一定値以上であること，これらの上記 4 つの条件を全て満たすと対象車ありと判定する．対象車の有無は制御内容を決定するために必要となる．先行車がいなければ設定された速度を保ちドライバへの通知を行い，対象車がいれば取得した補完車間距離，先行車相対速度，先行車速度から適切な加速制御量を決定し，制御を行う．
6	DCA 対象車の有無	DCA 対象車の有無は入力情報から判定された DCA 対象車の有無を保持するデータである．ACC と同様に制御内容を決定するために必要である．DCA は停止車両も対象車とする点と，ACC よりも補完車間距離が短い範囲内の車両を対象車とする点が ACC の判定条件と異なる．
7	PCB 対象車の有無	PCB 対象車の有無は入力情報から判定された PCB 対象車の有無を保持するデータである．検知確度 (レーダの連続検知時間) が一定値以上であること，車間距離が一定値以下であること，この上記 2 つの条件を満たすと対象車がいると判定される．

3.3　運転支援アプリケーション向けライブラリ設計

前節で述べた運転支援アプリケーションが必要とするデータを配信するライブラリについて述べる．図 6 に定義したライブラリを示し，また図 7 に各ライブラリを構成するオペレータを示す．3.1 節で述べたように，ライブラリはセンサ依存部，共通部，アプリケーションに分けて定義する．以下では各々で定義したライブラリについて説明する．

3.3.1　センサ依存部

表 3 にセンサ依存部のライブラリとその算出方法を示す．センサとしては車速，レーダ，ステアリングを持つため，その各センサの値を入力とし，整形した値を提供するライブラリを定義する．例えばレーダを入力源とするライブラリとしては，先行車の車間距離（表 3(2)），検知結果（表 3(3)）を配信するライブラリを定義する．車間距離，検知結果の情報は特定のレーダの情報に依存しない形式を取り，したがって 3.1 節で述べたようにレーダのデータ形式に関わらず同様の情報を配信するライブラリも定義可能とする．

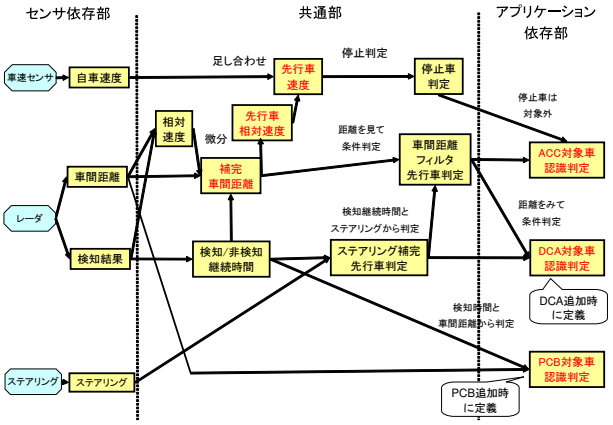

図 6　データアクセスライブラリの設計

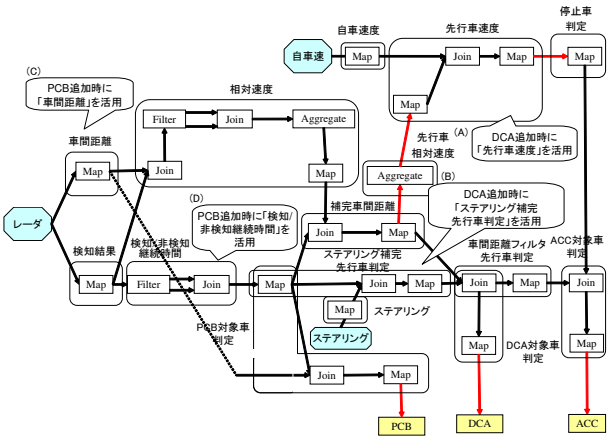

図 7　データアクセスライブラリのオペレータ

表 3　センサ依存部のデータアクセスライブラリ

ID	ライブラリ名	算出方法
1	自車速度	車速センサから自車の速度を抽出
2	車間距離	レーダから得られた情報から先行車との車間距離を算出
3	検知結果	レーダから得られた情報から先行車の検知結果を算出
4	ステアリング	ステアリングセンサから得られた情報から自車の舵角を算出

3.3.2　共　通　部

表 4 に共通部のライブラリとその算出方法を示す．共通部では特定のセンサやアプリケーションに依存しないライブラリを定義する．具体的には，アプリケーション ACC，DCA が必要とするデータである補完車間距離（表 4(3)），先行車相対速度（表 4(4)），先行車速度（表 4(5)）の算出処理は，特定のセンサやアプリケーションには依存しないため共通部のライブラリとして定義する．また補完車間距離の算出に必要となる相対速度の算出ライブラリ（表 4(1)）及び，検知/非検知時間の算出ライブラリ（表 4(2)）も共通部で定義する．さらに ACC の対象車判定に必要とする停止車判定結果（表 4(7)）や，車間距離でフィルタリングした先行車判定（表 4(8)），ACC 及び DCA

表 4　共通部のデータアクセスライブラリ

ID	ライブラリ名	算出方法
1	相対速度	現在及び一つ前の車間距離から先行車の相対速度を算出
2	検知/非検知継続時間	検知結果の履歴から，先行車を継続して検知または非検知し続けている時間を算出
3	補完車間距離	レーダが先行車を検知できなかった時間を相対速度で補完し車間距離を算出
4	先行車相対速度	現在及び一つ前の補完車間距離から先行車の相対速度を算出
5	先行車速度	自車の速度と先行車相対速度から先行車の速度を算出
6	ステアリング補完先行車判定	検知継続時間 2s 以上または，非検知継続時間が 2s 以内かつステアリング角度± 20 度以上なら先行車ありと判定
7	停止車判定	先行車の速度が 5m/s 以内なら停止車と判定
8	車間距離フィルタ先行車判定	ステアリング補完の先行車がありと判定かつ，補完車間距離が 100m 以内なら先行車ありと判定

表 5　アプリケーション依存部のデータアクセスライブラリ

ID	ライブラリ名	算出方法
1	ACC 対象車判定	ACC は停止車を対象としないため，車間距離フィルタの先行車があり，かつ停止車でないと判定されたら，ACC 対象車ありと判定
2	DCA 対象車判定	ACC とは異なり停止車を対象としまた対象とする車間距離が異なるため，ステアリング補完の先行車がありと判定かつ，車間距離が 80m 以内から DCA 対象車ありと判定
3	PCB 対象車判定	検知継続時間 2s 以上かつ車間距離が 50m 以内なら PCB 対象車ありと判定

の対象車判定に必要なステアリングで補完した先行車判定（表 4(6)）のライブラリも共通部に定義する．

3.3.3　アプリケーション依存部

表 5 にアプリケーション依存部のライブラリとその算出方法を示す．アプリケーション依存部ではセンサ依存部や共通部のライブラリの出力を入力とし特定のアプリケーションにのみ配信するデータを算出するライブラリを定義する．具体的には ACC，DCA，PCB が対象とする先行車を判定するライブラリ（表 5(1)〜(3)）は各々のアプリケーションに依存した処理であるため，アプリケーション依存部に定義する．

4.　評　　価

本稿では提案したライブラリ設計方式 SASLD の評価について述べる．

4.1　評 価 方 法

SASLD を活用し運転支援アプリケーション向けのライブラリを実現した場合の，アプリケーション追加時の開発コストを評価する．評価方法としては，ACC，DCA，PCB の順にアプリケーションを追加したと仮定しアプリケーション追加時のライブラリの追加コード行数を測定し階層型データ管理方式と比較する．

4.2　階層型データ管理方式によるライブラリ設計

比較に用いる階層型データ管理方式によるライブラリ設計について述べる．図 8 に定義したライブラリを示し，また図 9 に各ライブラリを構成するオペレータを示す．2.3.2 節で述べたように，ライブラリは第一層〜第三層に分けて定義する．

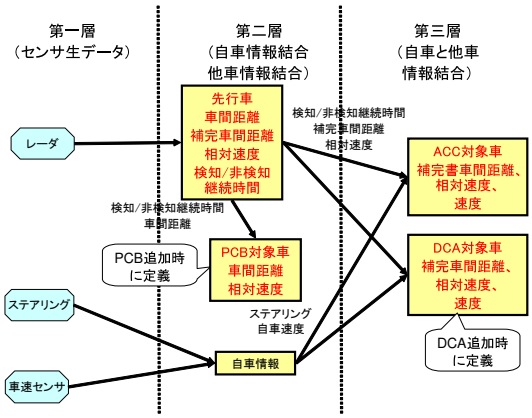

図 8　階層型データ管理方式によるライブラリの設計

- 第一層

第一層ではセンサの生データを定義する．センサとしてはレーダ（図 8(1)），ステアリング（図 8(2)），車速センサ（図 8(3)）があるためそのセンサの生データを定義する．したがってレーダの場合，SASLD のセンサ依存部ではレーダから算出する先行車の検知結果，車間距離を定義するのに対し，第一層ではレーダの値，すなわち前方 240 °の範囲の 0.3 °間隔ごとの距離情報の配列を定義する．

- 第二層

第二層ではレーダの情報から算出する先行車の情報を定義する．具体的には先行車との車間距離，補完車間距離，相対速度，検知/非検知継続時間を算出するライブラリ（図 8(4)）を定義する．また PCB 対象車を認識し，その車間距離，相対速度を算出するライブラリ（図 8(5)）も，レーダの情報のみを用いるため第二層に定義する．さらに同じく第二層でステアリング，自車速度の情報を結合し自車情報を算出するライブラリ（図 8(6)）を定義する．

- 第三層

第三層では補完車間距離などの先行車の情報と自車情報を統合し，ACC 及び DCA 対象車を判定し，その補完車間距離，相対速度，先行車速度を算出するライブラリ（図 8(7)(8)）を定義する．また SASLD とは異なりアプリケーション依存部を分離しないため，先行車速度やステアリング補完先行車判定などのライブラリは定義せず，前記の ACC 及び DCA 対象車の情報を算出するライブラリ内で処理を行う．

4.3　評価結果，考察

DCA 追加時の追加コード行数を表 6 に示す．SASLD では図 7（A）に示すように，ACC で活用したライブラリ「先行車速度」を DCA でも活用することができ，またライブラリ「補完車間距離」「先行車相対速度」も同様に活用できる．またライブラリ「DCA 対象車判定」の定義において，図 7（B）に示すように ACC のライブラリ「ステアリング補完先行車判定」を再利用することができる．一方，階層型データ管理方式では図 9

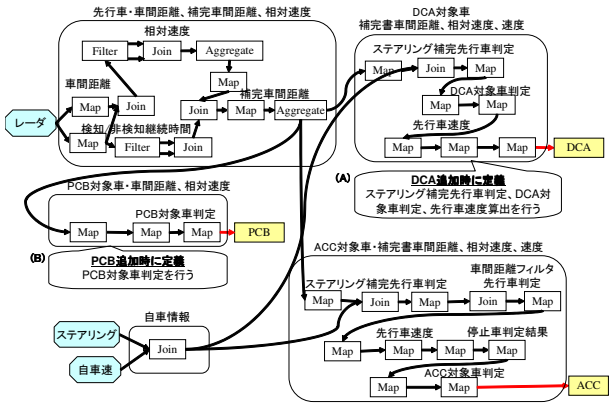

図 9　階層型データ管理方式によるライブラリのオペレータ

（A）に示すようにアプリケーション依存部を分離しないため，ライブラリ「先行車速度」（31 行）や「ステアリング補完先行車判定」（31 行）は定義されない．したがって SASLD の方針 2 により，階層型データ管理方式と比較した DCA 追加時のライブラリ開発コスト削減（追加行数計 62 行削減）を確認できた．

表 6　DCA 追加時の追加コード行数

評価対象	追加したコード行数	合計行数	変更割合
階層型データ管理方式	93 行	428 行	21.7%
SASLD	31 行	377 行	8.22%

また PCB 追加時の追加コード行数を表 7 に示す．SASLD では図 7（A）（B）に示すように，ACC や DCA で活用したライブラリ「車間距離」「検知/非検知継続時間」を PCB でも再利用し「PCB 対象車車判定」を定義することができる．一方，階層型データ管理方式では第三層の ACC 及び DCA 対象車の補完車間距離，相対速度，速度を算出するために第二層でライブラリ「車間距離」「検知/非検知継続時間」を定義するため，SASLD と同様に PCB で再利用できる．したがって PCB 追加時には SASLD と階層型データ管理方式の追加行数は変化が見られなかった．

表 7　PCB 追加時の追加コード行数

評価対象	追加したコード行数	合計行数	変更割合
階層型データ管理方式	32 行	460 行	6.96%
SASLD	32 行	409 行	7.82%

今後はアプリケーションやライブラリを増やし，アプリケーション追加時の開発コストを評価することで SASLD の有用性を確認する．なお今回の評価ではセンサを固定した環境で評価した．そのため SASLD の方針 1 で述べたレーダなどのセンサ追加時の開発コスト評価については今後の課題とする．

5. おわりに

本稿では車載データ管理 PF におけるライブラリの設計方式 SASLD を提案した．SASLD ではセンサに依存する処理を分離することにより，センサ追加時のライブラリの変更箇所を少なくする．またアプリケーションに依存する処理を分離することにより，アプリケーション追加時に多くのライブラリを再利用可能とする．本研究では 3 種類の運転支援アプリケーションを適用事例としライブラリを開発し，SASLD によりライブラリの開発コストが小さくなることを確認した．

謝辞　本研究の一部は科研費 (22240003) の助成を受けている．

文　献

[1] 浅沼信吉，加世山秀樹: 安全運転支援のための車両予知・予測技術のとりまく状況, 国際交通安全学会誌, 2006.
[2] W.D. Jones: Keeping Cars from Crashing, IEEE Spectrum, 2001.
[3] 藤田浩一，宇佐見祐之，山田幸則，所節夫: 衝突危険性のセンシング技術, 自動車技術, 2007.
[4] 西垣戸貴臣，大塚裕史，坂本博史，大辻信也: 予防安全の高度化を実現するセンササーフュージョン技術, 日立評論, 2007.
[5] AUTOSAR (AUTomotive Open System ARchitecture), http://www.autosar.org/.
[6] 山田真大，鎌田浩典，佐藤健哉，手嶋茂晴，高田広章: データストリーム管理機構を利用した車載データ統合モデルの提案と評価，自動車技術会論文集，Vol.42, No.2, 2010.
[7] 勝沼 聡，山口 晃広，熊谷 康太，本田 晋也，佐藤 健哉，高田 広章: 車載組込みシステム向けデータストリーム管理システムの開発，電子情報通信学会論文誌 Vol. J95-D, No.12, pp.2031-2047, Dec, 2012.
[8] 山口 晃広, 本田 晋也, 佐藤 健哉, 高田 広章: 車載システム向けデータストリーム管理システムにおけるクエリ自動構築手法, 第 11 回情報科学技術フォーラム講演論文集, Vol.2, pp.7-14, 2012.
[9] R. Motwani, J. Widom, A. Arasu, B. Babcock, S. Babu, M. Datar, G. Manku, C. Olston, J. Rosenstein, and R. Varma: Query Processing, Resource Management, and Approximation in a Data Stream Management System, Conference on Innovative Data Systems Research (CIDR), 2003.
[10] A. Arasu, S. Babu, and J. Widom: The CQL continuous query language: semantic foundations and query execution, The VLDB Journal, 2006.
[11] Borealis Distributed Stream Processing Engine, http://www.cs.brown.edu/research/borealis/public/.
[12] 山田 真大, 鎌田 浩典, 手嶋 茂晴, 高田 広章, 佐藤 健哉: データストリーム管理機構を利用した車載データ統合モデルの提案と評価, 自動車技術会論文集, Vol.41, No.2, 2010.
[13] Borealis Team: BOREALIS APPLICATION PROGRAMMER 'S GUIDE, http://www.cs.brown.edu/research/borealis/public/publications/borealis_application_guide.pdf, 2006.
[14] ZMP: RoboCarR 1/10 / ZMP RC-Z, http://www.zmp.co.jp/e-nuvo/jp/robocar-110.html.