Vol.2013-OS-127 No.9
Vol.2013-EMB-31 No.9
2013/12/3

アンドロイド系組込みデバイスと web クラウド上のリンクト・データとの融合

大橋　正[†1]

CPU,FPGA,DDR3,メインメモリー等のハードウェアから構成されるアンドロイド系組込み(eALD)デバイスに実装されたアンドロイド・プラットホーム上で走行できる所与のソフトウェア・プロダクツをクラウド上のリンクト・データを辿って組込みデバイスへ取り込むことで顧客の満足度を維持せしめる組込みデバイスと web のリンクト・データとの融合方式を検討した.

Fusion with Android Embedded Devices and Linked Data on web Cloud

TADASHI OHASHI[†1]

An Android embedded devices with Web Linked Data (eALD) in which CPU, FPGA, DDR3 etc. are mounted, is proposed to upgrade the devices by means of hardware components and software products. Operating system, hardware`s emulator, firmware (FPGA Data), applications are downloaded into eALD ‘s storage devices from web download sites through Web Linked Data. As results, the eALD computing may perform the excellent fusions with an Android Embedded System and Web Linked Data.

1. はじめに

Android のスマートフォン等のスマートデバイスは今日の社会システムの重要な下部構造を形成している.しかしシステム・サステナビリティは余り重要視されては来なかったが,近年になって環境保全等の見地から話題に上るようになった。一旦,ユーザがデバイスを購入するとハードウェアを始めとしてオペレーティング･システム(OS),エミュレータ,ファームウェア,アプリケーション・ソフトウェア等のデバイス･プロダクツの更新が不十分であり,購入デバイスの機能に対する顧客の満足度を維持継続することが難しく,新規機種の購入が避けられない場合が散見される.更に上記のデバイス・プロダクツの設計,製造,保守等のプロセスが国内外の多種業者に散在しており,購入デバイスから見たデバイス･プロダクツのソースが皆目,鳥瞰視ができない.購入デバイスの販売元も個々のユーザが自社の販売デバイスに実装されたデバイス・プロダクツの提供時の型番や仕様が判明できたとしても,他社で開発されたデバイス・プロダクツの改版,廃止などまで把握できず,これらの維持管理はできない.従いデバイス購入者自らが機能維持の持続性の為にデバイス・プロダクツの改版を自らが web 等を利用し,デバイス・プロダクツの改版をせねばならない.

以上の改善の為にデバイスをオープンソースの Android[1)]プラットフォームに限定し,Android 系組込みデバイスと web を融合させ購入デバイスの機能アップや設計過誤の是正のためのパッチ改版を購入デバイスに必要なプロダクツを自動的に実行ならしめる方式を検討した.

この方式はアップグレイダブルなコンピューティングシステム(Upgradable Computing)を形成するものであり,従来からのリコンフギュラブルなコンピューティング(Reconfigurable Computing)と同様に FPGA(Field Programmable Gate Allay)を実装し,オープンソースの Android プラットフォームを導入し,プロダクツのソース情報を Web 上の LOD[2)] (Linked Open Data)による融合を図り所与のプロダクツのソースを検索入手可能な eALD(Embedded Android with Linked Data)組込みデバイスを提唱するものである.

本論は以下の構成で検討結果の報告を展開する.
(1)当該分野に関わる現状技術と従来の組込みデバイスの現状課題を述べる.(第 2 章)

†1 アイリクト　iLICT Co.
(Homepage URL http://www.ilict.jp/,　Mail Address ohashi@ilict.jp)

Vol.2013-OS-127 No.9
Vol.2013-EMB-31 No.9
2013/12/3

(2)課題の解決に向けて. (第 3 章)
(3)現状課題と今後の予測されうる課題を解決する eALD デバイスの提唱.(第 4 章)
(4)eALD システムの効果について.(第 5 章)
(5)まとめ. (第 6 章)
(6)今後の課題.(第 7 章)

2. 現状技術と現状課題

当該分野に関わるハートウェア・デバイスやモジュールは以下の特徴や課題を持つ.

2.1 現状技術の歴史的背景

1980 年代に至っては,コンピュータはハードウェアとソフトウェアで構成されていた. そのコンピュータは紙面上に描画された回路図を見ながら設計製造を進めた. 設計者は設計要求仕様書や詳細設計書等を満たすことで設計製造の工程を終えることができる.この観点でハードウェアやソフトウェア設計製造に於いて図面は非常に重要な役割を担った.[3),4)]

2.2 現状技術

当該分野に関わる組込みデバイスは以下の 4 カテゴリから構成されている.

2.2.1 ハードウェア・コンポーネンツ

通常の組込みデバイスには ARM(Advance RISC Machines),CPU(Central Processing Unit),GPU(Graphics Processing Unit), DSP(Digital Processing Unit),DD3 SDRAM(Double-DataRate3 Synchronous Dynamic Random Access Memory)等のストレージコンポーネンツや各種センサ及び液晶表示パネルやタッチパネル等が装着されている.こうしたハードウェアは提供されるデバイスとして多岐にわたる.これらの管理はメーカ独自の型番や版数等で管理されるのが通常である。これらのコンポーネントはデバイスなどに組み込むと,その構成内容や各機能は固定される.機能向上には部品を追加又は置換えるか全く新しいデバイスを購入しなければならない.

2.2.2 オペーレーティング・システムズ

今日では多種多様のオペレーティング・システムが存在する.その必要条件はリアルタイム性,マルチプロセッサシング性等をクライテリアとして所与のデバイスに採用される.スマートフォンには iOS や Android が考えられるが,アップグレイダブルでオープンソースで提供される Android を選択して本論を説明している.

2.2.3 ファームウェア・プロダクト

ファームウェアは一般に組込みソフトウェアと称されるが,本論では組込みソフトウェアの語句は使用しない.当該ファームウェアにはハードウェアの FPGA ゲート等を制御するものであり,通常はデバイス内のストレージコンポーネンツに格納される.ファームウェアの初期段階では PROM(Programmable Read Only Memory)や PLD(Programmable Logic Device)等に格納されて市場の製品に適用される.ファームウェアの過誤修正のパッチ適用や機能アップ等はこれらのコンポーネンツに格納して販売製品内の該当品と置換える. しかし近年ではインターネットを介して適用できるようになった.

2.2.4 アプリケーション・プロダクト

ウェブブラウザ,アドレス帳,e メール等がアプリケーション・プロダクツの範疇に包含される.所与の製品が市場に投入される場合には製品にプリインストールされているか,若しくは購入ユーザがインターネットを介して App Store や Android マーケット(Google Play) からダウンロードする場合がある.

2.3 関連課題

以下の表 1 は上記の 2.1 で述べられた現状技術の歴史的背景と 2.2 で述べられた現状技術をまとめたものである.

表 1 組込みデバイスのカテゴリ別共通課題
Table 1 Common problems of an embedded system.

カテゴリ		プロダクツ	プロダクツ製品管理	プロダクツのリソース保管場所管理
C1.	ハードウェア	ARC,CPU,GPU,FPGA,PDL,GPS,DSP,Display,Sensors,etc.	プロダクツの製品管理(ソフトウェア名称,版数,リソース格納場所)は同一社内の部門別若しくは国内外の関連部署で独自に管理している.近年では各部門各社で独自にインターネットを介して管理することがある	プロダクツのリソース保管場所管理は同一社内の部門独自で管理されていたり国内外の関連部門が独自にリソース管理をしている.近年では各部門各社で独自にインターネットを介して管理することがある
C2.	オペレーティング・システム	オープンソースのAndroid OS		
C3.	ファームウェア	ゲートコントロール・ソフトウェア		
	エミュレータ	ハードウェアの一部をソフトウェアで代替させるエミュレータ		
C4.	アプリケーションズ	Android OS下で走行するソフトウェア		

表 1 の左側列は C1 から C4 の 4 カテゴリを示している.これらの開発,設計,製造,保守などは各部門,各社で個別のルー

ルの下で独自に行われている.完成したこれ等の製品管理は独自に行われている.更にソースの管理も各部署又は各企業で独自に行われている.近年ではインターネットを介して共通サーバを構築し効率化を図っている場合も散見できる.こうした背景がなぜ eALD が開発されねばならないのかの理由となっている.

2.4 各範疇の課題

ハードウェア・コンポーネンツ,リアルタイム・オペレーティング・システム,ファームウェア,エミュレータ並びにアプリケーションに於いては,設計文書類は重要な役目を担っている.更に上記の各プロダクツの扱うデータの保管管理も重要になるが各部門,国内外の各企業間での開発,設計,製造,保守スタイルが独自に行われることが多々発生する場合がある.ましてや全てのプロダクツが集積されて製品化され市場に出された以降では誰もが管理できず,購入者のみが独自の意思で管理しなければならない.製品過誤修正等が生じた場合,メーカ側で購入者をフオローするのが非常に難しく,特には規模の大きい通常の ICT システムと異なりスマートデバイスにあっては流通機構が複雑であり把握しきれない.

3. 課題の解決に向けて

上記で述べた如く,製品の企画,設計開発,製造,試験,保守等には全体を鳥瞰しホリステック(Holistic)に捉え,無駄なコンポーネンツやプロセスを省くことが必要である.この結果,コストダウンに繋がるばかりでなく種々の障害を未然に防ぎ,製品の信頼性を向上させることになり,メーカサイドからも顧客の販売製品をフォローすることが出来て,対顧客へのサービスアビリティが向上する.この結果,顧客の満足度が維持継続するサステナビリティ指向の製品供給が可能となる.この実現に向けて以下の可能性を探った.

3.1 関連分野の課題解決に向けて

上記の課題を解決するテクノロジィはLODとFPGAをスマートデバイスに積極的に適用させることを検討した.前者に於いてはリンクト・データ及びデータマッビッピングの著書[5]を参照した.

(1)オペレーティング・システムやファームウェア,エミュレータ,アプリケーションの製造元が千差万別なメーカが提供する方式を改め,クラウド上でホリステックに管理する方式を導入する.すなわち SPMW(Software Products Management on Web)の導入を提唱することである.

(2)この実現のためにクラウド上に RDF[6],RDFs[7],OWL[8]等 web オントロジィ[9]を構成してソフトウェア・リソースの管理し相互関係のリンクを用いて表現させ SPRMGR(Software Product Manager)が LODMGR(LOD Manager)を駆動しソースの位置検索と取得を行う.

(3)デバイス上にはFPGAを中心としたシステムがアップグーダブルとなるSUSMGR(Sustainability Manager)を構築する. 基本的にはリコンフギュラブルとは異なり,あくまでも購入製品の機能や品質を維持継続し顧客の満足度を維持継続させるコンセプトを導入したサステナブル指向でハードウェアを設計する.

(4)上記の(1)～(3)を満足する eALD デバイスを提唱する.

3.2 組込みデバイスのアップグレード

ユーザはデバイスのアップデートを開始すると以下のプロセスが実行される.

(1)eALD の SPRMGR は現在実装されているハードウェア・コンポーネント,オペレーティグシステム,ファームウェア,エミュレータやアプリケーションの名称や型番とその版数を認識する.

(2)eALD の SPRMGR はオペレータがアップグレードを要求している名称や型番とその版数を認識する.

(3)eALD の PRGMGR が LODMGR を介してオペレータがアップグレードを要求している名称や型番とその版数を RDF,RDFs,又は OWL を eADLre で検索する.

(4)希望するソフトウェア・プロダクトの格納ロケーションが LOD により抽出できたら,ソースを eALD の内部ストレージへダウンロードする準備を行う.

(5)eALD の SUSMGR が既存ソフトウェアプロダクツとそのデータ類を SPMW のデータストァーへ退避させる.

(6)eALD の SUSMGR がダウンロードしたプロダクツを格納する.

(7)eALD の SUSMGR がシステムの再起動を行う.

4. eALD システムの提唱

4.1 先行研究調査

リコンフィギュラブル・コンピューティングの専門用語はしばしばバーチャル・コンピューティング[10]と同義として扱われる.従い,先行研究調査にはリコンフィギュラブル・コンピューティングとバーチャル・コンピューティングの双方を調査対象としている.一方,本論のアップグレーダブル・コンヒューティングも前述専門用語と同一線上に存在すると捉えている.さて本論の先行研究調査では以上の 3 単語をキーワードとしてテーマに挙げて検討し, 以下

Vol.2013-OS-127 No.9
Vol.2013-EMB-31 No.9
2013/12/3

の表 2 にまとめた.

表 2 先行研究調査一覧

Table2. Preceding surveys for study research

	先行研究論文 1	先行研究論文 2	先行研究論文 3
タイトル	リコンフィギュラブル・コンピューティング	リコンフィギュラブル組込みミシステムの概要	AndroidにおけるJavaアプリケーションのFPGAアクセラレーション
発行年	1999	2006	2012
著者	末吉敏則,飯田全広	フィリップ ガルシア,キャサリン コンプトン,マイケル シャトル,エミリー ブレム,ウェニン フー	小池恵介,太田淳,大島浩太,藤波香織,那倉幸,竹本正志,中條拓伯
アーキテクチャ概要	①汎用エンジン方式(アタッチド・プロセッサ方式) ②リコンフィギュラブル・コプロセッサ方式 ③リコンフィギュラブ・プロセッサ方式	二つのアーキテクチャの例を挙げて解説 ①複数のFPGAの各々にSRAMを具備させたシステム ②複数のDSPに各々FPGAを具備させたシステム	Android端末上でのJava実行の高速化をFPGAで実現したリコンフィギュラブルAndroidについてハードウェア実行可能なJavaのソース部分をFPGA上で実行し全体の性能向上を実現したシステムについての研究
メリット	①ワークステーションなどのホストコンピュータに不可し,計算集約的処理やプロセッサの不得意な処理を任せられる ②汎用マイクロプロセッサと組合わて実現できる ③プログラマブル・デバイスを用いて実現できる	①低消費電力設計の実現 ②フォルト・トレラント・システムの実現 ③リアルタイムのサポートを実現 ④セキュリティ設計の実現	①リコンフィギュラブルなAndroid評価環境が構築された. ②Java アプリケーションが従来より,高速化処理される様になった. ③Android が業界で唯一初めてFPGAで実現できた.
デメリット	①リコンフィギュラブル・コンピューティングではハードウェアとソフトウェアを同時に開発する必要がある ②ハードウェアをソフトウェアで代替させると処理速度が問題となる ③現状のFPGAではリコンフィギュラブル・コンピューティング向けには対処していない	デメリットについて特に記載されていなかった	当該論文ではJTAGによるコンフィギュレーションの実行時間が明記されていなかった.
備考	当該資料は現状調査を解説.リコンフィギュラブル・コンピューティングの基礎を知るうえで役に立つ	リコンフィギュラブル・コンピューティングを設計する際に概念を捉えて設計する際には役に立つ	この論文には関連する索引が豊富であったのでアップグレイダブル・コンピューティングには非常に役に立つ

備考:○内数字は列挙番号であり,セル間の番号対応はされていない.

4.1.1 先行研究 1

当該論文 [11]は 3 タイプのリコンフィギラブル・コンピュータィングが調査されているので,基礎的な概要を知る上で有効であった.リコンフィギュラブル・コンピューティングには FPGA が必須であることが述べられている.処理速度の低下とコストアップとのトレードオフが重要な分岐点となる.

4.1.2 先行研究 2

当該論文 [12]では多種のアプリケーション環境について述べる.リコンフィギュラブル・コンピューティングの設計に於いて,留意すべき項目を例示している.例えばハードウェアの消費電力の低減,フォルト・トレーラントシステム設計,更にはセキュリティ問題を挙げている.ただの研究室内のリコンフィギュラブル・コンピューティングにとどまるのでなく製品化され顧客に手をされた時点にまで拡張して設計することが記述されていた.

4.1.3 先行研究 3

当該論文 [13]は以下の点で非常に有益な研究である.(1)リコンフィギュラル・コンピューティングの雛形を示した研究である.(2)その実現のために FPGA を用いて Java の実行をハードウェアで実行可能な Java のソース部分を FPGA 上で実行して全体の機能向上になっている点で参考となる.(3)リコンフィギュラブル・コンピューティングの様々な分野への可能性を示している.尚,関連文献[14][15]も併読して参考とした.

4.2 eALD システムの提唱

以上の先行研究調査を礎に種々多様な問題を解決ならしめ,独自のオリジナリティを発揮した eALD システムを提唱する. eALD 設計コンセプトは「web と eALD を融合し,web 上のソフトウェア・プロダクツを,eALD 内の FPGA に取込み,常にアップグレードを行うことでシステムの機能や品質を維持することでユーザの満足度を持続可能できるシステムを構築する」ことである.

4.2.1 リンクドデータとｅＡＬＤデバイスの融合

Web と eALD を融合させるためには LOD の導入が必要である. そこで関連書籍の引用を行う.タイトルは「Linked Data : Evolving the web into Global Data Space」[2]である.LOD は以下の各項目を満足しなければならない.

(1)あらゆるデータの識別子として URI(Uniform Resource Identifier)を使用する.

(2)識別子には(URI や他のスキーマではなく)HTTP(Hypertext Transfer Protocol)を使用し,参照やアクセスを可能とさせる.

(3)URI にアクセスされた際には有用な情報を標準的なフォーマット(RDF 等)で提供する.

(4)データには他の情報源における関連情報へのリンクを含め,web 上の情報発見を支援する.

「リンクト・データの基本的なアイデアはワールドワイドウェブ(WWW)の一般的なアーキテクチャをグローバル・スケールでの共有可能な構造に適用させることである」.このことは従来の web のアーキテクチャを理解することが重要となることを示す.ドキュメントを扱う web では簡単な標準ルールの集合として成り立っている HTML(Hyper Text Markup Language)は世界的に単一無比(ユニーク)な識別子の役割を担っている.一方単一無比のアクセス・メカニズムを具備している HTTP で HTML が呼ばれる.ウェブはウェブ・

ドキュメント間をハイパーリンクすることで構成されている.

4.2.2 ウェブとｅＡＬＤとの融合

eALD が web と融合する考えを拡張すると以下の図 1 の様に表現できる. 上部がウェブ上のクラウドを示している. ここには国内外のメーカの提供するプロダクツの管理情報とプロダクツを制御するゲートデータ等が分散して各メーカに散在しているデータを RDF 同志を結合することで容易にデータの所在を知りえる. そのデータは eALD 内のストレージコンポーネント例えば FPGA データとしてダウンロードすることで容易に結合が可能となる.

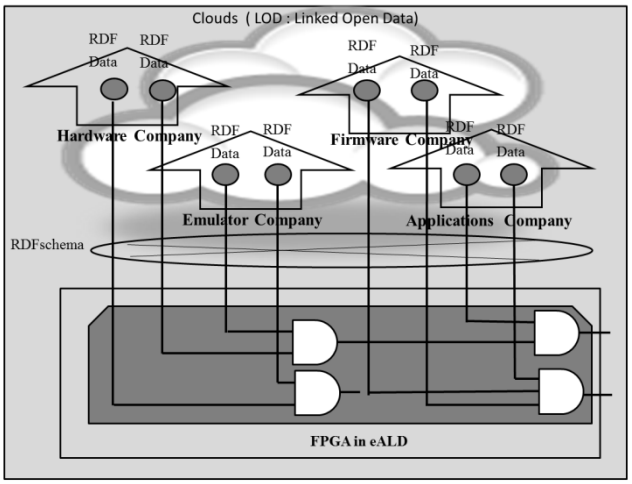


図 1 ウェブと eALD デバイスとの融合概念図
Fig. 1 Schematic concept between web and eALD

もし LOD を辿ってゲートデータの所在がわかればウェブが eALD のゲートを直接制御できることになる.

4.2.3 eALD のアーキテクチャ

eALD の語源は Yield なる語句からの造語である. イールドは「生み出す」とか「発生する」とかを意味し, システムのサステナビリティを実現することを意味する. すなわちシステムの機能向上に伴い不足したコンポーネンツを FPGA 等により機能を発生させるのが eALD の意味である.

所与の組込みデバイスのハードウェアは部分的に FPGA 内のエミュレータ等で置き換えられ部分的に代替することができる. 本論では一般的なリコンフィギュラブルとは概念の異なるアップグレイダブルはこの部分的に機能を生成させることを意味している.

図 2 に提唱する eALD システムの体系図を示し, 以下に説明する. eALD システムは大きく分けてクラウド上に存在する SPMW(Software Products Management on Web)及び eALD デバイスから構成される. eALD デバイスはハードウェアとソフトウェアで構成され, ソフトウェアは SPMW と接続された SPMD(Software Products in Device) と eACE(Embedded Android Control Engine)で構成される. 以下に eALD デバイスの詳細図を図 3. に示す.

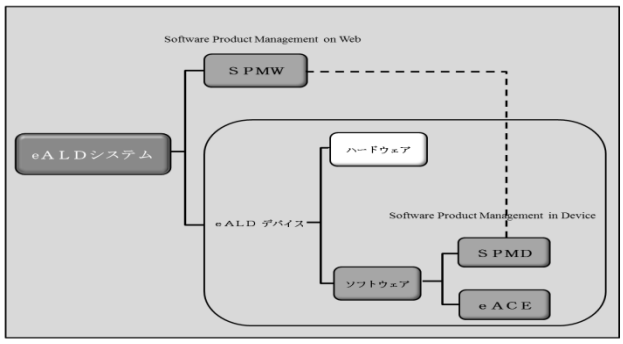
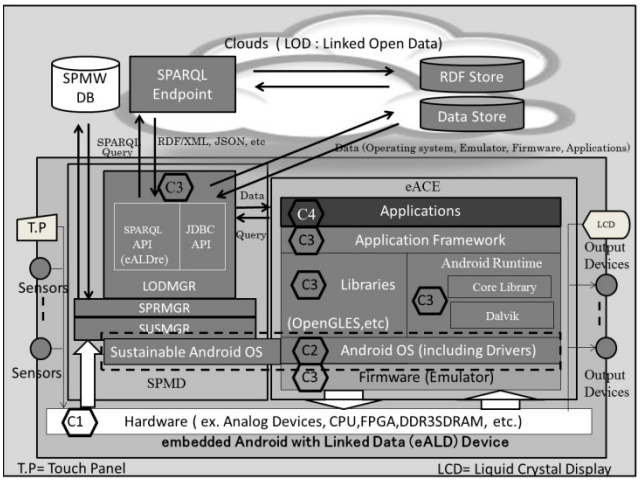


図２　eALD システム体系図
Fig. 2 Overall view of eALD system



備考:上記 C1～C4 は表 1 と対応している.

図３　eALD デバイス詳細図
Fig. 3 Detail view of eALD device

(1)クラウド上にはセマンティック・ウェブ [16]で構成される LOD が遍在する. LOD を辿って必要とされるソフトウェアプロダクツを収集する全体のメカニズムを総称して SMPW と呼称する. 詳細は事項で述べる.

(2)eALD デバイスのソフトウェアは SPMD(Software Products Management in Device)が在り, 大きく分けて以下の 4 ユニットから構成される.

①サステナブル Android OS は eACE に実装された AndroidOS の最低限機能を具備したもので, システムをアップグレードする際に動作する.

②SUSMGR(Sustainability Manager)はサステナビリティを実施するための全体的動作を制御する.

③SPRMGR(Software Products Manager)は eACE のソフトウェアのアップグレードを行う.

④LODMGR(LOD Manager)は eALDre(eALD Retrieval Engine)

の役目を担う LOD 検索を行う.
次に eACE の構成は図 3 で図示されている如く 6 項目から成る. 下部から上部に向かって説明する.
① ファームウェア(Firmware)はゲート制御データであり FPGA で使われる.
② Android OS はオープンソースのオペレーティング・システムである.
③ Android ランタイムはコア・ライブラリ Dalvik バーチャルマシーンから構成される.
④ OPenGLES などの様な各種ライブラリ.
⑤ アプリケーション・フレームワーク.
⑥ アプリケーション.
(3)eALD デバイスのハードウェアはアナログ部品, CPU, FPGA, DDR3 SDRAM, センサ [17] 等がある.

4.2.4 web 上のソフトウェア・プロダクト・マネージメント

前述の表 1 に区分された C1～C４はＬＯＤを扱うＳＰＭＷの処理対象になっていることを示す. 即ち C1～C４は共通の問題を抱えている. 製品の企画, 設計開発, 製造, 試験, 保守, 等のプロセスが千差万別となっていることである. この結果, ユーザは購入デバイスの最新版化を出来にくし, システムのサステナビリティの阻害要因になっていると考える. この解決法として, これを全てクラウドの同一レベルにソフトウェア・プロダクツのリソース保管場所（データストア）及びその管理情報（RDF ストア）を引き上げることが理想である. 具体的には RDF でプロダクト管理情報を表現しプロダクツ間の関係をリンク付けしＬＯＤ化することで解決できると考える. 以上の働きを SPMW で行う. SPMW は既存の LOD サーバ及び LOD の仕組みを用いて実現するもので特定なソフトウェアのシステム名称ではない. 以下の図 4 で詳細な働きを示す.

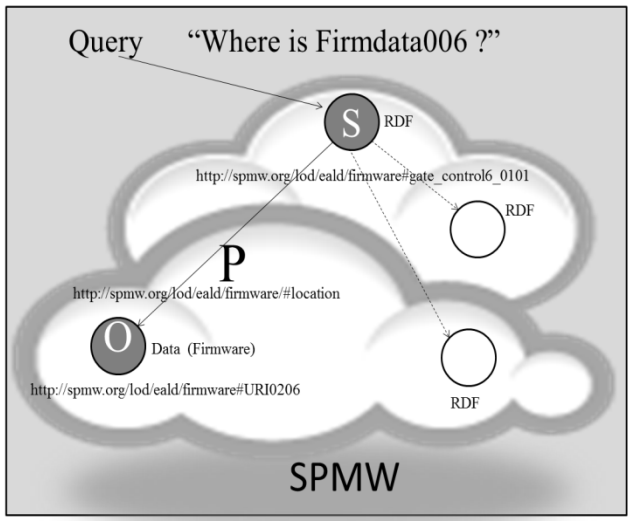


図４　SPMW の詳細図
Fig.4 Detail view of SPMW

「リンクト・データは web アーキテクチャと直結しておりグローバル・スケールでのデータ共有が可能である. 初めのリンクト・データの原理は URI を利用して特定化(ID 化)を行えることである. これは先に述べたウェブ・ドキュメントやデジタル・コンテンツを意味するものではなく, 実世界のオブジェクトや抽象的オブジェクトを特定化できることを意味している. これは図 4 で表象される.

図 4 では LOD の検索用エンジンとして eALDre を用いてファームウェアデータを取得する仕組みを示している SPARQL[18] を用いて,「Firmdata006 はどこに保管されていますか?」と尋ねる. この質問文はセマンティック・ウェブを用いて「S+P+O」で表現させる.「S」は主語(Subject)で Firmdata006 に該当する.「O」は目的語(Object)で「データの保管場所」であり「P」は述語で「存在する」に表現できる. この文章は Excel 等の表形式で作成すると RDF に自動変換し必要なサイト例えば Endpoint サーバに格納するウェブサイトが存在する. このサービスを利用することで他の RDF との関連性を検索でき, 所与の製品の全体的な構成表を検索することが出来る. 以上の仕組みを使用した結果, データが格納されている場所
(http://spmw.org/lod/firmware
#URI0206)が検索できる.

この実現のためには業界団体又は公的機関がソフトウェア・プロダクツのリソースのデータストア登録及びその管理情報の RDF 化を推進する必要がある. [19]

4.2.5 eALD システムのシーケンス図

図 5 に前述の動作を eALD システムとして SPMD と SPMW を関連づけてシーケンス図で示す
(1)SMPD ではユーザアップグレード要求により以下を実行する.
①SUSMGR が全体的なシーケンスの監視制御を行う.

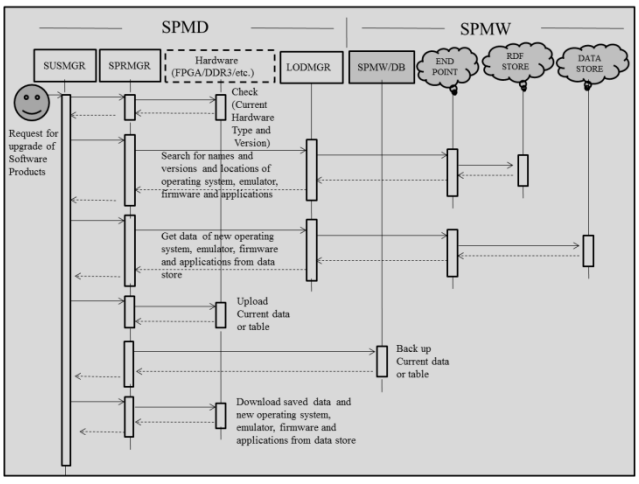


図５　eALD システムのシーケンス図
Fig.5 eALD system sequence chart

②SUSMGR は SPRMGR を通じて eALD デバイスに実装されているハードウェア・コンポーネントの型番,版数を把握する.
③SUSMGR はユーザに eALD システムの進行状況を eALD デバイスの LCD(Liquid Chrystal Display)を介してユーザに知らせる.
④もしユーザが正常な進行を確認し,アップグレード確認ボタンを押下することでアップグレード処理が行われる.
⑤SUSMGR は下位レベルの SPRMGR を起動し,SPRMGR が起動を終えた時点で,SUSMGR は eALD デバイスの再起動を行う.
(2)SPRMGR は LODMGR を駆動して LOD の検索を行う.
(3)LODMGR は SPARQL API(Application Programming Interface)を利用して LOD の収集を行う.
(4)SPMW は ENDPOINT プロセスの依頼を Endpoint サーバに依頼し所在情報により DATA ストアからデータの収集を終える.

図6には現在eALDデバイスにインストールされているソフトウェア・プロダクツの検索方法の詳細シーケンス図を示す.

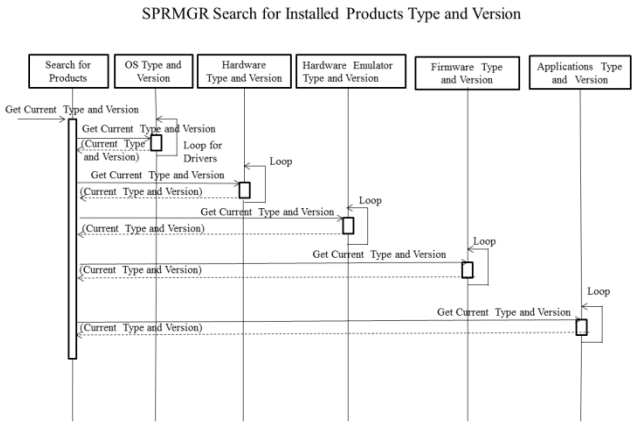


図６　SPRMGR の新プロダクツ検索シーケンス図
Fig6 A new product search sequence of SPRMGR

先ずオペレーティング・システムの名称及び版数が検索され,次にハードウェア・コンポーネントの名称及び版数をを検索する.ハードウェアの部分機能を代替するエミュレータの名称及び版数を検索し,ファームウェアの名称及び版数を検索し,最後にアプリケーションの名称及び版数を検索する.

以下に具体例を挙げて説明する.ユーザが Android のオペレーティング・システムの版数を 2.1(開発コードネームが Eclair)から 4.0.3(開発コードネーム:Ice Cream Sandwich)へアップグレードを行う場合に述べる.
(1) 4.0.3 を作動させるのに必要なハードウェア環境を調べる.
(2)現状の eALD デバイスのハードウェア構成を検索する.
(3)もしハードウェアの機能が部分的にエミュレータで可能か調べる.web の LOD 検索を行い,もし代替エミュレータが在れば取得する.もし存在しない場合は,この版数のアップグレードは行えないことになり,その旨ユーザに通知する.
(4)もし存在すれば Android OS 4.0.3 下で動作する.必要なオペレーティング・システム,エミュレータ,ファームウェア並びにアプリケーションの取得行う.

4.2.6 グレードアップに対する代替機能の可能性例

図 6 に示した SPRMGR のシーケンス図について述べる.ここでユーザが Android OS 2.1 から 4.0.3 へのアップグレードが出来るか否かが重要なポイントになる.代替可能であればハードウェアの部分機能の代替が行われる.その実際の可能性の対象を検討し図の 7 に例示した.

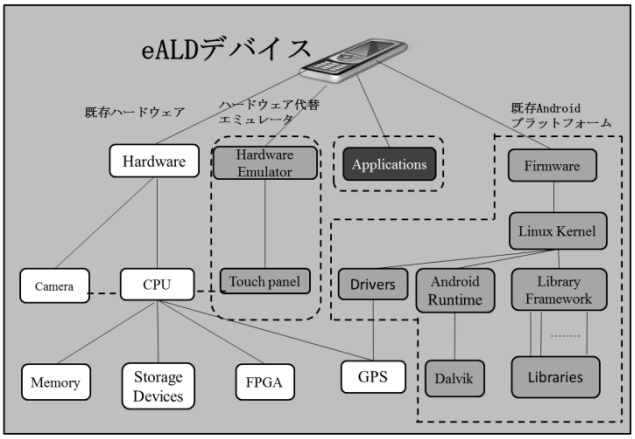


図 7　代替可能性検討図
Fig7 An alternative hardware functions

例えば 4.0.3 を動作させるにはメモリーが 1GB 必要であった場合は実装されているメモリーが不足しているかを判断し,不足していれば FPGA で代替させる必要がある.次にスクリーンショット機能が 4.0.3 に追加された.これは電源ボタンと音量ボリュームボタンを同時押下することで実現する.この制御も FPGA に仕掛けを組んでおくことで何とか実現できそうだと考える.以上の例で明確のとうり,代替機能がどの範囲まで実現できるかでアップグレードできる範囲が決まるので,今後はこの代替機能が容易となる様に事前にハードウェア及びソフトウェアのサステナブル設計が必要になる.

4.2.7 eALD の検索エンジン

eALD デバイスには LOD を検索するための eALDre なる検索エンジンが搭載されている.所与のプログラムから API(Application Programming Interface)により LDMGR のプログラムから走行できる.この設計方式は SPARQL を用い

て LOD を検索することで可能となっている. この検索概念は Tim Berners Lee のアイデアが基本となっている. [20)]

4.2.8 SPMW の web クラウド上リンクト・オープン・データ

リンクト・オープン・データは eALD システムにとって重要な情報源となっている. 多くのメーカは業界の LOD 推進団体の支援の下で標準化されたソフトウェア・プロダクツのリソースの web 格納とリソース管理情報の RDF 化を推進する必要がある. リソース管理情報ではプロダクツの名前, 版数, リソースの管理場所がポイントとなる. この管理情報は Excel の管理管表が在れば容易に RDF 化できるウェブサイトがあることを 4.2.4 で述べた. 本論では説明のために ソフトウェア提供メーカの 4 社に限定した. この具体例を表 3 から表 6 迄の 4 例を示した.

例 1)試行段階のオペーレーティング システム・・・・表 3
例 2)試行段階のハードウェア・エミュレータ ・・・・表 4
例 3)試行段階のファームウェア ・・・・・・・・・・表 5
例 4)試行段階のアプリケーション ・・・・・・・・・表 6

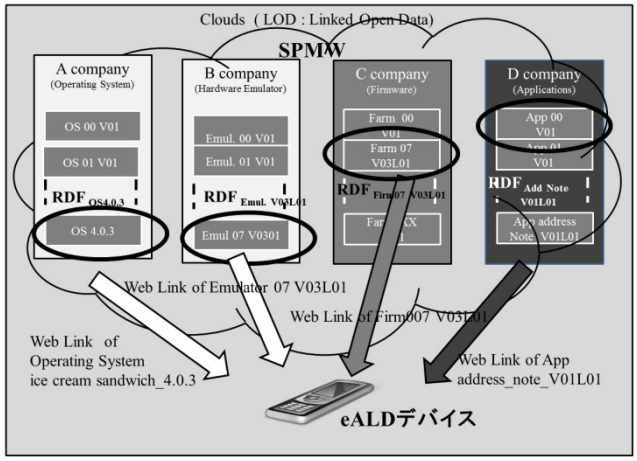


図 8 SPMW オープンデータの例示
Fig8 An example of SPMW open data

C1. オペレーティング・システム・テーブル (OST)

試行段階のオペレーティング・システム・テーブルをまず作成する必要がある. そこで以下の表 3 の如く Excel シートで作成し, RDF 変換フリー・サイトで変換させる準備を行う. このシートでは一番左側のセルに暫定的なオペーレーヘティング・システムのソフトウェア・プロダクト連番を記載している. 次のソフトウェア・コードネームはソフトウェア名称を示している. 次にはソフトウェアの版数を示し, API レベルも示してあるが本論では
参考情報に留めており利用いていない. 一番右側のセルには暫定的なソフトウェア・プロダクト・リソース格納場所を例示しており, 現時点では実在しない暫定的なドメイン名を示している.

表 3 オペレーティング・システムのリンクト・オープン・データ・テーブル例 (OST)
Table3. An example of Linked Open Data Table for Operating system (OST)

ソフトウェア・プロダクト連番	ソフトウェア・コードネーム(ソフトウェア名称)	ソフトウェア版数	API レベル	ソフトウェア・プロダクト・リソース格納場所*
Operation System 00	Eclair	2.1	7	URI0000
Operation System 01	Froyo	2.2	8	URI0001
Operation System 02	Gingerbread	2.3	9	URI0002
Operation System 03	Gingerbread	2.3.3	10	URI0003
Operation System 04	Honeycomb	3.0	11	URI0004
Operation System 05	Honeycomb	3.1	12	URI0005
Operation System 06	Honeycomb	3.2.X	13	URI0006
Operation System 07	Ice Cream Sandwich	4.0.1	14	URI0007
Operation System 08	Ice Cream Sandwich	4.0.3	15	URI0008
Operation System 09	Jelly Bean	4.1.X	16	URI0009
Operation System 10	Jelly Bean	4.2.X	17	URI0010

* URIの一例, http://spmw.org/lod/eald/os#icecreamsandwich_4.0.3
備考:上記のドメイン (spmw.org)は現時点で存在しない仮称である.

C2. ハードウェア・エミュレーション・テーブル (HET)

エミュレータのリンクト・オープン・データ・テーブル上記 C1 と同様に作成される. 一番左側のセルはエミュレータの暫定的ソフトウェア・プロダクト連番を示している. ソフトウェア名称やソフトウェア・プロダクト・リソース格納場所についても C1 と同様である.

表 4 ハードウェア・エミュレータのリンクト・オープン・データ・テーブル例 (HET)
Table4. An example of Linked Open Data Table for Hardware Emulator (HET)

ソフトウェア・プロダクト連番	ソフトウェア・コードネーム(ソフトウェア名称)	ソフトウェア版数	ソフトウェア・プロダクト・リソース格納場所*
Emulator00	Hardware00	V01L01	URI0100
Emulator01	Hardware01	V01L01	URI0101
Emulator02	Hardware02	V01L01	URI0102
Emulator03	Hardware03	V03L01	URI0103
Emulator04	Hardware04	V01L01	URI0104
Emulator05	Hardware05	V01L01	URI0105
Emulator06	Hardware06	V01L01	URI0106
Emulator07	Hardware07	V03L01	URI0107
Emulator08	Hardware08	V01L01	URI0108
Emulator09	Hardware09	V01L01	URI0109
Emulator10	Hardware10	V01L01	URI0110

* URIの参考例.
http://spmw.org/lod/eald/emulator#hardware07-0301
備考:上記のドメイン (spmw.org)は現時点で存在しない仮称である.

C3. ファームウェア テーブル (FWT)

ファームウェアのリンクト・オープン・データ・テーブルも上記 C1 と同様に作成される. 一番左側のセルはファームウェアの暫定的ソフトウェア・プロダクト連番を示している. ソフトウェア名称やソフトウェア・プロダクト・リソース格納場所についても C1 と同様である.

表 5 ファームウェアのリンクト・オープン・データ・テーブル例 (HET)

Table5. An example of Linked Open Data Table for Firmware (FWT)

ソフトウェア・プロダクト連番	ソフトウェア・コードネーム(ソフトウェア名称)	ソフトウェア版数	ソフトウェア・プロダクト・リソース格納場所*
Firmdata000	gata_control0	V01L01	URI0200
Firmdata001	gata_control1	V01L01	URI0201
Firmdata002	gata_control2	V01L01	URI0202
Firmdata003	gata_control3	V03L01	URI0203
Firmdata004	gata_control4	V01L01	URI0204
Firmdata005	gata_control5	V01L01	URI0205
Firmdata006	gata_control6	V01L01	URI0206
Firmdata007	gata_control7	V03L01	URI0207
Firmdata008	gata_control8	V01L01	URI0208
Firmdata009	gata_control9	V01L01	URI0209
Firmdata010	gata_contro1l0	V01L01	URI0210

* URIの参考例,
http://spmw.org/lod/eald/firmware#gate_control6_0101
備考:上記のドメイン (spmw.org)は現時点で存在しない仮称である.

C4. アプリケーション・テーブル (APT)

オペレーティング・システムの下で走行するアプリケーションのリンクト・オープン・データ・テーブルも上記 C1 と同様に作成される. 一番左側のセルはアプリケーションの暫定的ソフトウェア・プロダクト連番を示している. ソフトウェア名称やソフトウェア・プロダクト・リソース格納場所についても C1 と同様である.

表 6 アプリケーションのリンクト・オープン・データ・テーブル例 (HET)

Table6. An example of Linked Open Data Table for Application (APT)

ソフトウェア・プロダクト連番	ソフトウェア・コードネーム(ソフトウェア名称)	ソフトウェア版数	ソフトウェア・プロダクト・リソース格納場所*
Application 00	Web Browser	V01L01	URI0300
Application 01	E mailler	V01L01	URI0301
Application 02	Callendar	V01L01	URI0302
Application 03	Map	V01L01	URI0303
Application 04	Address Note	V01L01	URI0304
Application 05	Mouse support	V01L01	URI0305
Application 06	Software Keyboard	V01L01	URI0306
Application 07	Game00	V01L01	URI0307
Application 08	Game01	V01L01	URI0308
Application 09	Game02	V01L01	URI0309
Application 10	Game03	V01L01	URI0310

* URIの参考例,
http://spmw.org/lod/eald/application#address_note_0101
備考:上記のドメイン (spmw.org)は現時点で存在しない仮称である.

以上述べた C1～C4 の Excel シートは RDF の変換サイトのフリー・ツールを用いて, 容易に LOD データにすることが出来る.[21] 更に C1～C4 の Excel シートは以下の図 9 SPMW オープンデータの相互関係を例示できる. この結果 SPMW は eALD デバイスと融合して最新ソフトウェア・プロダクツを eALD デバイスへ提供できることになり顧客の満足度に対するサステナビリティを実現できる.

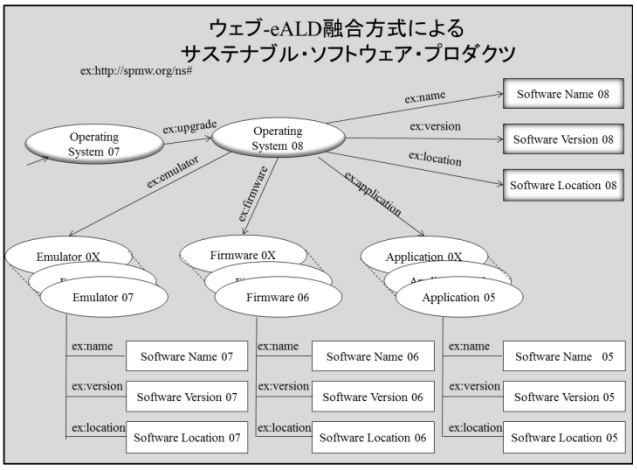


図 9 SPMW オープンデータの例示

Fig9 An example of SPMW open data

5. 新システムの効果

LOD は eALD 組込みデバイス上(オウン開発)だけでなくデバイス以外のシステムのデバッグ時又はシステム・ジェネレーション時(クロス開発)にも用いることが出来る. 従い eALD 組込みデバイス上だけで開発するのは思わしくない. 一方, アプリケーションはユーザ手元の eALD 組込みデバイスで例えば Google play で必要なアプリをダウンロードできる. 其のために eALD は検索エンジン eALDre を具備している. この様な機能の具備により以下のメリットを齎す.

(1) オペレーティング・システム, ファームウェア, エミュレータ及びアプリケーション・プロダクツが web 上で全てマッシュアップ(Mash-up)できるので, その結果, 提唱する eALD システムの SPMW はユーザが要求するソフトウェア・プロダクツを難なく, ユーザの所有する eALD デバイスに提供できる.

(2) ハードウェアの機能アップに関する要求も, エミュレータを eALD デバイスの FPGA へ書き込むことでユーザ要求を実現する方法が具備される.

(3) アプリケーションを開発時に, 多種多様な機種によるデバックが必要になるが, eALD デバイスのエミュレータを置き換えることで, 一機種の eALD デバイスだけで多種のデバイスをデバッグし, その機能を確認できる.

6. まとめ

eALD システムが本論文で提唱されシステムのサステナビ

リティの向上に実現性が在ることを見出した.しかし eALD システムはまだ幼稚期にある.現時点での実現される内容を以下に列挙した.

6.1 本研究での検討事項

本研究で検討された内容を以下に列挙する.
(1)eALD システムを構成する SPMW 上の LOD 及び eALD デバイスがユーザ購入機器の機能維持継続に対する満足度を持続可能とさせることがわかった.
(2)web 上の LOD と eALD デバイスの FPGA との融合がシステム・サステナビィリティに貢献することを示した.
(3)ハードウェアの機能追加がエミュレータでできることを示した.

6.2 今後の課題

eALD システムの実現に向けて今後の課題を列挙する.
(1)業界関係団体が集結しソフトウェア・プロダクツ管理情報の RDF を作成し LOD の蓄積を推進する必要がある.
(2)アップグレーダブルコンピューティングを実現する eALD デバイスの FPGA を中心とした回路設計の研究が必要である.
(3)サステナブル設計思想をハードウェア及びファームウェアを含むソフトウェア設計に浸透させ,この意識に立って設計するエンジニアを増強させることである.
(4)クラウド上のウェブ・リソースが eALD の FPGA を直接制御する技術の向上を目指す必要がある.

参考文献

1) 木島貴志,石丸宗平:アンドロイドプログラミング入門,情報処理,Vol.52,No.4・5,pp.527-539 (2011).
2)Tom Health, hristian Bizer: Linked Data: Evolving the Web into a Global Data Space, Morgan & Claypool Publishers (February 2011)
http://www.morganclaypool.com/
3)大橋　正:Android システム設計に於ける Semantic web の検討, Android Bazaar & Conference 2013 Spring/Tokyo (2013).
4)大橋　正: Web Linked Data の Android 系組込みデバイス設計への適用, 情報処理学会第 180 回 SE 第 29 回 EMB 合同研究会発表会
5)神崎正英:Linked Data とデータ,人工知能学会誌,Vol.27 No.2,pp.163-170,Feburaly(2012)
6) Resource Description Framework (RDF)
http://www.w3.org/RDF/ (2004)
7) RDF Vocabulary Description Language 1.0　RDFSchema (2004)
http://www.w3.org/TR/rdf-schema/
8) Web Ontology Language (OWL)
http://www.w3.org/2004/OWL/ (2004)
9)Vocabularies
http://www.w3.org/standards/semanticweb/ontology (2013)
10) Arnold, M., Fink, S.J. Grove, D., Hind, M. and Sweeney, P.F.: A Survey of Adaptive Optimization in Virtual Machine, *in Proc. IEEE,* Vol.93, No.2, pp.449-466 (2005)
11)末吉敏則, 飯田全広:ﾘｺﾝﾌｨｷﾞｭﾗﾌﾞﾙ・ｺﾝﾋﾟｭｰﾃｨﾝｸﾞ, *Magazine of Information Society of Japan*, Vol. 40, No.8, pp.776-782, (August, 1999)
12)Philip Garcia, Katherine Compton, Michel Schaelte, Emily Blem, Wenyin Fu: An Overview of Configurable Hardware in Embedded Systems, Hindawi Publishing Corporation *EURASIP Journal on Embedded Systems*, Volume 2006, Article ID 56320, pp.1-19 DOI 10.1155/ES/2006/56320 (2006)
13)小池恵介,太田淳,大島浩太,藤波香織,郡信幸,竹本正志,中條拓伯,Android に於ける Java アプリケーションの FPGA アクセラレーション, 情報処理学会論文誌, Vol.53 No.12 pp.2740-2751 (2012).
14)Lattanzi, E., Gayasen, A., Kandemir, M., Narayanan, V. Benini, L. and Bogliolo, A.: Improving java performance using dynamic method migration on fpgas, *Proc.18th International Parallel and Distributed Processing Symposium*, p.134 (2004)
15)Philip Faes, Mark Christians and, Dirk Stroobandt.: Interfacing java with reconfigurable hardware (2004).
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.124.7704&rep=rep1&type=pdf
16)Semantic Web
http://www.w3.org/standards/semanticweb/ (2013)
17)長野伸一:Linked Data とセンサネットワーク, 人工知能学会誌, Vo27,No.2 pp.200-206(2012).
18)SPARQL 1.1 Query Language
http://www.w3.org/TR/2013/REC-sparql11-query-20130321/ (2013)
19) 江上周作, 清水宏泰,谷口祥太,藤井章博 : ねじ LOD を基にしたマッシュアップアプリケーション,電子情報通信学会信学技報 IEICE Technical Report AI2013-17,SC2013-11(2013-08)
20) Christian Bizer, Tom Health, Tim Berners-Lee: Linked Data, *Magazine of Information Society of Japan*, Vol. 52, No.3, pp.284-292, (2011)
21)http://linkdata.org/ (2013)