分散インテントプラグイン: 組込み機器向け分散インテントの自動生成機構

長原裕希[†] 大山博司[††]
安積卓也[†††] 西尾信彦[†††]

スマートフォンが普及し，それらと家電との連携が多数提案されている．そこで我々は，Android 端末と組込み機器を連携するフレームワークで ある分散インテントを提案してきた．しかし，組込み機器の機能にあわせて通信機構を開発する必要があり，開発者の負担となってしまう．本論文では，TECS を利用して組込み機器の連携機構を自動生成する分散インテントプラグインを開発する．TECS は組込みシステム向けのコンポーネント システムであり，設計したコンポーネントに対応したコードを出力する．TECS がコード出力できない部分は，プラグインを開発・利用して生成できる．分散インテントプラグインを利用することで，開発者の負担を削減できることを確認した．さらに，生成された通信機構を評価し，分散インテントが満たす要件を維持していることを確認した．

DistributedIntentPlugin: Auto Generating System for Distributed Intent on an Embedded System

Yuki Nagahara ,[†] Hiroshi Oyama ,[††] Takuya Azumi [†††]
and Nobuhiko Nishio[†††]

We proposed a "Distributed Intent" framework which can cooperate with Android devices and embedded devices. However, the framework needs to implement a communication protocol for the cooperation, and it is the burden for a embedded system developer. In this paper, we propose a method to generate the framework by using the TECS generator. TECS is a component system based on a real-time OS. TECS is made by the TECS generator and the generator has plugins to generate codes. We implemented the plugin to generate the framework for an embedded system. We verified the plugin can reduce the burden for the developer dramatically, and the program that the plugin generated is keep requirements for the framework.

1. まえがき

近年では，Android や iPhone を筆頭にスマートフォンが急速に普及している．スマートフォンを利用し，ネットワークに接続された組込み機器を操作するためのサービスも多数提案されている．我々は，Android 機器を用いて，組込み機器を一部のプロトコルに依存せず，柔軟に操作できることを目的として,「インテント」を分散拡張する分散インテントフレームワークを提案してきた[1)]．

分散インテントを利用して想定されるサービスの例を，現行のサービスをもとに述べる．近年では，体に取り付けて健康管理などに利用するウェアラブル機器の開発が進み，普及しはじめている．ウェアラブル機器とインテントを用いた連携を行うことで，蓄積されたデータを Android 機器へ取り込んだり，Android 機器から手の届かない機器の操作を行ったりするサービスが期待できる．さらに，インターネットを利用した音楽配信サービスも普及してきている．インテントは命令の他に，付加情報を引数として送信できることから，音楽プレイヤーに対して URL と権利情報を送付することで，音楽再生が行えるサービスも期待できる．このように，多様な機能を持った組込み機器が存在するため，組込み機能の機能にあわせて機能通知を行う必要がある．しかし，現状は組込み機能ごとに分

† 立命館大学 大学院 情報理工学研究科
Graduate School of Information Science and Engineering, Ritsumeikan University
†† オークマ株式会社
OKUMA Corporation
††† 立命館大学 情報理工学部
College of Information Science and Engineering, Ritsumeikan University

散インテントを実装する必要があり，開発者の負担となってしまう問題がある．

TECS (TOPPERS Embedded Component System)[2] は，リアルタイム OS をベースとして，コンポーネント指向開発を行えるシステムである．TECS はコンポーネントに対応するコードを出力するジェネレータを利用して開発を行うが，すべてのコードを自動生成できるわけではないので，そのようなコードの自動生成機能をプラグインとして開発・追加できる．

本研究では，TECS ジェネレータのプラグイン機能を用いて，組込み機器が Android 機器とインテントで連携するコードを自動生成し，分散インテントに対応した組込み機器を容易に開発できる分散インテントプラグインを開発・実装する．分散インテントで利用する機能や機能数は，組込み機器ごとに異なるため，すべてをライブラリ化することは困難である．そこでまず，フレームワーク中の各組込み機器に依存せず共通で利用可能な部分と，本来開発者の実装を必要とする依存部分を明確にする．つぎに，本来開発者の実装を必要とする部分について，自動生成するプラグインを設計する．依存しないライブラリと自動生成コードにより，組込み機器を分散インテントに対応させることができる．以上から，分散インテントを組込み機器へ実装する際の，開発者の負担を削減できる．

本論文の章構成について述べる．第 2 章では，組込み機器を連携させるための既存の関連研究について解説し，その問題点を示す．第 3 章では，我々が提案する分散インテントフレームワークについて述べ，現状の課題を示す．第 4 章では，本研究と関係が深いシステムである TECS について解説する．第 5 章では，分散インテントプラグインに求められる要件を整理し，挙げた要件を満足するためのアプローチを示す．第 6 章では，提案プラグインの設計について述べる．第 7 章では，実装した分散インテントプラグインを用いて評価した結果を述べる．第 8 章では，本研究の結論を示す．

2. 関 連 研 究

本章では，組込み機器を連携するためのフレームワークについて解説し，問題点を示す．

2.1 CORBA (Common Object Request Broker Architecture)

CORBA[3] は OMG (Object Management Group) が定義した分散オブジェクト技術の標準規格であり，共通の呼び出し規約にしたがって動作するソフトウェアコンポーネント (オブジェクト) をネットワーク上の複数のデバイス上に配置し，それらを連携し動作させることでシステムの構築を行う．

ただし，CORBA はオブジェクトの生成に比較的高いリソースが求められる．したがって，リソース制約の厳しい組込み機器には向いていない．

2.2 UbiREMOTE

UbiREMOTE[4] は，家電を遠隔地から操作するサービスを提供するフレームワークであり，3D 仮想空間を利用して，ユーザに一元的かつ直観的な操作での制御を提供する．家電連携プロトコルには，UPnP (Universal Plug and Play)[5] が用いられている．UPnP はマイクロソフトが提唱する，家庭やオフィスに存在する組込み機器や PC などのデバイスを連携させるための技術仕様である．UPnP を用いることにより，デバイス同士をネットワークに接続するだけで連携できる．

しかし，連携には各デバイスで XML の解析が必要となる．更に，UPnP デバイスを操作する際は，操作用のアプリケーションを機能毎に用意する必要があるため，開発者の負担になる．

2.3 Cogma (Cooperative Gadgets for Mobile Appliances)

Cogma[6] は動的なネットワーク環境に組込み機器を適応させるためのミドルウェアであり，組込み機器を動的にフレームワークに追加し，連携できる．更に，Cogma は様々なプロトコルを柔軟に使い分けることで，組込み機器同士の連携が可能である．

しかし，連携する組込み機器は Cogma がインストールされている必要がある．したがって，Cogma をサポートしていない組込み機器を連携させることはできない．更に，プラットフォームとして Java VM が必要となるため，リソース制約の厳しい組込み機器には向いていない．

2.4 Chameleon

Chameleon[7] はスマートフォンを用いて家電を操作するためのフレームワークである．タグに触れる，カメラで写す，端末を振る，写真を選ぶ，ネットワークスキャンの 5 つの連携手法を状況によって切り替えできる．連携には矢印が印刷された RFID タグを利用し，RFID タグは自動発行システムで発行できる．

しかし，連携する組込み機器は Chameleon を利用するための RFID タグを用意する必要がある．更に，連携デバイスを統合管理するサーバが必要となるため，専用ハードウェアへの依存が大きい．

3. 分散インテント

分散インテント[1] は，Android と組込み機器とを

連携するフレームワークであり，Android 側サブシステム，組込み機器側サブシステムから構成される．

3.1 アプローチ

分散インテントは，関連技術の問題点を，以下のアプローチで解決している．

専用ハードウェアが不要: Android 機器から組込み機器へ連携を行うための処理は全て，Android アプリケーションとして実装している．したがって，RFID タグや連携サーバといった専用のハードウェアに依存しない設計となっている．

組込み機器側がプラットフォームに非依存: 組込み機器側サブシステムの実装は，多くの組込みソフトウェアで用いられている C 言語を用いて行う．よって，組込み機器側に Java VM を導入しなければならないといった制約がない．したがって，特定プラットフォームに依存しない設計となっている．

リソース制約の厳しい組込み機器に対応: Android 機器で行える処理は全て Android 側サブシステムで行うよう設計した．したがって，組込み機器の処理負荷を抑え，リソース制約の厳しい組込み機器へも導入が可能である．

3.2 フレームワークの動作

分散インテントの設計図を図 1 に示す．以下，図 1 をもとにフレームワークについて解説する．

3.2.1 組込み機器の検索・登録

フレームワークが起動すると，分散インテントが使える機器の調査のため，Android 機器は同一サブネット内に存在する組込み機器を検索する (図 1 I-1)．組込み機器側サブシステムは，分散インテントで操作できる機能の情報を保持しており，Android 側サブシステムにそれら機能の情報をサービス通知プロトコルを用いて送信する．今回の実装では，SSDP (Simple Service Discovery Protocol)[8] を用いた．SSDP を用いた組込み機器機能の検知に関しては，3.3 節 で詳しく述べる．Android が通知メッセージを受け取り，送信元の組込み機器が分散インテントを利用可能であれば，機能情報を組込み機器側から取得する．さらに，分散インテントの利用を想定して実装された組込み機器以外も，分散インテントを通じて一元的に連携可能にするために，手動で組込み機器の機能を入力する機構も用意している．(図 1 I'-1)

組込み機器が分散インテントで操作可能であることをフレームワークが把握すると，機能情報をデータベースのデバイス情報テーブルに登録する (図 1 I-2)．

3.2.2 命令の送信

Android 内のアプリケーションがインテントを発行し，フレームワークが受信すると (図 1 II-1, II-2)，フレームワークはインテントの処理を行える組込み機器をデバイス情報テーブルから探し出す (図 1 II-3)．1 つ以上の組込み機器が処理を行える場合，組込み機器のリストを表示し，ユーザに選択を促す．

処理を実行させる組込み機器が決定すると，プロトコルマネージャがインテントをシリアライズする (図 1 II-4)．今回の実装では，インテントは JSON にシリアライズされるが，JSON 以外の場合でも，組込み機器で受け取れるプロトコルを調査し，プロトコルに合わせたシリアライズが行われる (図 1 II-5)．シリアライズが完了すると，フレームワークは組込み機器の操作を実行する (図 1 II-6)．

3.2.3 操作命令の受信と解析

組込み機器は Android からメッセージを受け取ると，格納されていた各情報をとり出し，どういう処理を行うべきかを判断し，該当機能を呼び出して，処理を実行する (図 1 II-6，II-7)．

3.3 SSDP を用いた Android 側サブシステムによる組込み機器機能の検知

SSDP にはデバイスが自身の機能を LAN 上の他のデバイスに通知するための NOTIFY メソッドと，LAN 上に望みの機能を持つデバイスが存在するかどうかの調査を行うための M-SEARCH メソッドが存在する．組込み機器側サブシステムは，NOTIFY メソッドを利用し，Android 側サブシステムに機能通知を行う．Android 側サブシステムは M-SEARCH メソッドを用いて LAN 上に存在する組込み機器の機能を調査する．Android 側サブシステムは，分散インテントを利用可能な組込み機器が見つけると，デバイス記述，サービス記述と呼ばれる情報を組込み機器に要求する．デバイス記述は組込み機器の構成についての記述，サービス記述は組込み機器が持つ機能の操作に必要な引数などについて記載されている．

これらの記述を仕様通りに利用すると，UPnP 機器なのか分散インテントを利用できる組込み機器なのかを区別できない．したがって，分散インテントではデバイス記述中の操作命令の送信先を記述する controlURL の部分を “/intent” からはじまるパスで記述し，分散インテントが利用可能かどうか判断する．

3.4 課　　題

現状のフレームワークは，仕様にあわせて組込み機器側サブシステムを実装する必要があり，開発者の負担が大きい．先に述べた設計にあわせて実装したフレームワークは実在するが，プロトタイプであり，評価に利用した機能のみに対応している．よって，新し

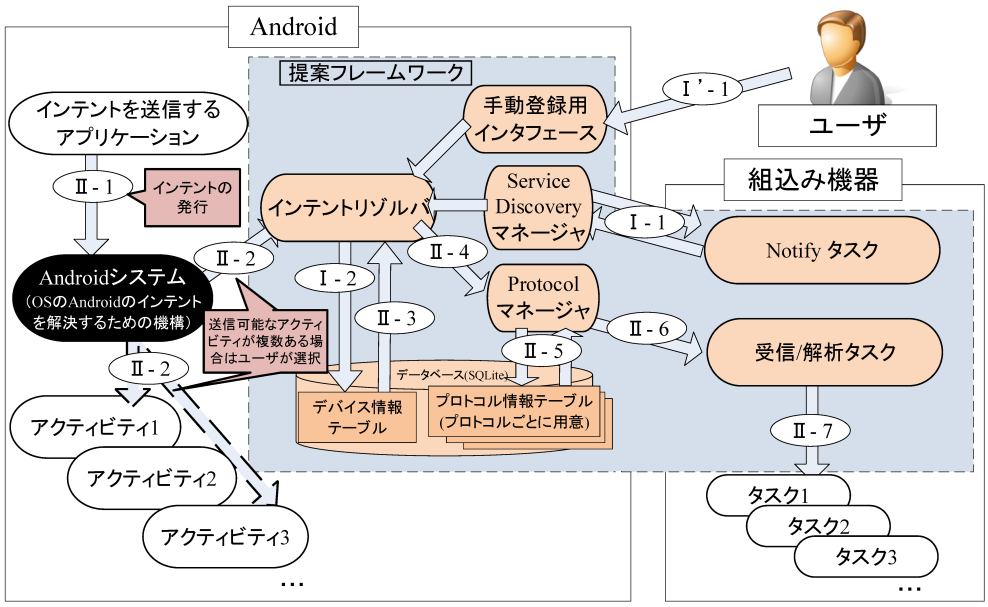

図 1 分散インテント設計図
Fig. 1 Design for a Distributed Intent Framework.

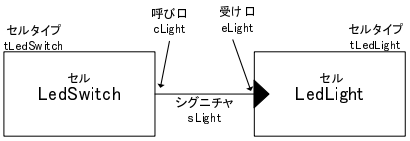

図 2 TECS コンポーネントモデル図
Fig. 2 Component Model for TECS.

く実装する組込み機器の機能にそのままでは適用できない．

4. TOPPERS Embedded Component System

本研究の設計，実装に利用した TECS (TOPPERS Embedded Component System)[2] について解説する．TECS は，大規模化する組込みシステムの開発の補助を目的とした，コンポーネントシステムである．従来のコンポーネントシステムと異なり，コンポーネントの静的な結合を行う特徴がある．コンポーネント通信時のオーバヘッドが無いため，リアルタイム性を確保できる特徴を持つ．

4.1 コンポーネントモデル

TECS コンポーネントモデル図を図 2 に示す．TECS コンポーネントは，セル（セルタイプ），シグニチャ，呼び口，受け口で構成される．

セルとは，TECS におけるコンポーネントの実体である．呼び口，受け口，属性，変数といった要素を持つ．**セルタイプ**とは，コンポーネントの型式を定義したものである．セルが持つ各要素の型やシグニチャは，セルタイプによって定義される．**シグニチャ**とは，関数ヘッダの集合体であり，セルとセルの間のインタフェースを規定するものである．シグニチャ自体も名前を持つ．**呼び口**とは，セルが他のセルの提供する機能を利用するための口である．呼び口を用いて，自らを利用することも可能である．**受け口**とは，セルが他のセルに機能を提供する口である．受け口にはシグニチャを定義し，呼び口が利用する関数群を対応付けする．

4.2 開発の流れ

TECS を利用した開発の流れを図 3 に示す．コンポーネント仕様開発者は，関数群の定義をするシグニチャ記述（図 3 (1)）およびコンポーネントの定義をするセルタイプ記述（図 3 (2)）を記述する．シグニチャ記述例を図 4，セルタイプ記述例を図 5 に示す．図 4 では，シグニチャ名を sLight とし，シグニチャの関数として on と off を定義している (図 4, 1-3 行目)．図 5 では，セルタイプ tLedLight は，先ほど定義した sLight を提供する受け口 eLight を定義している (図 5, 1-2 行目)．セルタイプ tLedSwitch は， sLight の呼び口 cLight を定義している (図 5, 5-6 行目)．

アプリケーション開発者は，コンポーネント仕様開

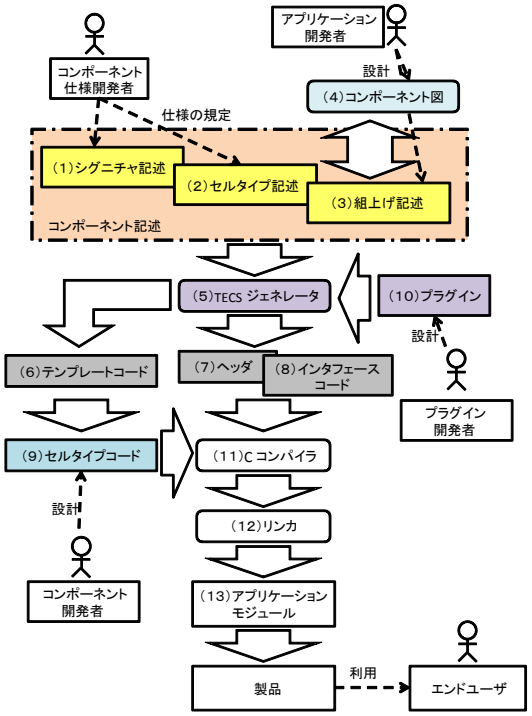

図 3 TECS による開発の流れ
Fig. 3 Development Flow on TECS

```
1 signature sLight {
2    void on( void );
3    void off( void );
4 };
```

図 4 Signature 記述
Fig. 4 Signature Description.

```
1 celltype tLedLight{
2     entry sLight eLight;
3 };
4
5 celltype tLedSwitch{
6     call sLight cLight;
7 };
```

図 5 Celltype 記述
Fig. 5 Celltype Description.

発者が生成したシグニチャ記述，セルタイプ記述をもとに，組上げ記述（図 3 (3)）を記述する．組上げ記述の実装には，コンポーネントモデル図（図 2）を用いてコンポーネントの設計を行うことが推奨されている（図 3 (4)）．組上げ記述の例を図 6 に示す．図 6 において，セル LedLight はセルタイプ tLedLight のインスタンスを表す (図 6, 1 行目)．LedSwitch セルは tLedSwitch セルタイプのインスタンスを表す (図 6, 4 行目)．sLight 型の呼び口 cLight を持つ LedSwitch

```
1 cell tLedLight LedLight {
2 };
3
4 cell tLedSwitch LedSwitch {
5     cLight = LedLight.eLight;
6 };
```

図 6 組上げ記述
Fig. 6 Build Description.

と，受け口 eLight を持つ LedLight とを結合する記述が書かれており，LedLight は LedSwitch の関数群を利用できる (図 6, 5 行目)．

TECS ジェネレータ（図 3 (5)）は，コンポーネント記述（シグニチャ記述，セルタイプ記述，および組上げ記述）をもとに，C 言語のテンプレートコード（図 3 (6)），ヘッダ（図 3 (7)），インタフェースコード（図 3 (8)）を生成する．テンプレートコードには，セルタイプが持つ関数の雛形が生成される．コンポーネント開発者は，コンポーネントの本体であるセルタイプコード（図 3 (9)）を，テンプレートコードをもとに実装する．

TECS のソフトウェア開発を容易にするために，TECS ジェネレータにはプラグイン（図 3 (10)）を利用できる．プラグインは，追加の機能を自動的に生成するために TECS ジェネレータに追加される．プラグイン開発者が Ruby を用いて開発する．プラグインを利用するためには，コンポーネント記述中にプラグイン記述を記述する．プラグイン記述により，TECS ジェネレータにプラグインが追加され，コードを自動的に生成できる．

最後に，C コンパイラ（図 3 (11)）およびリンカ（図 3 (12)）でコードをコンパイル・リンクし，アプリケーションモジュール（図 3 (13)）を生成する．

5. 要件とアプローチ

本章では，要件とアプローチを述べる．

5.1 要　　件

要件 1: 開発者の負担とならない設計

本研究では，分散インテントの実装を行うために，開発者の負担とならない方法で実装可能な設計にする必要がある．具体的には，開発者がインテントの通信機構の実装方法を知らなくても実装を可能とする．

要件 2: 分散インテントの要件を満たす設計

本研究では，組込み機器への適正を維持するため，分散インテント[1] の組込み機器側サブシステムの要件を満たす必要がある．組込み機器側サブシステムの要件を以下に整理する．

(1) Android から自動検出が可能であること.
(2) プラットフォームに依存していないこと.
(3) リソース制約の厳しい組込み機器へ対応できること.

5.2 アプローチ

本研究では，分散インテントの組込み機器側サブシステムを TECS ジェネレータで自動生成する分散インテントプラグインを設計・実装する．TECS におけるプラグインは，TECS ジェネレータから呼び出されて，通常開発者が実装するコードを自動的に生成するため，開発者の負担を軽減できる．なお，Android 側サブシステムに関しては，アプリケーションとして提供され，追加実装は不要である．よって，要件 1 を満たす.

分散インテントプラグインによって生成されるフレームワークは，分散インテントを踏襲する形で実装を行うため，サービス通知プロトコルをサポートする．よって，Android より自動検出が可能である．TECS のコンポーネントは多くの組込みソフトウェアで用いられている C 言語を用いて実装するため，特定のプラットフォームに依存しない．TECS はオブジェクトのリンク時に構成が決まる静的結合コンポーネントを採用しており，実行時に動的な結合を行わない．さらに，コンポーネント間結合の最適化が行われるため，メモリの使用量や実行時間のオーバヘッドを極力少なく抑えることが可能である．以上より，分散インテントの要件を満たすため，要件 2 を満たす.

6. 設　　計

本章では，分散インテントプラグインの設計について述べる.

6.1 語句の定義

設計の説明に利用する語句を定義する.

JSON インテント: インテントを JSON 形式にシリアライズしたものである．インテントは，そのままではネットワークを通じて送信できないため，分散インテントはプロトコルマネージャが組込み機器に合わせてインテントの変換を行うが，今回の実装では JSON 形式へ変換してインテントをやり取りする.

6.2 DistributedIntentPlugin

DistributedIntentPlugin(分散インテントプラグイン) は，分散インテントの，組込み機器側サブシステム (図 1 中「Notify タスク」および「受信/解析タスク」) を生成する TECS プラグインである.

6.2.1 設 計 方 針

本プラグインは，開発者が分散インテントを実装する負担の軽減を目的とする．具体的には，組込み機器ごとに実装を必要とする部分 (図 1 中，”Notify タスク”および”受信・解析タスク”) のコードを自動で生成する.

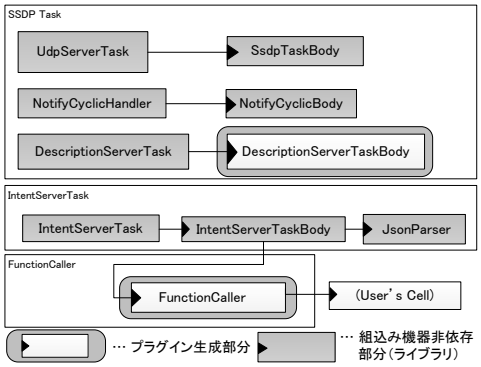

図 7 分散インテントのコンポーネント構成図
Fig. 7 Components for the Proposed Framework.

自動生成を行うには，TECS ジェネレータの実行前に，分散インテントの命令で実行する関数を，コンポーネント記述に書く必要がある．プラグインは関数の情報をもとに，組込み機器がどのような機能を持っているのかを SSDP で通知するメッセージと，インテント情報から関数を選択して呼び出すコードの 2 つを自動的に生成する．自動的に生成しないコードに関しては，組込み機器非依存のライブラリとして提供する.

本プラグインで生成されるソースコード，および組込み機器に非依存なライブラリは，TOPPERS/ASP カーネルと TINET 上で動作する．よって，特定のターゲットデバイスに対してライブラリが依存することなく，利用できる.

呼び出しの対象となる関数は，受信したインテントをグローバル変数で利用できる．したがって，関数呼び出しだけでなく，インテントのメッセージを用いた操作や引数を持ったインテントの利用も可能となる.

6.2.2 コンポーネント構成

本プラグインで生成，またはライブラリとして提供するコンポーネントの構成図を図 7 に示す.

UdpServerTask，**SsdpTaskBody**: SSDP メッセージを送信するタスクと動作するプログラム本体である．Android からの SSDP メッセージを受け取ると，組込み機器は自らの存在を通知する SSDP メッセージをブロードキャストする.

NotifyCyclicHandler，**NotifyCyclicBody**: SSDP メッセージを定期的にブロードキャストする周期タスクである．組込み機器は，自らの存在をネット

ワークに通知するため，周期的に自らの存在を通知するメッセージをブロードキャストする．周期タスクの実行間隔を設定することで，通知するタイミングを変更できる．

DescriptionServerTask，**DescriptionServerTaskBody**: 組込み機器が機能を紹介するメッセージである Description メッセージを送信するタスクである．Android がインテントで命令送信を行える組込み機器を見つけると，Android は組込み機器に対し Description メッセージを要求する．組込み機器は要求を受け取ると，Description メッセージを Android に送信する．

IntentServerTask，**IntentServerTaskBody**: Android から JSON インテントの受信を待機するタスクである．

JsonParser: Android から受信した JSON インテントをデシリアライズするタスクである．

FunctionCaller: Android から受信した JSON インテントの命令に対応した関数を実行するタスクである．

6.2.3 コンポーネント記述の変換

プラグインは，TECS のコンポーネント記述より情報を取得し，コード生成に利用する．生成する情報とセル情報の対応付けを図 8 に示す．生成する情報と対応するセル情報，配列名は以下のとおりである．各要素と配列番号を結びつけることで統一し，配列番号で同一関数の情報であることを判断している．

product_name_list[]: フレームワーク中で利用する表示名を，プラグイン呼び出し記述中の第 2 引数に指定する．引数を指定しない場合，セル名が指定される．

signature_list[], call_port_list[], entity_list[], function_list[]: 指定した関数を呼び出すため，FunctionCaller のインタフェース宣言を行う必要がある．よって，本プラグインは指定したプラグインを呼び出したセルの signature 名，call port 名，実体名，関数名を収集して利用する．

action_list[]: 分散インテントでは，インテントに書かれた命令を，組込み機器中のルーチンに変換する機構が必要である．本プラグインでは，Action 名と関数名を結びつけるために Action 名を生成する．Action 名は，接頭語に「DI_」を付け，関数名を利用する．

6.2.4 プラグインの処理の流れ

プラグインの処理の流れについて説明する．

図 9 は，DistributedIntentPlugin が持つメソッド

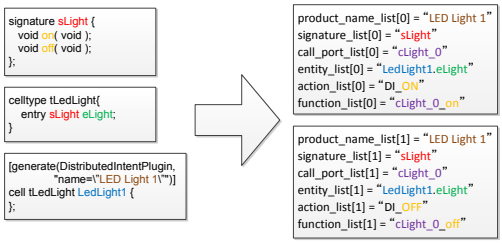

図 8 コンポーネント記述と生成情報の対応
Fig. 8 Handling from Component Description to the Material for DistributedIntentPlugin.

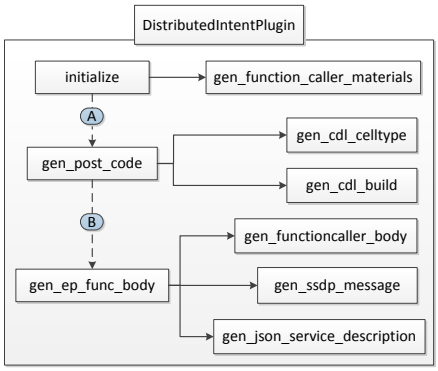

図 9 プラグインの処理の流れ
Fig. 9 Flow of the DistributedIntentPlugin.

の実行順序を示したプラグインの処理の流れである．プラグインが呼び出されると，initialize メソッドが実行される．DistributedIntentPlugin の記述が複数ある場合は，initialize メソッドは複数回実行される．プラグインの呼び出しをすべて確認すると，プラグインは処理 A および処理 B を開始する．各メソッドの説明を以下に示す．

initialize(cell, option): プラグインを実行したとき最初に呼び出される．引数の”cell”は，セルの名前，セルタイプ，セルタイプが持つ受け口・呼び口の情報を持つオブジェクトである．引数の取得後，get_function_caller_materials を呼び出す．

get_function_caller_materials(cell): FunctionCaller で関数，セルタイプ，セルを作成する際に埋め込んで利用する文字列を生成する．

gen_post_code(file): FunctionCaller のコンポーネント記述を作成するメソッドであり，gen_cdl_celltype(file) と gen_cdl_build(file) を順に呼び出す．

gen_cdl_celltype(file): コンポーネント記述中の，セルタイプ記述を行うメソッドである．Function-

表 1 Galaxy Nexus のスペック
Table 1 Specification of Galaxy Nexus.

項目	性能
Android バージョン	4.1 (Jelly Bean)
CPU	1.2 GHz dual-core ARM Cortex-A9
メモリ	1 GB
ストレージ	16GB

表 2 AKI-H8/3069F のスペック
Table 2 Specification of AKI-H8/3069F.

項目	性能
CPU	H8/3069RF 20MHz
Flash ROM	512KB
RAM	16KB
DRAM	16Mbit

Caller が各機能を呼び出すための呼び口記述を生成する．

gen_cdl_build(file): コンポーネント記述中の，組み上げ記述を行うメソッドである．FuncitonCaller と，呼び出される各受け口とを結合する記述を生成する．

gen_ep_func_body(file): FunctionCaller の C 言語ソースコードを生成するメソッドである．gen_functioncaller_body(file)，gen_ssdp_message( )，gen_json_service_descripton( ) を順に呼び出す．

gen_functioncaller_body(file): FunctionCaller 関数本体を生成する．FunctionCaller は，インテントを C 言語の関数呼び出しに変換する．

gen_ssdp_message( ): SSDP のメッセージを生成する．

gen_json_service_descripton( ): 組込み機器の機能説明記述を生成する．

7. 評　　価

本章では，実装したプラグインの評価結果を示す．

7.1 実 験 環 境

実験に利用した Android 機器は Galaxy Nexus である．マイコンボードは秋月電子の「AKI-H8/3069F フラッシュマイコン LAN ボード」を使用した．Android 機器のスペックを表 1，マイコンボードのスペックを表 2 に示す．

7.2 プラグインの評価

本節では，従来の分散インテントフレームワークと比較評価を行い，その結果を示す．評価には，無線 LAN アクセスポイントを利用して，Android 端末およびマイコンボードの間にネットワークを構築した．従来の分散インテントの評価には，「LED を点灯する」「LED を消灯する」の 2 種類の機能を実装したプロトタイプを用いた．さらに，プラグインで同等の機能を持つフレームワークを生成し，ソースコードの実装行数，ROM・RAM 使用量とオーバヘッドの比較を行った．

表 3 ライブラリ部分を除いた実装行数
Table 3 Implementation lines without the library.

項目	従来	プラグイン
SSDP メッセージ (行)	$87+\alpha$	0
機能呼び出し (行)	$7+\beta$	0
コンポーネント記述 (行)	0	1
合計 (行)	$94+\alpha+\beta$	1

表 5 インテントの受信から機能実行までの実行時間
Table 5 Execution Time of the Proposed Framework for an Embedded System.

項目	従来	プラグイン
平均 (ms)	1.61	1.58
最大時間 (ms)	1.87	1.86
最少時間 (ms)	1.48	1.47

7.2.1 実装行数による比較

実装行数による比較を行った結果を表 3 に示す．本研究で設計・実装したプラグインは，プラグインを呼び出す記述を入力するだけで，必要なファイルが生成されるため，1 行のソースコード記述で利用できる．一方，プラグインが自動生成する部分と同等のソースコードを開発者が実装する場合，SSDP メッセージの行数が 87 行＋α 行，機能呼び出しに必要な行数が 7+β 行となる．さらに，従来の実装では分散インテントの仕様や SSDP メッセージの仕様を開発者が理解する必要がある．したがって，プラグインは開発者の負担を軽減できる．

7.2.2 ROM・RAM 使用量

従来のプロトタイプの実行ファイルと，本プラグインで生成したフレームワークの実行ファイルを比較し，組込み機器への適正を評価した．結果を表 4 に示す．

ROM の使用量は増加しているが， 0.8% 程度の増加にとどまっている．RAM 使用量はほぼ同一であることから，フレームワークが動作中のリソース使用量に変わりはないことがわかる．したがって，従来のフレームワークと同等のリソース使用量で，フレームワークが実現できる．

7.2.3 実行時間比較

組込み機器がインテントを受信し，JSON をパースして機能を実行するまでの実行時間を 100 回ずつ測定した．結果を表 5 に示す．

従来のフレームワークの平均が 1.61 ミリ秒，プラ

表 4 フレームワークの ROM・RAM 使用量
Table 4 ROM and RAM Usage of the Proposed Framework for an Embedded System.

項目	従来手法 (ROM)	従来手法 (RAM)	提案プラグイン (ROM)	提案プラグイン (RAM)
SSDP (Byte)	5,968	7,914	6,518	8,278
Intent Server (Byte)	1,134	472	1,332	528
JSON Parser (Byte)	2,594	320	2,606	320
実行可能ファイル (Byte)	90,788	39,563	91,528	39,515

グインの平均が 1.58 ミリ秒となった．最も長い処理時間，最も短い処理時間を比較してみても，0.01 ミリ秒程度の差となり，大きな変化を確認することはできなかった．よって，組込み機器への適正は維持されている．

以上の結果により，分散インテントの要件を維持したまま，コードが生成できている．

8. ま と め

本論文では，Android を利用して，ネットワーク上にある機器と連携を行うために，組込みシステムが連携するフレームワークを自動生成するプラグインを提案した．はじめに，従来の関連研究について解説し，それらの抱える問題について述べた．つぎに，我々が提案する分散インテントフレームワークの概要について述べ，分散インテントの開発による開発者への負担が増加する問題について述べた．そして，問題を解決するための要件と，要件を達成するためのアプローチについて述べた．

アプローチを実現するために，インテントを分散拡張するフレームワークの組込みシステム部分を，組込み機器の機能に合わせて生成するプラグインを提案した．提案したアプローチを実現するために，プラグインを設計，実装して性能の評価を行った．結果，プラグインは分散インテントの要件を維持したまま組込みシステム部分のフレームワークを生成でき，なおかつプラグインによる生成によって開発者の負担を軽減できることを確認した．本論文では，評価には LED を点灯させる簡易な例を用いたが，LED の操作はフレームワークの外で行われる処理であり，フレームワークの実行時間に対して直接影響を与えない部分である．したがって，さらに高機能な処理が行われる場合でも，本フレームワークは適用可能である．ただし，本来インテントはローカルで実行されるため，実行までの時間については考慮の必要がなかったが，インテントを分散化したことにより，機能の実行までに時間がかかる可能性がある．したがって，開発者は組込み機器で機能が実行されるまでの時間差について，考慮する必要がある．

今後は，組込み機器の連携手法を充実させたり，ファイル送信を分散インテント内でサポートしたりすることで，分散インテントで利用できる機器の多様性についても追求する必要があると考える．

参 考 文 献

1) 伊藤孝宏, 長原裕希, 安積卓也, 西尾信彦: 分散インテント: Android アプリケーション間協調の分散拡張フレームワーク, マルチメディア，分散協調とモバイルシンポジウム 2012 論文集, Vol. 2012, pp. 1591–1600 (2012).
2) Azumi, T., Yamamoto, M., Kominami, Y., Takagi, N., Oyama, H. and Takada, H.: A new specification of software components for embedded systems, *Object and Component-Oriented Real-Time Distributed Computing, 2007. ISORC'07. 10th IEEE International Symposium on*, IEEE, pp. 46–50 (2007).
3) Vinoski, S.: CORBA: Integrating diverse applications within distributed heterogeneous environments, *Communications Magazine, IEEE*, Vol. 35, No. 2, pp. 46–55 (1997).
4) 清川皓太, 山本眞也, 柴田直樹, 安本慶一, 伊藤実: 3D 仮想空間を用いた情報家電のためのリモコンフレームワーク, 情報処理学会論文誌, Vol. 52, No. 2, pp. 596–609 (2011).
5) UPnP Forum: UPnP Device Architecture Version 1.0, Online, `http://upnp.org/` (2011).
6) Kawaguchi, N.: Cogma: A middleware for cooperative smart appliances for ad hoc environment, Proc. of 1st Int ' l Conf. on Mobile Computing and Ubiquitous Networking (ICMU2004), pp. 146–151 (2004).
7) 伊藤昌毅, 橘爪克弥, 河田恭兵, 生天目直哉, 伊藤友隆, 井村和博, 西條晃平, 中澤仁, 高汐一紀, 徳田英幸: Chameleon : 多様な状況下の機器指定を実現する複数インタラクション統合技術, 情報処理学会論文誌, Vol. 52, No. 4, pp. 1571–1585 (2011).
8) GOLAND Y. Y.: Simple Service Discovery Protocol/1.0 Operating without on Arbiter, `http://ci.nii.ac.jp/naid/10016572766/` (1999).