Vol.2014-SLDM-165 No.12
Vol.2014-EMB-32 No.12
2014/3/15

Anroidデバイス向けデータストリーム管理システムの試み

岡本 昌訓[1]　Mohammed Bhuiya[1]　中本 幸一[1]

概要：近年，自動車において安全運転支援等のための様々なシステムが登場している．これには，例えば，レーンキープアシストや自動駐車システム等がある．これらのシステムには様々なセンサが用いられているが，数多くあるセンサデータを処理するためのソフトウェアアーキテクチャが存在していない．この問題を解決するため，名古屋大学を中心とした産学連携プロジェクトで車載向けのデータストリーム処理機構 (eDSMS) が開発されている．本研究では，車載向けのデータストリーム処理機構で処理されたストリームデータを主にスマートフォンなどの Android を搭載したデバイスで扱うためのシステムの提案と試作を行っている．本稿では車載向けのデータストリーム管理システムから得られるストリームデータを Android デバイスで扱うための処理機構について述べる．

1. はじめに

自動車において低燃費化や走行性能の向上，運転支援等のための様々なシステムが開発されている [1]．近年これらのシステムは大規模化，複雑化が進んでいる．それに伴い，ECU(Electronic Control Unit) と呼ばれる制御用コンピュータが 1 台の自動車に搭載されている数も 1990 年代前半では 30 個程度だったものが，現在多いものでは 100 個を超えている．また，多くのシステムはミリ波レーダやステレオカメラ等，種々のセンサが直接的または間接的に用いられている．しかしながら，数多くあるセンサデータを処理するためのソフトウェアアーキテクチャが存在していない．この問題を解決するため，名古屋大学情報科学研究科組込みシステム研究センター (NCES) における産学連携コンソーシアムプロジェクトにおいて，自動車システム用のデータ中心のソフトウェアアーキテクチャが開発されている．ここではシステム全体からセンサ部を切り離し，センサから得られたデータに対して，ストリームデータ処理技術を利用し統合管理を行う車載データ統合アーキテクチャを構築している．自動車システム用のデータ中心ソフトウェアアーキテクチャの開発の一環として車載向けのデータストリーム処理機構が開発されている．

本研究では，車載向けのデータストリーム処理機構で処理されたストリームデータを，主にスマートフォンなどの Android を搭載したデバイスで扱うためのシステムの提案と試作を行っている．本稿では車載向けのデータストリーム処理機構 (eDSMS) の概要と，Android 向けデータストリーム管理システムについて述べる．

[1] 兵庫県立大学大学院応用情報科学研究科
Graduate School of Applied Infomartics, University of Hyogo

2. Cloudia: 車載データ統合プラットフォーム

本研究において車載側で用いられている組込みシステム向けデータストリーム管理システム eDSMS が含まれている組込みソフトウエアプラットフォーム (Cloudia: Cloud technology for Data Integration Architecture) について述べる．Cloudia のシステムは図 1 のように構成されている．複数のアプリケーションにおいてそれぞれ管理されていたデータを物理的あるいは仮想的にデータ空間として統合管理し，センサを切り離す構成を採る．現行のシステム構成では，1 つ以上の ECU においてアプリケーションプログラムがデータ処理を行い，その結果，アクチュエータの操作を行う．そのため，センサの構成や設置状況の違いにより，アプリケーションプログラムを変更する必要が生じ，また，複数アプリケーションの設計や開発を統合することが困難となる．

車載データ統合アーキテクチャでは，周辺監視のためのレーダやカメラなどのセンサ情報から得られる物体の存在情報を確率として合算し，車速/車輪速センサやステアリングセンサ，加速度/角速度センサなどから得られる車両の運動状況を示すデータを統合化することで，複数のアプリケーションから共通に利用することが可能となる．

リアルタイム性に関する要求の高いデータに対して有効であるストリーム処理技術をセンサデータに適応することで，様々なデータ処理の共有化によるコスト削減しつつ，そのデータ処理結果の高速な配信を実現できると考えられる．そこで車載データ統合アーキテクチャのデータに対

Vol.2014-SLDM-165 No.12
Vol.2014-EMB-32 No.12
2014/3/15

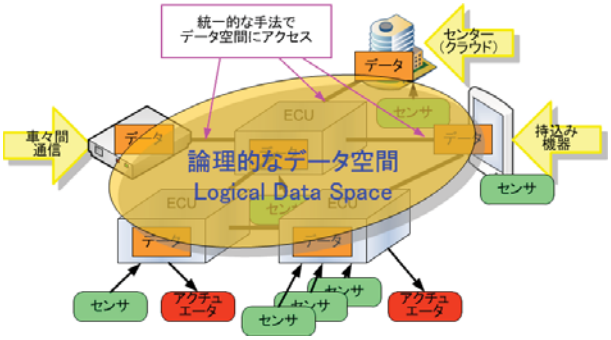

図 1　Cloudia: 車載データ統合プラットフォーム

してデータストリーム管理システム (DSMS) を適応する．DSMS は従来の蓄積型データベース管理システム (DBMS) と異なり，時々刻々センサから送られてくる一過性のデータを逐一データベースに格納した後に処理するのではなく，クエリを逐次に実行する．

2.1　eDSMS

本研究において車載側で用いられている組込みシステム向けデータストリーム管理システム eDSMS について述べる．eDSMS[2][3] は Borealis[4][5] を参考に開発された組込みシステム向けデータストリーム管理システムである．特に車載システムをターゲットしている．データストリームに対するクエリは，ストリームクエリ言語で記述される．ストリームクエリ言語には CQL やデータフロー言語があり，eDSMS ではデータフロー言語で用いられており，Filter，Map，Aggregate 等のオペレータを用いて手続き的に記述する．オペレータの種類及びその説明を表 1 に示す．eDSMS は組込みシステムを対象としているため，クエリは静的に定まるものとし，図 2 にあるようにターゲットソフトウェア開発時に C/C++言語に変換して，必要なライブラリのみを使ってリンクし，ターゲットソフトウェアに組み込む．

eDSMS のランタイムは，ストリーム管理部 (固定長キュー/可変長キュー)，オペレータ実行部 (Filter /Map /Union /Join /Aggregate)，ウインドウ管理部 (個数/時間/キー毎ウインドウ)，集計関数 (Count /Sum /Average /Max /Min)，スケジューラ (入力順/時刻順/優先度付)，データ通信部 (TCP /UDP /CAN /FlexRay /Lin) のコンポーネントグループ (コンポーネント) で構成される．

表 1　eDSMS のオペレータ

オペレータ	説明
Filter	単一の入力ストリームから指定された条件に合致したデータを抽出し出力ストリームに出力する
Map	単一の入力ストリームに対して操作や関数実行を行いストリームデータとして出力する
Unite	複数入力ストリームをマージし，単一の出力ストリームに出力する
Join	入力ストリームからのデータを一定期間メモリに保持し，条件にあったデータを結合させ，単一の出力ストリームに出力する
Aggregate	入力ストリームからのデータを一定期間メモリに保持し，そのデータに対して Aggregate 関数を実行し単一の出力ストリームに出力する

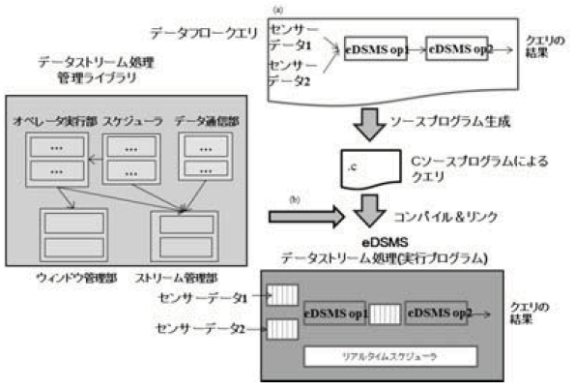

図 2　eDSMS

2.1.1　スキーマデータ

eDSMS におけるスキーマデータについて述べる．スキーマデータの定義は，ストリーム定義やクエリ定義と共に XML 形式で記述されている．出力ストリームのスキーマ定義の例の一部を図 3 に示す．

```
<schema name="SchemaOut1">
    <field name="timestamp" type="int"/>
    <field name="signalId" type="int"/>
    <field name="currentStatus" type="int"/>
    <field  name="latitude" type="double" />
    <field  name="longitude" type="double" />
 </schema>
```

図 3　スキーマ定義の例

これは，出力されるストリームデータの，1 つ目が “timestamp” という名前の int 型であるということである．2 つ目は “signalId” の int 型である．3 つ目以降も同様であり，合計で 5 つのデータが eDSMS から SchemaOut として出力されることを示す．

3.　Anroid デバイス向けデータストリーム管理システム (AeDSMS)

3.1　AeDSMS

Anroid デバイス向けデータストリーム管理システム (以下，AeDSMS と呼ぶ) は組込みシステム向けデータストリーム管理システム eDSMS（以下，車載 eDSMS と呼ぶ）から取得したストリームデータを Android デバイスで扱うためのシステムである．

Android デバイスにおいて，AeDSMS と車載 eDSMS 間

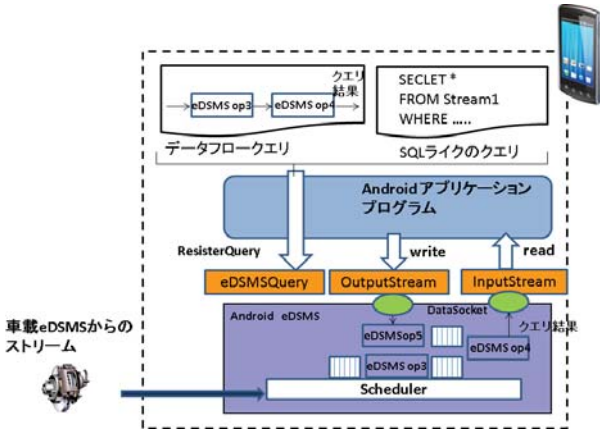

図 4　AeDSMS アーキテクチャ

3.2　AeDSMS におけるクエリ

AeDSMS においてクエリ言語として 2 種類ものが考えられる．1 つは eDSMS と同様のデータフロー言語で記述するというものである．もう一方は SQL に類似した記法によるもので記述するというものである．本稿では，Android アプリケーションプログラムで扱いやすくするため，後者の SQL に類するものを使用している．以下に車載 eDSMS から図 3 で示したスキーマに基づくストリームデータから必要なデータを取り出すクエリの例を図 5 に示す．

```
select *
where currentStatus = 0   AND signalId < 2
```

図 5　クエリの例

の利用ケースとして以下ような 3 つが考えられる．

ケース 1： 車載 eDSMS データストリームを利用して運転手や乗客にサービスを提供するケース．例えば，車載 eDSMS からのストリームデータを取得し，必要な情報を取得したストリームデータから取り出し運転者等に提示することが挙げられる．

ケース 2： Android デバイスに入力された情報を車載 eDSMS のデータストリームへ入力するケース．例えば，運転者や同乗者が Android の GUI 等を通じて運転者が判断した状況を車載 eDSMS へ入力し，それを通じて他の車両に通知するといったことが挙げられる．

ケース 3： Android デバイスのリソースを利用する目的で，車載 eDSMS のデータフロークエリの一部を Android プラットホームにオフロードするケース．これによって処理時間の短縮が可能となる．

以上のように車載 eDSMS で処理されたストリームデータを Android デバイスで利用をできるようにすることで，ドライバーなどの利用者に対してより簡便なサービスを提供することが可能となる．本稿では，ケース 1 を考慮した Android デバイス向けのデータストリーム管理システムについて述べる．

次に AeDSMS のアーキテクチャを図 4 に示す．AeDSMS では車載 eDSMS から入力されてきたストリームデータに対してクエリを実行する．クエリにマッチしたデータに対して Android の Java アプリケーションにおいて次の 2 つのアクションを取ることが考えられる．

(1) Borealis と同様に Java のアプリケーションのコールバック関数 (リスナー関数) を呼ぶ
(2) マッチしたデータのみを Java の InputStream として出力する

ここでは (2) のみを採用した．Stream クラスでマッチしたデータを受け取る方が連続したデータ処理が容易になると考えたからである．

これは，車載 eDSMS から取得したストリームデータのうち，where 節で指定した条件のみを取り出すクエリである．今回使用しているクエリは，一般的な SQL とは一部異なっており，select 文は，`select *`のみとしている．実際にストリームデータのうち，一部分のデータが必要な場合は，Java の InputStream から指定したデータのみをアプリケーションプログラム側で取り出すものとしている．

3.3　AeDSMS クエリ処理

3.3.1　処理の流れ

AeDSMS がストリームデータを処理する流れについて述べる．ストリームデータを処理する流れは図 6 にあるように，まず車載 eDSMS からストリームデータを取得し，取得したストリームデータの各値を Java オブジェクトへ変換する (Receive Stream Data)．次にパーサを用いてクエリ処理を行う (Query Parser)．クエリ処理では主にストリームデータをフィルタリングする役割を担う．最後にクエリ処理を行ったデータをバイト配列へ変換し，Java の Stream として出力する (To Byte array)．

3.3.2　eDSMS スキーマデータ依存ファイルの生成

AeDSMS においてストリームデータを処理するにあたり，車載 eDSMS のスキーマデータに依存するプログラムが存在する．そこで，それらのプログラムを効率的に作るため，スキーマデータを利用して必要なプログラムファイルを生成するすることにした．スキーマデータ定義については 2.1.1 で述べた．ここで生成することにしたファイルは図 7 にあるように主に 5 つのファイルである．そのうち，4 つのファイルは Java ソースファイルである．これを実現するために，ストリームデータの定義等の情報が記載されたスキーマデータを読み込み，eDSMS から出力されるデータの名前および型の情報を取得し，必要なファイルを生成する．各ファイルの説明と，各ファイルの図 6 における利用モジュールを表 2 に示す．

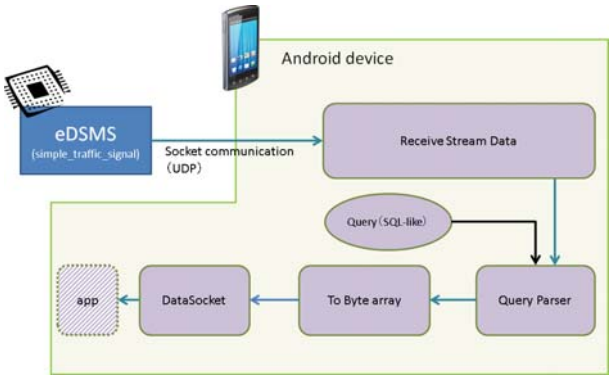

図 6　処理の流れ

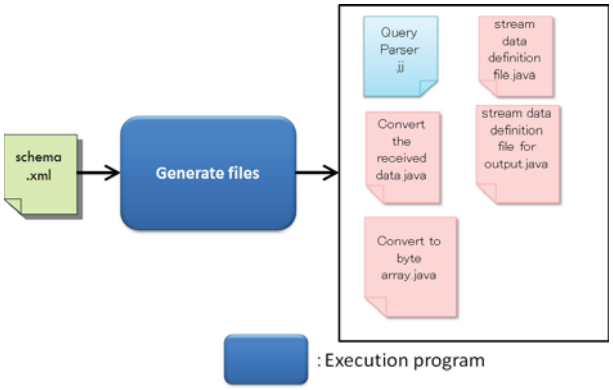

図 7　ファイル生成

表 2　生成するファイル

ファイル名	説明	利用モジュール
Query Parser.jj	パーサ作成のソースファイル	Query Parser
stream data definition.java	ストリームデータの定義	Receive Stream Data, Query Parser
Convert the received data.java	eDSMS から受信したストリームデータを変換する	Receive Stream Data
stream data definition for output.java	出力用ストリームデータの定義	Query Parser, To Byte array
Convert to byte array.java	ストリームデータをバイト配列へ変換	Query Parser

3.3.3　ストリームデータの受信

ストリームデータ受信部は eDSMS からストリームデータを取得する機能を担う部分である．ストリームデータを取得するために eDSMS と Android デバイスは通信を行う必要があり，現状では無線 LAN を用いている．

また，車載 eDSMS からストリームデータを取得する際にデータフォーマットについても考慮する必要がある．データ送信する際に用いられる主なフォーマットとして，XML(Extensible Markup Language) や JSON(JavaScript Object Notation) がある．それぞれのフォーマットは以下のような特徴を有している．

XML

- マークアップ言語であるため，要素の定義などが柔軟
- ファイルサイズが大きくなりやすい
- 可読性がよくない

JSON

- ファイルサイズが比較的軽量
- 多くの言語で簡単に扱うことが可能
- xml に比べると柔軟性に乏しい

本システムではデータが，常に送信されるというデータストリームの性質があるため，1 回あたりのデータが少量であるという観点や汎用性の高さなどを理由に JSON 形式を用いた．

3.4　Java オブジェクトへの変換

現状の AeDSMS では，3.3.3 で述べたように，車載 eDSMS との通信時のデータフォーマットは JSON 形式を用いている．JSON 形式のストリームデータは文字列であるので，プログラムで扱いやすくするために Java のオブジェクトへ変換する必要がある．Java において JSON を扱うためのライブラリは複数存在するが，本システムでは，比較的簡単なコードで Java オブジェクトへ変換することができるという点から，google-gson[6] を用いて JSON 形式のストリームデータを変換した．

3.5　クエリパーサ

クエリパーサは SQL に類する形式で記述されたクエリに従いストリームデータを処理する部分である．SQL に類する形で記述されたクエリを扱うにはパーサが必要となる．そこでパーサを作成するためにパーサジェネレータを用いた．

3.5.1　パーサジェネレータ

パーサジェネレータはパーサを作成するプログラムである．作成されるパーサが構文解析を行う際の手法には，上向き構文解析法 (bottom-up parser) と下向き構文解析法 (top-down parser)[7] の大きく 2 種類あるが，本研究では下向き構文解析を用いたパーサジェネレータである JavaCC を用いた．

JavaCC(Java Compiler Compiler) は元々は米国サンマイクロシステムが開発した Java ベースのコンパイラ・コンパイラである．2003 年には BSD ライセンスに基づいたオープンソースとなり，ソースの取得も可能である [8]．現在の最新バージョンは 6.0 である．JavaCC の機能的な特徴として以下のことがある [9]．

- 生成されるファイルはすべて純粋な Java コード
- 入力は拡張 Backus-Naur 記法
- 文法は下向き構文解析法の 1 つである LL(k)（デフォルトでは LL(1)）

● JavaCC に付属している JJTree で構文木を作成可能

このような特徴を鑑みてパーサジェネレータとして JavaCC を用いた．特に 1 つ目に挙げた生成されるファイルはすべて純粋な Java コードであるという特徴は，生成されたパーサを Android で用いる際に修正・追加をせずに移植が可能であり，別途必要なライブラリ等がないということは大きなメリットである．また，JavaCC は主に Android アプリや Java などの開発等多く用いられる統合開発環境である Eclipse のプラグインが存在するためコーディングや実行を比較的簡便に行うことができるという特徴も有している．

パーサは作成したソースファイルを JavaCC でコンパイルすることで，自動生成される．今回作成したパーサの機能は，クエリと与えられた文字列をトークンと比較し，マッチした場合はそれぞれのトークンのごとに事前に指定した処理を行う物である．準備した主なトークンには，“+”や“-”，“<”等の算術演算子や，“AND”,“OR”といった論理演算子等がある．

3.6 バイト配列化

Datasocket が InputStream で取得する際にバイト形式で行うため，出力用のデータストリームをすべてバイト配列へ変換する必要がある．そのため，出力データのバイト配列への変換用ファイル (Convert to byte array.java) では，各ストリームデータをバイト配列に変換する処理を行う．その際，すべてデータを 1 つのバイト配列にする．

3.7 Datasocket

Datasocket は Android アプリケーションとストリームの通信方法を管理し，車載 eDSMS のストリームデータと Java の stream クラスライブラリの間をブリッジするものである．DataSocket のクラス定義を以下に示す．

```
class DataSocket {
  DataSocket DataSocket(eDSMSQuery,
                                int StreamId);
  inputStream getInputStream();
  outputStream getOutputStream();
}
```

DataSocket クラスは以上のクラス定義に示すようなメソッドを持つ．getOutputStream() メソッドは，eDSMS クエリで指定した条件に合致したストリームデータを OutputStream で取り出すようにするメソッドである．getInputStream は車載 eDSMS へデータを渡すための InputStream を用意するものである．現在のところ，Android アプリケーションプログラムはスキーマデータ定義に対応したクラスを準備し，InputStream からバイト配列のデータを読み込み，スキーマに対応したデータフィールドを取り出すクラスを用意すると想定している．

図 8 サンプルアプリケーションのイメージ

4. 利用例

4.1 サンプルアプリケーション

車載 eDSMS と AeDSMS の利用の一例として，Android アプリケーションの例のイメージを図 8 に示す．このアプリケーションは周辺にある信号のデータを車載 eDSMS が取得できるという前提のもと，自車の前方にある信号の状態や次の状態に変化するまでの時間といった情報を地図上に表示する．このアプリケーションによって，自車の前に大型車が止まった際等に前方の信号が見えないといったことによる事故の防止や，ドライバーのいらいらの軽減が可能になると考えられる．

4.2 アプリケーションプログラムでの利用

今回の試作では AeDSMS のクエリ処理を行う際は，Android アプリケーションにおける Activity とは別のスレッドで処理を行っている．これは，アプリケーションと同じスレッドでクエリ処理を行った場合，アプリケーションからクエリ処理が呼び出されない限り，ストリームデータを取得し処理することができない場合が考えられるためである．そのようなことが起こると，クエリの処理がアプリケーション側に大きく依存してしまい，逐次的に送信されるという性質を持つストリームデータを扱う処理機構としては適切ではないからである．但し，スレッドをスタート

させるためのトリガーとしての機能はアプリケーションが持っている．

5. 議論

今回の試作を通して幾つかの問題点が明らかになった．ここでは，その問題点と現在施している解決策を述べる．

- アプリケーション側がクエリを変更を行う場合，別のスレッドで実行しているクエリ処理に対してクエリの変更を動的に反映させるケースがある．解決策として，クエリの変更を行う際にはスレッドを停止させている．
- Android アプリケーションが車載 eDSMS のスキーマに依存している．このスキーマ情報をどのように Android アプリケーション側に提供するかの問題がある．現在は事前に配布することを想定しているが，より柔軟な方法が望まれる．

6. おわりに

本稿では，Android デバイス向けのデータストリーム管理システムとして車載 eDSMS のストリームデータを Android デバイスで利用するための処理機構におけるクエリ処理部について述べた．そして，SQL に類する形でクエリを記述し処理を行う箇所の設計及び実装を行い Android 上での動作を確認した．

しかしながら，ストリームデータを処理するに当たり以下のようなか課題が存在している．

- 5 章で述べた課題への対応
- 車載 eDSMS へデータを送る機能の追加
- データフロー言語によるクエリの追加

2 点目については AeDSMS から車載 eDSMS へストリームデータを送るために，車載 eDSMS の対応が必要である．同時に，AeDSMS もクエリ処理したものを eDSMS へ送信する機構を追加する必要がある．今後はこれらの課題を踏まえた上でより汎用性の高いシステムにしていく予定である．

謝辞 eDSMS を開発している名古屋大学組込みシステム研究センター Cloudia プロジェクトメンバーに感謝します．なお，本研究の一部は，SCOPE(0159-0036)，JSPS 科研費 (25240007, 24500045) の助成を受けている．

参考文献

[1] 国土交通省自動車交通局 先進安全自動車推進検討会, “先進安全自動車（ＡＳＶ）推進計画報告書－第４期ＡＳＶ計画における活動成果について－”, http://www.mlit.go.jp/jidosha/anzen/01asv/resourse/data/asv4pamphlet_seika.pdf,2011.

[2] 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 D，vol.J95-D，no.12，pp.2031–2047，2012.

[3] 佐藤 健哉, 勝沼 聡, 山口 晃広, 島田 秀輝, 本田 晋也, 中本 幸一, 高田 広章，“Cloudia：車載データ統合プラットフォーム－基本コンセプト－”，情報処理学会研究報告 (組込みシステム), Vol.2011-EMB-24, No.18,pp.1-6, 2012 年 3 月.

[4] Daniel J. Abadi, Yanif Ahmad, Magdalena Balazinska, Ugur C， etintemel, Mitch Cherniack, Jeong-Hyon Hwang, Wolfgang Lindner, Anurag S. Maskey, Alexander Rasin, Esther Ryvkina, Nesime Tatbul, Ying Xing, and Stan Zdonik, “The Design of the Borealis Stream Processing Engine”, CIDR Conference, 2005.

[5] Brown Computer Science,“The Borealis project”,http://cs.brown.edu/research/borealis/public/, 参照 2013 年 12 月.

[6] “google-gson”,https://code.google.com/p/google-gson/, 参照 2013 年 12 月.

[7] 今城哲二, 布広永示, 岩澤京子, 千葉雄司, “コンパイラとバーチャルマシン”, オーム社, 2004.

[8] 五月女健治, “JavaCC-コンパイラ・コンパイラ for Java”, テクノプレス, 2003.

[9] Java Compiler Compiler,“JavaCC”, https://javacc.java.net/, 参照 2013 年 12 月.