千葉工業大学
博士学位論文

IoT アプリケーションを開発するための エージェントプラットフォームに関する研究

平成 30 年 3 月

顧　優輝

指導教員　菅原　研次

論文要旨

Kevin Ashton 等により提唱されたモノのインターネット (Internet-of-Things/IoT) は，センサやアクチュエータ技術の発展とコストの低減化，スマートフォンの爆発的普及，組み込み型コンピュータの性能の大幅な向上と低コスト化などが進み，社会を支える重要な基盤技術として注目を浴びている．

モノに対してセンサやアクチュエータ等のデバイスを内蔵及び外付けし，インターネットにより現実空間とクラウドを接続することで，ユーザの必要とするサービスを提供するシステムのことを IoT アプリケーション（以下，IoT-App）と呼ぶ．IoT-App は，物理空間や人の社会と密接に関係するため，それぞれの物理空間の特性や社会の特性に合わせて設計することが必要であり，さらに環境や状況の変化に合わせて動作条件が変化する．また，クラウドにおける大規模データの高度な処理から物理空間のデバイスの制御などの幅広い分野の技術を組み合わせるので，様々な機能や特性が異なる多数の要素の相互作用により目的とするサービスが実現される．しかしながら，これまでのシステム開発方法論により，このような環境の変化に対処するシステムの開発する場合，開発・運用者に大きな負担がかかることが予想されているため，IoT-App のための新たな開発技術が必要となっている．

そこで，本研究では，環境や状況の変化に対応することが可能な IoT-App のためのエージェント指向開発方法論を提案し，これに基づいたエージェントプラットフォームを開発することを目的とする．次にこれを利用して，オフィスワーク支援のための IoT-App を試作し評価実験を行い，提案した開発方法論の妥当性の評価を行う．

まず，クラウドのデータの分析などを行うプロセスと物理空間のデバイスの制御プログラムなどの異なる要素の相互作用を柔軟に行うために，IoT-App のエージェント指向開発方法の検討を行った．IoT-App を構成する要素に対して，エージェントの性質を付加する操作をエージェント化と呼ぶ．デバイス，データおよびプログラムをエージェント化するために，本研究ではリソースコネクタと呼ぶエージェントモデルの提案を行った．リソースコネクタを用いてデバイスをエージェント化することにより，クラウドのプログラムとデバイスの制御プログラムを独立に設置することができ，これらがエージェントメッセージにより協調を行うことにより，アプリケーションプログラムからデバイスの機能を動的に呼び出すことが可能になった．これにより，状況の変化や要求の変化に対して，アプリケーションがデバイスを選択して利用することが可能になった．

次に，リソースコネクタのモデルに基づいてデバイス，データおよびプログラムをエージェント化するためのツールとして，リソースコネクタを設計・実装・運用するためのエージェントプラットフォームを開発した．これにより，IoT-App の要素をエージェントとして実装し，動的に協調を行うことができる基盤を実現した．

そして，リソースコネクタモデルに基づいて，IoT-App を開発するための構成方法の明確化を行った．この構成方法は，エージェント化される要素間の通信回線（これをチャネルと呼ぶ）とエージェント同士の通信回線を独立に構成できることに特徴がある．このことにより，IoT-App の機能や性能をエージェント間の協調により動的に変更することが可能になり，IoT-App に必要な環境や状況の変化に適応する仕組みが実現できる．

最後に，上述のエージェントプラットフォームの実験による評価を行うために，テストベッドとしてオフィスワーク支援の分野を選択した．オフィスワーク支援の典型的な例として，マルチメディアの要素がより重要視される遠隔地でのブレインストーミング会議の支援に焦点を当てた．実験の結果，クラウドの要素と物理空間のカメラなどのデバイスの要素が動的に連携することを確認した．また，チャネルの概念に基づき，同期的・非同期的なチャネルを状況や環境に応じて適応的に追加・削除することを可能とし，スケーラブルなアーキテクチャの実現が確認できた．そして，チャネル間の連携を行うことで資産化したブレインストーミングの成果の再利用が可能である事を確認した．

以上により，本研究のエージェント指向開発方法論に従ったエージェントプラットフォームは，IoT-App を設計し実装するために有効であることを明らかにした．これまでの開発方法論では，変化する環境の中で多数の要素が複雑な相互作用を行うシステムを記述するためには，設計者に大きな負担を伴っていた．これに対して，提案した開発方法論では，要素の機能や利用法を内部データとして用い，メッセージにより相互調整するエージェントのモデルを採用することにより，クラウドの要素とデバイスの要素の連携を自律的に行わせることが可能となる．インダストリー 4.0 やインダストリアルインターネットなどのように，産業基盤や日常生活の基盤に IoT-App が浸透していくことが予想されているが，本研究で提案した方法論は，IoT アプリケーションの効率的な開発に貢献する．

Abstract

Internet-of-Things (IoT) proposed by Kevin Ashton et al. is attracting attention as an important generic technology supporting activities in society, because of development and cost reduction in sensors and actuator technologies, explosive spread of smart-phones and drastic improvements in embedded computer performance.

The IoT can connect the real space and the cloud space using the Internet by embedding devices to objects. The IoT application (IoT-App) is a system that provides necessary services for people using IoT technology. As IoT applications are closely related to change on the environment and situation, it is necessary to design them based on their characteristics. It combines technologies having different functions and characteristics, including advanced processing of large-scale data in the cloud and control of devices in physical space. However, it is expected that developing a system to cope with such environmental changes using the existing system development methodology will be a heavy burden on development and operators. Therefore, new development technology is required for IoT-App.

In this paper, an agent-oriented development methodology for IoT-App that can cope with changes in the environment and situation is proposed, and an agent platform based on this methodology is developed to support developers. Next, using this platform, IoT-App for supporting office work using this agent platform are developed to evaluate the validity of the proposed development methodology by conducting the evaluation experiment.

First, in order to flexibly interact with different elements such as process of analyzing cloud data etc. and device control program of physical space, the agent-oriented development methodology of IoT-App is examined. The addition of the property of the agent to the elements constituting the IoT-App is called agentification. In order to agentify devices, data and programs, an agent model called resource connector is proposed. Cloud programs and device control programs can be installed independently by agentifing the devices using the resource connectors. It became possible for them to collaborate by agent messages and to dynamically call the function of the device from the application program. Therefore, it becomes possible for the application to select and use the device according to the change of the situation and the request.

Next, an agent platform for designing, implementing and operating resource connectors as a tool is developed based on the resource connector model. As a result, IoT-App elements were implemented as agents, and they realized a dynamically cooperative infrastructure.

And a configuration method for developing IoT-App based on resource connector model is formalized. In this configuration method, a communication line (referred to as a channel) between elements to be agentified and a communication line between agents can be configured independently. Therefore, it becomes possible to dynamically change the function and performance of IoT-App by collaboration among agents, and it is possible to realize a mechanism required for IoT-App adapting to changes in environment and situation.

Finally, the feasibility of this agent platform with a test bed is validated using a test bed of a meeting support service in office work. As a result of the experiment, it was confirmed that the elements of the cloud and the devices of the physical space can dynamically cooperate.

In conclusion, it was confirmed that the agent platform using agent-oriented development methodology of this research is effective for designing and implementing IoT-App.

目次

図目次

表目次

第 1 章

序論

1.1 ユビキタスコンピューティングと IoT

ユビキタスコンピューティング（Ubiquitous Computing）とは，社会や生活のいたるところにコンピュータが存在し，いつでもどこでも使える状態であることを表す概念のことである [76]．ユビキタスコンピューティングは．1990 年代にマーク・ワイザー博士により提唱され，コンピュータネットワークやユーザインタフェースの分野を含めた広い分野の研究者や企業から注目を浴びた．しかしながら，インタフェースデバイスやネットワーク技術のコストや性能などの問題により，アプリケーションの実用化や普及の進展には至っていない．

一方，1999 年に Kevin Ashton 等により提唱されたモノのインターネット（Internet-of-Things）[4] は，限定された領域ではあるが，流通分野の電子タグ技術とインターネットを用いた製品の追跡など，一定の利用が行われてきた．

2010 年前後には，センサやアクチュエータ技術の発展とコストの低減化，スマートフォンの爆発的普及，Raspberry Pi などの組み込み型コンピュータの性能の大幅な向上と低コスト化などの現象が起きた．同時に，インターネットの高速化や社会への浸透が進み，無線ネットワークの普及が進むことにより，家電，車両，運輸，流通，工場などの製造システムなどの情報がインターネットに直接つながるための技術が成熟してきている [73, 29]．

この状況を受けて，米国ではインダストリアルインターネットやサイバーフィジカルシステムなどの産業や日常生活を巻き込む技術の提案と開発がすすめられてきた．ヨーロッパでは，インダストリー 4.0 と呼ばれる製造業を中心とした製造設備や製品の情報をインターネットにつなぎ，動的に製造ラインを管理する技術の開発と産業の育成が，国家の枠組みとして取り組まれている [31]．

ユビキタスコンピューティングも IoT も，インターネットと物理空間のデバイスに関連した技術分野としては同じような分野ととらえることもできる．ただし，ユビキタスコンピューティングが，科学技術の理論的な側面からインターネットやデバイスの研究を行っていることに対して，IoT は，社会や産業の側面からインターネットやデバイスの技術をとらえるという違いがある．

本研究では，より実践的な対象として新たなデバイス等を利用したシステムアーキテクチャを実現するために，IoT のコンテキストにおいて研究を行うこととする，

1.2 IoT の研究動向

Internet-of-Things(IoT) とは，図 1.1 に示すように，モノに対してセンサやアクチュエータ等のデバイスを内蔵及び外付けすることでモノを介した現実空間のセンシング及び制御を可能とし，それらをインターネット

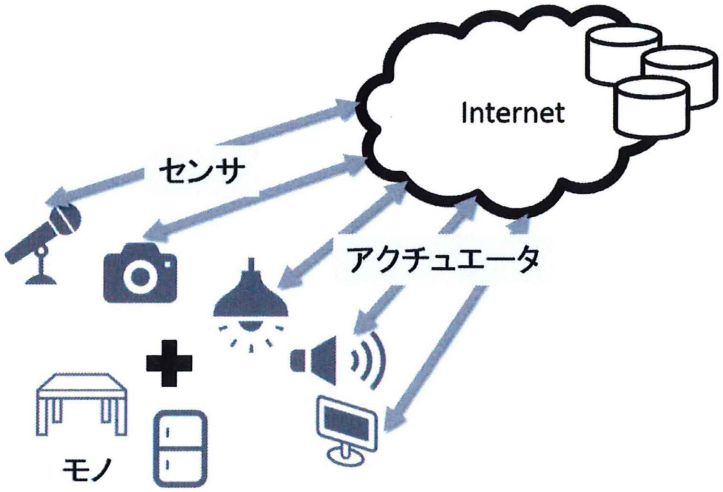


図 1.1　Internet-of-Things(IoT) の構造

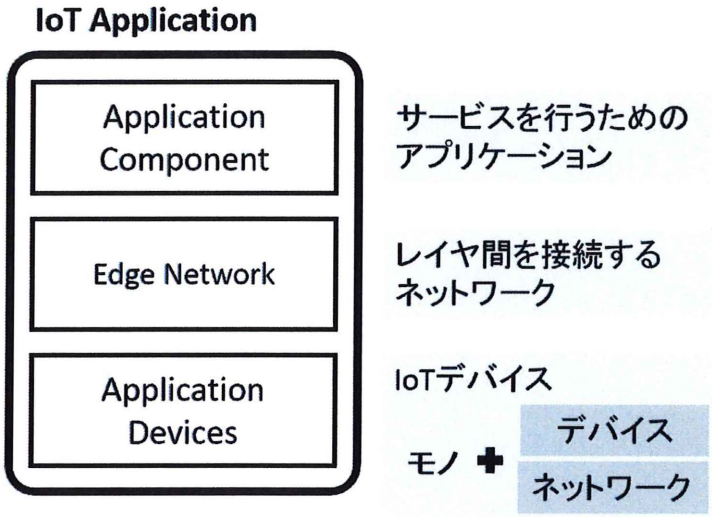


図 1.2　IoT アーキテクチャのレイヤ構造

に接続することで単体のモノではリソースの問題でなし得なかったシステムを実現する為の枠組みのことである [73, 29].

現在，考案されている IoT アーキテクチャは，多くが 3 層モデルをベースとしたものであり，図 1.2 のような形となっている [2]．3 層モデルは Application Component Layer(Application Layer)，Edge Network Layer(Network Layer)，そして，Application Devices Layer(Perception Layer) で構成される．

IoT アーキテクチャの Perception Layer には，IoT デバイスが存在し，それによりモノをインターネットに接続可能としている．IoT デバイスとは，モノに対してセンサ及びアクチュエータを付与し，それから得られるデータを処理する機構と処理後のデータを IP ネットワークによる通信部分を利用して Application Layer のアプリケーションとやり取りを行う構造を持つ．

これらの構造の IoT のアーキテクチャを実現するために，様々な IoT プラットフォームの開発が行われてい

る [48]．また，これらの IoT アプリケーションのためのプラットフォームに関する仕組みは発展途上であるため，様々なミドルウェアが提案され，研究されている [60]．標準のプラットフォームは存在しないものの，これらミドルウェア等のプラットフォームのおかげで，IoT アプリケーションのデバイス利用の相互運用性の確保がある程度のレベルにおいて可能となっている．

その中でも，Razzaque らのミドルウェアのサーベイの中では，IoT アプリケーションを実現する為のエージェント型プラットフォームが紹介されており，エージェント型とすることでより有効な相互運用性が確保可能であると考えらえる．また，適応的な動作がプラットフォーム上で可能であり，ネットワークリソース管理，各エージェントの非同期および自律実行，可用性と信頼性による分散処理の実現などが達成できるとされている [60]．

しかしながら，これらのミドルウェアにおいても IoT アーキテクチャの問題点としてこれらの IoT デバイス依存で IoT アプリケーションが実現されているので，IoT デバイスの変更や拡張に対する Flexibility が足りないとされており，内部に組み込まれたアプリケーションが更新することすらできないという問題点が指摘されている [65, 58]．これは，IoT デバイス上において動作するアプリケーションがデバイスに依存した形で実装されており，ラッパーアプローチによって組み込まれた状態で実装されているからであると考えらえる．

また，IoT-A と呼ばれるヨーロッパでの標準化の動き等も存在し，様々な国において研究が盛んである [43, 63]．さらに，IoT を利用してソーシャルネットワーキングの概念を IoT に統合するための研究も行われており，モノから得られる情報により社会的関係を含んだ情報を生成する Social IoT と呼ばれるアーキテクチャも存在する [5]．

そして，これらの研究における IoT の枠組みを一般向けに提供するために多くの IoT プラットフォームが提供されている．例えば，Digi の XBee ZigBee Cloud Kit[*1]や Microsoft の Azure IoT[*2]，Amazon の AWS IoT[*3]，そして IBM の IBM IoT[*4]等が存在する．

しかしながら，これらの IoT の仕組みを実現するためには，Network Layer の存在からわかる通り，IoT デバイスが IP ネットワークへの接続機能を持つことが必要不可欠であり，多量の IoT デバイスが効率的なネットワークの接続状態でセキュアな状態で接続する必要が有る．

さらに，IoT では，IoT アプリケーションにおいて扱われるデータは多種多様であるため，マルチメディアの要素を扱う事も重要な課題である．しかしながら，マルチメディアの要素を扱うためには，様々な問題も存在する [3, 14]．Alvi らによれば，IoT に関するシステムをマルチメディアに対応させる場合の問題点として，“マルチメディアベースのサービスは，ユーザの好み，デバイスの能力，QoS，および QoE に従って，重要な構成およびカスタマイズを必要とする” としており，また，“異なるユーザ装置およびネットワーク特性に適応することができるマルチメディア適応機構を考案する必要がある” としている [3]．つまり，IoT におけるマルチメディアの要素を実現する為には，各層において実現されるサービスが状況に応じて構成できる必要が有り，特に各 IoT デバイスにおけるネットワークを必要に応じて適応的に構成可能な仕組みを現状のプラットフォームにおいて実現する事ができていないと考えらえる．

これは，各層が異なるインフラストラクチャ上で実装されており，制御を行うためのインタフェースもそれぞれ異なるため，層に存在する要素が動的に連携する事が難しいためであると考えられる．この全ての層の要素を連携することを垂直連携と呼ぶ．動的な垂直連携を行うためには，図 1.3 に示すような全ての層を連携する事が可能な仕組みが必要である．

[*1] https://www.digi.com/blog/community/xbee-tech-tip-connecting-to-the-iot-with-xbee-zigbee-cloud-kit/

[*2] https://docs.microsoft.com/ja-jp/azure/iot-hub/iot-hub-device-management-overview

[*3] https://aws.amazon.com/jp/iot/

[*4] https://console.bluemix.net/catalog/?category=iot

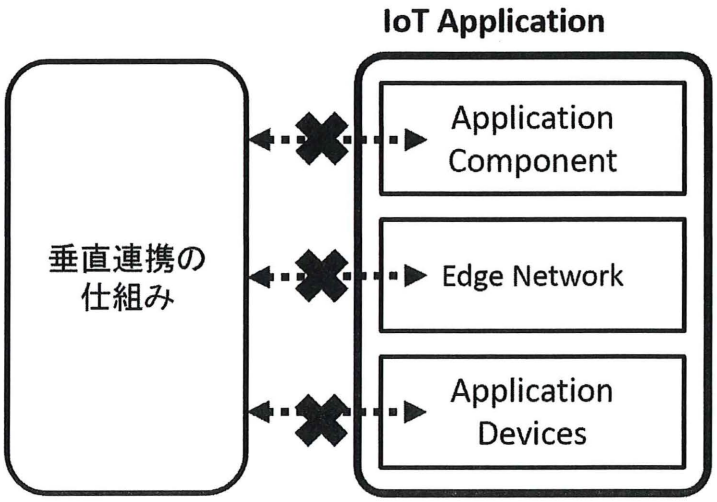


図 1.3　垂直連携のアーキテクチャ

そこで，本研究では，この動的な垂直連携の仕組みを実現するために，相互運用性と適応的な動作が可能な前述のエージェントの仕組みが有効であるのではないかと考えた．しかしながら，既存のエージェント型 IoT プラットフォームにおいては，上述のラッパーアプローチの問題点が存在する．また，主に IoT デバイスに対してエージェントから扱う仕組みは存在するものの，IoT アプリケーションの各層に対して全てに対応可能なプラットフォームは存在していないという問題点が存在する．このエージェントから要素を扱う事を可能とするための行為をエージェント化と呼び，ラッパーアプローチではエージェントに対してモノに付随するデバイスの制御プログラムを組み込むことを指す．

また，Cisco によれば，2020 年にはインターネットに接続された IoT デバイスが 500 億個以上となると推計されており [24]，IoT プラットフォームが IoT デバイスや層のコンポーネントやプログラムなどの要素が多量に存在する大規模な IoT アプリケーションに対応する必要が有る．

1.3　エージェント指向システム開発技術の動向

システム開発技術は，1980 年前後に提案されたオブジェクト指向技術 [27] により，質的にも量的にも急速な発展を見せている．システムの要素をオブジェクトにより抽象化し，オブジェクト間の相互作用をメッセージ通信でモデル化することにより，大型のシステムを短期間で効率的に開発することに多大な貢献を行っている．しかしながら，オブジェクトは，何らかの仕組みにより制御されることが必要で，あらかじめ設定された条件の中で，設計された動作を行っている点に，システム開発方法論としての限界があるといわれている．

エージェントの定義は，自律性，反応性，協調性，自発性のいずれか，あるいはそれらのすべての性質を持ったシステムとして定義されている．これらのエージェントの集団が，相互に関係しながら，定められた目標を達成するシステムを，マルチエージェントシステムという．エージェントは，外部環境を認識し，その変化に対して反応する．反応の仕方は，エージェント内部のなんらかの知識を利用するが，自分の中に無い知識や機能は，他のエージェントと協調することにより，マルチエージェントシステムとして，より柔軟な処理を行うことが期待できる．

以上のような枠組みを用いて，オブジェクト指向システム開発方法論の限界を超えて，外界の変動に自発的に対処する機能を実現することが，エージェント指向システム開発方法論の狙いになっている [80].

エージェント指向開発方法論を踏まえて，エージェントのモデルとそれに基づいた開発支援機能と運用環境を提供するシステムをエージェントプラットフォームという．エージェントプラットフォームとして，JADE[13]，OMAS[9], DASH[88]，JASON[16] などが開発されている．また，移動エージェントやノマディックエージェント [85] が開発されている．

本研究では，エージェント指向開発方法論によって大規模な IoT アプリケーションを開発するためのプラットフォームを実現する為に，1.2 節で示したエージェント型 IoT アプリケーションの問題点を解決し，動的に垂直連携可能な大規模 IoT アプリケーションのエージェント指向開発方法論の提案を行う．

1.4　本研究の対象とする問題点

本研究では，大規模 IoT アプリケーションのエージェント指向開発方法論の提案を行うために，以下の問題点を対象として，それぞれの問題点の解決を図ることを目的とする．

1.4.1　(A) IoT アプリケーションのエージェント指向開発方法論

IoT アプリケーションは 3 階層のアーキテクチャに従う．これらの各層は，機能や特性が異なる要素から構成される．したがって，エージェント指向開発方法論は，各層の特性に基づいたエージェントのモデルを必要とする．ただし，それらのエージェントは，共通のプロトコルで，協調可能であることが必要である．しかしながら，これらの条件を満足するエージェントモデルを持つエージェント指向開発方法論が存在しない．

1.4.2　(B) IoT アプリケーションのエージェント指向開発方法論に基づくプラットフォーム

IoT アプリケーションのアーキテクチャの各階層のエージェントは，異なるインフラストラクチャの上で動作することが必要である．たとえば，クラウドのエージェントプラットフォームは，アプリケーションの論理動作を高速に処理し，クラウド内の通信プロトコルで効率的にメッセージ送信を行う．ネットワークを構成する要素を実現するエージェントは，大量のストリームを送受信したり，多数のエージェント間の安定したメッセージの総配信を行う機能を持つ必要がある．デバイスを制御するエージェントは，直接物理デバイスを制御するための仕組みを必要とする．このように，IoT アプリケーションの各階層に対応して異なる特性を持つエージェントを実装するためのプラットフォームが存在しない．

1.4.3　(C) IoT アプリケーションの垂直連携型アーキテクチャ

IoT アプリケーションシステムは，クラウドアプリケーション，ネットワーク，デバイスの 3 階層のアーキテクチャと考えられる．ネットワークは，クラウドのインフラストラクチャ，インターネット，デバイスとインターネットを接続する無線ネットワークから構成される．デバイスは，カメラなどのセンサ，ディスプレイなどのアクチュエータおよび移動端末などから構成される．

IoT アプリケーションの各階層はたくさんの要素から構成される複雑なシステムであり，これらが連携して目的とするサービスを実現する．このような階層間の要素の連携を垂直連携と呼ぶ．IoT アプリケーションにおける垂直連携は，サービス利用者の位置や状況で要素が変化するため，的確なサービスを実現するためには，動的に垂直連携を構成することが必要であるが，このための効率的な仕組みが存在しないことが，IoT アプリケーション開発の方法論の課題である．

1.4.4 (D) スマートオフィスの実現に向けたテストベッド

オフィス業務は，書類や資料の作成，会議などを繰り返し行い，それぞれの作業の成果を共有したりするワークフローにより実現されている．これらを支援するためには，高度に連携するマルチメディアシステムを実現することが必要である．しかしながらオフィス業務支援ツールは，個別に動作するツールの寄せ集めであり，これをワークフローの各段階に合わせて利用するためには，各ツールの調整が利用者の大きな負担になる．この負担を軽減するための IoT アプリケーションの構成法が課題である．

1.5 本研究の目的

本論文では，適応的に動作が可能で分散処理にも優れるエージェント型 IoT プラットフォームを対象として，IoT デバイスの問題点を解決するために情報リソースとそれらをアプリケーションによって扱うことを可能とするための仕組みであるリソースコネクタを提案し，IoT のためのエージェント型プラットフォームの問題点を解決する．また，マルチメディアチャネルという概念を提案することでネットワークを適応的に構成可能な仕組みを提案し，IoT におけるマルチメディアの要素の実現を図る．さらに，これらの実装により評価及び検証することで実現可能性を示し，これらを問題点の解決することを目的とする．本論文において，ユビキタス型システムは IoT アプリケーションの実装のために存在するものとして扱い，ユビキタスシステムの問題点を IoT での問題として捉えることとする．

1.6 本論文の構成

本論文の構成を以下に述べる．まず，2 章では，Internet-of-Things とその問題点，そして，各章の関連研究について述べる．次に，3 章では，IoT アプリケーションのエージェント指向開発方法論の課題を解決するために，IoT アプリケーションを開発するためのエージェントプラットフォームの提案を行う．そして，4 章では，3 章において提案したプラットフォームの実現を確認するために，IoT アプリケーションを開発するためのエージェントプラットフォームの設計と実装を行う．さらに，5 章では，IoT アプリケーションの垂直連携型アーキテクチャを実現する為にエージェントプラットフォームを利用した IoT アプリケーションの設計方法の提案を行う．また，6 章では，スマートオフィスの実現に向けたテストベッドを対象として，エージェントプラットフォームを利用したブレインストーミング支援システムの設計を行う．さらに，7 章では，6 章のテストベッドに対する IoT アプリケーションを拡張し，マルチメディアチャネルに対応したブレインストーミング支援システムの実現を行い，エージェントプラットフォームの評価を行う．最後に，8 章において，本論文におけるまとめと本研究の今後の展望を述べる．

論文の構成における有審査論文の貢献については，3 章では “ユビキタスデバイスとデジタルリソースを論理エージェントに接続するためのエージェントプラットフォームの開発”[89]，4 章は “An IoT Application Connecting Edge Resources and Cloud Resources using Agents”[39]，5 章から 7 章は “Capitalization of Remote Collaborative Brainstorming Activities”[52] 及び “Agent-based System Architecture Supporting Remote Collaboration via an Internet of Multimedia Things Approach”[40] による．

第 2 章

関連研究

本章では，以降の各章における関連研究を示す．

2.1 IoT アプリケーションを開発するためのエージェントプラットフォーム

本節では，IoT アプリケーションを，エージェント技術を用いて実現した具体的な例について記述する．

2.1.1 ラッパーアプローチによる情報リソースのエージェント化

本研究では，IoT アプリケーションを構成する各層の要素である，IoT デバイス，プログラム，データを情報リソース (Resource) と呼ぶ．

エージェント型の IoT アプリケーションでは，情報リソースをエージェント化するために，ラッパーアプローチが採用されている．図 2.1 にラッパーアプローチの概要を示す．ラッパーアプローチとは，推論などの論理的処理を行う論理機能，情報リソースに接続し制御するためのリソースアクセス機能，そして利用する情報リソースを各エージェントプラットフォームの開発言語によりエージェントの内部に直接組み込む方式である．以下で述べるエージェント型 IoT アプリケーションはこのラッパーアプローチを利用して設計・開発されている．エージェント型 IoT アプリケーションには，エージェントプラットフォームを利用しないものと利用するものに分けられる．

2.1.2 エージェントプラットフォームを利用しないエージェント型 IoT アプリケーション

センサからのデータを利用して，知覚や認知の機能を実現し，人間の社会活動を支援するエージェント型 IoT アプリケーション技術の研究が進められている．

松山らが行った分散協調視覚に関する研究プロジェクト [94] では，カメラが結合された複数の画像処理装置を地理的に分散して配置し，それらの装置が協調的に追跡タスクを実行する．分散協調における追跡タスクは，多数の自律した知能的処理機構（マルチエージェントシステム）として構成されている [98]．

この研究では，契約ネットプロトコルや交渉プロトコルなどのエージェント技術が IoT 機能の連携に効果的に使われている．また，情報リソースであるカメラの制御をエージェントに直接組み込むラッパーアプローチが採られている．

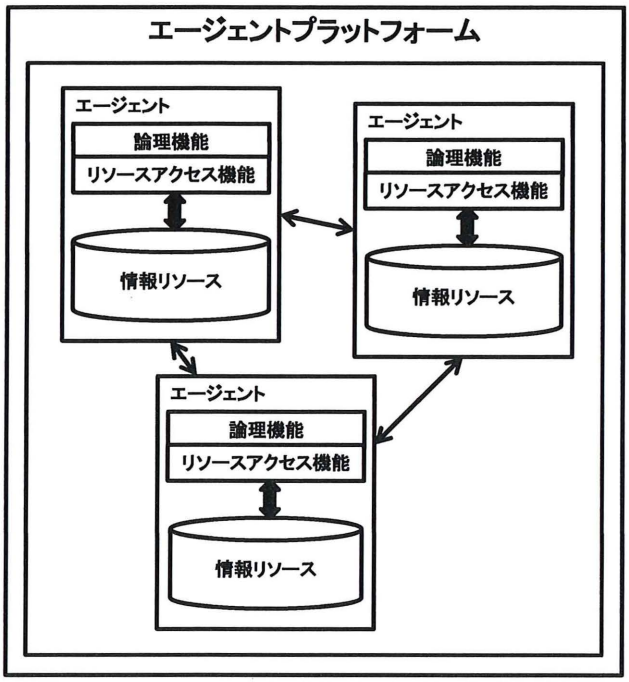


図 2.1　ラッパーアプローチ

2.1.3　エージェントプラットフォームを利用した IoT アプリケーション

JADE

JADE[13] は Java 上で実装された FIPA 仕様準拠のエージェントプラットフォームである．これを利用して，IoT アプリケーションを作成するための様々なフレームワークが開発されている．例えば，CONSORTS は，セマンティック Web の概念を実世界に適用し，コンテンツ流通の支援を行うマルチエージェントシステムを構築するためのフレームワークとして提案され，様々なアプリケーションの開発が進められている [90]．CONSORTS では，JADE の ACL(Agent Communication Language) やエージェント間協調プロトコルを使用することができ，エージェント型 IoT アプリケーションの開発をより容易にしている．また，Sebbak らは，JADE と OSGi フレームワークを用いて，プラグイン構造により情報リソースをエージェントから利用する枠組みを提案している [62]．さらに，Fortino らは，IoT デバイスのエージェント化のために JADE を利用して実装を行う Smart Objects を提案している [26]．これらのフレームワークを用いることにより，情報リソースをエージェントに結び付けることが可能である．しかし，Java のクラスを用いてリソースアクセス機能を記述し，エージェントに組み込む必要がある．

OMAS

OMAS（Open Multi-Agent System）[9] とは，Lisp で実装され，フレームシステム型の知識ベースを内部に持つエージェントを生成し，KQML に基づく ACL で通信することで，インターネット／ LAN 上でタスク指向の協調プロトコル（OMAS プロトコル）を提供するプラットフォームである．OMAS は，知識記述モデルと推論機能を持ち，COTERIE と呼ばれるエージェント集合の概念を用いてエージェント間のメッセージのやり取りを行う知的なマルチエージェントシステムを構築することを支援する．これまで，OMAS を用いて，

利用者と対話を行うエージェント [10]，ニュース共有システム [11] が開発されている．

OMAS では，Lisp を用いて情報リソースをエージェント化できるが，リソースアクセス機能を個々の情報リソースに合わせて Lisp でプログラムする必要がある．

JASON

JASON[16] は，Java で実装され，KQML に基づく ACL で通信を行い，AgentSpeak に基づく協調プロトコルで通信を利用する BDI アーキテクチャのマルチエージェントシステムを構築するためのエージェントプラットフォームである．

JASON では，JADE と同様に Java のクラスを用いてリソースアクセス機能を記述し，エージェントに組み込む必要がある [15]．

DASH/IDEA

DASH[88] は，Java で実装されたエージェントプラットフォームであり，リポジトリと呼ばれるエージェント集積機構を持つ．また，DASH は，エージェントの利用／再利用プロトコルに基づいて，エージェント動作環境上にマルチエージェントシステムの部品となるエージェント群を呼び出すことにより，マルチエージェントシステムの構築を支援する．IDEA[72] は，この方式でマルチエージェントの設計／改善を支援する開発環境である．DASH 同士の接続及び通信には Java の機能である RMI が用いられる．DASH ／ IDEA を用いて，ケアサポートシステム [66] やスマートグリッドシステム [42] などの IoT アプリケーションが開発されている．DASH/IDEA では，JADE 及び JASON と同様に，Java のクラスを用いてリソースアクセス機能を記述し，エージェントに組み込む必要がある．

2.1.4　ラッパーアプローチの問題点

ラッパーアプローチで情報リソースをエージェント化する場合，以下の 2 つの問題点が存在し，エージェント型 IoT アプリケーションの開発者に対して負担が大きくなることが問題として挙げられる．まず，(1) 情報リソースのエージェント化のために，エージェント開発者が情報リソースに関する開発知識，そしてエージェントプラットフォームに関する知識の両方を持ち，その開発作業を行わなければならないという問題がある．次に，(2) ラッパーアプローチにおける情報リソースのエージェント化では，一つのアプリケーションでエージェント化された情報リソースを他のエージェントプラットフォームのアプリケーションから再利用することが困難であるという問題がある．

2.2　IoT アプリケーションの設計方法

2.2.1　IoT アーキテクチャ

Atzori ら [5] および Razzaque ら [60] によれば，IoT は 3 つのコンポーネントから成り立っている．それは，モノ自体と適応されたミドルウェア，そしてそのセマンティクスで構成される．最初の 2 つのコンポーネントが汎用的であると考えると，ミドルウェアを介して交換されるデータのセマンティクスの側面は，システムのアプリケーションドメインに依存するということを前提として考える必要がある [29]．

IoT-A と呼ばれる欧州の研究プロジェクトで定義された Architectural Reference Model（詳細は，ARM[*1] を

[*1] IoT の最終的なアーキテクチャ参照モデル：http://www.iot-a.eu/public/public-documents/d1.5/view

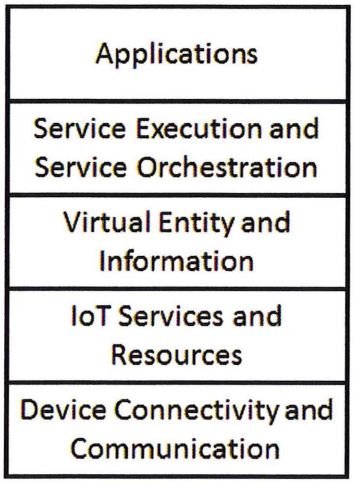


図 2.2　Alvi らの調査による IoT アーキテクチャ [3]

参照）では，3 つのタイプのデバイス（センサ，タグ，アクチュエータ）から構成され，異なる外部システムにデータを提供することができる．このモデルで提案されている物理的エンティティは，物理的環境の識別可能な部分であり，ほぼすべてのオブジェクトまたは環境にすることができる．また，仮想エンティティは，物理的オブジェクトの仮想的な対応物であり，それらの機能を定義する．これらの機能は，機能の特性を満たすためのデータを受け取り，提供を行う．

Alvi らは [3] において，IoT システムのためのアーキテクチャを提案した．図 2.2 において，5.5.2 節において説明する提案する独自のアーキテクチャと比較するために，アーキテクチャにおける各層の名称のみを同一とし，Alvi らの論文の 91 ページに示されている構造との対応を示す．Alvi らによれば [3]，“IoT Services and Resources レイヤはサービス発見機能と検索機能を提供することを目的としており，モノは単一の特定の垂直方向の層の要素にデータを提供することが制限されず，接続されたモノの協調のために外部システムが利用できるようになっている．” としている．しかしながら，本研究では Resource の要素は，ビデオストリームを送信するときや，アプリケーションのストリームが自身の属するアプリケーションのコアの要素に向かって垂直方向に通信するときなど，他のリソースと水平方向に通信する必要があると考えている．また，Alvi らは [3]，“仮想エンティティは，クラウドによって実装された追加機能を使ってモノの機能を補強する” としている．しかしながら，各 Resource の要素はサービスやエージェントなど，クラウドにある他のエンティティとの通信に必要なインタフェースをラッパーアプローチのように直接実装するのではなく，リソースコネクタと呼ばれる適切なトランスレータを使用することが重要であると考えている [89, 39]．

Alessio Botta ら [17] は，IoT アーキテクチャ内にクラウドのエンティティの役割を導入した．彼らはその中で，用語としての “CloudIoT paradigm” を作り，CloudIoT アプリケーションの展望を示した．このパラダイムには，新しいタイプのアプリケーションが含まれており，IoT アプリケーションが示された統合レベルに到達するためには新しい進歩が必要であるとされている．現在の調査研究の大部分は，クラウドによる IoT アプリケーションの統合の利点として，通信，ストレージ，および計算処理を考慮している．この観点から見ると本研究の場合，プラットフォームの実装には，主にストレージ（ビデオ，ドキュメントなど）と計算処理が含まれる．また，ストリーム転送のための通信は，ピアツーピア通信を利用する通信と多対多の接続による配信システムを考慮する必要が有る．

2.3　ブレインストーミング支援システム

2.3.1　リモートブレインストーミング支援システム

ブレインストーミングとは，参加者同士で意見を出し合うことにより，創造的な効率を高め，問題点に対する解決策を模索し，アイデアを生成する会議手法である．

ブレインストーミングを支援する研究としては，TATIN-PIC[51, 37] や KUSANAGI[99], Groupgarden[71], GKJ[50]，ScriptStorm[21]，そして，Firestorm[20] などが存在する．

また，遠隔地の多拠点間でブレインストーミングを行う手法として，リモートブレインストーミングが存在する．リモートブレインストーミングを支援する研究としては，宗森らの群元 [93] や GUNGEN-SPIRAL II[100]，そして，Forster らの IdeaStream[25] などが存在する．さらに，リモートブレインストーミングを支援する Web サービスとして，Bubbl.us[*2]，MindMeister[*3]，Stormboard[*4]，XMind[*5]，そして，Mindomo[*6]が存在する．

2.3.2　同期型ブレインストーミングと非同期型ブレインストーミング

Johansen は拠点間の距離と時間においてグループウェアによる機能を分類するために Time/Space Matrix の定義を行い [34]，Baecker がこの定義を参考に CSCW 技術やそのための機能を分類するための表として再定義を行った [6]．表 2.1 に CSCW 技術を分類するための表を示す．表の各項目は，コミュニケーション及びそれらの持つ通信機能の違いと捉えることもできる．これに基づいて，リモートブレインストーミングを分類した場合，同期型ブレインストーミングと非同期型ブレインストーミングに分けられる．同期型ブレインストーミングとは，Face-to-Face と同様な作業環境を目指し，遠隔拠点との情報共有のリアルタイム性を満たす協働作業である．また，非同期型ブレインストーミングとは，ブレインストーミングの行為の発生時刻と行為の結果を利用する時刻が参加者によって独立に選択できる環境で行うブレインストーミングのことである．同期型ブレインストーミングを支援するツールとして普及し利用されているものに Skype[*7]がある [78]．しかしながら，このツールはブレインストーミング行為の記録が蓄積されないため，その行為の記録の再利用をできず，非同期型ブレインストーミングの支援ができない．表 2.1 のすべてのブレインストーミングの行為を支援するアプリケーションは現在存在しないため，参加者は複数の支援アプリケーションを立ち上げることが必要になるが，この操作は参加者の負担になっている．

2.3.3　マルチメディアコンテンツによるブレインストーミングの支援

Marlow らは，リモートコラボレーションのサーベイ [46] の中で，“参加者に会話だけでなく，異なるタイプのマルチメディアコンテンツを遠隔で共有し，共同で参照する状況を満たすためには，現在のツールやアプローチが不十分である” としており，さらに，“参加者は単一のビデオ会議ツールがすべての要求に対応するのに十分な機能を備えていないので，様々なタイプの会議で様々な要求を支援するために，様々なツールを使用

*2 https://bubbl.us/
*3 https://www.mindmeister.com
*4 https://stormboard.com/
*5 http://xmind.net/
*6 https://www.mindomo.com/
*7 https://www.skype.com/

表 2.1　CSCW 技術における Time/Space Matrix [34], [6]

	One meeting site (same place)	Multiple meeting sites (different place)
Synchronous Communication (same time)	Face to Face Interactions	Remote Interactions
Asynchronous Communication (different time)	Ongoing Tasks	Communication and Coordination

する必要がある" としている．そして，ブレインストーミングのためのツールにおいても，それぞれ機能が異なるメディアを支援する為のツール（Skype や Google Hangouts[*8]，Google Docs[*9]等）をユーザが複数同時に利用する必要がある [78]．また，多くの音声や映像の会議支援アプリケーションはデータを蓄積し再利用する機能を持たないことが多く，非同期型ブレインストーミングへの支援が不十分である．

2.3.4　リモートブレインストーミングのリソースの資産化

リモートブレインストーミングにおけるリソースとは，ブレインストーミングに使用するデータ，資料，記録を含むオブジェクトと定義する．遠隔拠点間の場合，場の情報を完全に共有することは難しい．リモートブレインストーミングにおいては．ユーザが情報の共有に必要なアプリケーションを起動することでブレインストーミング中の情報であるリソースの共有を行っている [78]．しかしながら，それらのアプリケーションは連携することがなく，ブレインストーミング中に起こった様々なイベントを後で利用することが難しい．

例えば，Skype for Business[*10]は全体の会議映像を保存できるなどの機能が利用できるため，そのレベルで再利用が可能であるが，他のメディアの要素と関係付けることが難しい．また，Google Hangouts[*11]を利用することで，複数の機能を用いながら会議の映像を YouTube に保存することが可能であるが，保存される会議の要素が映像と作成されたドキュメントのみであり，会議中のイベントを把握することができない．すなわち，構造化が容易なデータを扱っているものの，構造化ができないデータについての検討を行っていない事が問題点としてあげられる．

本研究では，ブレインストーミングの結果を蓄積し非同期で利用するための資産化 (Capitalization) の概念を [52] において提案を行った．

2.3.5　コラボレーション支援のためのインタラクティブディスプレイ

本研究における IoT の捉え方としては，センサなどの単純なオブジェクトの使用よりもはるかに広い IoT オブジェクトのカテゴリを含んでいる．また，あらゆるタイプのタッチスクリーンデバイスは，潜在的に IoT オブジェクトであると考えている．しかし，これらは入力デバイスと出力デバイスの両方に対応することができ

[*8] https://hangouts.google.com
[*9] https://docs.google.com/
[*10] https://www.skype.com/business/
[*11] https://hangouts.google.com/

るため，アプリケーションでの統合がより困難となる．

IoT デバイスであるマルチタッチディスプレイは，オフィス環境および他の公共エリアに統合され，利用されている [30]．また，マルチタッチディスプレイは水平と垂直の 2 種類のディスプレイが存在するため，どちらも検討する必要がある．大型の垂直設置型タッチディスプレイ（ボード）は設置が容易であり，その設置方向によりユーザディスプレイとして主に使用され，シンプルなスクリーン用に設計されたアプリケーションを表示することができる．これらの人が触れることによりインタラクションする平面の IoT デバイスをサーフェスと呼ぶ．

A. Jones らは [36] において，プロジェクトの暫定的デザインを支援するためのプロジェクトである TATIN-PIC プロジェクトを紹介している．プロジェクトマネジメントのアクティビティは，大規模なタッチデバイスを使用する場合，多人数でインタラクションが可能となることで様々な支援を受けることができると考えられる．TATIN では，水平設置型マルチタッチディスプレイの IoT デバイスであるテーブルトップに何かを書く場合，仮想キーボードを開く必要がある．しかしながら，複数人がテーブルトップを操作する状況においては，シンプルなテキストでも容易に編集することが難しい [51]．創造的な活動をより自然に行うためには，これらに対して技術的な進歩が必要である．さらに，これらのテーブルトップ等のサーフェスと同時にインタラクションを行い，その機能補完する機能を持つモバイルデバイスの使用が必要である [7]．

2.3.6　テレビ会議ツール

本節では，コミュニケーションツール（Skype，WebEx など）が本研究における要件すべてを満たしていない理由を示す．

Skype[*12]は，プロフェッショナルなコミュニケーションではなく，個人または家族とのコミュニケーションを目的としている．これにより，すでにプラットフォームに登録されている 1 人以上の個人をリンクし，コミュニケーションを行うことができる．また，チャットやメッセージングシステムにより，コメントやテキスト情報の送信，文書の転送が可能である．しかしながら，主な欠点の 1 つとして，文書やビデオが異なるコミュニケーションラインを通じて分散され，会議全体のデータの直接の資産化ができないということがあげられる．

WebEx[*13]は，オンライン会議を一般の人々にもたらした．このオンライン会議には，画面共有の表示モードによるビデオ会議が含まれる．このシステムでは，プレゼンターを切り替えたり，デスクトップを共有したり，仮想ホワイトボードでアイデアをスケッチしたり，そして，ミーティングを記録したりすることが可能である．WebEx は，閉鎖されたシステムであっても，本アーキテクチャの要件の大部分を達成することが可能な非常に興味深いツールである．しかしながら，リモートコラボレーションの支援ツールは，それぞれのアイデアを独立した方法で管理できるシンプルな仮想ホワイトボードよりも多種多様なデバイスやアプリケーションであるさまざまなカメラや同期データを提供するアプリケーションからの複数のビデオストリームを必要とするため，支援が不十分であると考えられる．

2.3.7　テレビ会議システムの測定方法

ビデオ会議中は，コミュニケーションの際に快適に感じる必要がある．それらの上で行われる会話の発言は明瞭さを持って伝達するべきであり，人々が話しているときにリアルタイム性が高い同期が可能であるべきで

[*12] Skype の機能：https://www.skype.com/en/features/
[*13] https://www.webex.com/

ある．したがって，ビデオ会議アプリケーションの品質を測定することは研究者にとって重要であると考えらえる．さらに，これらのアプリケーションは，音声品質を保証する必要がある．

Bartoli らは [12] において，IP ネットワーク上のビデオ会議サービスにおける音声/ビデオストリーム内の同期とストリーム間同期を多くのシステムにおいて検証を行っている．彼らは，音声とビデオのストリームの品質とアプリケーションの動作を確認するための測定アルゴリズムを開発している．

ビデオ会議アプリケーションの品質を測定することは，ユーザがこれらのアプリケーションを介して通信するときに快適に感じる必要があるため，研究者にとって重要な課題である．この時，システムのパフォーマンスを分析し，その特性と経験の質のさまざまな側面を示すことは必須である．Lu らは [44] において，ビデオ会議アプリケーションを設計する際に，次の点を検討することを推奨した：

1. 限られた帯域幅をよりよく使用し，安定した会話を行うためのトラフィック負荷制御/バランス
2. 全体的なトラフィックを制限するためにストリームの再符号化

また，Boyaci らは，[18] において，キャプチャ及びエンコードからの送信と受信からデコード及びディスプレイへの表示までに実行される流れでのキャプチャから表示までの待ち時間（CDL）をソースコードを変更せずに特別なハードウェア要件を必要とせずにフレームレートを測定することを推奨している．

そして，[47] では，ビデオストリームにおける次の 3 つのレベルの解像度とフレームレートは以下ように分類されている：

1. 標準 320 × 240 ピクセルのビデオ解像度と 15fps のフレームレート
2. 解像度が 640 × 480 ピクセル，フレームレートが 30fps の高画質
3. 最高解像度 1280 × 720 ピクセル，同じフレームレート 30fps の HD 画質

本研究では，その中でも標準的な画質とフレームレートである 2 番目の画質を検討する．

第 3 章

IoT アプリケーションを開発するためのエージェントプラットフォームの提案

IoT アプリケーションは 3 階層のアーキテクチャに従う．これらの各層は，機能や特性が異なる要素から構成される．したがって，エージェント指向開発方法論は，各層の特性に基づいたエージェントのモデルを必要とする．ただし，それらのエージェントは，共通のプロトコルで，協調可能であることが必要である．しかしながら，これらの条件を満足するエージェントモデルが存在しない．なぜなら，現在のユビキタス及び IoT におけるエージェント型ミドルウェアによる開発方法によれば，下層レイヤを扱うことのみに終始しており，且つ，下層レイヤの IoT デバイスはその内部に持つモノのセンシング及び制御のためのアプリケーションプログラムが強固にラッピングアプローチを用いて結びついてしまっており，切り替えることが難しい構造を成しているからである．

本章では，ラッピングアプローチに対するエージェント型アーキテクチャの構成方法としてリソースコネクタと呼ぶ構造の提案を行う．リソースコネクタは，ラッピングアプローチと違い，上述の IoT デバイス内のアプリケーションプログラムを構造上分離し，プラグインプログラムとして扱うことで再利用と切り替えを可能とする．また，リソースコネクタは扱うことが可能な対象として IoT デバイス・データ・プログラムである情報リソースをエージェント化することが可能であり，これにより IoT アーキテクチャにおける各層のコンポーネントをエージェント化し，エージェントプロトコルによる通信を可能とすることで，それぞれのコンポーネントをエージェントにより扱うことが可能となる．このリソースコネクタにより，各層のコンポーネントはエージェント空間において共通のプロトコルにより協調可能となった．

3.1 IoT に関するエージェント研究の動向

近年，利用者の行動や状況を知覚するためのセンサデバイスの信号から利用者の行動や環境の状態を推測する様々なデータがクラウドなどのデジタル空間に蓄積されている．また，蓄積されたデータに基づき，利用者に適切な情報やサービスを提供する方法が研究されている [95]．本章では，実世界の様々な場所に設置可能なデバイスとして，利用者の行動や状況を知覚するためのカメラや加速度センサ，マイクロフォンなどの入力デバイス，適切な情報を提供するときに使われるディスプレイやスピーカなどの出力デバイスを IoT デバイスと呼ぶ．また，デジタル空間のデータ及びそれらを処理・分析するためのプログラムをデジタルリソースと呼ぶ．これに加えて，IoT デバイスとデジタルリソースを合わせて情報リソースと呼ぶ．蓄積された様々な社会的データやプログラムは，IoT デバイスからの実時間情報と組み合わされることによって，これまでには存在しなかった新しい高度なアプリケーションの創出に利用されている [84, 86]．これらの高度なアプリケーショ

ンを創出するための一つのアプローチとして，エージェント技術，IoT 技術，インターネット技術を組み合わせることにより，多様で大量のデータの獲得，蓄積，流通の仕組みを実現することが期待されている [98, 90]．このためのエージェント型アプリケーションを構築する開発環境として，様々なエージェントプラットフォームが提供されている [79]．エージェントプラットフォーム上で動作するエージェントプログラムには，人工知能に基づく様々な推論，意思決定，プランニング，認識などの機能が組み込まれており，記号化された情報の論理的な処理機構を実現しやすいという長所が存在する．本章ではこのようなエージェントを論理エージェントと呼ぶ．論理エージェントが情報リソースを利用して情報の蓄積・流通・利用者へのサービス提供を行う方法の一つとして，情報リソースのエージェント化と呼ばれる技術がある [92, 88]．エージェント化を実現するためには，IoT デバイスからの信号を取得・分析し記号に変換する処理や，エージェントの意思決定の結果を現実空間のデバイス制御信号に変換する処理といった情報リソースにアクセスする機能を組み込んだエージェントの開発が必要である．しかしながら，情報リソースをエージェント化する体系的な仕組みの提供が行われていないことが問題となっている [38]．例えば，これまでのエージェント化の方法は，エージェントプラットフォームを開発したプログラミング言語で，直接エージェントプログラムの内部に情報リソースへアクセスするプログラムを記述するラッパーアプローチがとられていた．そのため，情報リソースの制御などに関する知識をエージェント開発者が持つ必要があり，この事が開発者の負担となっていた [79, 13, 9, 15] ．さらに，エージェントプラットフォームごとに，アドホックに情報リソースのエージェント化が行われていた．そのため，他のエージェントプラットフォームからエージェント化した情報リソースの再利用が困難であることが問題となっていた．

本章では，エージェント型の IoT アプリケーションを開発するラッパーアプローチの開発者の負担軽減と情報リソースの再利用を実現するリソースコネクタ方式を提案する．この方式は，エージェントプラットフォームから，情報リソースへのアクセス機能と論理エージェントの機能を分離し，開発に関する知識を分離することでエージェント開発者の負担を軽減する．また，情報リソースへのアクセス機能を複数のエージェントプラットフォームに接続可能とする新しい機能を提供することで，情報リソースの再利用を実現する．また，この分離により，IoT アプリケーションの各層のコンポーネントのような多種多様な情報リソースを共通のプロトコルによって協調可能とする．図 3.1 に提案するエージェントプラットフォームによる多種多様な情報リソースの協調の仕組みを示す．

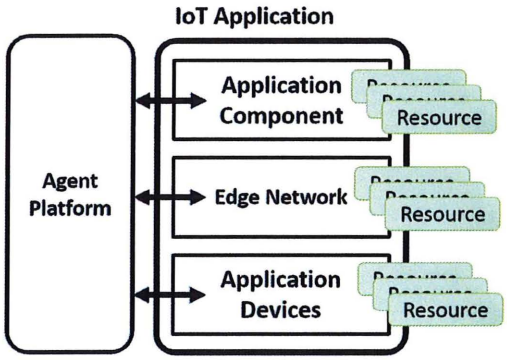


図 3.1　提案する IoT アプリケーションを開発するためのエージェントプラットフォームによる多種多様な情報リソースの協調

まず，関連研究の 2.1 節において，エージェント型 IoT アプリケーションとエージェントプラットフォームについて述べ，本章で対象とした問題点について述べた．以降，3.2 節では，IoT アプリケーションの開発者の

負担を軽減するエージェント化の提案方式であるリソースコネクタ方式について述べる．3.3 節では，リソースコネクタ方式でエージェント化を行うためのリソースコネクタプラットフォームの設計と実装について述べる．3.4 節では，リソースのエージェント化を論理エージェントのプラットフォームと独立させることにより，リソースのエージェント化を行う開発者の負担の軽減の評価を行う．

3.2　IoT アプリケーションを開発するためのエージェントプラットフォームの提案

本節では，2.1.4 節で示したラッパーアプローチの問題点を解決するためにリソースコネクタ方式を提案する．

3.2.1　リソースコネクタ方式の提案

提案方式では，今までエージェント内部に組み込まれていたリソースアクセス機能と情報リソースを論理機能と分離する．図 3.2 にラッパーアプローチに対するリソーコネクタアプローチの違いを示す．この時，リソースアクセス機能及び情報リソースを内部に持つエージェントを情報リソースエージェントと呼ぶ．情報リソースエージェントはエージェントプラットフォームの外部に存在するが，論理エージェントから見るとエージェントプラットフォーム内の論理エージェントとみなすことができる．つまり，情報リソースエージェントは論理エージェントとして同一のエージェントネットワーク内に存在するように振る舞う．図 3.3 に情報リソースエージェントの構造を示す．情報リソースエージェントは，リソースコネクタエージェントと情報リソースとして実装される．リソースコネクタエージェントは，プラットフォーム通信機能と情報リソース制御機能，そしてリソースコネクタ機能という 3 つの機能と，エージェント通信仕様と情報リソース仕様という 2 つの仕様から構成され，情報リソースのエージェント化を行うための機構として動作する．エージェントプラットフォームと情報リソースエージェントの通信は，接続するエージェントプラットフォームで定義された ACL（Agent Communication Language）に基づくメッセージ通信によって実現する．また，この通信を実現する機能部分であるプラットフォーム通信機能は，プラグインプログラムとして実装されている．これにより，一度接続するエージェントプラットフォームに対するプラットフォーム通信機能を開発すれば，エージェント通信仕様の変更のみで自動的に複数のエージェントプラットフォームへ切り替えることが可能な設計となっている．さらに，情報リソースにアクセスする機能部分である情報リソース制御機能もプラグインプログラムとして実装されており，情報リソース仕様に基づいて容易に変更が可能な設計となっている．

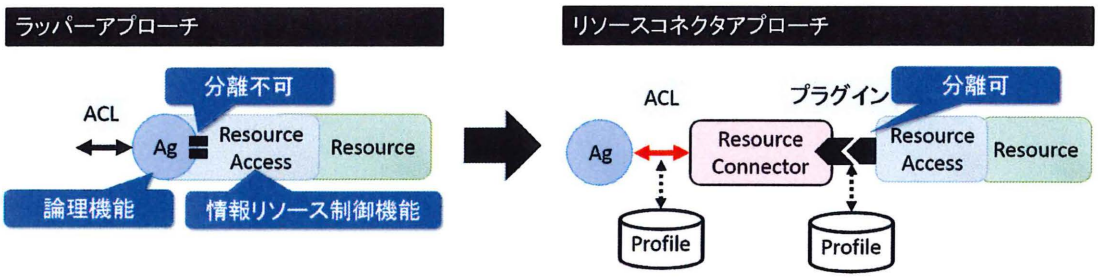


図 3.2　ラッパーアプローチとリソースコネクタアプローチの違い

本章では，論理エージェント群を開発する者を論理エージェント開発者と呼ぶ．また，リソースコネクタエージェントを開発する者を情報リソースエージェント開発者と呼ぶ．提案方式は，2.1.4 節で示した 2 つの

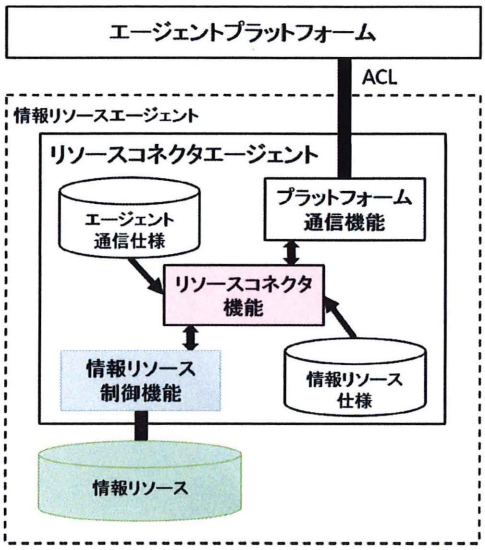


図 3.3　情報リソースエージェントの構造

問題点を解決する枠組みを提供する．まず，問題点 (1) に対しては，論理エージェントと情報リソースエージェントを分離したことにより，それぞれの機能を分担して独立に開発することが可能な枠組みを提供する．問題点 (2) に対しては，エージェント通信仕様を書き換え可能にすることによって，異なるエージェントアプリケーションの論理エージェントから情報リソースを利用可能な枠組みを提供する．さらに，プラットフォーム通信機能を書き換え可能にすることにより，異なるエージェントプラットフォームのエージェントから情報リソースを利用することが可能な枠組みを提供する．

3.2.2　エージェント通信仕様と情報リソース仕様

まず，エージェント通信仕様は，プラットフォーム通信機能のために用意された他のエージェントプラットフォームと通信するための記述である．エージェント通信仕様には，他のエージェントプラットフォームに接続する際の設定が記述されており，これに基づいてリソースコネクタエージェントが論理エージェントとメッセージ通信を行う．次に，情報リソース仕様は，情報リソースの入出力定義とロケーションの記述である．入出力定義は，リソースコネクタエージェントが制御する情報リソース機能を表す入力元と出力先の詳細の記述である．ロケーションは，入出力デバイスの設置場所，プログラムがインストールされているコンピュータの IP アドレスやデータの URI の記述である．これらの仕様は，情報リソースエージェント開発者によって記述され，論理エージェント開発者が情報リソースエージェントの機能とロケーションを知るために利用される．また，情報リソースエージェントが自身の機能とロケーションを論理エージェントに提供するために利用される．

3.2.3　プラットフォーム通信機能と情報リソース制御機能

まず，プラットフォーム通信機能は，リソースコネクタエージェントがエージェントプラットフォームと独立して動作するためのプログラムであり，ACL に基づくメッセージの解析や構成を行い，エージェントプラットフォームと通信するための機能を持つ．この機能は，情報リソースエージェント開発者により開発される．次に，情報リソース制御機能は，情報リソース仕様に従って情報リソースを制御するためのプログラムである．

具体的には，入力デバイスを利用した認識処理や，出力デバイスの制御，またアプリケーションの制御やデータの取得等の処理を実現するプログラムのことである．この機能は，情報リソースエージェント開発者により開発される．

3.2.4　リソースコネクタ機能

リソースコネクタ機能は，2 つの機能と 2 つの仕様を必要に応じて読み込み，それによって生じる処理を取り纏める中核的な役割を果たすプログラムである．このため，リソースコネクタ機能は，情報リソースからの入力によって起こる処理と論理エージェントからのメッセージ受信によって起こる処理の 2 つの流れを制御する．情報リソースからの入力が生じた場合，まず，情報リソース制御機能が動作する．そのとき，リソースコネクタ機能は，情報リソース制御機能から情報を受け取り，情報リソース仕様を読み込む．そして，情報リソース仕様に基づき，メッセージ通信を行う対象の論理エージェントを決定する．その後，エージェント通信仕様を読み込み，プラットフォーム通信機能に情報を渡す．最終的に，プラットフォーム通信機能は，渡された情報に基づきエージェントプラットフォームに対してメッセージ通信を行う．一方，論理エージェントからのメッセージ受信が起こった場合，まず，プラットフォーム通信機能が動作する．そのとき，リソースコネクタ機能は，エージェント通信仕様を読み込み，プラットフォーム通信機能から情報を受け取る．そして，情報リソース仕様を読み込み，それに基づき，制御を行う対象の情報リソースを決定する．その後，情報リソース制御機能に情報を渡す．最終的に，情報リソース制御機能が渡された情報に基づき，情報リソースの制御を行う．

3.2.5　リソースコネクタプラットフォーム

図 3.4 に示す，3 つの機能と 2 つの仕様から構成されるエージェントプログラムを読み込んで動作させるツールをリソースコネクタプラットフォームという．図 3.4 に示すように，プラットフォームにより読み込まれ動作を開始したエージェントは，情報リソース仕様の記述に基づいて情報リソースを起動・制御する．また，リソースコネクタエージェントは，エージェント通信仕様の記述に基づいて，指定されたエージェントプラットフォームで動作するエージェントのように振る舞うことができる．そのため，論理エージェント開発者は，そのエージェントプラットフォーム上の通信プロトコルに従って，マルチエージェントシステムを構築することができる．

3.3　提案したリソースコネクタプラットフォームの試作

本節では，提案方式に基づいたリソースコネクタプラットフォーム rConnector を開発した．また，rConnector を用いて情報リソースをエージェント化するリソースコネクタエージェントを rConnector エージェントと呼ぶ．rConnector を用いて情報リソースをエージェント化するのは情報リソースエージェント開発者である．

rConnector エージェントの構成要素は，PlatformProfile，AgentListener，AgentPacket，ResourceAgentProfile，ResourcePlugin，EventConnector の 6 つである．これらと図 3.4 で示した提案方式の構成要素との対応を表 3.1 に示す．AgentListener と AgentPacket，及び EventConnector は rConnector によって提供される機能である．また，各機能はすべてマルチスレッドにより動作し，必要に応じて連携が行われる．

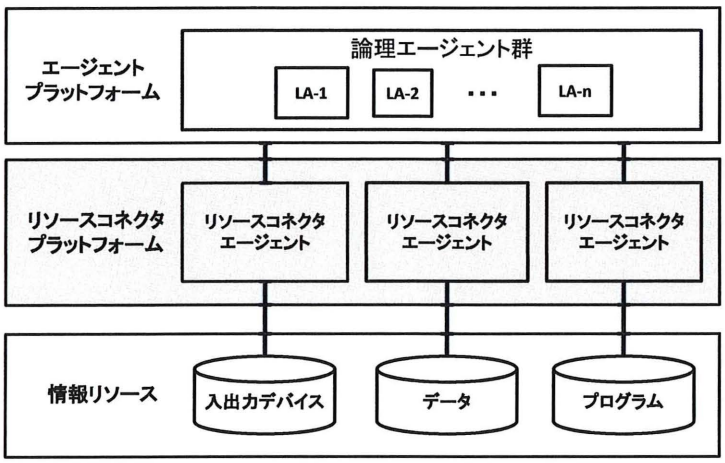


図 3.4　リソースコネクタプラットフォーム

表 3.1　提案方式と rConnector の対応関係

図3.2の構成要素	rConnectorの構成要素
エージェント通信仕様	PlatformProfile
プラットフォーム通信機能	AgentListener, AgentPacket
情報リソース仕様	ResourceAgentProfile
情報リソース制御機能	ResourcePlugin
リソースコネクタ機能	EventConnector

PlatformProfile

Name	プラットフォーム名
Version	バージョン制限
Description	本設定についての詳細
Log	ログレベルと通信ログの出力先設定
CoterieConfig	対象ネットワークの設定
ProtocolType	プロトコルの種類(UDP/TCP)
Broadcast	ブロードキャストの設定(True/False)
LocalIPAddress	自己IP設定(Autoの場合，現在のPC)
Port	ポート番号
TimeoutCount	タイムアウト設定(ミリ秒指定)

図 3.5　PlatformProfile の記述構造

3.3.1　PlatformProfile

接続するエージェントプラットフォームの情報を記述した設定ファイルである．XML 形式により定義され，CoterieConfig というエージェントプラットフォームにおける対象ネットワークの設定や IP アドレスの設定，ポート番号の設定等が記述されている．図 3.5 に記述構造，図 3.6 に記述例を示す．この設定ファイルは，対象とするエージェントプラットフォームが OMAS であり，エージェントは UDP のプロトコルを利用し，ポート 50000 番で通信するエージェントの記述例を表している．PlatformProfile は情報リソースエー

```
<?xml version="1.0"?>
<PlatformProfile xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <Name>OMAS</Name>
        <Version>1.0</Version>
        <Description></Description>
        <Log>
                <LogLevel value="3"/>
                <Path>./rConnector/networkLog/</Path>
        </Log>
        <CoterieConfig>
                <ProtocolType>UDP</ProtocolType>
                <Broadcast>True</Broadcast>
                <LocalIPAddress>Auto</LocalIPAddress>
                <Port>50000</Port>
                <TimeoutCount>3000</TimeoutCount>
        </CoterieConfig>
</PlatformProfile>
```

図 3.6　PlatformProfile の記述例

ジェント開発者が記述するが，論理エージェント開発者が情報リソース利用の際に記述を変更することもできる．PlatformProfile を変更することにより，様々なエージェントプラットフォームに対応することが可能となる．具体的には，PlatformProfile の Name タグの値に対応するプラットフォーム通信機能のプラグインプログラムである AgentListener 及び AgentPacket が読み込まれることで，対応可能な設計となっている．

3.3.2　AgentListener

PlatformProfile に記述されているエージェントの設定情報に基づき，対象となるエージェントプラットフォームのネットワークに対して接続を行うプラグインプログラムである．AgentListener は，対象ネットワークのエージェントプラットフォーム上に流れるメッセージを構成するパケットを取得し，送信することができる．取得したパケットは AgentPacket に送られる．

3.3.3　AgentPacket

AgentListener から受け取ったエージェントメッセージを構成するパケットをパースし，プログラム内で利用しやすい形に加工する機能を持つプラグインプログラムである．また，ACL のメッセージを構築し，検証する機能を持つ．

3.3.4　ResourceAgentProfile

情報リソースを定義するために，rConnector エージェントの情報としてエージェント名等の情報やデータの入出力の定義情報を扱う設定ファイルである．XML 形式により定義され，入出力を行う情報リソースが機能毎に記述されており，論理エージェントが機械的に処理可能な形式で記述されている．図 3.7 に記述構造を，図 3.8 にマイクロフォンをエージェント化した際の記述例を示す．この記述例においてエージェントは，マイクロフォンからの入力を受け付け，音声認識機能を用いて，論理エージェントに対してテキスト形式で認識結果を送る．ResourceAgentProfile は情報リソースエージェント開発者が記述し，論理エージェント開発者が利用の際にこれを閲覧する．

3.3.5　ResourcePlugin

ResourcePlugin は，情報リソースにアクセスするためのプラグインプログラムである．ResourcePlugin は，情報リソースに関する様々な認識処理やデバイス制御処理を行うために，シグナルデータと記号データの相互変換処理を行う．また，本機能は情報リソースエージェント開発者が開発を行う．

3.3.6　EventConnector

EventConnector は，エージェント及び情報リソースからの 2 つの流れを制御するために，rConnector 内の各機能を連携させ動作を行う．具体的には，まず，PlatformProfile と ResourceAgentProfile のシリアライズ及びデシリアライズをすることにより，プロファイルの読み書きを行う．そして，その内容に基づき，AgentListener と AgentPacket，及び ResourcePlugin をプラグインプログラムとして読み込み，それらを連携させることにより，2 つの流れを制御する．この時，複数のエージェントからのメッセージを同時に処理する必要があるため，内部の各機能はマルチスレッドで動作する．

ResourceAgentProfile	
Name	rConnectorエージェント名
Version	バージョン
Description	rConnectorエージェントの注釈
Log	ログレベルとログの出力先設定
Profiles	プロファイルの集合
Profile	一つのプロファイル記述
Id	プロファイルID
Name	プロファイル名
Summary	機能の注釈
Description	プロファイルの詳細
Source	入力設定
Name	入力対象名
Action	入力アクション名
Type	入力デバイス or エージェント
ContentType	入力タイプ： 文章，音声，認識
Language	言語コード
Location	ロケーション
Destination	出力設定
Name	出力対象名
Action	出力アクション名
Type	出力デバイス or エージェント
ContentType	出力タイプ： 文章，音声，認識
Language	言語コード
Location	ロケーション

図 3.7　ResourceAgentProfile の記述構造

3.4　提案したリソースコネクタプラットフォームの評価実験

rConnector による様々な情報リソースのエージェント化の実装例と評価実験について以下に示す．また，本節では，IoT アプリケーションを開発する能力が十分かどうかを各種 IoT デバイスをエージェント化し，動作確認を行うことで評価を行う．

```
<?xml version="1.0"?>
<ResourceAgentProfile
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <Name>VOICE-INPUT</Name>
      <Version>1.0</Name>
      <Description>音声認識</Description>
      <Log>
            <Path>./rConnector/resourceLog/</Path>
            <LogLevel value="3" />
      </Log>
      <Profiles>
         <Profile>
            <Id>1</Id>
            <Name>SpeechToText</Name>
            <Summary>音声認識を行う</Summary>
            <Description>
                  <Use>Microsoft Speech Platform 11</Use>
                  <Link>http://msdn.microsoft.com/en-
                  us/library/hh362873.aspx</Link>
            </Description>
            <Source>
                  <Name>Microphone</Name>
                  <Action />
                  <Type>Device</Type>
                  <ContentType>Audio</ContentType>
                  <Language>ja-JP</Language>
            </Source>
            <Destination>
                  <Name>CONNECTOR-PA</Name>
                  <Action>CONSOLE-INPUT</Action>
                  <Type>Agent</Type>
                  <ContentType>Text</ContentType>
                  <Language>ja-JP</Language>
            </Destination>
         <Profile>
      <Profiles>
</ResourceAgentProfile>
```

図 3.8　ResourceAgentProfile の記述例

3.4.1　実装したエージェント群

本章で提案したリソースコネクタプラットフォームである rConnector 及びその上で動作する rConnector エージェントについては Microsoft Visual C#を用いて実装を行った．これらの rConnector エージェント群は，表 3.2 に示す 4 つの rConnector エージェントであり，OMAS 上で動作する論理エージェントと通信できるように実装した．実装を行った 4 つの rConnector エージェントは，それぞれ情報リソースの典型例をエージェント化したものである．まず，Voice Input エージェントは，IoT デバイス（入力装置）の例としてマイクロフォンをエージェント化したものである．次に，TTS エージェントは IoT デバイス（出力装置）の例としてスピーカをエージェント化したものである．そして，Web Access エージェントは，デジタルリソース（データ）の例として，Web 情報をエージェント化したものである．最後に，iTunes エージェントは，デジタルリソース（プログラム）の例として，アプリケーションプログラムである iTunes をエージェント化したものである．まず，マイクロフォンをエージェント化した Voice Input エージェントは，マイクロフォンからの音声信号を入力として受け取り，音声認識機能により認識結果をテキストに変換し，その情報を ACL のメッセージとして論理エージェントに送る動作を行う．音声認識機能は，Windows Speech Recognition を用

いて実装した．次に，スピーカをエージェント化した TTS エージェントは，論理エージェントからの発話テキストを含む ACL のメッセージを受け取り，そのテキストを音声発話機能により音声信号に変換し，スピーカから出力する動作を行う．音声発話機能は，Microsoft Speech Platform を用いて実装した．そして，Web 情報をエージェント化した Web Access エージェントは，論理エージェントから HTTP に基づくリクエストを含む ACL のメッセージを受け取り，そのリクエストに基づき Web 情報取得機能を用いて HTTP サーバにアクセスし，その結果を ACL のメッセージに変換して論理エージェントに送る動作を行う．Web 情報取得機能は，.Net Framework 上のモジュールを利用して実装した．最後に，アプリケーションプログラムである iTunes をエージェント化した iTunes エージェントは，論理エージェントからのプログラム制御に関するリクエストを含む ACL のメッセージを受け取る．続いて，そのリクエストに基づきアプリケーションプログラムである iTunes を音楽管理機能により制御（音楽の再生・停止・頭出し・曲名取得等）する動作を行い，結果情報を ACL のメッセージに変換して論理エージェントに送る動作を行う．音楽管理機能は COM（Component Object Model）を利用して実装した．

各エージェントは図 3.4 に基づいて実装されている．ここで Voice Input エージェントを例としてエージェント通信仕様，情報リソース仕様，プラットフォーム通信機能，情報リソース制御機能の具体的な実装方法について記述する．まず，エージェント通信仕様は XML 形式のファイルであり，図 3.6 に示した内容で記述されている．次に，情報リソース仕様はエージェント通信仕様と同様に XML 形式のファイルであり，図 3.8 で示した内容で記述されている．そして，プラットフォーム通信機能は，ダイナミックリンクライブラリの形式で提供されるプラグインプログラムとして実装されており，エージェント通信仕様に基づいて呼び出される．最後に，情報リソース制御機能は，プラットフォーム通信機能と同様にダイナミックリンクライブラリの形式で提供されるプラグインプログラムとして実装されており，情報リソース仕様に基づいて呼び出される．

表 3.2　rConnector エージェントの実現例

エージェント名	エージェントの機能	ResourcePlugin開発に用いた技術
Voice Input	音声認識機能	Windows Speech Recognition
TTS	発話機能	Microsoft Speech Platform
Web Access	Web情報取得機能	.NET Framework上のモジュール
iTunes	音楽管理機能	COM（Component Object Model）

3.4.2　評価実験及び実験結果

本節では，3 つの評価実験を行った．3.4.2 では，rConnector エージェントの動作確認実験を行った．3.4.2 と 3.4.2 では，リソースコネクタ方式の有効性を評価するために，それに基づくリソースコネクタプラットフォームである rConnector の評価実験を行った．

rConnector エージェントの動作確認実験

rConnector により 3.4.1 で実装を行った rConnector エージェントの動作確認を行った．具体的には，図 3.3 で提案した方式に対応して，OMAS 上の論理エージェントが rConnector エージェントを OMAS エージェントとみなし，相互にメッセージ通信が行えるかを検証した．検証のために，Voice Input エージェント及び TTS エージェントと通信を行う論理エージェントとして TA エージェント，Web Access エージェントと通信を行う論理エージェントとして SOCIAL-CONNECTOR エージェント，iTunes エージェントと通信を

行う論理エージェントとして MUSIC-MANAGEMENT エージェントをそれぞれ OMAS 上に実装した．検証項目は以下の 4 点である．

1. Voice Input エージェントは，マイクロフォンからの音声認識結果を TA エージェントに正しく送ることができるか
2. TTS エージェントは，TA エージェントからの発話テキストを正しく受け取ることができるか
3. Web Access エージェントは，SOCIAL-CONNECTOR エージェントからの URL 情報を受け取ることができるか．また，URL 情報に基づき，Web 情報を取得し，SOCIAL-CONNECTOR エージェントに正しく送ることができるか
4. iTunes エージェントは，MUSIC-MANAGEMENT エージェントからのプログラム制御要求を正しく受け取ることができるか．また，プログラム制御結果を取得し，MUSIC-MANAGEMENT エージェントに正しく送ることができるか

図 3.9 に 4 つの検証項目に基づく各エージェントの動作結果を表す OMAS のメッセージダイアグラムを示す．メッセージダイアグラムとは，エージェントのメッセージ通信を時系列で表示するためのデバッグ用ツールである．メッセージダイアグラムの上端には，OMAS 上で動作するエージェントの名前が表示されている．また，メッセージダイアグラム内の横に伸びる矢印はメッセージ送信を表し，その色の違いはメッセージの種類を表している．論理エージェント群として囲まれている領域に存在するエージェントは論理エージェントを表している．また，rConnector エージェント群として囲まれている領域に存在するエージェントは rConnector エージェントを表している．

図 3.9 を見ると，実装した 4 つの rConnector エージェントが，OMAS のメッセージダイアグラム上で 3 つの論理エージェントと同様に表示されており，OMAS 上のエージェントとしてみなされていることが分かる．また，メッセージダイアグラム内の横に伸びる矢印により，rConnector エージェントと論理エージェントが OMAS に基づくメッセージ通信を行っていることが分かる．そして，4 つの検証項目に対する実際の動作（音声認識，音声発話，Web 情報取得，プログラム制御）についても正しく行われることを確認した．さらに，動作確認実験中にハードウェアの停止等で情報リソースが異常終了した場合や，OMAS 上の論理エージェントが異常終了した場合も，rConnector は動作し続けることを確認した．このことより，提案方式に基づく rConnector は，入力装置・出力装置・データ・プログラムといった多様な情報リソースのエージェント化が可能であり，適切に動作することがわかった．

独立に開発した論理エージェントと rConnector エージェントの接続実験

論理エージェント開発者と情報リソースエージェント開発者がそれぞれ独立に開発したプログラムを，提案方式に基づいて接続し，IoT アプリケーションが開発できるか実験を行った．開発を行った IoT アプリケーションは音声対話アプリケーションである．まず，論理エージェント開発者が開発を行ったプログラムは，サッカーの領域についてテキストベースで日本語対話を行うことができる対話制御プログラムである．このプログラムは OMAS の論理エージェントを用いて開発された．一方，情報リソースエージェント開発者が開発を行ったプログラムは，rConnector 上で実装された Voice Input エージェントと TTS エージェントである．従って，Voice Input エージェントと TTS エージェントは OMAS の論理エージェントから利用される前提で開発を行った．お互いの開発プログラムを接続するために，共有することが必要な情報は ACL の構成に必要な情報であり，本実験では以下の 3 点の情報である．

1. コンテンツのデータ形式

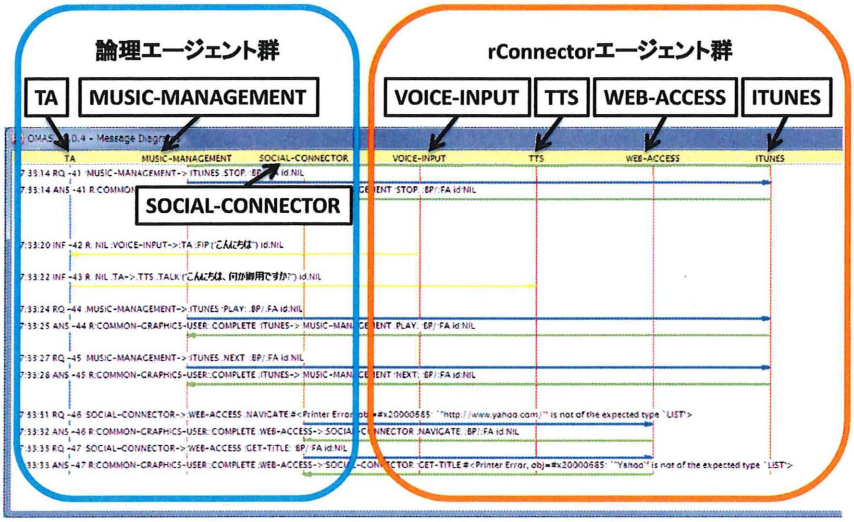


図 3.9　OMAS のメッセージダイアグラム

2. エージェント名
3. アクション名

コンテンツとは実際に情報リソースエージェントと論理エージェント間でやり取りされる情報のことであり，そのデータ形式は，コンテンツを記述する形式のことである．エージェント名は，論理エージェント名，または，rConnector エージェント名のことであり，実際に ACL に基づくメッセージの送受信を行うエージェントの名前である．アクション名は，ACL に基づくメッセージの送受信において呼び出されるエージェントの機能名のことである．これら 3 つの情報は，図 3.8 に示した ResourceAgentProfile に記述される．コンテンツのデータ形式は，Source または Destination タグ内の ContentType タグに対応し，エージェント名は，Source または Destination タグ内の Name タグに対応する．また，アクション名は，Source または Destination タグ内の Action タグに対応する．音声認識の場合を例にとり，これらの 3 つの情報に従って ACL に基づくメッセージが構成される流れを説明する．まず，ユーザがマイクロフォンに向かって発話を行った際に，音声認識機能が音声認識結果を生成する．次に，rConnector エージェントが，ResourceAgentProfile の Destination タグの情報に基づき，音声認識結果を送信する対象のエージェント名とアクション名及びデータ形式を取得する．最後に，rConnector エージェントは，データ形式に従って音声認識結果をコンテンツとした ACL に基づくメッセージを構成し，論理エージェントにメッセージを送信する．論理エージェント開発者は，上記 3 点の情報により rConnector エージェントを利用し，音声認識を行った結果を受け取ることが可能な論理エージェントを開発することができる．情報リソースエージェント開発者は，ResourceAgentProfile を作成する際に 1〜3 の情報を用いたが，これ以外の論理エージェントに関する知識を持たない条件で実験を行った．また，論理エージェント開発者は，1〜3 の情報をテキストベースの入力エージェントとテキスト表示エージェントの名称を変更する際に用いたが，これ以外の音声認識・音声発話プログラムに関する知識を持たない条件で実験を行った．結果として，情報リソースエージェント開発者が開発した Voice Input エージェントと TTS エージェントを論理エージェント開発者が開発した対話制御プログラムに接続することができ，テキストベースの日本語対話プログラムを音声認識と音声発話に対応させることができた．　このことから，IoT アプリケーションを作る上で，論理エージェント開発者の負担が軽減されたことは明らかである．また，情報リソースエージェント開発者は，rConnector を用いたことで 1〜3 の情報のみで論理エージェントに接続可能な情報リ

ソースエージェントを作成することができた．従って，エージェントプラットフォームの知識を理解する必要がなく，情報リソースをエージェント化する際の負担が軽減できたと言える．すなわち，論理エージェント開発者と情報リソースエージェント開発者双方の負担が軽減でき，ラッパーアプローチにおける問題点 (1) を解決できたと言える．

異なるエージェントプラットフォームにおける情報リソース再利用実験

OMAS の論理エージェントを接続対象として rConnector で実装した Voice Input エージェントと TTS エージェントを別のエージェントプラットフォームである JADE の論理エージェントから再利用可能であるか実験を行った．具体的には，Voice Input エージェントと TTS エージェントが持つ情報リソース制御機能を，JADE に対応したものに作り替えることなく，JADE の論理エージェントがそれぞれの機能を利用できるか検証した．再利用するために行った手順は，まず，JADE の論理エージェントと rConnector エージェントとの通信を可能とするために，JADE に対応する AgentListener と AgentPacket を実装した．次に，実装した AgentListener と AgentPacket を呼び出すために PlatformProfile の記述を修正した．最後に，Voice Input エージェント及び TTS エージェントと通信を行う JADE の論理エージェントを実装し，互いに正しく通信が行えるように ResourceAgentProfile の記述を修正した．上記の手順に基づく変更を行った Voice Input エージェントと TTS エージェントが JADE 上の論理エージェントと通信可能であるか実際に動作させ確認した．その結果，エージェントメッセージのログ記録により，正しく通信及び連携が可能なことがわかった．このことから，情報リソース制御機能については，Voice Input エージェント及び TTS エージェントで利用している ResourcePlugin を変更することなく，ResourceAgentProfile の記述の修正のみで情報リソースを再利用することができたと言える．これは，ResourcePlugin がダイナミックリンクライブラリの形式で提供されるプラグインプログラムであるため可能となった．一方，プラットフォーム通信機能については，JADE の論理エージェントとの通信を実現するために AgentListener 及び AgentPacket を実装したが，これらのプログラムは，ダイナミックリンクライブラリの形式で提供されるプラグインプログラムであるため，一度作成すれば PlatformProfile の記述の修正のみで利用が可能となり，再利用性が高いと言える．従来までに用いられているラッパーアプローチでは，異なるプラットフォームに対応させるためにはリソースアクセス機能部分をすべてコーディングし直す必要があり，多大な知識とコストを必要としていた．それに対して，本実験では，情報リソース制御機能に対応する C#プログラムは変更することなく，異なるプラットフォームである JADE に対応することができた．今回は OMAS と JADE で動作確認を行ったが，rConnector はその他のエージェントプラットフォームに対しても情報リソースを再利用可能な枠組みを提供していると考えられる．その他のエージェントプラットフォームとしては，例えば FIPA-OS[57] や SAGE[1]，Zeus[55]，Bee-gent[96]，KODAMA[83, 91] が存在する．これらのエージェントプラットフォームは，RMI や IIOP，HTTP を使用しており，通信プロトコルとして TCP や UDP を用いている．rConnector では，TCP や UDP のプロトコルであれば対象とするエージェントプラットフォームのメッセージ通信の処理を実現するプラットフォーム通信機能を実装することができる．また，情報リソース制御機能は，プラットフォーム通信機能とは独立に開発されるプラグインプログラムであり，エージェントプラットフォームに依存することなく，再利用することができる．このことから， rConnector エージェントは異なるエージェントプラットフォームに対して容易に再利用可能であり，ラッパーアプローチにおける問題点 (2) を解決できたと言える．

3.5　まとめ

本章では，IoT アプリケーションをエージェント指向開発方法論を用いてエージェント型 IoT アプリケーションを開発する方式の一つであるラッパーアプローチの問題点を解決するために，情報リソースをエージェント化するエージェントモデルであるリソースコネクタ方式を提案した．また，提案方式に基づくエージェントプラットフォームである rConnector を開発し，入出力それぞれの IoT デバイス・プログラム・データに対応する 4 つの rConnector エージェントを実装した．実験の結果，rConnector は多様な情報リソースをエージェント化できることを示した．また，独立して開発した論理機能と情報リソースを接続して，IoT アプリケーションの開発ができ，開発者の負担軽減ができることを示した．さらに，異なるエージェントプラットフォームの論理エージェントが，rConnector でエージェント化した同じ情報リソースを利用可能であることを示した．このことは，rConnector が論理機能を中心とするエージェントアプリケーションを IoT 型へと容易に拡張可能であることを示している．また，本提案方式においては，プラグインプログラムであるリソース制御機能とプラットフォーム通信機能を一度作成すれば，新しいアプリケーションを作るたびに特定のエージェントプラットフォームのパケットに対応したプログラムを作り直す必要がなく，どちらかを入れ替える事で，他のエージェントプラットフォームや他のリソースに対応できる．このことから一度作成したプラグインプログラムを利用できるため，再利用性が非常に高いと考えられる．

しかしながら，本章における提案はエージェント指向開発方法論における IoT アプリケーションに関するエージェントを開発する際のエージェントモデルの提案としては十分であるものの，特に IoT アプリケーションを構成する 3 層目の IoT デバイスを中心としており，エージェントプラットフォームを IoT システムとして実現するための実装技術としての検討としては不十分であると考える．また，利用するエージェントプラットフォーム自体が IP が指定された TCP や UDP による接続を行っており，大規模な IoT プラットフォームにおいては不十分であると考える．よって，次章では，IoT アプリケーションのアーキテクチャの各階層に全体に対して，設計及び実装の検討を行う．また，同時に，大規模な IoT アプリケーション開発に対するエージェントプラットフォームの対応を模索する．

第 4 章

IoT アプリケーションを開発するためのエージェントプラットフォームの設計と実装

IoT アプリケーションのアーキテクチャの各階層のエージェントは，異なるインフラストラクチャの上で動作することが必要である．たとえば，クラウドのエージェントプラットフォームは，アプリケーションの論理動作を高速に処理し，クラウド内の通信プロトコルで効率的にメッセージ送信を行う．ネットワークを構成する要素を実現するエージェントは，大量のストリームを送受信したり，多数のエージェント間の安定したメッセージの総配信を行う機能を持つ必要がある．デバイスを制御するエージェントは，直接物理デバイスを制御するための仕組みを必要とする．このように，IoT アプリケーションの各階層に対応して異なる特性を持つエージェントを実装するためのプラットフォームが存在しない．

本章では，3 章で実現したエージェント指向開発方法であるリソースコネクタを実際の開発において利用するためのプラットフォームとして実現するために，その開発方法におけるモデルを実現する為のエージェントプラットフォームであるリソースコネクタプラットフォームの設計と実装を行った．リソースコネクタプラットフォームは，IoT アプリケーションの各階層に対応して異なる特性を持つエージェントをリソースコネクタの概念に基づいて実装を行うことで，物理デバイスを制御することが可能な仕組みを提供する．そして，プラットフォームの支援機構により，開発方法論を実現する為の様々な支援ツールの開発を行った．さらに，プラットフォームの実現を確認するために，複数の特性を持つ情報リソースを対象としてプラットフォームによる IoT アプリケーションの実装を行う事で検証を行った．この実装とその確認により，エージェント指向開発方法論に基づく，異なるインフラストラクチャ上で動作可能なプラットフォームの実現できた．また，IoT アプリケーションの各階層に対応して異なる特性を持つエージェントを実装することが可能なプラットフォームを実現できた．

4.1 エッジコンピューティングとクラウドコンピューティングの支援に関するネットワーク技術

情報通信技術（ICT）を利用することで，人々はクラウド内の情報を簡単に送信（保存）および受信（アクセス）することができる．さらに，ユビキタス技術の急速な発展は，情報システムがスマートフォンなどのモバイル通信デバイスによって現実空間（Real Space/RS）にアクセスすることを可能にする．また，様々な種類のセンサデバイスが，RS と多量のユーザの多種多様な情報を持つ場所に配置され，その周囲の情報を取得してクラウドに保存することができる．この情報を利用して，「いつでも」「どこでも」などの非常に便利なサービスは，日常的にユーザに広く提供されている [45].

しかしながら，重要な情報がクラウドの膨大な情報に隠れる可能性があるため，ユーザにとって必要な情報や支援をすばやく見つける際に重い負担となっている [87]．ユーザが何らかのイベントを監視しなければならない時，クラウドが提供する情報を通じて，誰かが情報機器を監視し続ける必要がある可能性が存在する．スキルを持たないユーザが情報システムを利用するスキルを持っていれば，彼らがそのような状態となる機会を防ぐことが可能である．

エージェントベースの技術は，RS のクラウドから未熟なユーザに対して情報を配信する機能を実現するためのソリューションの 1 つである [61]．適切に設計されたエージェントの集まりであるエージェント空間（エージェントスペース/AS）は，クラウド上で適切な社会情報を利用して RS でのユーザの要求を満たすプラットフォームとなることができる [64]．ユーザの要求は，ユーザの社会活動に応じて瞬間的に変化する．一方，クラウド上におけるソーシャル情報の内容や構成は，RS における社会の変化に応じて変化する．しかしながら，従来の静的アルゴリズムを用いてソーシャル情報によりユーザの要求を満たすという課題を自律的に解決することは非常に困難である [67]．これは，ユーザがコンピュータとネットワークを便利に扱うためのスキルを必要とするためである．また，熟練したユーザであっても，クラウドの Web サイトやデータを見守るのに時間を費やす必要がある．

近年，ユビキタスデバイスやユーザの近くで動作するネットワークなどの計算機資源は，高度な ICT アプリケーションの開発において益々必要とされてきている．これらのようなリソースとネットワークを使った情報処理は，エッジコンピューティングと呼ばれている．

そして，エッジコンピューティングとクラウドコンピューティングを統合するための新しいテクノロジとしても，IoT が注目されている．IoT は，生活の質の向上と世界経済の成長に貢献する事に対して重要な家庭用および業務用アプリケーションを提供することが期待されている [2]．現在，IoT に関する最新の研究は，エッジコンピューティングとクラウドコンピューティングをどのように接続するかに焦点を当てている．しかしながら，Oihui Wu らは，単にそれらを接続するだけでは不十分であると主張しており，それを超えて，モノは，身体的および社会的な実世界を自分自身で学び，考え，理解する能力を持つべきであるとしている [81]．エッジとクラウドの両方において，オフィスでユーザの活動を支援するためにパーソナルアシスタントエージェント（PA）とサービスエージェント（SA）を含む AS に接続するための IoT アプリケーションのアーキテクチャが開発されている [9]．

本章では，デバイスによってキャプチャされたデータやそのセンサ等をエッジリソースと呼ぶ．さらに，センサを制御し，ローカルコンピュータでそれらを分析するプログラムも，エッジリソースと呼ぶこととする．エッジリソースによって取得された生のデータと生のデータから変換された分析データは，クラウド上のデータベースに保存可能である．これらのデータを含む Web データおよびプログラムは，クラウドリソースと呼ぶ．IoT アプリケーションは，これらのエッジリソースとクラウドリソースを使用して，生活におけるユーザの活動を支援することが期待されている．

図 4.1 に本章で示すリソースコネクタプラットフォームが実現する各層を支援可能な大規模なエージェント型 IoT アプリケーションを示す．本章では，3 章のリソースコネクタアプローチを用いて全ての IoT アプリケーションの層の要素を制御及び共通のプロトコルにより協調可能するために，Application Component 層の要素であるクラウドリソースと Application Devices 層の要素であるエッジリソースに対する実装による実現可能性の検討を行う．しかしながら，IoT アプリケーションのコミュニケーション部分を司る Edge Network については 5 章において検討を行う．

また，本章では，上述の各層に対するリソースコネクタプラットフォームの実現と共に，大規模なエージェント型 IoT アプリケーションを開発可能とするために，インターネットとクラウドの最新技術をエージェントプラットフォームに適応し，大規模開発に対応したエージェントモデルとそのプラットフォームの実現をする．

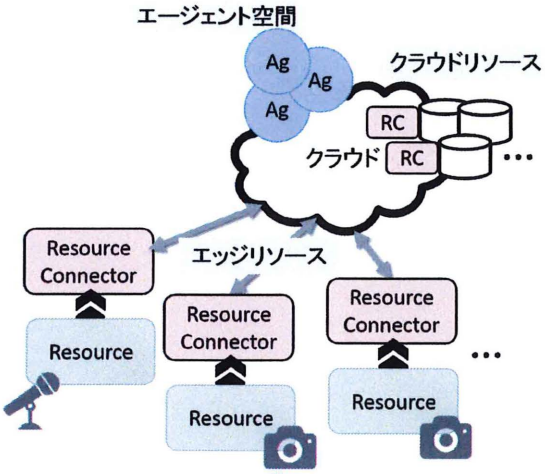


図 4.1　各層を支援可能な大規模なエージェント型 IoT アプリケーション

本章の構成は以下のとおりである．まず，4.2 節では，エッジリソースとクラウドリソースを接続する IoT アプリケーションのアーキテクチャを提案する．次に，4.3 節では，提案されたアーキテクチャにより，エージェントがリソースコネクタ（RC）という名前のプログラムを介してリソースを制御することが可能な AS の設計を行う．最後に，4.4 節では，IoT アプリケーションのプロトタイプとして，ユーザとのボーカルインタフェースと Web インタフェースを備え AS のパーソナルアシスタントエージェントを使用して，オフィスで作業するユーザ支援するアプリケーションの開発を行った．

4.2　エッジリソースとクラウドリソースを接続するエージェントスペースの設計

AS は，図 4.2 に示すように，エッジリソースとクラウドリソースによって推薦された有用なサービスに，ユーザがアクセスできるように設計されている．本章では，エージェントをエッジリソースまたはクラウドリソースに対して接続するプログラムを RC と呼ぶ．RC はエージェントメッセージをエージェントに送信し，エージェントのメッセージを AS 内のエージェントから受信することを可能とする．ユーザを支援するアプリケーションを実現するために，エッジリソースとクラウドリソースを接続するエージェントと RC で構成される集合は，図 4.2 に示すようにコテリ (Coterie) と呼ぶ．

AS 内の各エージェントは，知識ベースのフレームワークに基づいて行動することを決定し，社会的知識を使用し，ユーザの好みや活動に応じて，AS 内の他のエージェントと協調してユーザの要求を実現する．LAN およびインターネットを介して接続されたコンピュータで動作する一連のエージェントランタイムシステムのことを AS プラットフォームと呼ぶ．通常，AS プラットフォームはエージェントプラットフォームによって実現される．AS プラットフォームは，エージェントと RC が互いに通信するための一連の基本協調プロトコルと基本オントロジのセットを提供する．エージェントと RC のグループは，アプリケーション指向の協調プロトコルと基本プロトコル，そして基本的なオントロジを使用して設計されたオントロジを使用してアプリケーションロジックを実現する．このオントロジは，AS 内で定義された共通の記号を用いてリソースとエージェントをリンクするためのものであり，デバイスやプログラムの仕様に従って記述されたリソースの詳細と，エージェントによるリソースの使用状況に応じて記述されるエージェントの詳細からなる．

本章では，ユーザのアクティビティやそれらを取り巻くオブジェクト，そして，クラウドでデータを読み書きするいくつかのサービスエージェント (SA) を監視するように設計されたパーソナルアシスタントエージェント（PA）等を紹介する．PA は，特定のユーザからキーボードインタフェース，ボーカルインタフェース，ジェスチャインタフェースなどを介して取得した要求を処理するために機能する．SA は，開発者が指定したタスクに従って特定のドキュメント，Web ページ，またはデータベースを読み書きし，他のエージェントに対して情報を提供する．

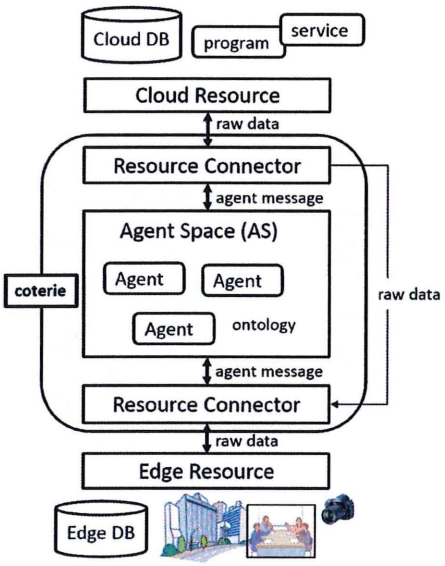


図 4.2　エッジリソースとクラウドリソースを接続する IoT アプリケーションのアーキテクチャ

RC は，部屋に配置された特定のデバイスまたは Web（クラウドリソース）の特定のプログラムからのデータを，エージェントに送信されるメッセージを構成する記号に変換するように設計されている．この時送信されるメッセージは，エージェントがコンポーネントとして実現するアプリケーションロジックによって定義されたプロトコルとオントロジのセットに基づいて構成される．たとえば，Speech-To-Text エージェントと呼ばれる RC には，マイクからの音声信号を認識し，それをテキストに変換するプラグインプログラムが含まれている．次に，RC はその文を含むメッセージをそれと話している特定のユーザを監視する PA に送信する．

4.3　エージェントスペースとリソースコネクタで構成されるコテリの設計と実装

4.3.1　エージェントスペースのためのエージェントプラットフォーム

エージェントプラットフォームは，LAN とインターネットを介して接続されたエージェントのランタイムシステムと開発支援システムの集合である．OMAS エージェントプラットフォーム（OMAS-P）は，Allegro Common Lisp を使用して開発され，様々な種類のエージェントベースのシステムの開発のためにエージェントのプラットフォームとして使用されている [9]．また，OMAS-P は，知的なアプリケーションロジックを実現する知識ベースのマルチエージェントシステムの開発に適している．PC 上で実行される OMAS-P のランタイム環境はコテリと呼ばれ，各 OMAS-P に配置されたいくつかの通信エージェントを使用してインターネット経由で他の OMAS-P に接続することができる．したがって，いくつかの OMAS-P は，エージェント

が相互にエージェント名をインデックスし，蓄積することによってエージェントメッセージを送受信できるため，複数の AS を統合することが可能である．また，エージェントプラットフォームを使用する開発者は，必要に応じて動的にエージェントをプラットフォームに追加することができる．したがって，コテリの概念はより大規模な AS を開発するのに適している．

本章では，RC のプラットフォームとして，3 章のリソースコネクタプラットフォーム (RC-P) を拡張して開発を行った．図 4.3 に示す環境において，Windows OS 上にインストールされたプラグインモジュールを使用し，デバイスやソフトウェアをコントロールするための提案システムである．また，RC-P は C#（.NET Framework）及び Python を用いて開発を行った．

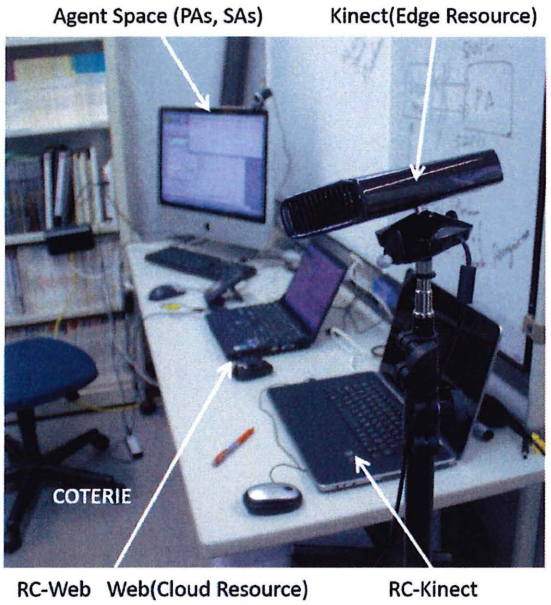


図 4.3　Web リソースとエッジリソースを接続するコテリの実験システム

図 4.3 の RC-P は，OMAS-P と同じ通信プロトコルで接続されており相互に通信可能となっている．また，クラウド及びエッジのリソースを接続するために，インターネットを介した接続を確立するように設定を行った．しかしながら，これらのコテリの概念をインターネットを介したネットワークにおいて構築する際，クラウド及びエッジのリソースにおいては様々な問題が発生することが分かった．

4.3.2　大規模開発における既存エージェントプラットフォームと試作したリソースコネクタプラットフォームの問題点

コテリを形成する既存のエージェントプラットフォームを大規模開発に対応させ，図 4.1 のような大規模な IoT アプリケーションにおけるエージェント指向開発に対応させるためには，上述の OMAS や JADE 等の 3 章や関連研究で示した既存のエージェントプラットフォームにおいては，エージェントメッセージを交換するためのエージェントメッセージ空間に複数の問題点が存在する．

図 4.4 に既存エージェントプラットフォームにおけるエージェント及びエージェントメッセージ空間の接続に関する図を示す．既存のエージェントプラットフォームは基本的に外部のエージェント空間とエージェントメッセージ空間を接続するために，Transfer Agent(XA) のような外部のエージェント空間と接続を行う仕組

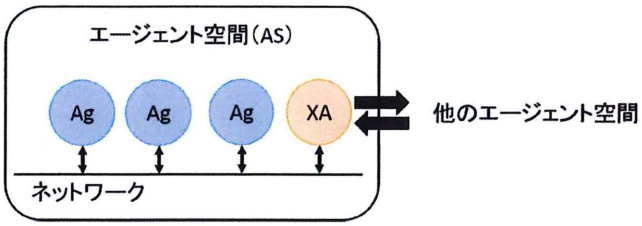


図 4.4　既存エージェントプラットフォームにおけるエージェント及びエージェントメッセージ空間の接続

みが必要となる．しかしながら，この接続によって形成されるエージェントメッセージ空間は大規模開発に対応した実用的な仕組みを目指すに際して，以下の問題点を持つ：

P1. ネットワーク及びインターネットを介してエージェント空間が接続する場合，接続が主にローカル LAN ベースの考え方をしており，グローバル IP を利用し，UDP のブロードキャストや TCP によるポートを複数指定した相互接続，そしてネットワーク自体を RMI(Remote Method Invocation) ベースの仕組みを用いた接続を行う事でエージェント同士の通信を透過させている．そのため，実装がローカル LAN の M2M(Machine to Machine) 向けとなっている．

P2. 通信のセキュリティが担保で来ておらず，メッセージの暗号化が出来ていない

P3. エージェントの仕組みは分散処理に向いているものの，P1 の問題も含めて，実装上既存のエージェントプラットフォームはスケーラビリティを確保することが難しい

P4. エージェントネットワークシステムの実装としてメッセージを保持する仕組みが存在しないため，永続性の確保が難しい

上述の問題点を解決し，大規模開発に対応可能な新たなエージェントメッセージ空間を構築するために，インターネットとクラウドの最新技術をエージェントプラットフォームに適応することで，大規模開発に対応したエージェントモデルを実現し，大規模 IoT アプリケーションにおけるエージェント指向開発のためのプラットフォームの実現を目指す．

4.3.3　大規模な IoT アプリケーションを開発するためのエージェントプラットフォームの提案と設計

新たなエージェントメッセージ空間の形成のためには，以下をエージェントメッセージ空間の要件として持ち，それぞれの問題を解決する必要が有る．これらは前節で述べた問題点を解決するための手法としてそれぞれが対応する．

S1. 既存のエージェントプラットフォームのエージェントメッセージ空間を同じエージェント空間にいるように透過的に接続し，クラウド上のメッセージ空間で共有

S2. TLS(Transport Layer Security) 1.2 を用いたエージェントメッセージの暗号化通信によるセキュリティの確保

S3. 多量にメッセージを処理可能であり，スケーラビリティを確保できるメッセージングモデルの採用

S4. エージェントメッセージ空間のエージェントメッセージレベルでの永続化

また，これらを実現する為には，実装の上でセキュリティを確保し，スケーラビリティを確保しやすいモデルが必要である．本モデルには，Publish/Subscribe モデルが適当であると考え，問題の解決に利用する．図

4.5 に Publish/Subscribe モデルの図を示す．Publish/Subscribe モデルとは，非同期のメッセージング手法のことであり，スケーラブルで疎結合であることが知られている [23]．メッセージを送出する Publisher と送出先の Broker 上のメッセージ空間である Topic，そしてそれを購読し，配信されたメッセージを Topic 経由で受け取る Subscriber で構成される．Publisher と Subscriber は疎結合であり，Broker サーバ及びトピック（Topic）に対して通信さえできれば，ネットワーク構成を互いに知る必要がないという利点が存在する．また，1 つの Topic に対して複数の Publisher 及び Subscriber を接続可能であるため，多対多のメッセージ空間を構築可能である．

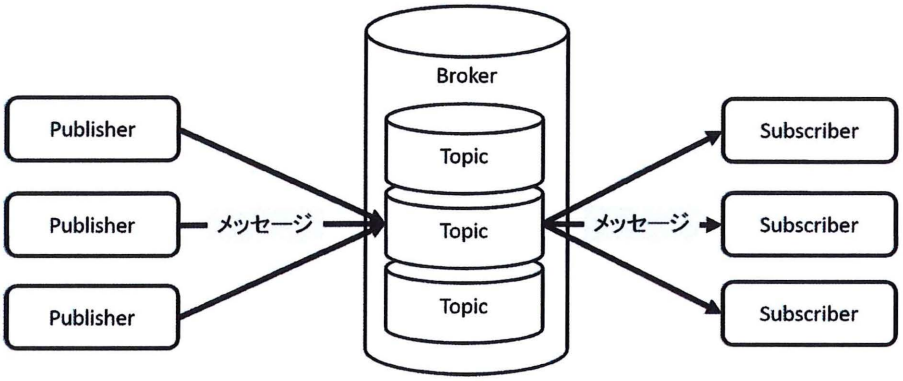


図 4.5　Publish/Subscribe モデル

本章では，上述の Publish/Subscribe モデルを用いて，大規模な IoT アプリケーションを開発するためのエージェントプラットフォームを実現する為のエージェント空間の形成のために，図 4.6 に示すエージェントプラットフォームの機構を提案する．

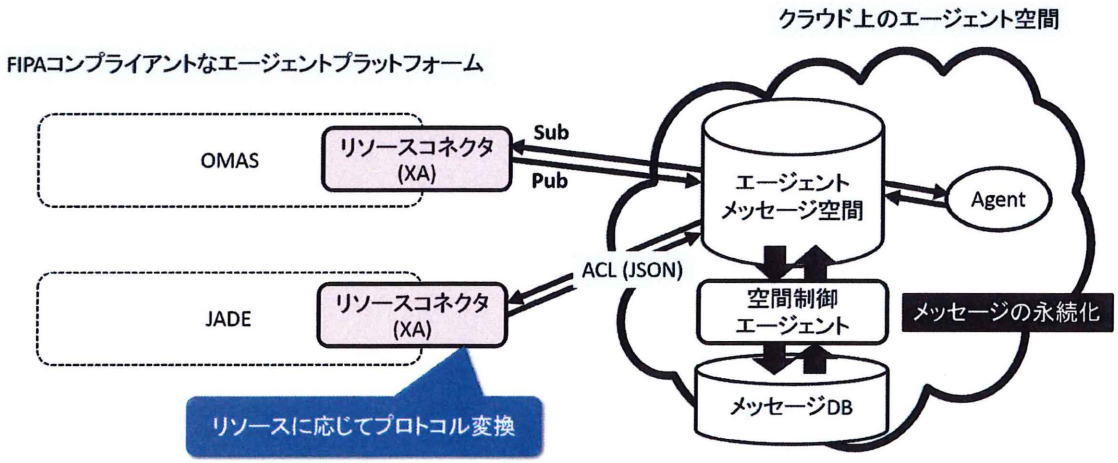


図 4.6　提案する大規模な IoT アプリケーションを開発するためのエージェントプラットフォーム

本エージェントプラットフォームは，RC を用いて接続する対象の FIPA コンプライアントなエージェントプラットフォームのメッセージ空間を情報リソースとして捉え，それぞれのプラットフォームに XA を配置することで，クラウド上のエージェント空間においてエージェントメッセージを同一のメッセージ空間にいるように透過的に扱う事を可能とする．このことにより，情報リソースとなっている複数のエージェントプラットフォームと共に各エージェントのメッセージ交換を可能とする．提案するエージェントメッセージ空間は，

FIPA-ACL 互換の JSON 形式のプロトコルによってエージェントメッセージが通信される．そのため，リソースコネクタ (XA) は，対象エージェントプラットフォームのエージェントメッセージ空間に応じたプロトコルの変換を行う．また，リソースコネクタ (XA) とエージェントメッセージ空間の接続は，Publish/Subscribe モデルのメッセージング手法を用いて接続することで，多地点での通信を可能とし，同時に TLS を用いることでセキュアな暗号化通信を実現する．これらの動作により，S1 から S3 による問題点の解決を行う．

また，エージェントメッセージ空間のエージェントメッセージの永続化のために，空間制御エージェントは，常時エージェントメッセージ空間を監視し，メッセージをメッセージ DB へ保存する．また，同時にエージェントメッセージ空間からの要求に従い，エージェントメッセージ空間に対してメッセージ DB に保存されたメッセージを送信する．空間制御エージェントは，デーモンプロセスとしてエージェントのコテリ以上の数が起動することでそれ自体が停止することを防ぐ．また，メッセージ DB は多量の書き込みを同時に行う必要が有るため，KVS（Key-Value Store）型のデータベースとその制御プログラムによりシステムの構築を行う．そして，エージェントプラットフォームやエージェントがエラーにより異常終了した場合，メッセージ DB よりエージェントに対してメッセージを与え，再度状態を戻す機能を持つ．さらに，特定のエージェントメッセージが来た際に，メッセージ DB に保存されたエージェントメッセージを再度送信することが可能とする機能を持つ．空間制御エージェントのこれらの機能とメッセージ DB により，エージェントメッセージ空間におけるメッセージの永続化を実現することで，S4 による問題点の解決を行う．

4.3.4　大規模な IoT アプリケーションを開発するためのエージェントプラットフォームの実装

本節では，4.3.3 節において提案した各機構を実現する為に，各機能の実装を行った．各機能を実装する際のライブラリや実装方法の選定では，それぞれがスケーラブルであるかセキュアである点を考慮し，選択を行った．

エージェントメッセージ空間として，VerneMQ[*1]を利用することで MQTT Broker の Topic としてメッセージ空間の構築を行い，エージェントメッセージのための Publish/Subscribe モデルのプロトコルとして MQTT を用いて実装を行った．この MQTT Broker は TLS1.2 に対応しているため，セキュアな通信を行うことが可能であり，スケーラブルであることが分かっている．さらに，リソースコネクタ (XA) は，C#及び Python を用いて実装を行い，空間制御エージェントは Python と MQTT を扱うためのライブラリである Paho[*2]のライブラリ群を利用した．

そして，空間制御エージェントにおいては，メッセージ DB を扱うために，ライブラリとして redis-py[*3]を用いた．メッセージ DB は，Python によってインタフェースの実装を行い，KVS の機能と内部で利用する Publish/Subscribe のためのブローカ機能を利用するために Redis[*4]を，さらに永続化を補完するために MariaDB[*5]を用いて構築を行った．

以降の実装と評価では，それぞれのエージェントプラットフォームが本エージェントプラットフォームの機構を利用してセキュアなメッセージ空間を実装により実現する．この実装により評価することで，本エージェントプラットフォームの実現を確認する．

---

*1 https://vernemq.com/
*2 http://www.eclipse.org/paho/
*3 https://github.com/andymccurdy/redis-py
*4 https://redis.io/
*5 https://mariadb.com/

4.3.5　オフィス業務を支援するエージェントベースの IoT アプリケーションの実験的設計

次に本節では，エッジリソースとクラウドリソースを接続する RC を使用して提案されたアプリケーションの機能を検証するために，簡単なテストベッドとしてオフィス作業支援のタスクを選択した．本システムは，提案された方法によって構築され，以下のシナリオを達成するソフトウェアを構築できるかどうかを評価することで検証を行った．まず，シナリオとしては，旅行代理店に勤務するユーザがクライアントと同僚と共にリゾートエリアに旅行する計画を立てるタスクを達成することとした．ユーザは多くの Web ページを検索し，優れた航空会社，鉄道マップ，レンタカー，ホテル，サイトを確認し，顧客のさまざまな種類の要求を非常に迅速に満たすことができる必要がある．

図 4.7 にオフィス作業支援のためのシナリオ用に試作された RC と AS で構成されるコテリの単純な構造を示す．このコテリには，上記のシナリオでユーザを支援する PA，データベースサービスを提供する 2 つの SA，PA，そして PA のための Web ページ検索サービスが含まれる．

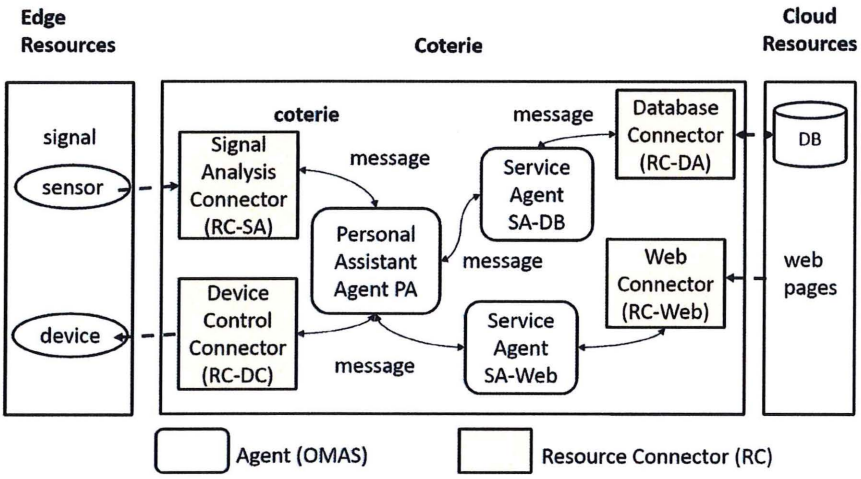


図 4.7　RC とエージェントで構成されるコテリの構造

図 4.8 に PA と協働する PA と RC の実験設計を示す．それらはオフィス作業を支援するために，図 4.7 に示されているエージェントベースの IoT アプリケーションで動作する．PA は，Decision エージェント，Voice Conversation エージェント，User-Watch エージェント，および Task Support Agent という名前の OMAS エージェントで構成されるマルチエージェントシステムである．ビデオ会話エージェントは，RC-Speech-To-Text から音声テキストを含むメッセージを受信し，メッセージを RC-Text-To-Speech に送信する．User-Watch エージェントは，ナレッジモデルで定義されたユーザのアクションを含むメッセージを受信することができる．RC-Kinect という名前の RC は，Kinect デバイスによってキャプチャされた信号からエージェントが認識可能なアクションの記号を生成することができる．Task-Support エージェントは，RC-Visio にコマンドを含むメッセージを送信してディスプレイされる項目を生成及び値を書き込み，ユーザが RC-Visio からディスプレイに何かを書き込むときに Visio プログラムによってキャプチャされるイベントを含むメッセージを受信する．そして，RC-iTunes は，Decision Agent（DA）ディレクトリからメッセージを受信して，ユーザにビデオ映像を表示するように iTunes アプリケーションを制御する RC である．

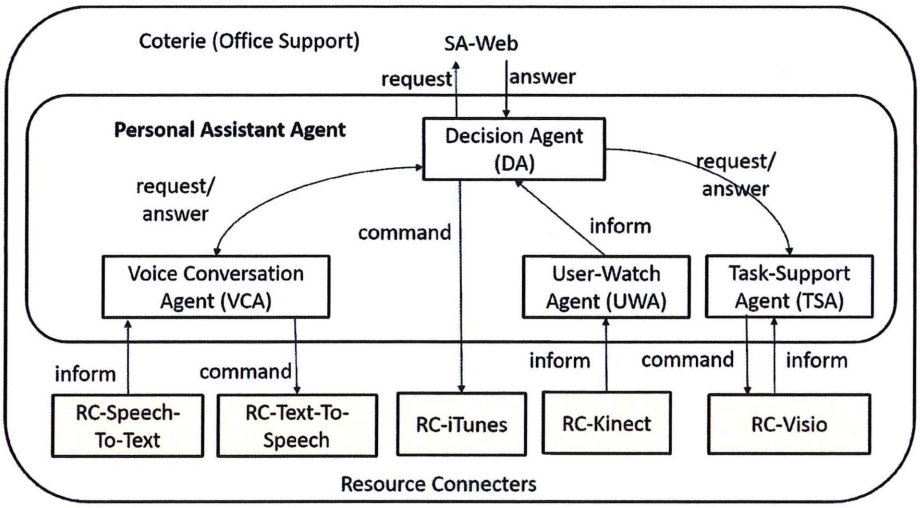


図 4.8　OMAS エージェントと RC で構成されたパーソナルアシスタントエージェント（PA）の構造

4.3.6　OMAS-P と RC-P の統合開発環境

OMAS-P は，エージェント開発者のためのプログラム開発とデバッグを支援するグラフィカルユーザインタフェースを提供する．図 4.9 に，OMAS 支援機能の一部を示す．支援機能には，エージェント間の相互作用を確認するための OMAS-Message-Diagram，エージェントが送受信するメッセージの内容を監視するための OMAS-Message-Panel，各エージェントのアクションを制御およびチェックするための OMAS-Control-Panel が存在する．RC-List には，RC によって現在コンピュータ上で利用可能なリソースのリストが表示される． RC-Control / Message-Panel では，リソースの現在の操作の設定，データ送信者/受信者の構成およびステータスログが表示される． OMAS-Message Diagram では，AS で交換されたすべてのメッセージが表示され，シーケンス図によって送信者/受信者のステータスログとフローが確認できる．エージェント開発者は，これらの機能によって支援される．本章では，OMAS-P を OMAS-P と同様の方法で RC の活動を観察，制御，確認を行う事が可能とするために，RC-P の支援までカバーするように拡張を行った．

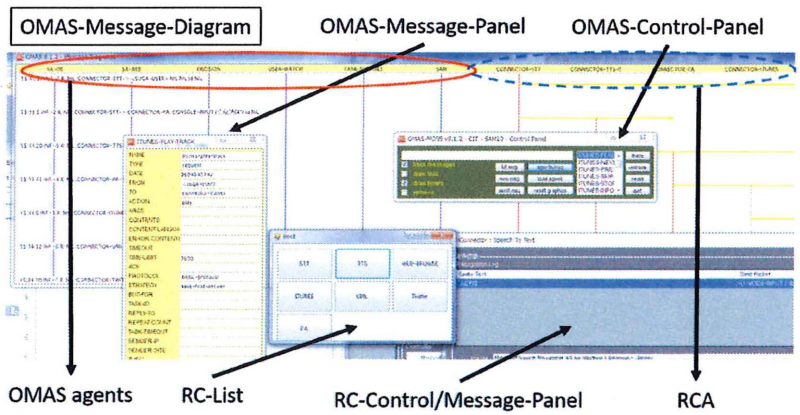


図 4.9　OMAS-P と RC-P の開発者向けツール

4.4　IoT アプリケーションによるエージェントプラットフォームの評価

本節では，対象とするシナリオ及びシチュエーションにおいて IoT アプリケーションを実装することができるかを確認することで評価を行った．

4.4.1　デスクで働くユーザを支援するシナリオ

このシナリオで PA のユーザである作業者は，図 4.10 のようにクライアントの旅行計画を立てるためにデスクで働いていることとする．同僚がユーザの机に来て，彼らが担当している計画についてユーザに尋ねる際，ユーザはなぜスケジュールが彼らが整理したものに対して遅れているのかを示し，説明したいという状況であるとする．まず，フランスのクライアントに対して「東京で 5 日間」という計画を提示したいと考えている場合，ユーザは次のように音声でエージェントと会話する壁の大きなディスプレイに PA を表示するよう PA に依頼する．

```
User> Agent. Show me the document about hotel reservations
      that I made yesterday afternoon.
PA>   This is a list of documents you made yesterday afternoon.

文書のリストは，ユーザだけが見ることができるタブレット PC に表示される．

User> Show us the Visio document titled Trip-plan-2016-11-11.
PA>   Is this right?
User> OK.

Visio ドキュメントは，2 人のユーザが同時に見ることができる大きなディスプレイに表示される．
```

次に，ユーザは，ホテル内のホテルのリストの部屋の合計価格が，総旅行予算に対してあまりにも多くの費用をかけていることを同僚に伝える．その後，ユーザは同僚の意見に基づいてより良いホテル予約にリストを変更したいと考えて行動する．そのために，ユーザは昨日の午後にホテル予約サイトのいくつかのブックマークを用意し，掲示を行う事とした．

```
User> Agent. Show me a list of bookmarks that I created in the late afternoon.
PA>   This is a list of web pages you opened between 4 pm and 5 pm.
User> Show us number 5.

そして，ホテル予約サイトの Web ページが大きなディスプレイに表示される．
```

4.4.2　音声会話エージェントの実装

図 4.11 に示す通り，PA のコンポーネントエージェントとして機能する Voice Conversation Agent（VCA）は，知識ベースのエージェントであり，トピック領域に関する拡張階層状態遷移表とオントロジのセットを

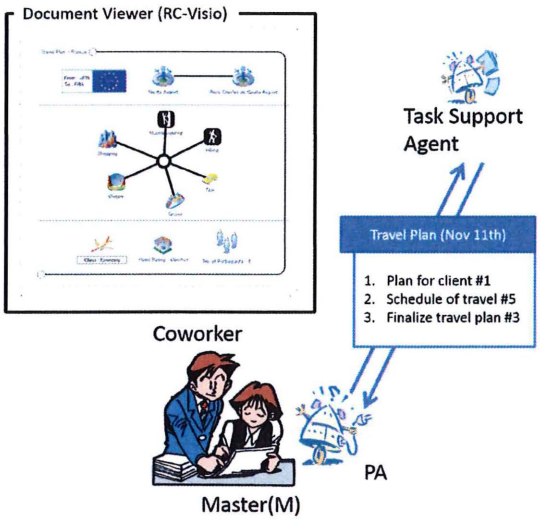


図 4.10　デスクで作業するユーザを支援するシナリオ

使用して，タスク指向の会話によってユーザの要求を検出する [70]．特定のユーザの VCA は，ユーザ，ユーザの同僚，仕事に関する専門知識などのオントロジを持ち，自分の要求を見つけることができる．VCA が要求をタスクとしてクリアすると，VCA はそのタスクを PA 内の Decision エージェントに送信する．その後，Decision エージェントは，VCA の結果に答えるために問題を解決する方法を決定する．たとえば，VCA によって送信されたタスクが「12 : 00/26/03/2016 と 24 : 00/24/03/2016 の間に作成したすべてのドキュメントを見つける」場合，Decision エージェントは，オフィスワークシナリオにおいて VCA に文書名のリストを返す．

VCA は RC-STT（Speech-To-Text）と RC-TTS（Text-To-Speech）を制御して，音声でユーザと対話する．RC-P 上で動作する RC-STT は，ユーザが話す文を認識し，認識結果のテキストを VCA の入力変数に送る．VCA がマスタと会話する文章を作成すると，VCA は OMAS プロトコルで RC-TTS にメッセージを送信する．

この例では，ユーザが Decision エージェントに「同僚のリゾート地のビデオを再生する」という要求を送信すると，Decision エージェントから RC-P 上で実行されている RC-iTunes にコマンドメッセージが送信される．

4.4.3　RC-Kinect の実装

PA は，音声会話以外のマスターのアウェアネスを用いて，マスターとのやりとりを通じてユーザの要求を理解する必要がある．そのため，ユーザとその周辺を見るために，Kinect センサデバイスとライブラリとして Open NI を用いたソフトウェア開発環境をこの実験で使用し，実装を行った．RC-Kinect エージェントは，図 4.12 に示すように，OMAS エージェントへの接続機能（Kinect-Connect）と Kinect 用 OpenNI を用いて作成されたプログラムをラッピングするラッピングモジュール（Kinect Open NI）で構成される．これらは，図 4.13 の設計に従い，必要なプロファイルとリソースに対応するプラグイン・プログラム，シーケンス図に示された操作が可能なエージェントと RC を開発した．

図 4.13 の右側には，User-Watch エージェントと Kinect Agent の間のメッセージダイアグラムが示され

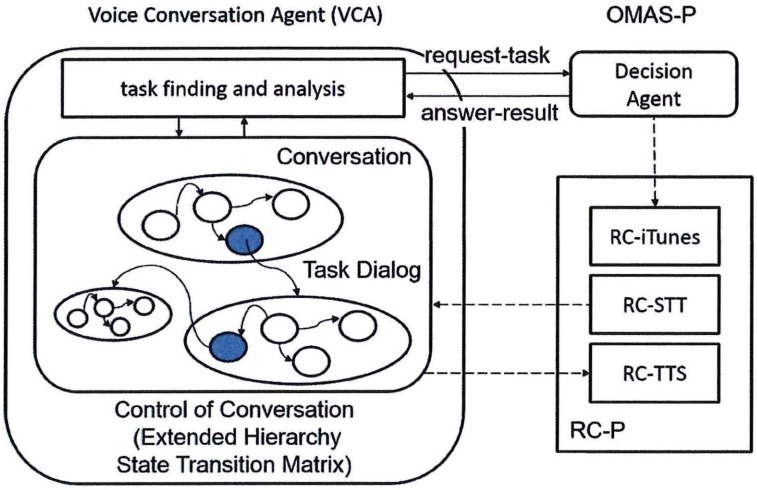


図 4.11　OMAS-P の音声会話エージェントと音声会話用リソースコネクタ

ている．図 4.13 は，Kinect Open NI が Kinect エージェントに接続されたときに，「Person-1 が出現」と「Person-1 が退出」というイベントに対応する User-Watch エージェント（この場合は OMAS PA）へのレポートメッセージを示している．RC-Kinect ウィンドウ用の RC-P-IDE には，Kinect で検出された人物のスケルトンと画像，および現在検出されている人数などの情報がデバッグ用に表示される．

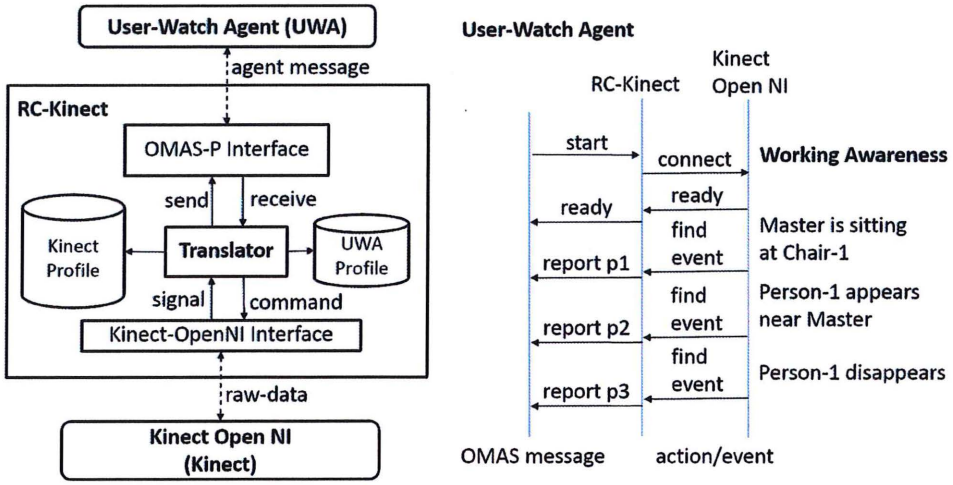


図 4.12　RC-Kinect の設計

4.4.4　Web ページエージェントの実装

図 4.14 の Web エージェントは，SA-Web からの要求に応じて Windows OS 上で実行されるプラグインモジュールを使用し，Web ページをデータの集まりとしてマイニングを行う RC である．Web エージェントは特定のデータマイニングアルゴリズムを使用して，Web ページ内のいくつかのテキストとイメージを処理し，Web エージェントと SA-Web エージェント間のプロトコルに基づいて設定されたオントロジによって定義さ

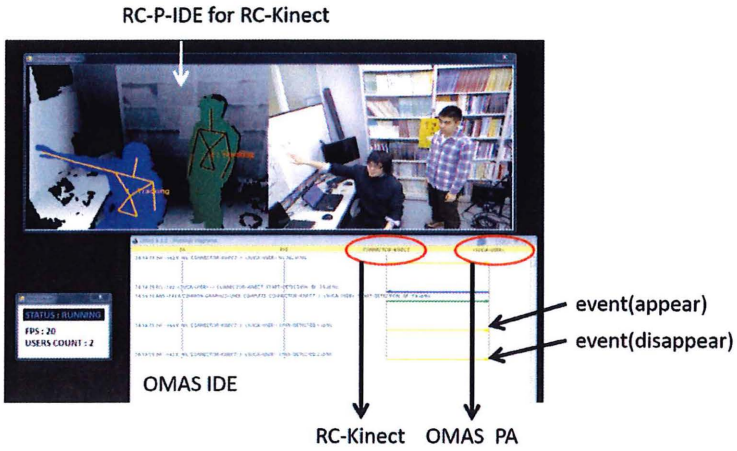


図 4.13　RC-P の開発支援ツール

れた知識表現を含むメッセージに変換を行う．

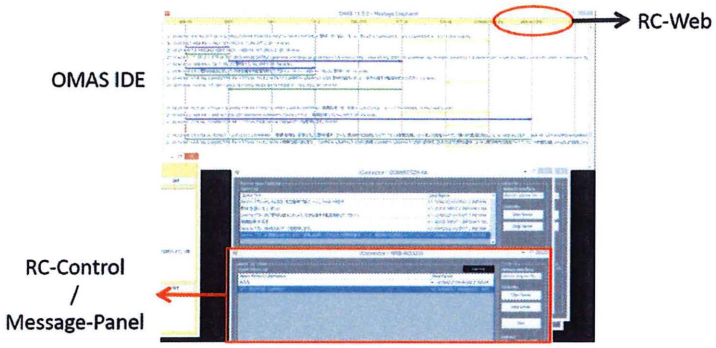


図 4.14　Web エージェントのデバッグビュー

4.5　まとめ

本章では，センサデバイスとそれによって取得されたデータをエッジリソースと呼ぶ．さらに，それらを制御し，ローカルコンピュータでそれらを分析するプログラムもまたエッジリソースと呼ぶ．エッジリソースによって取得された生データと生データから変換された分析データは，クラウドデータベースに保存可能である．Web データとプログラム，および前述のデータは，クラウドリソースと呼ぶ．IoT アプリケーションの開発における困難な問題の 1 つは，センサやパターン認識プログラムなどのエッジリソースをデータベースや Web ページなどのクラウドリソースに接続するためのアーキテクチャを設計することである．そこで本章では，エッジリソースとクラウドリソースをエージェントに接続する IoT アプリケーションを開発するためのエージェントベースの開発方法論を提案した．

図 4.2 は，AS と RC を使ってインテリジェントな IoT アプリケーションを実現するために提案されたアーキテクチャを示している．AS は，IoT アプリケーションのサービスを提供するさまざまな種類のマルチエージェントシステムで構成される．たとえば，センサを使用してユーザの行動や環境を監視したり，クラウドリ

ソースからユーザの知識を検索することができる．マルチエージェントシステムは，スキルを持たないユーザがボーカルインタフェースやジェスチャインタフェースを使用してクラウド上の多くのサービスを利用するの事を支援できることが期待される．

まず，シナリオによる実装では，本研究の提案に従って，機能の実装及び実装の確認ができたため，シナリオの目的を達成するのに十分な機能を備えていることが確認された．本章において，提案された方法の 1 つの強みは，クラウドとエッジの各リソースを再利用できることである．

次に，ユーザが要求した Web ページを表示するために，エッジリソースの音声インタフェースを使用しているユーザと会話する PA の IoT アプリケーションを紹介した．PA の備える機能は限られているが，提案されたアーキテクチャでエージェントと RC（リソース）を簡単に置き換えることができるため，PA の構造は柔軟であると考えられる．

さらに，新しいエージェントメッセージ空間の設計と開発により，大規模開発に対応可能な新たなエージェントメッセージ空間を構築できた．また，エージェントプラットフォームのセキュリティ，スケーラビリティ，永続性の向上が実現できた．よって，大規模開発に対応したエージェントモデルが実現できたと言える．

また，本章では，対象とするシナリオ及びシチュエーションで IoT アプリケーションを実装することができるかを確認することで評価を行った．実装により，各種 IoT アプリケーションは本エージェントプラットフォームにおいて実装ができたことを確認した．よって，IoT アプリケーションを構成するための 3 層のモデルにおいてすべての層をカバーし，エージェントプラットフォームによる支援ができたと言える．このことから，IoT アプリケーションを実装するためのエージェントプラットフォームとして十分な機能を備えていることが確認できた．

実装とその確認により，エージェント指向開発方法論に基づく，異なるインフラストラクチャ上で動作可能なプラットフォームの実現できた．よって，IoT アプリケーションの各階層に対応して異なる特性を持つエージェントを実装することが可能なプラットフォームを実現できたと考える．

第5章

エージェントプラットフォームを利用したIoTアプリケーションの設計方法

IoTアプリケーションシステムは，IoTアプリケーションの3層構造を参考として，クラウドアプリケーション，ネットワーク，デバイスの3階層のアーキテクチャと考えられる．ネットワークは，クラウドのインフラストラクチャ，インターネット，デバイスとインターネットを接続する無線ネットワークから構成される．デバイスは，カメラなどのセンサ，ディスプレイなどのアクチュエータおよび移動端末などから構成される．

IoTアプリケーションの各階層は，多数の要素から構成される複雑なシステムであり，これらが連携して目的とするサービスを実現する．このような階層間の要素の連携を垂直連携と呼ぶ．IoTアプリケーションにおける垂直連携は，サービス利用者の位置や状況で要素が変化するため，的確なサービスを実現するためには，動的に垂直連携を構成することが必要であるが，このための効率的な仕組みが存在しないことが，IoTアプリケーション開発の方法論の課題である．

本章では．3章と4章において提案，設計，実装を行い，開発したエージェントプラットフォームであるリソースコネクタプラットフォームを用いて動的に垂直連携が可能なIoTアプリケーションを開発するための設計方法について述べる．

5.1 Internet of Multimedia Things(IoMT) とIoMTにおける垂直連携

近年，Alviらは[3]において，スマートな異種のマルチメディアのモノが相互にやりとりし合い，インターネットに接続された他のモノと相互作用し協調して，マルチメディアベースのサービスやユーザにグローバルに利用可能なアプリケーションを容易に提供することができるIoTの新しいパラダイムとしてInternet of Multimedia Things（IoMT）を定義している．

例えばIoMTにおいて，マルチメディアなリソースの例として，ビデオ会議ツールは以下のような特徴を持つコンポーネントである．

- エンドツーエンドの遅延：ネットワーク特性によって，マルチメディアコンテンツの配信に許容可能なレートが確保されている必要がある
- いくつかの視点の提供：いくつかのマルチメディア機器（マイク，カメラなど）をプラグアンドプレイ方式で追加し，それぞれのチームの観点からさまざまな視点を提供する必要がある
- アクセス：マルチメディアコンテンツは，同期アクセスと非同期アクセスのために処理され，保存されるべきである．

2 つの遠隔チーム間の IoMT アプリケーションに関する最も挑戦的な研究課題の 1 つは，目的，状況，規模に応じてユーザのニーズを満たす動的に垂直連携可能なアーキテクチャモデルの設計である．システムのコンポーネントは，IoMT アプリケーションのコンテキストの変化に応じて特性を適応させるために動的に追加または削除する必要がある．

そして，特に IoMT の分野で垂直連携は必要とされている．なぜならば，カテゴリの違う種類のリソースの制御と，リソース間の連携を行う必要が有るからである．そこで，本章では，IoT をより拡張したパラダイムである IoMT において，動的に垂直連携可能な IoMT アーキテクチャを実現する為に，IoT アプリケーションの設計方法の提案を行う．

まず，この問題に取り組むために，前章で示したエージェントプラットフォームを用いて，マルチメディアデバイスをサーバ内で動作するプログラムに組み込むいくつかの IoT アプリケーションを試作した．このアプリケーションでは，センサとアクチュエータをエージェントベースのサブシステムにモジュール化して IoT アプリケーションに動的に組み込む構造とした．また，Alvi の IoMT アーキテクチャ [3] を参考とした IoT アプリケーション用の 5 層モデルを定義した．

しかしながら，IoT アプリケーションのアーキテクチャモデルは，マルチメディア通信設備とクラウド機能をアプリケーションに組み込み制御することができなかったため，リモート環境下におけるモデル [52] には適用できないものであった．そこで本章では，IoMT システムのアーキテクチャにマルチメディア通信リソースとクラウドリソースを組み込み，設定し，制御するための新しいチャネル概念を導入することで，動的に垂直連携可能なアーキテクチャとその設計方法を実現する，

このチャネル概念は，リソース間の通信とそれらの制御を行うことが可能なエージェント間の通信を明確に分けることで，記号としての制御と生のデータをやり取りするストリームを明確に分けることが可能となり，各要素が扱う対象のリソースのプロトコルに依存することがなく，協調することを可能とすることができる．この概念により，各層の動的な垂直連携のために必要な Edge Network 部分の共通のプロトコルによる制御が可能となり，IoT の 3 層の全ての要素が共通のプロトコルにより協調可能となる．

5.2　リソースをモジュール化するための IoT アプリケーションのエージェント型アーキテクチャモデル

5.2.1　リソースのモジュール化のためのリソースコネクタ

今回，3 章の構造に対して，マルチメディアデバイスを定義を追加することで，リソースと異なる部分を分離した．その定義から見ると以下のように設計できる．以降は，本定義を利用し，アーキテクチャの実現を図る．

図 5.1 に，モジュールの一般的な構造を示す． これは，アプリケーションに組み込まれたマルチメディアデバイス (Multimedia Device)，デバイスを制御するリソース（プラグインプログラム），およびリソースがアプリケーションのサービス実行エージェントと通信できるようにするリソースコネクタによって構成される．リソースコネクタ (Resource Connector) は，サービス実行エージェントとエージェントメッセージを交換し，プラグインプログラムとしてリソースを組み込むエージェントプログラムである． リソースコネクタは，リソースの開発者が記述したように，リソースおよびマルチメディアデバイスの仕様を保存するための知識ベースを有する． サービス実行エージェント (service execution agent) は，アプリケーションのユーザによって要求される要求仕様を含む要求メッセージをブロードキャストすることによって，協調するためのエージェントプロトコルを使用して適切なマルチメディアデバイスを見つけることができる [89, 39].

例えば，図 5.1 において，入力モジュールはマルチメディアデバイス（マイクロフォン）からリソース

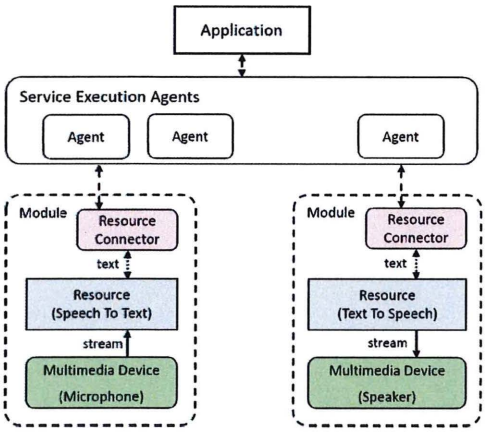


図 5.1　リソースコネクタとサービス実行エージェントを介してリソースを IoT アプリケーションに組み込むモジュールの一般的な構造

(Speech-To-Text(STT) エンジン) に向けられたストリームを有する．また，リソースコネクタは，リソースコネクタの知識ベースで定義された形式で認識されたテキストを含むエージェントメッセージを作成する．

また，出力モジュールは，サービス実行エージェントから，テキストを含むエージェントメッセージを受信し，リソースコネクタによって入力されるリソース (Text To Speech(STT) エンジン) を介してマルチメディアデバイス (スピーカ) に向かってストリームを生成する (図 5.1 の右側を参照)．

すなわち，各リソースコネクタは，その知識ベースを使用して，マルチメディアデバイスの特性に対応するストリームにサービス実行エージェントから，またはサービス実行エージェントへの，エージェントメッセージを変換する．

モジュール複合アプリケーションは，複数のモジュールを使用する．そのコアはエージェント空間と通信し，各モジュールはリソースコネクタを介してエージェントと通信を行う (図 5.1 を参照)．アプリケーションは，サービス実行エージェントとリソースコネクタのおかげで，デバイスを動的に追加および交換可能である．リソースは，場合によっては，コマンドやデータを別のリソースに送信する場合もある．

5.2.2　IoT アプリケーションにモジュールを組み込むエージェント層

図 5.2 に，アプリケーション (Applications) 層内で動作するアプリケーションに対してモジュールを組み込むためのリソースコネクタエージェント層を含む完全な IoT アプリケーションの 5 層アーキテクチャを示す．図 5.1 はこの 5 層のシステムアーキテクチャの下位層を示す．エージェントは，図 5.1 に示すように，IPC プロトコルを介してアプリケーションと通信し，エージェントベースのプロトコルを介してリソースコネクタと通信する．

ここでは，IoMT システムの動的に垂直連携可能なエージェントベースのアーキテクチャを提案する前に，IoT アプリケーションの 5 層アーキテクチャを説明を行う．まず，デバイス (Device Connectivity and Communication) 層は，物理空間内のユーザおよびオブジェクトと対話するデバイスで構成される． 次に，リソース (IoT Services and Resources) 層は，デバイスを制御するプログラムと，これらのデバイスによってキャプチャされたデータによって構成される． 最後に，リソースコネクタ (Resource Connectors) 層は，サービス実行エージェント (Service Execution Agent) 層のエージェントと通信するエージェントプログラムからなる．

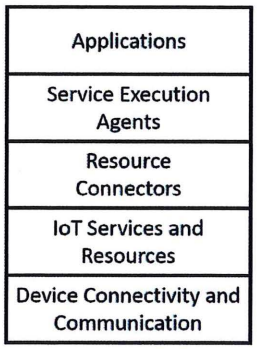
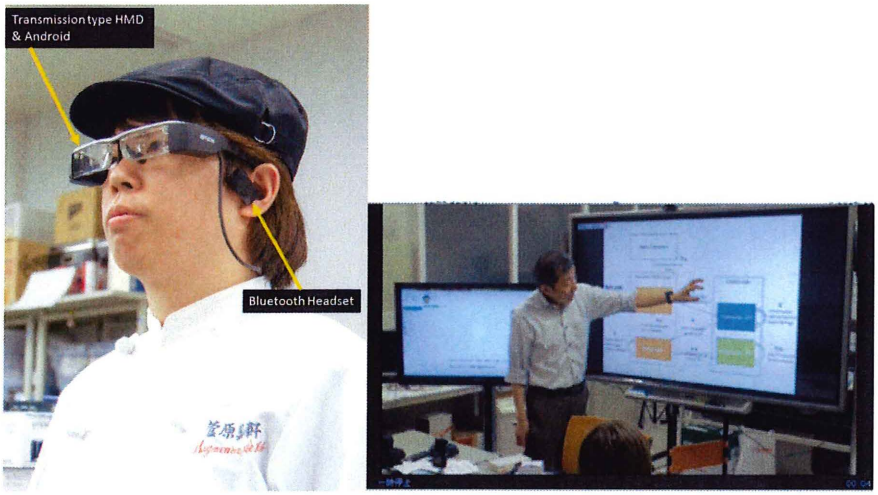


図 5.2　モノのインターネットのためのエージェントベースのアーキテクチャ



(a) Instruction support system　　(b) Conference tool

図 5.3　提案されたエージェントベースのアーキテクチャに基づいて開発された IoT アプリケーション

5.2.3　エージェントベースの IoT アプリケーションのプロトタイプ

本節では，システムの拡張という概念を示す．システムは拡張可能であることとは，ユーザが必要とする新しい種類のリソースが，動作中に新しいサブシステムによって管理することが可能であることを指す．

システムを拡張可能にする方法を検討するために，提案したエージェントベースのアーキテクチャに基づいて構築されたいくつかの小さなアプリケーションの試作を行った．各アプリケーションには，その目的を達成するために，デバイスとサービス実行エージェントにリンクされた 1 つ以上のモジュールが関係している．

Instruction support system（図 5.3a）　このアプリケーションでは，ユーザはアプリケーションから受け取った指示に従って何らかの動作を行う．ユーザは，指示を受け取るためにヘッドマウントディスプレイメガネとヘッドセットを着用する．この時，ユーザは TTS コンポーネントやスマートグラス（スマートグラスの AR コンポーネント）を介して書き込まれた各表示や指示を受け取る．指示された各ステップが達成されると，ユーザはそれをシステムに報告し（STT/音声認識コンポーネントのおかげで），次のステップを提示する．

Human detection system　Kinect のカメラは，部屋の中にいる個人の存在を検出し，この存在にフォローすることができる．各モジュールの動作により，動きと手と腕のジェスチャの検出が行われ，パーソナルエージェントに基づいてモジュールに送信されるイベントを生成する．それぞれの行動は，人工知能アルゴリズムによってこれらの通知から解釈される．

Human bearing on a chair　椅子に座っている人の位置を認識し，分析を行う（椅子の圧力センサのおかげで）．アプリケーションは，これらのデータを使用して重心を測定し，人がどのように座っているかを認識することができる．さらに，これらの結果に応じて，アプリケーションは 3D 仮想ディスプレイ上に座った状態を再現する．

Conference support　タッチスクリーン上に表示されたスライドショーを使用する会議発表者は，カメラによって撮影される（図 5.3b）．会議への出席者は，システムによって保存されたメッセージを送信することができ，スライドショーの使用に関するイベント情報は一部のエージェントによって分析される（会議中の各スライド表示時間など）．会議後，アプリケーションによってビデオ視聴者は，会議中に送信されたコメントおよびスライド解析のデータ（スライド要素内の文および単語）と共にビデオのレビューを可能となる．

5.2.4　リソースの拡張性について

前節で説明した各アプリケーションには，リソースコネクタとサービス実行エージェントによって接続された複数のモジュールが含まれる．例えば，2 番目と 3 番目のアプリケーションからのサービス実行エージェントを追加することで，1 番目のアプリケーションを統合アプリケーションに拡張することができる．

これらのリソースを 1 番目のアプリケーションに直接手動で追加することでアプリケーションを展開する場合，その都度，1 番目のアプリケーションを変更してリソース仕様に従ってデータフォーマットとプロトコルを調整する必要がある．しかしながら，提案されたエージェントベースのアーキテクチャを使用する場合，複数のエージェントメッセージを送信することによって，1 番目のアプリケーションをリソースコネクタによって他のアプリケーションと容易に接続することが可能であった．

さらに，4 番目のアプリケーションでは，複数のサイズのマルチタッチディスプレイを制御し，2 人のユーザ間の会議の前に起動するリソースコネクタを開発している．会議中に他の 3 人のメンバが参加すると仮定した場合，次に，4 番目のアプリケーションはエージェントメッセージを迅速に送信することによって，小型マルチタッチディスプレイでの動作を，他に設置されており，アプリケーションが動作可能な大きなマルチタッチディスプレイに対して動作する対象をすばやく交換することが可能となった．

したがって，提案されたエージェントベースのアーキテクチャは，アプリケーションを拡張可能にすると考える．次節においては，このエージェントベースの IoT アプリケーションアーキテクチャの IoMT アプリケーションアーキテクチャへの拡張を検討する．

5.3　IoMT におけるマルチメディアチャネルの設計

5.3.1　IoMT アプリケーションを実現するための機能

遠隔チームのための IoMT アプリケーションには，システムのアーキテクチャを構成するさまざまな種類の機能が必要である．両サイト間で発生する同期アクティビティに関する主な機能は次のとおりとする．

コミュニケーションと認識機能：　1 か所にある 1 つ以上のカメラは，ビデオストリームを別の場所に送信す

る必要がある．また，1つのステーションは，他の場所でキャプチャされたビデオストリームを受信して表示する必要がある．

データ機能： ある場所で特定の同期ツールで作成されたデータは，他の場所で同じツールを使用してアクセス可能である必要がある．

ドキュメント機能： ある場所で選択された興味のある文書は，他の場所でアクセス可能でなければならない．

例のとして図5.8bに，ビューアの一部を示す．これは，1つの場所からの1つのカメラビューと，他の場所からの1つのカメラビューを表示する，開発したアプリケーションである．複数のカメラが同じコンピュータに接続されている場合，同じ画面で複数のビューアを開くことができ，リモートなIoMTアプリケーションにおける場所の雰囲気をよりよく認識することができる．

その他の機能は，非同期アクティビティの支援に関係する機能である．例えば，IoMTシステム利用時にその場にいない人は，動作中に登録されたビデオの視聴に興味がある可能性がある．これらの種類の機能には2つの要素が必要である．第1にアーキテクチャは，ビデオおよび交換された文書を格納するための情報システムを所有していなければならない．例えば，PDF文書やJPEG画像などのユニバーサル形式に準拠している場合は，問題なく配信することが可能である．第2に，彼らそれらは適応するビューアを必要とする．本システムでは，動画はアノテーションとともに保存されているため，IoMTシステム動作中に発生した重要なイベントを簡単に検索することが可能である．

5.3.2　マルチメディアチャネルの概念とIoMTシステムの動的な垂直連携

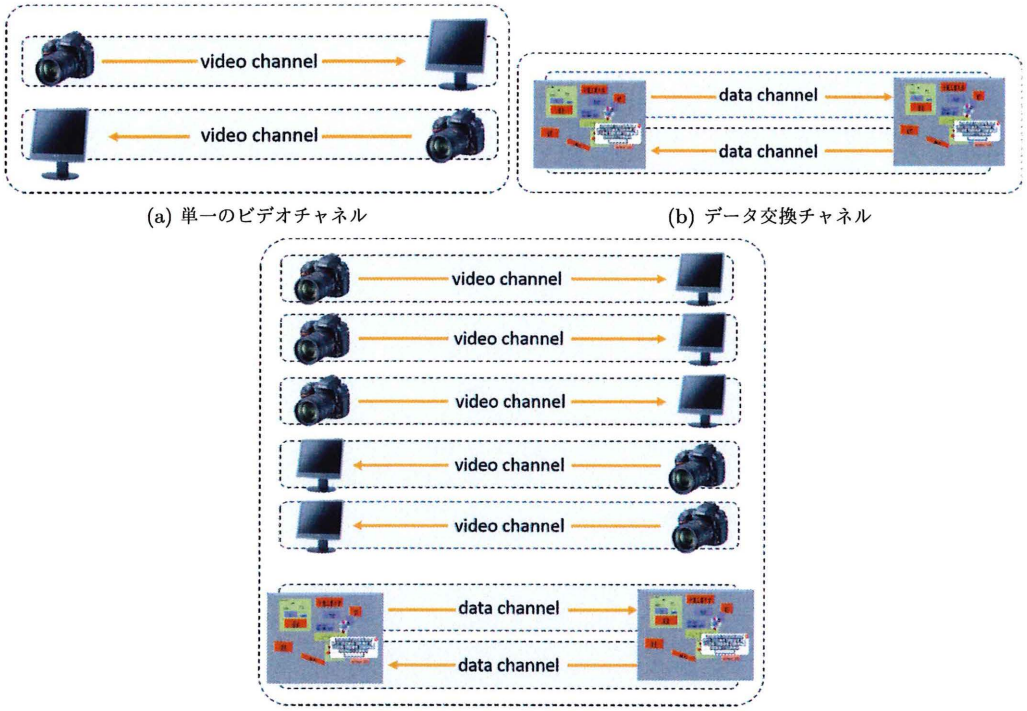
これらの要件をすべて満たすために，論理的にアーキテクチャのストリームをある場所から別の場所に送信することが可能な並列で有向なチャネルとして構成する．この時，ストリームは連続的であっても離散的であってもよい．連続的なストリームはビデオストリームであるが，離散的なストリームはデータとイベントが交換されるストリームである．この仕組みは，利用可能なカメラの数に応じて新しいビデオチャネルを追加することができ，ドキュメントおよびデータチャネルの新しいインスタンスをアクティブにできるという意味で迅速に利用可能であり，動的に垂直連携可能である．また，新しいタイプのチャネルを追加することが可能であるが，チャネルの構造に従って開発する必要がある．IoMTシステムの利用中に開かれるチャネルの数は，接続されているデバイスの数に応じて，偶数または奇数である場合がある．図5.4aは単純なビデオチャネルを示し，図5.4cは7つのチャネルが開いているより複雑なアプリケーションを示している．

5.4　動的に垂直連携可能なIoMTシステムのチャネル構造の設計

図5.5では，図5.4bに示すような，リモートコラボレーション支援システムのサブシステムを含む一対の同期データ共有（synchronized data sharing/SDS）アプリケーションを示す．ローカルサイトで動作するSDSアプリケーションは，入出力サービス実行エージェントと連携し，動作を行う．

入力サービス実行エージェント (Input Service Execution Agent) は，入力および送信者 (sender) のリソースコネクタと協調して動作する．また，出力サービス実行エージェント (Output Service Execution Agent) は，受信者 (receiver) および出力のリソースコネクタと協調して動作する．

入力のリソースコネクタは，入力リソース (Input Resource) を制御して，マルチタッチディスプレイ上のイベントを取得し，送信リソースコネクタによって制御される送信リソース (Sender Resource) にイベントデータを送信する．受信リソースコネクタは，受信リソース (Receiver Resource) を制御してデータストリームを受信し，それらを出力リソース (Output Resource) に送信して，出力のリソースコネクタによって制御される



(a) 単一のビデオチャネル　(b) データ交換チャネル

(c) 組み合わせた複合チャネル

図 5.4　動的に垂直連携可能な IoMT アプリケーション：エージェントによってオーケストレーションされたビデオチャネルとデータチャネルで構成される IoMT システム

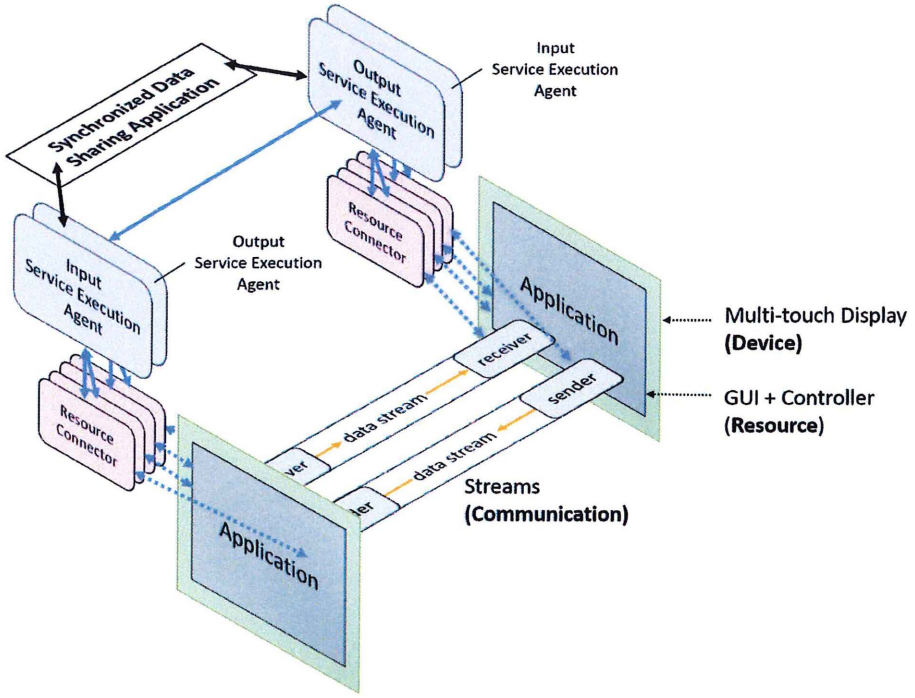


図 5.5　リモートコラボレーション支援システムにおける動的に垂直連携可能な IoMT システムのチャネルの構造

ストリームに記述されたイベントを表示する．

5.3.2節で示したように，動的に垂直連携可能なチャネルの概念は，リモートコラボレーションを実現するために必要である．しかしながら，各チャネルを動的に垂直連携するためには5.2節で提案されたアーキテクチャでは対応できない．このようなチャネルを実装するために，提案するIoMTアーキテクチャでは，サービス実行エージェント層 (Service Execution Agent Layer) とマルチメディア装置と通信層 (Device Connectivity and Communication Layer) との間のデータの流れを分離する．このデータ分離により，リソースコネクタより上のレイヤでリソースのデータストリームを処理する必要がなくなる．したがって，このアーキテクチャは，様々な形態のマルチメディアリソースデータに適応可能な構造を実現することができる．

また，エージェントメッセージに生のデータが混在しない状態となり，共通のプロトコルによって扱うことが可能なエージェントの空間が形成可能となる．さらに，ネットワークの調整をエージェントが可能となり，IoTの3層のレイヤ構造で示していたEdge Networkの各要素をリソースとして扱うことが可能となる．

5.5　エージェントアーキテクチャに基づく動的に垂直連携可能なIoMTアプリケーションの設計方法

IoMTシステムは，IoTアプリケーションで構成されたシステムである．5.4節で説明されているように，IoMTシステムに必要なアプリケーションの性質を考慮するために，5.2節に示すIoTアーキテクチャを適応させる必要がある．一部のモジュールは，リモートサイトで実行されている同様のアプリケーション，たとえば主にタッチまたはマルチタッチディスプレイを使用するアプリケーションによって生成されるエフェクトの同期に特化している．

5.5.1　リソースコネクタによるエッジ/クラウドリソースのモジュール化

図5.6に示すように，エッジリソースは環境およびユーザ（例えば，インタラクティブディスプレイ），または，インターネットおよびローカルネットワークの通信設備とコミュニケーションするデバイスを直接制御する．したがって，エッジリソースは，リアルタイム処理を実行し，主に同期して動作を行う．クラウドリソースは，クラウド内のオブジェクト（例えば，マルチメディアデータベース）を制御する．クラウドリソースは，主に非同期操作を実行する．クラウド内のリソースとオブジェクトのモジュール化も，5.2節で解説を行ったリソースコネクタによって構成される．

システムは，データを交換する必要のあるさまざまなコンポーネントとサブシステムで構成される．いくつかのエッジリソース（例えば，sender, receiver）は，ストリームを直接かつ同期して送信するために，対応するエンドポイントのアドレスを与えなければならない（図5.6参照）．このアドレスは，サービス実行エージェントによってエッジリソースに与えられ処理される．他のタイプのエッジリソースは，データをクラウドリソースに送信してクラウドに格納し，非同期に読み込まれる．図5.6に示すように，ストリームを直接対応するエッジリソースに送信する同期通信と，非同期通信で構成された複合チャネルを設計し，クラウドリソースに格納を行う．これらのタイプのチャネルは，パフォーマンス要件，コスト，および実行するエージェントとリソースコネクタを使用した共有を用いて，アプリケーションにより選択可能である．

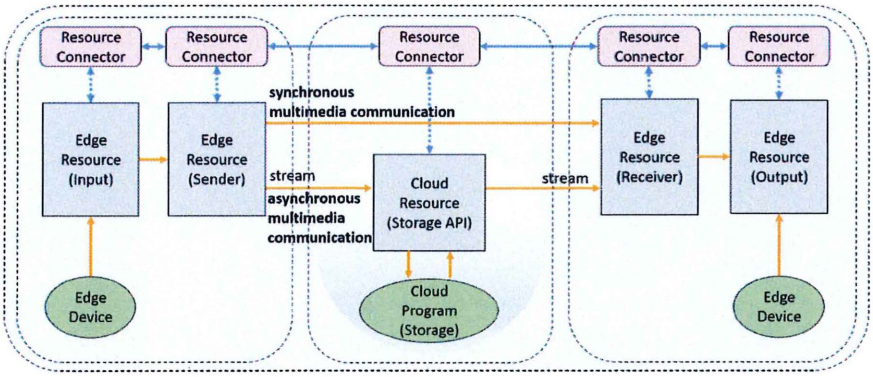


図 5.6　同期マルチメディア通信および非同期マルチメディア通信のための複合チャネルの構造

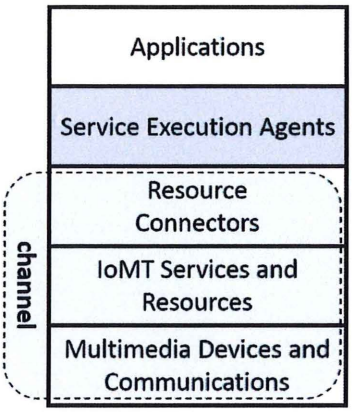


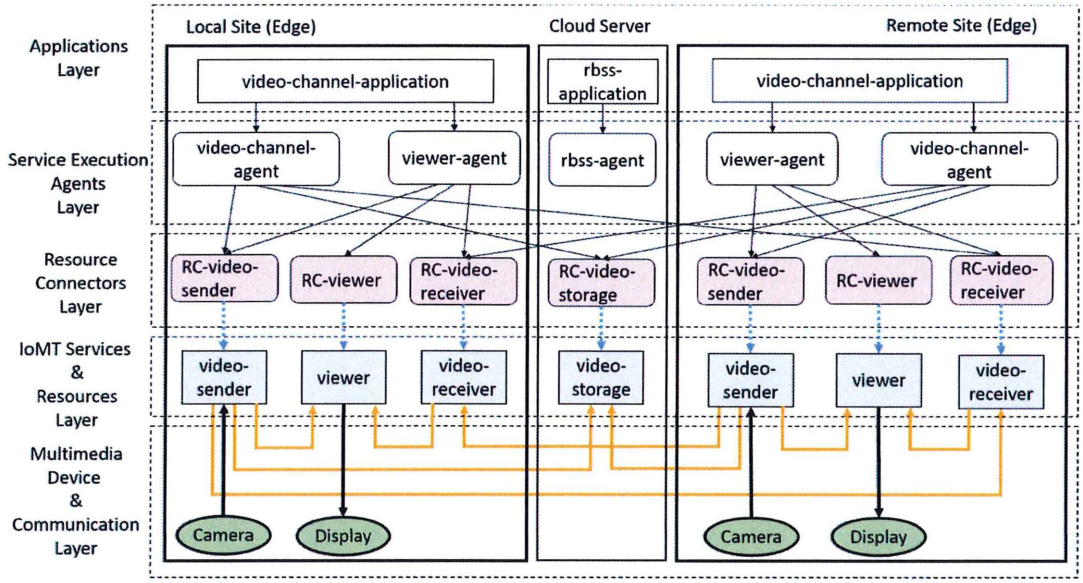
図 5.7　提案する IoMT システムの動的に垂直連携可能なアーキテクチャ

5.5.2　エージェントベースの IoMT アーキテクチャに基づくリモートな IoMT システムの設計

本研究では，Alvi らが提案した IoMT アーキテクチャに，5.2 節で示したエージェントベースのアーキテクチャと 5.3 節で示したチャネル概念を導入した新しい IoMT アーキテクチャを提案する．本エージェントベースのフレームワークは，マルチメディアデバイスとその制御プログラムをエージェントベースのモジュールにすることができ，アプリケーションの実行要件を満たすサブシステムとして動的にサービス実行エージェントを構成することができる．また，サブシステムのチャネルのコンポーネントは，リソースコネクタによって制御されるモジュール間でデータの交換を行う．

図 5.7 に，IoMT アプリケーションを開発するために提案されたエージェントベースのアーキテクチャを示す．また，図 5.8a に本アーキテクチャを用いて構成した例としてビデオチャネル全体のアーキテクチャの階層を含めて示す．そして，以下に提案する IoMT システムの動的に垂直連携可能なアーキテクチャの各層の要素について詳細を示す．

Multimedia device and communication layer　5.5.1 節の様々なタイプのエッジデバイス（例えば，カメラ，ディスプレイ，およびタッチおよびマルチタッチディスプレイ）は，この層の構成要素である．タッチデバイスは，チャンネルの入力開始ポイントと出力終了ポイントの両方として使用可能である．異なる



(a) ビデオチャネルの全体アーキテクチャ

(b) ディスプレイ上のビューア：ローカルチームのエッジリソース

(c) ディスプレイ上のビューア：リモートチームのエッジリソース

図 5.8　提案された IoMT アーキテクチャに基づくビデオチャネルアプリケーション

マルチメディア通信設備およびプロトコル（例えば，無線ネットワーク，MQTT，および WebRTC）もこの層の構成要素である．

IoMT service and resource layer　リソースは，チャネルおよびアプリケーション内のリソースコネクタを通じて，モノや他のリソースと通信することができる．また，1 つの特別なタイプのリソースは，データ，知識，ビデオ，およびアノテーションベースを管理するクラウドリソースである．

Resource connector layer　リソースとエージェント間のインターフェースである．リソースの内部データ表現を，エージェントの意味モデルに準拠したフォーマットに変換することができる．そして，これらのインタフェースをリソースコネクタと呼ぶ．それらは，[41] で詳細に示されている．また，リソースコネクタはエージェント言語プロトコルに適合したエージェントメッセージを使用してエージェントと通信する機能を備えている．

Service execution agent layer　アプリケーションのシステム制御のためのエージェントベースの動的に垂直連携可能なセマンティックミドルウェア．サービス実行エージェント (Service Execution Agent) とは，リソースコネクタとの間でエージェントメッセージを送受信し，アプリケーションの要求に応じてモジュールを起動および操作するプロセスである．サービス実行エージェントは，アプリケーションが要求を送信し，応答を受け取るための API を提供する．この層は同種のすべてのエージェントが同じフレームワークで実装され，通信のためのメッセージモデルを共有する．

Application layer　アプリケーションのロジックを保証するアプリケーションコアである．

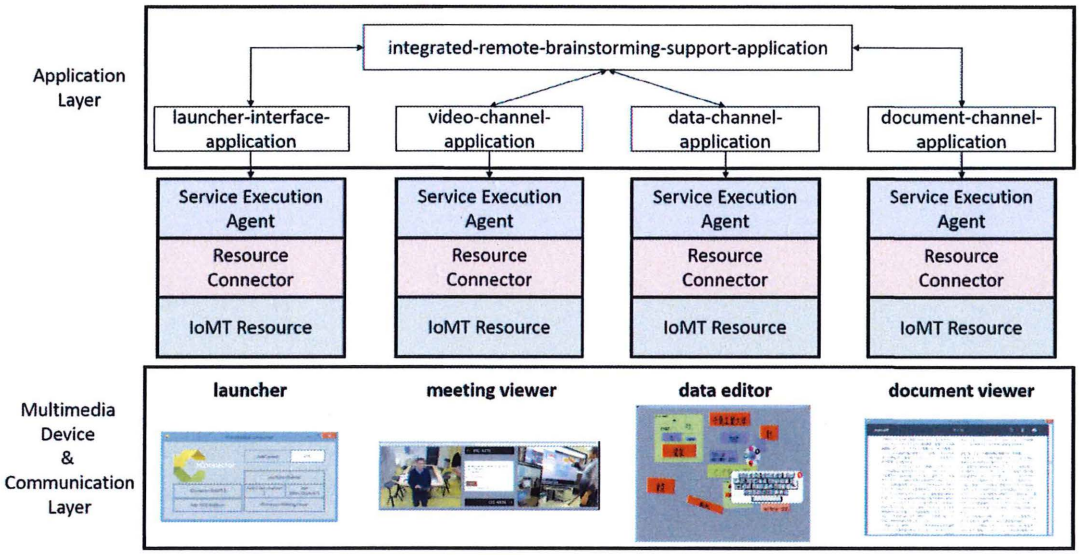


図 5.9　IoMT システムの動的に垂直連携可能なアーキテクチャにより構成された IoT アプリケーションの例

リソースコネクタ，リソース，およびデバイスまたは通信の 3 つをモジュールと呼ぶ．同期チャネルは，いくつかの通信設備を介してデバイス間でストリームを伝達するメカニズムを持つ．そして，非同期チャネルはデバイスからクラウド内のオブジェクトに，または，クラウド内のオブジェクトからデバイスにストリームを伝達するメカニズムを持つ．

複合チャネルは，図 5.6 に示すように，両方のチャネルタイプを組み合わせることで成り立つ．これらのチャネルは，サービス実行エージェントからの要求によってモジュールで動的に構成される．

さらに，提案されたアーキテクチャは，アプリケーションとチャネルに応じていくつかの異なる表現が存在する．図 5.8a にいくつかのモジュールと両方の複合チャネルの方向の通信から構成されるコラボレーション支援システムのビデオチャネルサブシステムを示す．モジュールは 2 つのローカルサイトとクラウドサーバのアプリケーションコンポーネントからの要求に応じて，サービス実行エージェントによって起動される．

また，このアーキテクチャは，いくつかの異なるサブシステムの表現になる場合がある．図 5.9 は，さまざまな同期アプリケーションのタイプとランチャアプリケーションの例を示している．各同期アプリケーションには，チャネルのエンドポイントである 2 つの同様のインスタンスが必要である．図のデバイスレイヤに表示されるイメージは，ミーティングビューア（ビデオチャネルアプリケーションの場合），データエディタ（データチャネルアプリケーションの場合），ドキュメントビューア（データチャネルアプリケーションの場合）である．ランチャは，それぞれのサイトでアプリケーションのインスタンスを開くために使用される．

5.6　まとめ

本章では，IoT を拡張した IoMT アプローチに基づいたリモートな IoMT システムの革新的で動的に垂直連携可能なアーキテクチャを提案した．このようなタイプのアプリケーションには，カメラ，マイク，センサなどの基本的な IoT 要素およびデバイス，ならびにマルチメディア通信回線が含まれる．それらは，IoT，IoMT，クラウドコンピューティング，ビデオ会議，コラボレーション支援など，いくつかのドメインが交差するところに位置している．

本章で示した，チャネルの概念は，IoT アプリケーション開発における課題である動的な垂直連携を実現す

る為に有効であり，各階層間の要素の連携を適応的にエージェントによって構成する支援を行うことが可能である．そのため，この概念を用いた拡張をリソースコネクタプラットフォームに施すことで，エージェントプラットフォームにより，IoT アプリケーション開発方法論の課題の解決を行った．

さらに，このチャネル概念は，リソース間の通信とそれらの制御を行うことが可能なエージェント間の通信を明確に分けることで，記号としての制御と生のデータをやり取りするストリームを明確に分けることが可能となり，各要素が扱う対象のリソースのプロトコルに依存することがなく，協調することを可能とすることができる．この概念により，各層の動的な垂直連携のために必要な Edge Network 部分の共通のプロトコルによる制御が可能となり，IoT の 3 層の全ての要素が共通のプロトコルにより協調可能となった．

次章からは，本章で示したアーキテクチャの実装による評価を行う．

第 6 章

エージェントプラットフォームを利用したブレインストーミング支援システムの設計

オフィス業務は，書類や資料の作成，会議などを繰り返し行い，それぞれの作業の成果を共有したりするワークフローにより実現されている．これらを支援するためには，高度に連携するマルチメディアシステムを実現することが必要である．しかしながらオフィス業務支援ツールは，個別に動作するツールの寄せ集めであり，これをワークフローの各段階に合わせて利用するためには，各ツールの調整が利用者の大きな負担になる．この負担を軽減するための IoT アプリケーションの構成法が課題である．本研究では，オフィス業務における会議支援に焦点を当て，中でもマルチメディアの要素がより重要視される課題として，遠隔地でのブレインストーミング会議の支援を行うシステムにおける構成をテストベッドとして扱う．

本章では，第 5 章で示した IoT アプリケーションの垂直連携型アーキテクチャを用いて設計・実装されたエージェントプラットフォームであるリソースコネクタプラットフォームを用いて IoT アプリケーションの構成を行うことで，オフィス業務における会議支援での負担の軽減を図る．

6.1 既存のブレインストーミング支援システムとその問題点

インターネットの普及に伴い，ソフトウェア開発における国際的な共同作業が日常的な企業活動として行われている．そのような遠隔拠点間の開発プロセスにおいては，企画や要求定義の段階を支援することが重要であることが指摘されており [28]，この段階のコラボレーションを支援する技術としてブレインストーミングの技法が使用されている．ブレインストーミングの技法を，国際的な共同開発作業に適用するために，様々なマルチメディア情報をインターネットを利用して開発拠点の間で共有する仕組みの研究開発が重要な課題になっている [78]．

例えば，単独の開発拠点で KJ 法に基づくコラボレーションの支援システムの研究として，マルチタッチディスプレイによるインタラクティブなデバイスによるアイデアの提示・集約の作業支援システムが提案され，その支援機能の有効性が検証されている [36]．また，遠隔地の開発拠点間で，KJ 法に基づくコラボレーションを支援する方法論の研究がすすめられている [100, 93]．さらに，協調作業のためのデータをクライアントサーバ型で共有するシステムの開発が行われており（Inspiration[*1]，Bubbl.us[*2]，Mindmeister[*3]，そして

[*1] http://www.inspiration.com/Inspiration
[*2] https://bubbl.us/
[*3] http://www.mindmeister .com/

Trello[*4]等），遠隔地間で協調作業データの共有の重要性が確認されている．

本研究におけるブレインストーミングのアクティビティの定義とリソースの定義，そして Capitalization の定義を以下に示す．ブレインストーミングのアクティビティとは，行動や状況を指し，本章においてはブレインストーミングにおけるイベントを指す．リソースとは，3 章で示したデバイス・データ・プログラムのことであり，実際に現実空間に対してインタラクションを行うものを指し，変化によって生じたデータ系列のことを指す．そして，Capitalization(資産化) とは，現実世界のイベントをリソース化する行為あるいは処理である．このリソース化・資産化が行われることにより，現実世界のイベントが保存，利用，そして再利用可能となる．本章においては，アクティビティというブレインストーミングで発生するイベントをリソース化している．

本章における目的は，ブレインストーミングのアクティビティを支援し，特に遠隔チーム間のコラボレーションにおける資産化を実現するツールの設計と開発である．

遠隔チーム間ではリソースを作成し，共有することが必要であり，それらについて通信を行う．この時，共通のブレインストーミングのセッション中にユーザが作成することができるデジタルノートのようなアクティビティの具体的な結果や会議のビデオ及びビデオストリームのおかげで得られるあらゆる種類のイベントを資産化することが不可欠である．

様々なプロジェクトの初期段階では，チームは新製品を設計する際などにおいて，いくつかのミーティング中に新しいコンセプトを定義する必要がある．チームは，参加者間の議論から出現した主なアイデア・問題・ソリューションなどのブレインストーミングのアクティビティを導出するまでの間，協調して取り組む．これらのアクティビティの結果は，活用されるべきであり，議事録は，その結果から作成されなければならない．

本実験におけるシナリオは，日本とその文化的側面のいくつかを発見したいフランスの訪問者のための有用な Web サイトを設計する必要があり，2 つのリモートチーム（日本のチームとフランスのチーム）により実施される．設計者は，議論やアイデアの生成，同意を可能とし，ノートの記述やクラスタに分類されたいくつかの結果を整理する．この時，会議参加者はまた，それぞれの持つ文化的な概念について議論し，その意味を説明する必要がある．

これらのアクティビティは，通常，ペンと紙メディアを用いた伝統的なミーティングルームで行われる．会議の結果は，一般的にポストイットノートに書かれ，デジタルでエクスポート可能な形式にこれらの手書きのメモを変換するためにはかなりの労力が必要とされる．

既存のブレインストーミングに対するコンピュータ支援による各ソリューションは，単一のシステムではなされていなかった．それらは参加者の間の自然なコラボレーションが難しい状況となってしまうため，追加の機器を導入する必要が有った（例えばビデオチャットのためのアプリケーション等）．

さらに現在，インタラクティブなテーブルトップやインタラクティブなホワイトボードおよびユーザが持つデバイス（タブレットやスマートフォン）などの技術は，現在広く普及している．

大型インタラクティブなデバイスは，従来のペンと紙による物理的な環境を代替することができ，それらは仮想的なポストイットノートをソフトウェアによって実現できる．その際，マルチタッチジェスチャに基づいた入力は，通常の入力（キーボードやマウスなど）を再現し，同じ場所に参加しているユーザのグループ間の協調を行うことが可能となっている．このアプローチの欠点は，グラフィカルマルチユーザインタフェースの設計が単一のユーザインタフェースよりも複雑となることである．これらは一般にインタフェースのウィジェット等のマッシュアップにおいて結果として生じることがある [74]．また，Web ベースのコラボレーションシステムでは，異なる役割を持つユーザ・グループは，専門のインタフェースを介して協力を行う．コラボレーションのためのユーザインタフェース構造は，通常，かなり複雑であり，そして設計はその中に重要な役

[*4] https://trello.com/

割を有している [32].

ブレインストーミングのアクティビティにおける資産化は 2 種類の結果が含まれる．まず，アイデア・提案等は，デジタルノートに書かれている場合，容易にセマンティックインフォメーションシステムに格納することが可能である．支援ツールは作成・構造化・共有，そしてそれらノートの保存のみを可能とする必要が有る．次に，人々はｓそれらを用いずとも議論することが可能であるため，一部のイベントはまた，参加者間の合意のように発生する可能性がある．そのため，ブレインストーミングのセッションを撮影し，ビデオを保存することが，これらの結果を活用すること唯一の方法である．

本研究では，リモートのチーム間のコラボレーションおいてブレインストーミングの支援システム以外に，参加者が通信するテレビ会議システムを持っている必要があるため，このビデオ会議システムにおける資産化も行う．

6.2　ブレインストーミングにおけるアクティビティとアウェアネス

参加者は，遠隔地間のチームに所属し，そのアクティビティの中で異なるタイプのデバイスを使用する．また，このようなシステムは必然的に分散して遠隔地に存在する．

コラボレーションのアクティビティに役立つことができるが他のツールは存在するものの，同じ部屋に集められたユーザのグループに対して支援するものであり，且つ非常に少ない数しかツールが存在しない．

このような分散システムをモデル化し，協調アクティビティについて説明する際には，正確なユースケース図とシナリオを考慮した上でアクティビティの種類を記述する必要がある．[59] は，ソフトウェアの設計のためのいくつかのパターンを提案している．

それらのシナリオは，アクティビティの種類が関連していくつかのステップで構成されている．本章ではアクティビティを 3 種類であると考える．

1. 参加者が 1 人であり，コラボレーションセッションを準備する．
2. 参加者は 1 つ以上のチームによるコラボレーションセッション中に作業する．
3. いくつかの参加者はコラボレーションセッション中にビデオストリームに対してアノテーションを付与する．

他の次元は，同じアプリケーションを使うことができる参加者の数を示す．また，他のケースとして，2 人が異なる瞬間または同時に同じアプリケーションを使用した場合や，2 人が同期の問題につながる別の場所で同じアプリケーションを使用したケースにおいても協調作業を可能する必要が有る．

6.2.1　単一ユーザの状況における協調アクティビティ

ブレインストーミングセッション中に，参加者は進行に従っていくつかの定義やいくつかの決定に同意することができる．この時，ディスカッションでは，外部リソースの調査や獲得が必要な場合が存在する．しかしながら，チームにより達成された結果は，何らかの方法で活用しなければならない [56]．これらのセッションの結果は前章で示したカードに類似するクラスタとカードのセットによってグループ化されたノートの形を取ることを提案する．

単一の拠点におけるブレインストーミングセッションでは，参加者が同じ部屋に位置しており，それらのアクティビティが常に同期している．少なくとも 1 つのブレインストーミングにおけるサーフェス（ブレインストーミングのために用いられるインタラクションツールが提供される場所）がすべての参加者がチームにとっ

て興味を持つノート（仮想化されたポストイットやクラスタ）を見ることができるように共通の支援機能持つように設計されている．

ノートは，このサーフェス上で直接作成することが可能であるが，本システムの主な利点として，サーフェス以外のタブレット PC 等から追加することができることとする，この機能は現在のディスカッションの場を乱すことなく，サーフェスである共通のディスプレイに送信することができるという利点が存在する．

また参加者はアクティビティの効率的なデジタルサポート [51] のための大規模なインタラクティブサーフェス上に表示されたノートを操作する．参加者間のコミュニケーションを強化するデジタルデバイスは，ブレインストーミングセッション中のデータの格納を可能するため，ブレインストーミングセッションに参加するチームのために必要な支援である [77, 75].

このような大規模なコラボレーションのためのデバイスを利用することで，新しいリソースを生成するためにメンバを助け，支援することができる [19]．このようなブレインストーミングにおける支援が存在しない場合，より多くの労力が紙のシート上に書き込まれたものを再度デジタルデータに書き込むために必要となる．

リモートブレインストーミングセッション中においては，いくつかの遠隔地のチーム（少なくとも 2 つは）が同じブレインストーミングの活動に参加している．彼らの行動は，環境に応じてソフトウェアアプリケーションによって支援される必要が有る [68].

例えば，同じアプリケーションの 2 つのインスタンスが 2 つの離れた部屋に位置する場合，通信環境及び機能を備えたピアシステム上で実行する必要がある．

これらのセッションに関与するチームがリアルタイムで通信して同じタイプのグラフィカルユーザインタフェース (GUI) とインタラクションし，同期することで，すべてのアクションが共通のリモートサーフェス上に再現される．

また，本システムが持つロックシステムは，すでに他の場所で選択したのと同じ要素上の同時操作を防ぐことが可能である．ロックシステムにおけるカギのグラフィカルな表示は，リモートチーム内の他の誰かがすでに要素を選択したことを理解しやすいように表示される形となっている．

6.2.2　リモートブレインストーミングにおける他の参加者のアウェアネス

各チームメンバとの間には，空間的，時間的，文化的分離によるコミュニケーションと協調の問題が存在する．支援システムに対して期待される効果は，リソースに対するより多くの効率性を上昇させるとともに柔軟なアクセスの改善が期待される．組織は，可能な限り効果的に既存のリソースを再利用する必要があり，また，異なるサイトから，世界全体をまたいでリソースを採用する必要がある．

また，異なる文化を持つチームが協力する必要がある場合，いくつかの問題が表出することがある．[49] は分散ソフトウェア開発の特定のドメイン内の文化的問題に関する対応を提案している．開発プロジェクトのケースにおいては，人々は他のチームに一つの特定の文化のいくつかの概念を説明する必要が存在する可能性がある．本ツールは，異なる言語でのメモ及びノートを記述することができ，そのノートを自動的に翻訳することで，参加者において共通の理解を助けることができる．

この時，エコシステムの観点から考えると，題目を識別し，違う種類のリソースを作成および共有し，コメントを行い，同一のエコシステムに参加することについて人々がコミュニケーションすることが必要であると考えらえる．

6.3　エージェントプラットフォームを用いたブレインストーミング支援システムの提案と設計

まず，上述のブレインストーミング支援のためのコラボレーションのためのツール群と資産化のための機能を持つブレインストーミング支援システムを 3 章と 4 章の仕組みとリソースコネクタプラットフォームにより実現するために提案と設計を行う．

6.3.1　アプローチ

リモートブレインストーミングセッション中においては，1 つのチームは，他のチームが生成した要素を見る必要があり，またその逆も同じく見る必要が有る．このことから，チーム間でのリソースの同期（これをノートチャネルと呼ぶ）を構成する必要が有る．また，チームのメンバは，他のチームのメンバとローカルでも通信する必要がある．そして，カメラより取得されたリアルタイムの動画の送信は，リモートチーム間の両方向の交換を可能にするためにインストールされている必要がある．この時，遠隔地において何が行っているかを知るためにも，人々が通信し，それらが同じ部屋にいるかのように議論することが必要である．この機能をビデオチャネルと呼ぶ．図 6.1 にそれらの構成を示す．

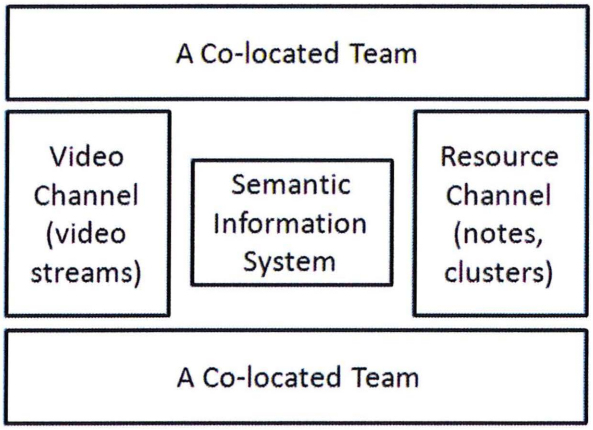


図 6.1　リモートブレインストーミングセッション中のリソースとビデオの共有とそのためのセマンティッククインフォメーションシステムの構造

6.3.2　ノートチャネルの設計

ブレインストーミングのアクティビティを支援するためのアプリケーションの開発を行った．この中で，最も重要なものは，大規模なタッチデバイスによるインタラクティブなテーブルやホワイトボードのいずれかにより実行される．これらの異なるデバイスを用いる場合，人々の間のインタラクションは同一ではない，例えば，テーブルによるデバイスを用いる側では，表示とインタラクションを行うために水平のディスプレイを持ち，テーブルの周りに人々が立つ状況で実施される．また，ボード（図 6.2 を参照）[*5]によるデバイスを用い

[*5] 実験中に使われた言語に関する言及は本章の範囲外である．そして，参加者間のコミュニケーションは英語行った．技術的にノートは，英語，日本語，そしてフランス語の 3 つの異なる言語によって記述することができる．また，互いのチームはそれぞれの言

る側では，垂直に設置したインタラクティブディスプレイを持ち，参加者はボードを見ることが可能な位置に座って参加することとなる．そして，図中の上部の図は，大規模なインタラクティブなサーフェスを用いたモデレータによる日本語の文字で書かれたノートを移動する行為を示している．移動中はノートがロックされており，誰かがこのノートという要素に対してインタラクションを行っていることを示すために遠隔地に存在する対応するノートに対してもロックマークが表示される．下部の図は，参加者が日本語で文字を記述する際に利用できるタブレット PC を示している．タブレット PC 上で記述されたノートはボード側に送信及び共有される．これらにより，機能の持つ特徴とマルチタッチによる特性により，同一拠点における参加者のインタラクションを可能とする．この時，両方の立場において，議論をまとめるためにモデレータが必要である．

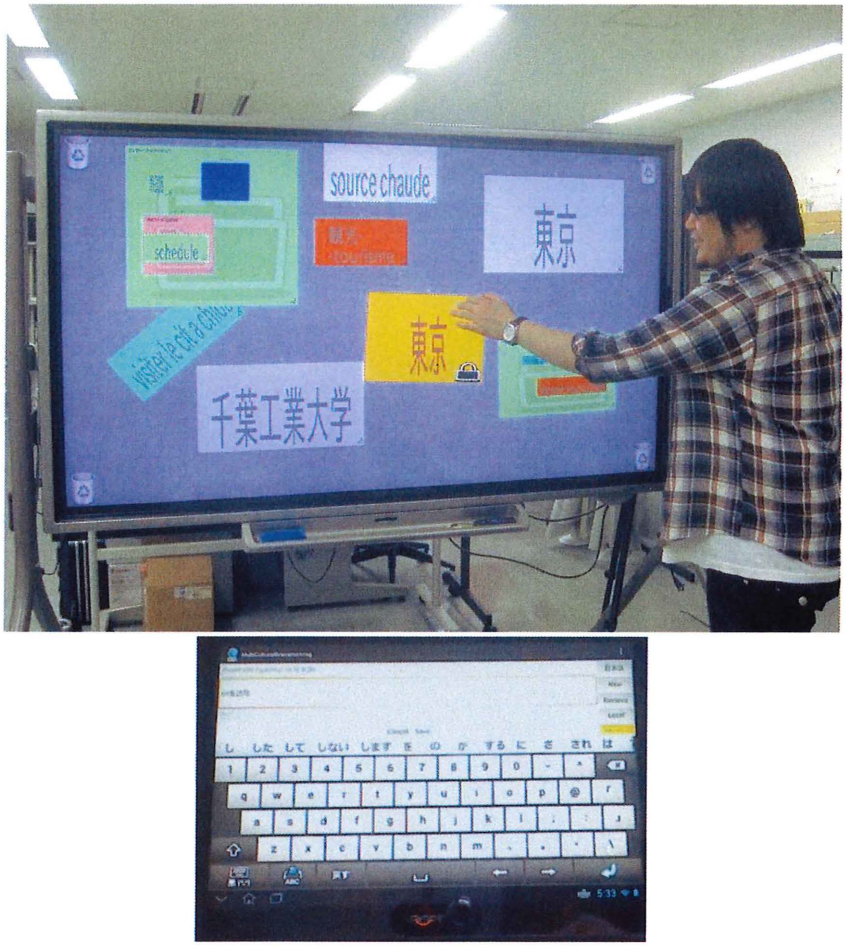


図 6.2　大規模なインタラクティブサーフェスを用いたモデレータによるノートの移動と参加者による日本語入力のためのタブレット PC

アプリケーション群はインタラクティブボードやインタラクティブテーブル，タブレット PC などの異なるデバイス上で実行される（図 6.3 を参照）．2 つの遠隔地における会議が同時に発生した場合，2 つの類似するモジュールを持つアプリケーションが異なる場所で同時に実行されることにより，会議が行われる．アプリケーションとその機能（8 人の参加者と参加者は 10 個までのアプリケーションを利用可能）との間の通信は，

語によって要素を追加することが可能である．そして，ノートはそれぞれの言語によって表示することができる．

マルチエージェントシステムにより管理される [35]．各アプリケーションは独立しており，少なくとも 1 つのエージェントを含んでいる．

エージェントはコンテナに内包されることでグループ化される．メインコンテナでは，そのプラットフォームとシステムにおけるコミュニケーションのコントロールを提供するエージェントが動作している．それらは，マルチエージェントシステムにおけるホワイトページやイエローページを維持し，それらがエージェントメッセージを配達可能な場所に存在する異なるエージェントについてのすべてを知ることができる．また，それらもまた，エージェント間のメッセージの良好なコミュニケーションを行うために寄与している．

次に，エージェントに振られた数字と図 6.3 の中の数字を参照し，解説を行う．本システムでは，特定のフェデレータエージェント（n°1）は，遠隔地に存在する同等のエージェントシステムとの通信に関する機能を担当している．このエージェントは，他のプラットフォームと接続可能であり，メインとなるインタラクティブデバイス上の参加者の行動を表すメッセージを転送することが可能である到達可能なステーションのアドレスを持つ．

いくつかのエージェントは，インタラクティブサーフェスへリンクされたステーション上で動作し，ユーザとのインタラクションが可能なアプリケーションと共に動作する．

また，ミーティングエージェント（n°2）はユーザのアクションを認識する．このエージェントは，それらを伝播するロジスティックエージェントにこれらのユーザのアクションについてのメッセージを外部のプラットフォームに対して配信することができる．

認識されるユーザの主なアクションは，ノートやクラスタの作成，要素の移動，クラスタへのノートを追加，要素を削除などから成る．永続化エージェント（n°3）は，ローカルなコンテンツや大規模なインタラクションサーフェス上に表示されたノートやクラスタポジション（座標，大きさ，向き）を保存することができる．これは，各ユーザ/参加者のアクション後，ブレインストーミングに関するコンテンツを保存することを可能とする．アクションとブレインストーミングのコンテンツは以下の JSON[*6]形式で記述される：

```
{
 "action" : "create", "state" : "UNIQUE",
 "object" : {
  "type" : "postit",
  "id" : "c8c62904-aa52-4237-85be-acceb130dc21",
  "author" : {
    "id" : "4ac99de8-d66b-40ae-a72d-39f082817c96",
    "name" : "claude", "alias" : "cm", "color" : "#f27982ff"
   },
  "created" : 1381743259385, "color" : "#f27982ff",
  "content" : {"content" : {"FRENCH" : "UTC\nGI" }},
  "textFont" : null, "textSize" : 0, "textColor" : "#000000ff",
  "title" : { "content" : { } },
  "matrix":{"x" : 928.0, "y" : 295.99988, "sizeX" : 243.99994,
          "sizeY" : 177.99986, "angle" : 1.3223116} }
}
```

別のエージェント（n°4）はブレインストーミングの結果を格納し，インデックスを作成するためのセマンティックインフォメーションシステムとの通信を担当している．

タブレット PC 上で動作するアプリケーションは，ノートとクラスタの作成，および，ユーザが共同作業の

[*6] http://www.json.org/

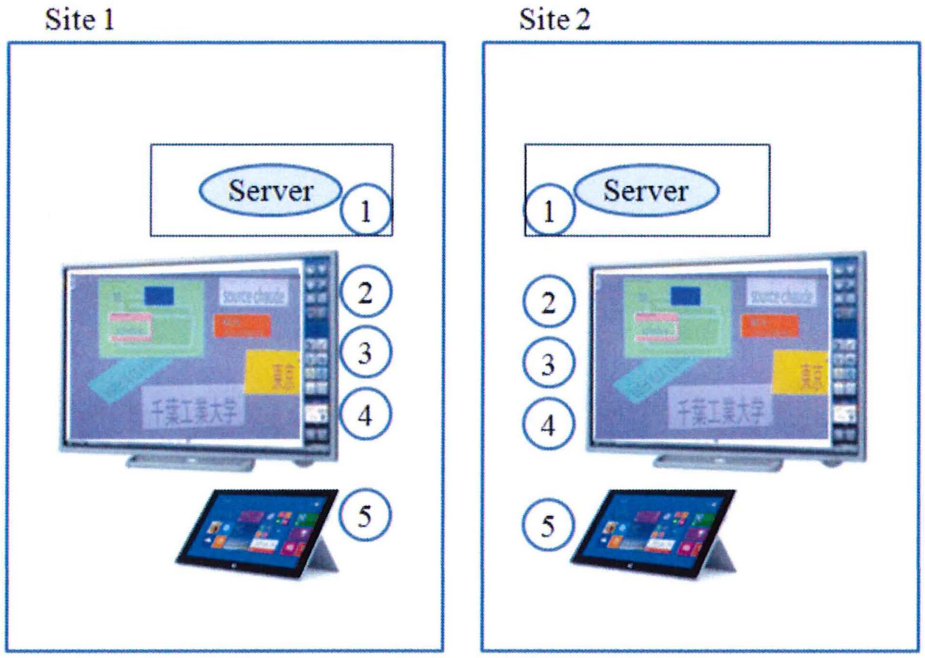


図 6.3　異なるデバイス上のリソースを管理する分散エージェント

ためのセッションの準備を行う事ができる．これはパーソナルアシスタントエージェント（n°5）[*7]に含まれる．その主な役割は，対応する会議のエージェントにタブレット PC 上で書かれたポストイットノートを送信することである．

このようにノートを作成する場合，2 つの利点が存在する．まず，1 つ目の利点として，より快適な方法で会議中にメモを追加することができるという点である．タブレットの仮想キーボードは，PC 上のローカル言語に適応されているため，日本語のように複雑なインプットが求められるようないくつかの言語に属する文字列を書くためには有効な方法である（IME が必要となる処理の場合，ハンドリングが難しいためである）また，2 つ目の利点として，アプリケーションが大画面のサーフェス上で実行されている場合，プログラミングによっては文字を書くための唯一の方法は仮想的なキーボードを開くことであるが，OS によっては使用可能なローカル言語を制限される可能性があり，それを解決するためにもこの方法が有効である．

タブレット PC からメインのサーフェスに要素が追加されるとき，特定のパケットにより追加に関するメッセージが生成される．これらのメッセージによって追加されたノートは，自動的にメインとなるインタラクティブデバイス（同期ステーション上に表示される）に表示されず，ミーティングエージェントは，メールボックスにそれらを格納し，任意の適切なタイミングでそれらを選択しアプリケーション上に表示することができる機能を持つ．この操作は，一般的にメディエータであるユーザがメールボックスを開き，適切なタイミングで選択し，ノートやクラスタとして追加する形で操作を行う．

[*7] 各参加者は，タブレットを所有し，適切なアプリケーションを動作させている．システムに接続されたエージェントなどのような多くのパーソナルアシスタントエージェントが存在する．

6.3.3　ビデオチャネルの有用性

遠隔のブレインストーミングセッション中，チームのメンバは他のチームのメンバと通信する必要がある．ビデオ送信は，遠隔のチーム間の双方向のコミュニケーションを可能にするためにインストールする必要がある．参加者全員が同じ部屋にいた場合と同じように，参加者全員がローカル，遠隔地ともにポストイットノートで書かれた共有するアイデアについてコミュニケーションをとり，議論することができる必要が有る．また，人々は，ビデオチャネルのおかげで遠隔の部屋で何が起こるかを確認することが可能となる．それらの状況を取得するために，いくつかのカメラを配置し運用することは有用であり，それらのカメラの表示が移り変わり，他のチームでの状況の変化やイベントの起こりに応じて切り替わることも有効であると考えらえる．技術的な観点から見ると，同じ部屋の複数のビデオストリームを一つのストリームに合成し，ブロードキャストを行う形となっている [8].

ビデオ送信を録画することも，将来の再利用には非常に重要である．なぜなら，会議中に時には重要なことが発言される際，必ずしもメモに移されるとは限らず，保存された動画内で検索することで確認可能とする必要がある．また，後で不足したり忘れられた情報を追加することも可能となる．ストリームの送信者側がこれの処理を担当し，保存及びその利用を可能とする．

6.3.4　ビデオチャネルの設計

ビデオチャネルは，インターネットを介してビデオとオーディオのストリームを送受信することを可能とし，図 6.4 のような Video Streamer というコンポーネントのペアで構成される．Video Streamer は，部屋に設置された複数のカメラからキャプチャされた画像を合成してコンポジット画像に変換したものを，ネットワーク上でビデオストリームを送受信し，ビデオストリームを保存する機能を持つ．Video Streamer を使用した場合，カメラを選択して切り替えるための Video Switcher を操作することも可能である [82]．この時，ビデオスイッチの操作は記録され，ビデオにアノテーションを付けるために使用される．ブレインストーミングセッションの後のビデオのレビュー行為により，Video Switcher 機能が使用された理由（新しい参加者のプレゼンテーションなど）を示すことが可能となる．

また，Video Streamer はビデオストリームを Video Streaming Server に送信し，共同配置された 2 つのチーム間で映像を交換する．Video Streaming Server は，ビデオキャッシュとビデオストレージで構成されている．Video Streamer の Sender 機能はビデオストリームをビデオキャッシュに送信し，Receiver 機能はビデオキャッシュからビデオストリームを受信することにより，リアルタイムで Viewer に映像を表示する．ビデオアノテーションは，「Video Streamer を開始」，「Video Streamer を停止」，「カメラ画像を画面右下に置く」などの Video Streamer に関する操作のアノテーションのことであり，Video Annotation DB は，Video Streamer によって送信されたこれらのビデオアノテーションを保存するサーバである．

最初のタイプ（図 6.4 の Video Streamer-1）は，メディエータと複数の参加者からなるチームによって使用される．また，ブレインストーミングセッションでのさまざまな視覚情報を遠隔のチームに伝えるために，いくつかのカメラが導入されている．この Video Streamer には，Video Switcher 機能が搭載されており，複数のビデオ画像を動的に合成する．Video Operator インタフェースは，メディエータが Video Switcher を操作することを可能にする．

Video Operator のアクションは，Video Annotation DB によって保存されるアノテーションを生成する．それらの記述は Video Streaming Server によって保存されている対応するビデオストリームと関連付けることを可能にする．Video Annotator は，Video Operator からアクション記述を受け取り，アクションの起こっ

た際のタイムスタンプを含むアノテーションを Video Annotation DB に保存する．これらの機能により，ブレインストーミングセッションの後，参加者は特定の Viewer を使用してビデオをレビューし，アノテーションが作成された瞬間を取得することが可能となる．

2 つ目のタイプの Video Streamer（図 6.4 の Video Streamer-2）は，1 人または 2 人のメンバからなるチームによって使用される．この場合，他の遠隔地との通信には 1 台のカメラで十分である．このような Video Streamer は，ハードウェア装置によって実現されるビデオスイッチャを，ソフトウェアによってのみ実現されるビデオキャプチャ機能に置き換えた，よりシンプルなシステムとなっている．

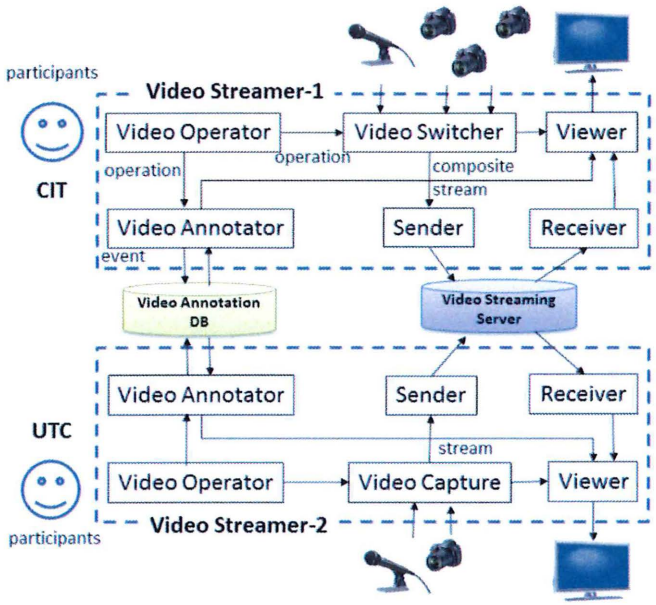


図 6.4　Video Streamer で構成されたビデオチャネルの設計

6.3.5　セマンティックインフォメーションシステムの設計

セマンティックインフォメーションシステムは，ブレインストーミングアクティビティに関連するデータを格納するためにも必要であり，MEMORAe Information System[22] を利用する．MEMORAe の利用を選択した理由として，そのセマンティックモデルが本研究において適応するものであったからである．このモデル概略図を図 6.3.5 に示す．MEMORAe モデルは非常に豊富な機能を含むが，図中では本章に関連する概念だけを提示する．図中の接頭辞の mc2 は，モデルを示す．また，アノテーション，ノート，クラスタ及びその上位概念のみが表示される．

本アプリケーションは，ノートとメモのクラスタを作成するシンプルなアプリケーションである．これらはそれぞれ `SimpleResource` と `CompositeResource` を継承する．`SimpleResource` の主な種類は `Document` である．それらは，PDF ドキュメント，アーカイブ，テキストドキュメントなどのリソースをカプセル化することができるが，あらゆる種類のドキュメントの仕様を持つことは必要ではない．本モデルでは，`Note` には，`Resource` という本体がある．これはモデルが，画像やその他のものであることを可能とするためである．本章のケースでは，ノート本体は匿名のテキスト文書である．匿名とは，この文書が対応するアノテーションによってのみアクセス可能であることを意味する．同様に，`NoteCluster` の本体はテキスト文書である．

NoteCluster は，Note と NoteCluster の composedOf の関係にある．

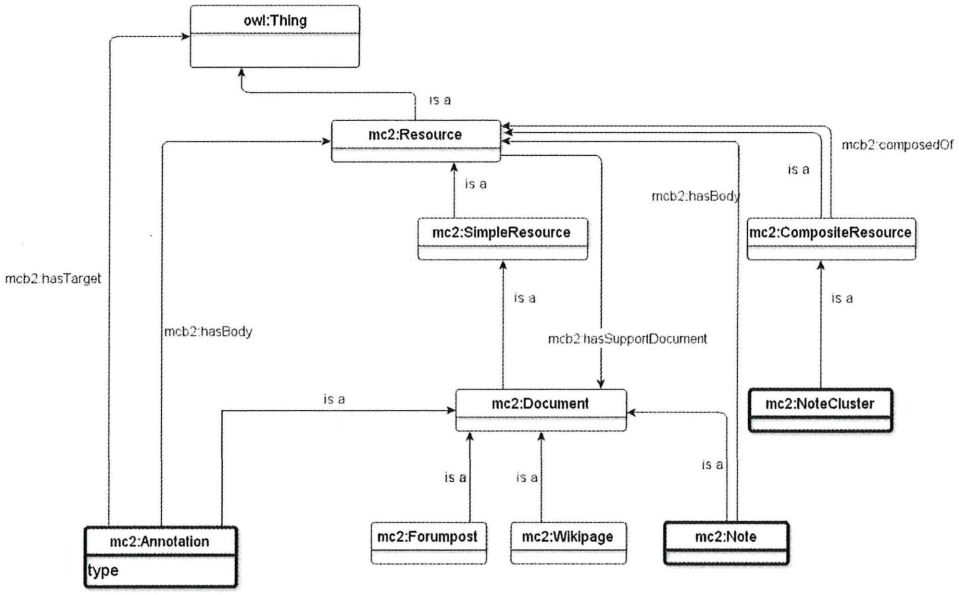


図 6.5　MEMORAe によるリソースの記述モデル

すべてのリソースは，1 つまたは複数のアプリケーションによって表示される必要がある．たとえば，リソースは Web アプリケーションとタブレット PC 上で実行されているアプリケーションで表示可能である．多くの場合，ある 1 つのデバイスで作成され，他の異なるデバイスで表示される状況となる．この時，対話式のテーブルトップディスプレイとホワイトボードを使用しているため，Note と NoteCluster を保存する必要がある．また，これらのデバイスで実行されるアプリケーションによって，表示される色やリソースの検索に役立つ項目などのデータが追加される．これらのデータはアプリケーションに非常に依存しており，モデル上では記述されないため，新しいアプリケーションを作成するたびにモデルを変更する必要はない形とする必要が有る．この目的を達成するために，モデルは各リソースがこれらのデータを含む 1 つ以上のドキュメントにアクセスすることを可能とした．ドメインと範囲が mcb2:hasSupportDocument であり，Resource と Document の関係は，一般的にテキスト文書をリソースに関連付け，JSON[*8]構造体により構成される．

ブレインストーミングセッション中に撮影されたビデオは，保存され，セッション中に生成されたデータ（Note 及び NoteCluster）に何らかの形でリンクされなければならないドキュメントでもある．それらは，Document タイプの下に格納することができる．図 6.3.5 にはない Event の概念は，Event としてのブレインストーミングセッションを記述し，それに関するリソースにリンクすることを可能とする．特に，セッション中に生成されたすべてのメモとクラスタは，セッション中のイベントに添付された一意の親クラスタ内にカプセル化される．MEMORAe モデルは，他の文書としてだけでなく，作成されたイベントからもメモを見つける機会を与える．

[*8] JSON site: —http://www.json.org/

6.4　エージェントプラットフォームを用いたブレインストーミング支援システムの拡張

グループウェアを分類するために Time/Space Matrix が提案されている [6]．これに基づいて，遠隔の拠点間のブレインストーミングを分類すると，同期型と非同期型に分けられる．同期型ブレインストーミングとは，Face-to-Face と同様な環境で行う作業であり，非同期型ブレインストーミングとは，ブレインストーミングの行為の発生時刻と行為の結果を利用する時刻が参加者によって独立に選択できる環境で行う作業である．例えば，カメラからの映像を遠隔のディスプレイに表示する機能は，リアルタイムの条件を満足するときに同期型ブレインストーミングを支援することができる．この映像を，クラウドのストレージに蓄積し，ブレインストーミング終了後に議事録として再利用する機能は，非同期型ブレインストーミングを支援する．しかしながら，通常，同期型と非同期型を同時実現するマルチメディア通信機能を実現することは困難である [46]．

6.3 節では，エージェントプラットフォームを用いたブレインストーミング支援を行うシステムを提案及び設計した．しかしながら，この同期型と非同期型の仕組みを同時実現し，且つユーザの要求に応じて動的に組み合わせることはできていない．そのため，本節からは，5 章で示したアーキテクチャにより，システムを動的に垂直連携可能とすることでこの問題点を解決することを目的とし，システムの実装及び評価を行う事で動的に垂直連携可能なアーキテクチャの評価を行う事とする．

データを取得するためのデバイス，データを表示するためのデバイスおよびその間の通信回線を，本章ではチャネル要素と呼ぶ．また，様々なデバイスからデータを取得する複数のチャネル要素をまとめた機構をマルチメディアチャネルと呼ぶ．遠隔の拠点間をマルチメディアチャネルで接続することにより，遠隔のブレインストーミングを支援できる．このとき，非同期型ブレインストーミングを支援するために，蓄積したマルチメディア情報の関係を定義するデータをアノテーションと呼ぶ．コラボレーションの支援において会議内容の再利用は重要な課題であり，会議内容にアノテーションを付与することで再利用を促進することが可能であることが分かっている [97, 54]．このアノテーションを用いることにより，蓄積したデータの再利用が促進される．

6.5　エージェントプラットフォームを用いた動的に垂直連携可能なマルチメディアチャネルを持つブレインストーミング支援システムの提案

6.5.1　アプローチ

2.3.1 項で示した通り，同期型ブレインストーミングと非同期型ブレインストーミングを同時に実現することが十分にできておらず，同期型ブレインストーミングの過程や結果のデータを，非同期型ブレインストーミングやその後に発生した同期型ブレインストーミングで再利用することが困難である．また，マルチメディアに対応するために複数のメディアに対する支援ツールを同時に利用する必要があるが，その際に連携可能な仕組みが存在しない．そこで，本研究では，6.5.2 項において，各メディアを連携するために同一のインタフェース構造を持ち，マルチメディアの通信機能に対応した同期型および非同期型のブレインストーミングを同時に支援するチャネル要素とチャネル要素を組み合わせたマルチメディアチャネルの概念を提案する．

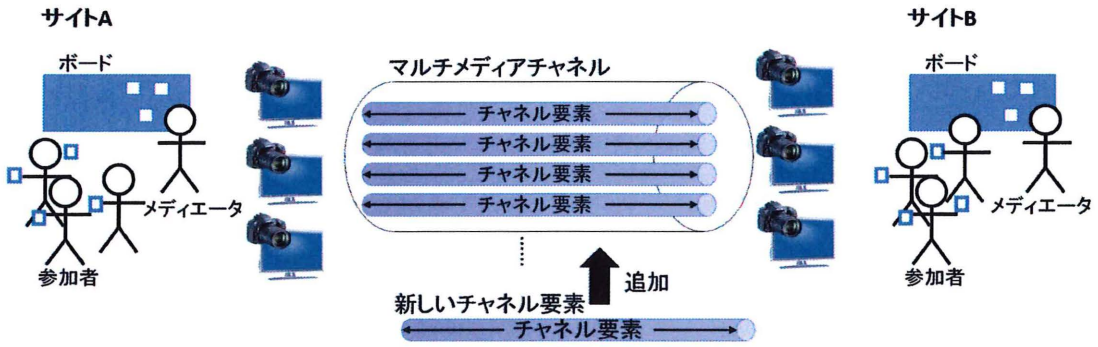


図 6.6　リモートブレインストーミングのためのマルチメディアチャネル

6.5.2　リモートブレインストーミングのための同期／非同期型マルチメディアチャネルの提案

図 6.6 にマルチメディアチャネルの概念を示す．リモートサイト間の会議中はリアルタイムにデータの共有を行い，場の情報をより多く共有することで互いの状態を共有し，ブレインストーミングを行うことが必要である．遠隔拠点間のリソース毎の通信を資産化の要求に従って実装するためにアプリケーションの通信の単位をチャネル要素という概念によって定義する．また，リモートブレインストーミングに必要なチャネル要素をマルチメディアチャネルに統合することにより，ブレインストーミングのアクティビティの情報をより多く共有することを目指す．マルチメディアチャネルはチャネル要素を容易に追加できる構造とし，以下の機能を持つこととする．

- 同期型ブレインストーミングを支援するリアルタイム通信機能
- 非同期型ブレインストーミングを支援するリソースの蓄積と資産化機能
- 資産化したリソースを再利用するためのアノテーション作成と検索機能

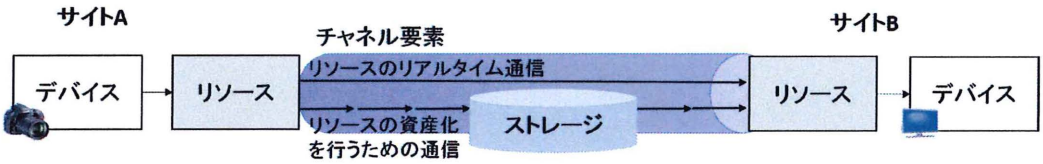


図 6.7　ブレインストーミングを支援するためのチャネル要素の構造

図 6.7 に同期・非同期のブレインストーミングを同時に支援するチャネル要素の構造を示す．同期型ブレインストーミングはリアルタイムの通信により，主にブレインストーミング中の場の情報を共有するために利用される．ITU-T G.114 では，インタラクティブな会議を円滑に行うために必要なレイテンシは 400ms 以下とされている [33]．そのため，本研究においてもリアルタイム通信の制約条件としてレイテンシは 400ms 以下とする．また，リソースの資産化のための通信はリアルタイムに通信する必要はないが，確実にストレージに保存できる必要がある．

6.5.3　リモートブレインストーミングのためのチャネル間アノテーション機能の提案

図 6.8 にチャネルのアノテーション機能の詳細を示す．本章では，[52] の論文で示したアノテーションの自動生成を行うために，イベント認識機能とアノテート機能を追加する新しいアノテーション機能の設計を行った．チャネル間アノテーション機能は，チャネル要素を流れるデータからブレインストーミングのアクティビティに関連付けることができるイベントを抽出し，他のチャネル要素を流れるデータに対して，タイムスタンプを用いてイベントからデータへのリンクを生成する機能である．具体的には，以下の 2 つの工程により実施される．まず，各チャネル内に流れるストリームのデータを非同期型の通信より取得し，イベント認識モジュールによりチャネル毎に異なるブレインストーミング中に起こるイベントを認識する．この時，イベントの認識によって得られたイベントの情報と，取得時のタイムスタンプが生成される．これらの情報をアノテーションと呼ぶ．次に，アノテートモジュールにより他のチャネルを流れる同時刻のストリームのデータに対して，アノテーションによる意味づけを行う．

以上により，チャネル間アノテーション機能は，ストリームを流れるデータに対してアノテーションによる意味づけを行うことができる．チャネル間アノテーション機能によって生成され，付与されたアノテーションは，チャネル毎のストリームに対してタイムスタンプを起点としたリンクを可能とする．よって，このリンクを用いることで，チャネル同士が連携することを可能となる．

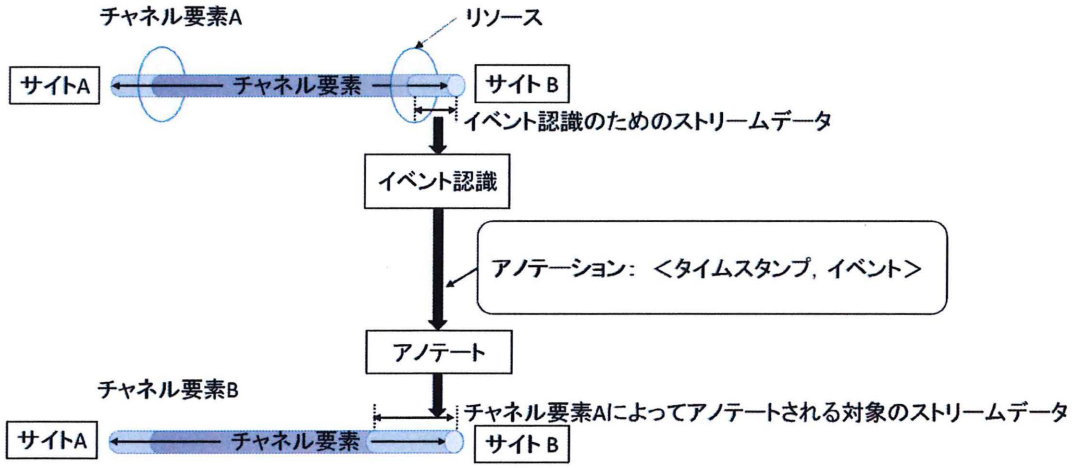


図 6.8　チャネル間アノテーションの機能

6.6　エージェントプラットフォームを用いた動的に垂直連携可能なマルチメディアチャネルを持つブレインストーミング支援システムの設計

6.6.1　リモートブレインストーミング支援システムの概要

図 6.9 に 6.5 節で提案した概念を実現するためのシステム全体の設計を示す．本システムは，ノートチャネルとビデオチャネルと呼ばれるチャネル要素により同期型/非同期型ブレインストーミングを支援する．ノートチャネルとは，2.3.1 節に述べたブレインストーミング支援機能を，同期型および非同期型へ同時に適用可能にした新しいチャネル要素である．また，ビデオチャネルは，2.3.3 節において述べた異なるマルチメディ

アコンテンツであるビデオチャットを同期及び非同期に支援するための新しいチャネル要素であり，カメラとディスプレイが 1 個ずつ含まれる単純型と，複数のカメラがビデオスイッチャにより一つ選択されディスプレイに送信するスイッチ型の 2 種類が実現されている．カメラ切り替えの行為は，イベントとしてアノテーション機能によって抽出され，ビデオストリームデータとノートデータにリンクされる．ビデオチャネルは，複数のチャネル要素としてマルチメディアチャネルに組み込まれる．ノートチャネルとビデオチャネルの設計について以下に述べる．

また，各チャネル要素は 5 章で示したアーキテクチャにより，エージェントプラットフォームを用いて動的に垂直連携可能とすることで組み合わせをリアルタイムに行う事を可能とする．

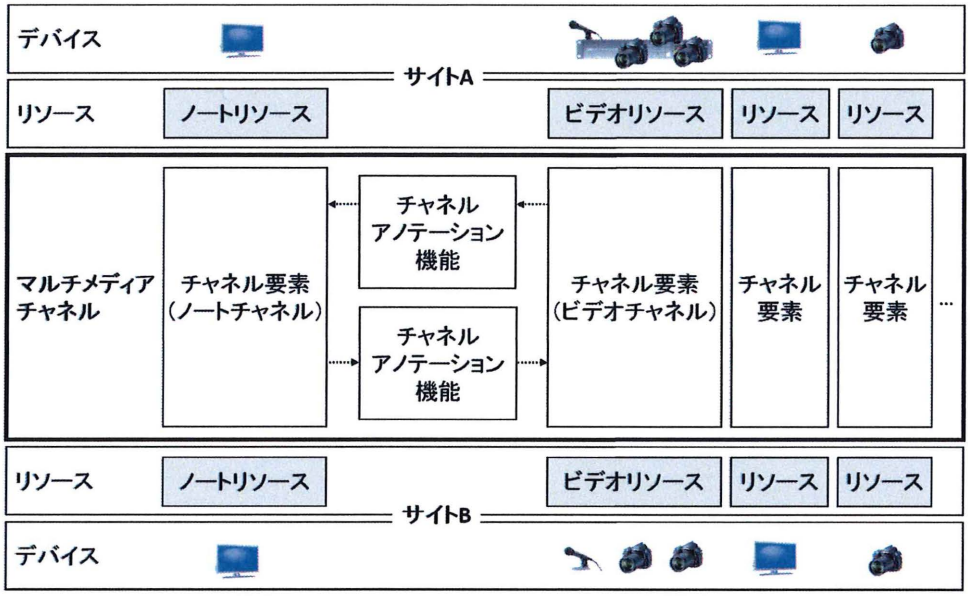


図 6.9　リモートブレインストーミング支援システムの全体設計

6.6.2　リモートブレインストーミングのためのノートチャネルの設計

図 6.10 にノートチャネルの設計を示す．ノートチャネルは，ノートとクラスタ等のオブジェクトである Note-Resource をタッチパネル式の共有ボードとタブレットからキャプチャし，サイト毎に配置された共有ボードに表示するための機能を持つ Note-Interface と，その通信先である Note-Channel-Server，そして，それらで起こったイベントの蓄積と再利用を行うためのストレージである Note-Cluster-Repository によって構成される．ノートは，参加者のタッチパネル式タブレット及びメディエータの共有ボード上で作成される．タブレットで作成されたノートは，「送信」の操作によってボードに送信される．参加者は，共有ボードにいるメディエータの操作によってアイデアと解決策をより分けるために，ノートとノートの間の関係を議論し，改善し，反映する．Note-Interface は，作成，送信，移動，変更，削除などのオブジェクトに対する操作のためにタッチパネルデバイスを制御するための機能と，アイデアや結果を表示するために相互に接続されたノート，クラスタからなるダイアグラムを表示する機能を持つ．Note-Channel-Server は，ノート及びクラスタオブジェクトをインターネット経由で送受信し．それらの蓄積及び再利用とチャネル間アノテーション機能と連携する機能を持つ．Note-Cluster-Repository は，Note-Channel-Server を介して送信されるすべてのリソースを格納する．

図 6.11 において，破線の矢印は非同期の通信を表し，実線の矢印は同期の通信を表す．ノートチャネルは，ボード上での操作に起因するノートおよびクラスタの変更を送受信し，各サイトのボード上のダイアグラムを書き換えることにより表示内容を同期する．この際，非同期の通信を Note-Channel-Server を経由して行い，同期の通信を Note-Interface 間で行う．

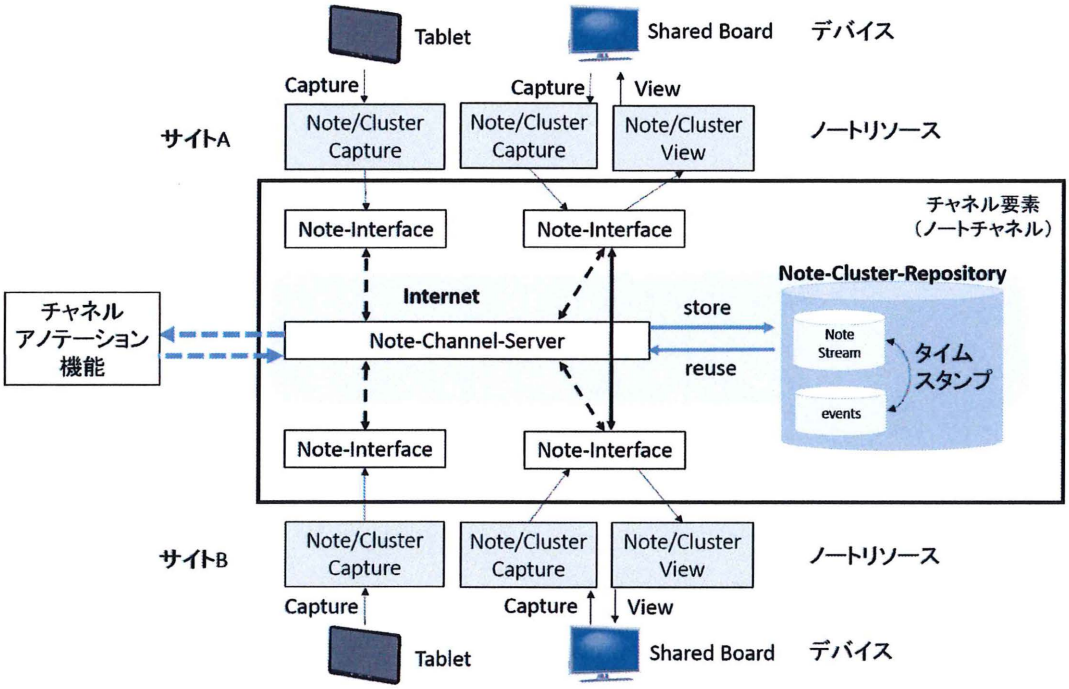


図 6.10　リモートブレインストーミングのためのノートチャネルの設計

図 6.11 にノートチャネルのユーザインタフェース (Note-Interface) の設計を示す．ユーザインタフェースにより参加者が自サイトで移動したノートは，同期されている遠隔サイト上でノートチャネルを通じて自動的に移動される．また，ノートを分類するためにクラスタ上でグループ化することが可能である [69, 53]．参加者は仮想キーボード機能により，ノートを編集する．この機能は，マルチタッチディスプレイ上での実装となっているため，複数人による同一画面での操作を可能とする．そして，画面をタップ&ホールドすることで，Note-Interface の機能にアクセスするためのメニューを開くことができる．

6.6.3　リモートブレインストーミングのためのビデオチャネルの設計

図 6.12 に，ビデオチャネルの設計を示す．ビデオチャネルは，ビデオストリーム (Video Stream) やそれに添付されたタグ (Tag) 等のオブジェクトである Video-Resource を扱うためのチャネルである．このチャネルは，マルチタッチディスプレイとカメラに接続されたローカルコンピュータで動作するコンポーネントである Sender，Receiver，および Tag-Interface と，タグを生成するためのプログラムである Annotator，そして，クラウドサーバ上で動作する Video-Channel-Server と Video-Repository から構成される．タグオブジェクトは，ビデオのシーンを識別するために，参加者がタブレットまたはボード上の Tag-Interface を操作することによって生成され，イベント名とビデオストリームオブジェクトを関連付け，タイムスタンプを伴ってイベント名を記憶する．Video-Repository はストレージとしての機能を持ち，すべてのビデオストリームオブジェクトを蓄積及び検索する機能を持つ．

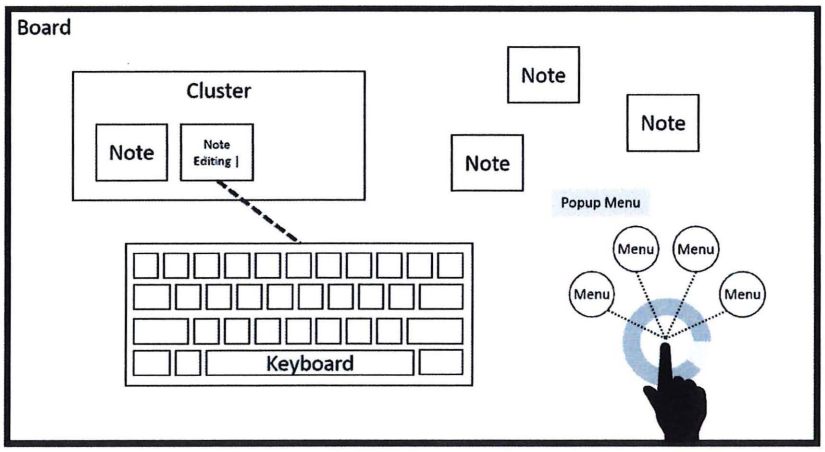


図 6.11　ノートチャネルのユーザインタフェース

また，参加者またはメディエータによってビデオストリームに追加されたタグも，検索のための重要な情報である．Annotator は，メンバが操作するカメラの切り替えイベントに応じて，タイムスタンプ付きの Tag を作成するためのインタフェース機能である．Tag-Interface は，それを Video-Repository に送信してビデオストリームオブジェクトにタイムスタンプと共に関係付ける．

Sender と Viewer は，ビデオストリームオブジェクトの送受信を行う機能を持つビデオチャネルの重要なコンポーネントである．図 6.13 に Sender と Viewer の機能を持つビデオチャネルのユーザインタフェースの設計を示す．Sender と Receiver の通信のために，暗号化によりセキュアな通信を行うことが可能でリアルタイム性を高くするためにピアツーピアの通信とクライアントサーバ通信を切り替えることが可能なプロトコルを採用する．

Video-Channel-Server は，ビデオストリームオブジェクトをインターネット経由で送受信し．それらの蓄積及び再利用とチャネル間アノテーション機能と連携する機能を持つ．また，ビデオチャネルでは，非同期の通信を Video-Channel-Server を経由して行い，同期の通信を Sender と Viewer 間で行う．

さらに，サイト A 側のみに存在する Video Switcher は，複数のカメラを切り替えることが可能であり，ビデオストリームをシステムに対して入力するデバイスである．通常，ビデオストリームにおけるイベントは構造化されていないデータであるため認識することが難しい．しかしながら，この機構を挟むことにより，カメラのどのカメラからの入力の制御が可能となり，現在表示されているのか，そして，いつカメラが切り替わったか等の情報を取得可能となり，ある程度の資産化が可能となる．

6.6.4　チャネル間アノテーション機能の設計

6.5.3 節の機能を実現するために，チャネル間アノテーション機能を設計する．本機能は [52] で示した資産化とその検索を行う機能である．図 6.14 に，チャネル間アノテーション機能の設計を示す．本機能は，各チャネル要素において認識されたイベントをアノテーションとして保存し，検索することができる．Annotation Generate モジュールは各チャネル要素からのイベント情報を受けてアノテーションの形式で Annotation DB に保存する機能を持つ．この時，アノテーションは JSON 形式により記述され，アノテーションの種類によって記述形式が変化する．また，Annotation Reuse モジュールは Annotation DB に保存されたアノテーションを各チャネル要素に対して提供する機能を持つ．このモジュールは，チャネル要素において保存されたアノテーションを利用する際に呼び出され，過去に保存されたアノテーションを呼び出すことで，資産化された

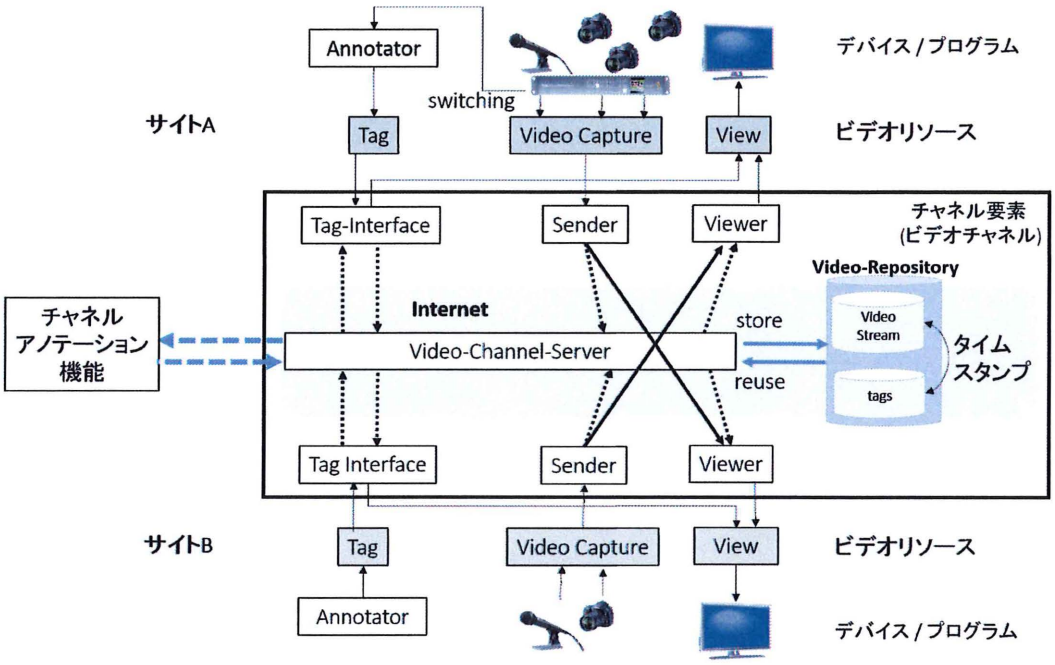


図 6.12　リモートブレインストーミングのためのビデオチャネルの設計

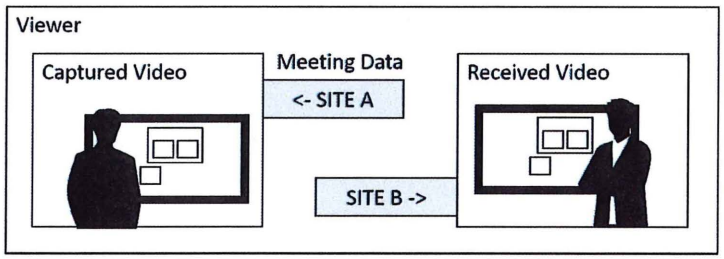


図 6.13　ビデオチャネルのユーザインタフェースの設計

データの再利用を可能とする．

図 6.15 にチャネル間アノテーション機能のユーザインタフェースの設計を示す．Annotation Table には各チャネルのサーバ経由で得たイベント情報により生成された Annotation が表示される．また，検索のための機能として，各行をユーザが選択する事により，選択された Annotation が生成された時刻まですべてのチャネルの時刻が戻り，すべてのチャネルで履歴を操作する動作が可能である．

6.6.5　チャネルランチャ機能の設計

図 6.16 に，マルチメディアチャネルに対してチャネル要素を追加するためのインタフェースの画面設計を示す．「Add Channel」を行うことで必要に応じて簡単にチャネルを追加することができる．Site ID を設定することで自分の拠点に関する ID を設定する．初期状態である場合，接続されたサーバより自動的に ID を振られることとする．また，設定を保存する場合は Apply ボタンを押下する．各拠点におけるランチャは設定を行った Topic ID によってすべて接続されており，チャネルの追加された際，すべての拠点においてチャネルが追加される．さらに Change Viewer Mode ボタンを押下する事により，接続されているすべてのチャネル

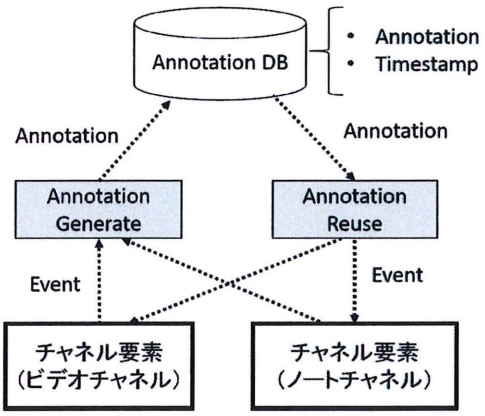


図 6.14　チャネル間アノテーション機能の設計

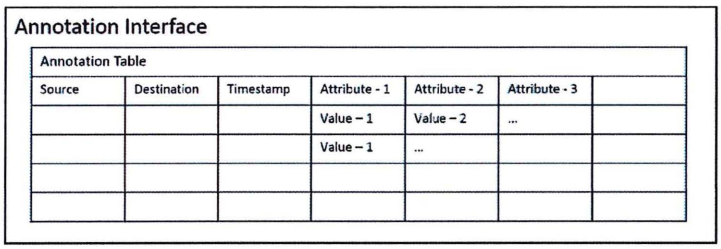


図 6.15　チャネル間アノテーション機能のユーザインタフェースの設計

が非同期ブレインストーミングの状態となり，議事録の閲覧やアノテーションによる各チャネルの確認を行うことができる状態となる．また，上述の制御による結果を把握するために，チャネルランチャ機能が起動する際，同時にチャネル間アノテーション機能が起動する事とする．

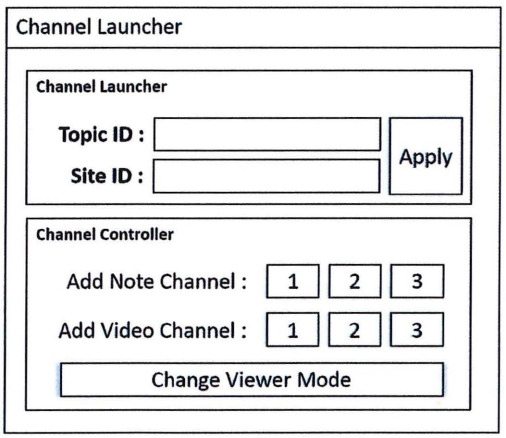


図 6.16　チャネルランチャ機能のユーザインタフェースの設計

6.7　まとめ

本章では，まず，遠隔地間におけるブレインストーミングアクティビティの資産化について重要な 3 つの特徴について提案を行った．セッション中に生成されたリソースはノートチャネルに格納する必要があり，また，セッションがビデオチャネルに記録されなければならず，同時に，それらのデータを永続的なシステムにおいて保存を行い，すべての要素をリンクできるようにする必要がある．これは，ブレインストーミングのセッションの円滑な進行とリソースの資産化の両方を可能にするアプリケーションを必要とする．ビデオにおいて起こるイベントのアノテーションは，ビデオストリームと共に他のアクティビティにおけるイベントに関連付けられ，ビデオのリプレイをするなど再利用をする際に会議中に起こったイベントの原因や本質に関する情報を得るために利用可能とした．さらに，エージェントプラットフォームを用いたブレインストーミング支援を行うシステムのメインアプリケーションとしてこれらを提案し，設計を行った．

しかしながら，上述のシステムでは，同期型ブレインストーミングと非同期型ブレインストーミングを同時に実現することが十分にできておらず，同期型ブレインストーミングの過程や結果のデータを，非同期型ブレインストーミングやその後に発生した同期型ブレインストーミングで再利用することが困難であったあり，マルチメディアに対応するために複数のメディアに対する支援ツールを同時に利用する必要があるが，その際に連携可能な仕組みが存在しないという問題点があった．そこで，各メディアを連携するために同一のインタフェース構造を持ち，マルチメディアの通信機能に対応した同期型および非同期型のブレインストーミングを同時に支援するチャネル要素とチャネル要素を組み合わせたマルチメディアチャネルの概念の提案を行い，設計においては，5 章で示したアーキテクチャにより，システムを動的に垂直連携可能とすることでこの問題点を解決することを可能とした．

次章では，上述の 2 つの提案と設計に対して実装と実験による評価を行う．

第 7 章

ブレインストーミング支援システムによるエージェントプラットフォームの評価

本章では，6 章で示されたテストベッドにおける IoT アプリケーションの提案と設計を用いて，さらに 5 章で示したマルチメディアチャネルの概念の提案を用いてより具体化した実装を行う事で，会議支援システムとしての機能の拡張と評価を行った．

機能の拡張として，同期型および非同期型のブレインストーミングの成果を表現するマルチメディアデータを遠隔拠点間で共有するためのチャネル要素を提案し，チャネル要素を利用者の要求に対応して組み合わせたマルチメディアチャネルにより，ブレインストーミングの型にあった支援を可能とした．さらに，チャネル要素間のデータをリンクするアノテーション機能により，これまで資産化したブレインストーミングの成果を再利用することが可能とした．

本章では，評価実験として，実装されたアプリケーションを利用して，日本とフランスの二つのチーム間で行われたリモートブレインストーミング実験により，目的とする支援機能が実現されていることを示すために，実装による検証と，レイテンシ等の測定による実装した全体システムの実験による検証を行った．

実験の結果，チャネルの概念に基づき，同期的・非同期的なチャネルを状況や環境に応じて適応的に追加・削除することを可能とし，各チャネル要素を実現している IoMT アプリケーションにおける動的な垂直連携が可能なアーキテクチャの実現が確認できた．そして，チャネル間の連携を行うことで資産化したブレインストーミングの成果の再利用が可能である事を確認した．

7.1 エージェントプラットフォームを用いたブレインストーミング支援システムの実装

本節と次節では，6 章で示したブレインストーミング支援のためのコラボレーションのためのツール群と資産化のための機能を持つブレインストーミング支援システムをリソースコネクタプラットフォームにより実現するために実装と評価実験を行う．

7.1.1 ノートチャネルの実装

マルチエージェントシステムは JADE プラットフォーム[*1]によって実装した．サーバアプリケーションはメインコンテナに含まれ，他のアプリケーションはメインコンテナにリンクされたセカンダリコンテナにビル

[*1] JADE stands for: JAVA Agent DEvelopment Framework. http://jade.tilab.com/

ドされる．これらのコンテナ群によりエージェントシステムが形成される．JADE プラットフォームは非常に堅牢であり，動的に新しいコンテナや新しいエージェントを統合し，エージェントの消失を観察・監視することができる．

メインアプリケーションは大きなインタラクティブデバイス上で動作し，MT4J ライブラリを用いた Java 上で実装を行った．このライブラリは，アプリケーション上のグラフィック要素に結合させることができるタッチイベントのための高レベル API を所有している．このことから，ミーティングエージェントはすべてのジェスチャに関するイベントを取得することが可能である．

GUI とエージェントとの間の通信は，オブザーバ設計パターンに基づいた Java Bean アプローチで実装した．例えば，エージェントがユーザインタフェースに関するメッセージを受信したとき，シーンによって受信された通知を起動し，その内容に応じてプロパティの一つを変更することが可能である．

7.1.2　ビデオチャネルの実装

まず，Skype と Ustream のサービスを使って，Sender，Receiver，Viewer，Capture の要素の試作を行った．しかしながら，この最初の実装ではシステムに統合することが困難な閉鎖されたサービスを使用していたため，提案された設計に準拠した実際のシステムに拡張することはできないことが分かった．そこで，これらの要素の新たに実装を行った．

提案する Video Streamer は，従来のライブラリを使用し，Microsoft Windows 上で Visual C#を用いて開発を行った．各コンポーネントは，Red5 Media Server[*2]を Video Steaming Server，Open Broadcaster Software[*3]を Sender コンポーネント，Web ブラウザーコントロールとして Shockwave Flash Object[*4]を Receiver コンポーネントとして実装を行った．各部の通信プロトコルとしては，HTTP はセッションの開始と終了に使用し，RTMP をビデオストリーミングの送受信に使用した．

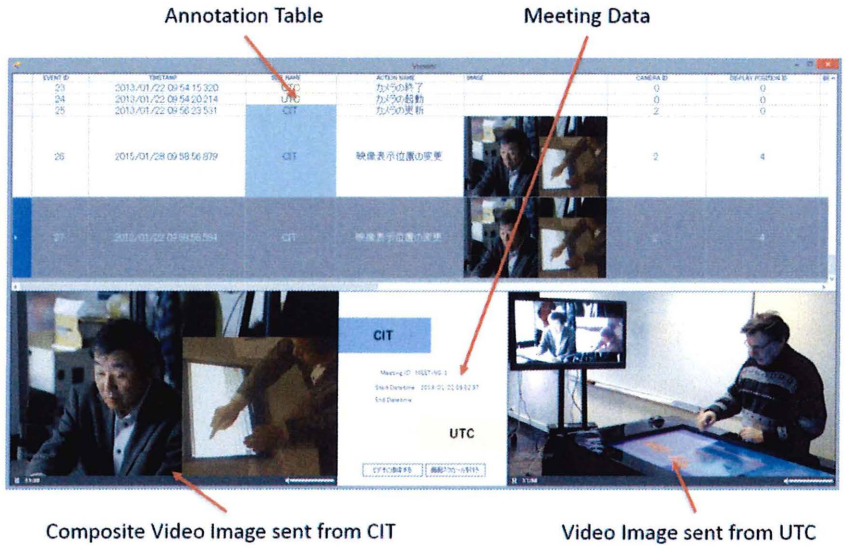


図 7.1　CIT 側チームによるブレインストーミング会議の録画映像の視聴実験

[*2] http://red5.github.io/
[*3] https://obsproject.com/
[*4] http://www.adobe.com/products/flash.html

ビデオスイッチャは主にデバイスとして Blackmagic Design ATEM Television Studio[*5]を使用して実装され，カメラからの複数のビデオストリームをビデオストリーミングサーバに送信される 1 つのビデオストリームに構成する．また，ビデオストリームを構成するための Video Operator 要素を開発し，この要素によってメディエータまたは参加者が操作するためのユーザインタフェースを提供する．これらの機能によって起こったアクションの説明が Video Annotator に送信され，Video Annotation DB によって MySQL データベースに格納されたアノテーションに変換される．

図 7.1 にブレインストーミングセッションのビデオを見直すことができる機能を持つ Video Viewer の表示画面のスクリーンショットを示す．同時刻に撮影された両チームのビデオ画像[*6]が下部のパネルに表示される．図 7.1 の上部のパネルは，アノテーション（タイムスタンプ，操作名，カメラ ID，ビューの表示位置等）のデータを表示する Annotation Table である．ユーザは，テーブルの行内で 1 つのデータ（タイムスタンプや操作など）を選択し，セッション中に対応するイベントが発生したときと同じようにビデオを再生することができる．

7.2　エージェントプラットフォームを用いたブレインストーミング支援システムの評価実験と考察

7.1 において，フランスのコンピエーニュ工科大学（UTC），および日本の千葉工業大学（CIT）で同時に実施された実験に関する詳細を説明する．実験シナリオとして，遠隔地の 2 つのチーム（UTC および CIT）が，日本に滞在するフランスの旅行者のためのウェブサイトの予備的なデザインについて協力して作業することと決めた．

両チームは，リモートブレインストーミングセッションに関与しており，上述のシステムを使用している．各チームにおける参加者はアイデアを作成し，このプロジェクトについてアイデアを分類する必要がある．アイデアは，クラスタによってグループ化し，インタラクティブデバイスであるボード上に表示され，様々なアイデアがメモとして記述されている．インタラクティブデバイス上の各要素はズームなどを適用することができるとともに，移動等も行う事ができ，一方のチームのアクションは，他のチームのデバイスにリアルタイムで送信されている．UTC 側では，人々はテーブルトップ型の対話型テーブルの周りでメンバが作業を行っており，テーブルの上に向けられた単一のカメラには，テーブルの上の要素を利用して作業を行う人々を示すシーンを送信していた．これにより，CIT 側のメンバがこの作業風景をビデオストリームにより確認を行い，作業者の音声を聞くことが可能であった．

UTC 側では，CIT 側からのビデオストリームは大きいスクリーンに表示された．日本の CIT 側におけるシステムは単一のカメラを扱う UTC 側のシステムと違い，3 つのカメラのストリームを統合する機能が存在し，その機能により，複数のカメラの映像を選択的に送信することが可能であった．これは，セッションのモデレーターは大型ディスプレイを操作し，日本語でノートを記述するためには他の座っている参加者が操作可能で表示可能な小型デバイス利用することでシステムを構成していた．

また，この時，3 つのカメラの向きやビデオストリームに統合される映像が議論に応じて変更された．この時，UTC 側においても参加者はテーブルの上において CIT 側の行動がデバイス上でリアルタイムに更新されることを確認できた．また，誰もが共通のサーフェスと直接インタラクションすることができるため，本当の意味でのメディエータは存在しない状態であった．

[*5] https://www.blackmagicdesign.com/products/atemtelevisionstudio/

[*6] 実験中，ビデオストリームのディレイは非常に低く，画像を同期させるのに必要十分な状態であった

CIT 側では，場は 1 つのインタラクティブホワイトボードがインタラクションを行うために表示され，利用できる状態であった．CIT 側では，これが 2 つの指によって拡大，縮小，回転のような移動に関するアクションを行うことができるあったが，ノートはズームと回転を許すために設計されており，ノート上部のハンドルと右下の四角のゾーンを 1 回タッチすることでそれらのアクションを行う事が可能であった．図 7.2 において，これらの特性を持つクラスタとノートを示す．

メディエータは，他のチームメンバとの話し合いに応じて，ホワイトボードの移動の近くに立ち，表示されている要素をグループ化することができる（図 6.2 を参照）．モデレータが新しいメモを作成することは，仮想キーボードを開き，キーをタイピングするなどを行わなくてはならず，簡単に入力できるものではないため，好ましくない入力方法であるともいえる．しかしながら，他の参加者はタブレット PC を持つことにより，それらの上で動いている特定のアプリケーションのおかげで，メモを作成することができ，ホワイトボードに対してそれらの作成した結果を送ることができたため，この問題点を回避できたといえる．

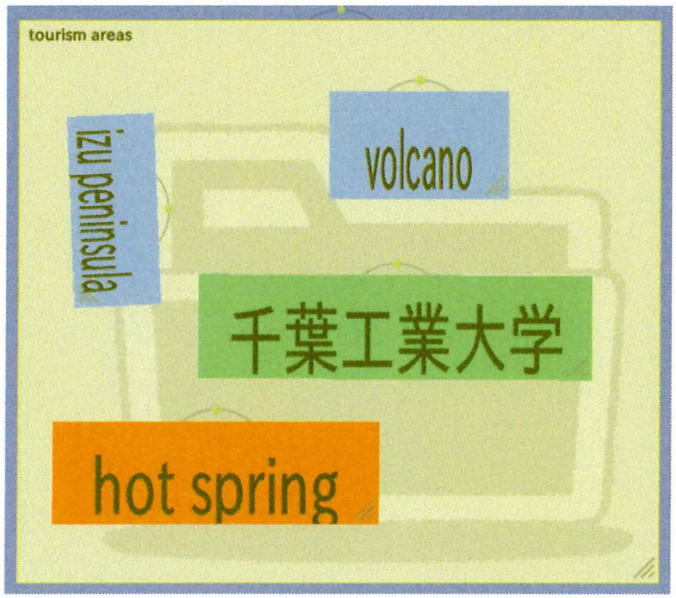


図 7.2　ズームと回転のための操作オブジェクトと四角い領域で表示されるクラスタ及びノートの例

実験時におけるセッションは，各サイトの間の様々なコミュニケーションの問題やネットワークの問題が起こらず約 2 時間続けることができた．よって，この観点においては満足できる結果であると考えている．また，ノートチャネルはビデオチャネルより早い速度で更新されることが分かった．最初のセッションにおいては，参加者は遠隔地の参加者に知らせるために彼ら自身のアクションを説明することが必要であると考えていた（口頭による「私は”伊豆”と書かれたノートを動かし，私は”山”クラスタなどを削除する」などの説明）．セッション開始後，しばらくして，それらはそこまで必要ではないという結果となった．まず，最初に参加者はアイデアの推敲より多くの種類の要素のマニピュレーションに集中することとなった．

主要な問題はノートで作成し続け，日本側のチームはそれのためのそれらのタブレット PC を使うことができるだけである．それは，フランス側のチームがバーチャルなキーボードでしようとしたようにノートをテーブルに直接タッチして入力することで作成するより快適であったようである．

しかしながら，それらの大きなサーフェスとタブレット PC について比較するのは難しく，どちらにも利点が存在する．片方のチームはテーブルトップのテーブルであり，また他方のチームはボード型のサーフェスを

用いているためである．直立のボート型サーフェスの場合，水平型のテーブルよりも通常の会議時のホワイトボードを利用した形と近いため，使いやすいと考えられる．また，これらのデバイスはメディエータによって議論を整理するために重要な役割を持つことが分かった．

7.3　動的に垂直連携可能なマルチメディアチャネルを持つブレインストーミング支援システムにおけるエージェントプラットフォームを用いた実装

本節と次節において，5章で示したアーキテクチャにより，システムを動的に垂直連携可能とすることでこの問題点を解決することを目的とし，システムの実装及び評価を行う事で動的に垂直連携可能なアーキテクチャの評価を行う事とする．

7.3.1　ノートチャネルの実装

Note-Channel-Server，Note-Cluster-Repository およびノートチャネルの4つの Note-Interface は，インターネット経由でノートやクラスタを含むエージェントメッセージを通信するためのマルチエージェントシステムを提供する JADE エージェントプラットフォームを使用して実装を行う．Note-Interface は，タッチパネルを制御する Java プログラムをエージェントにラップするために，実装された JADE エージェントプラットフォームで動作するエージェントとして実装を行う（図 6.10 を参照）．

本研究では，各エージェント群やリソースとサーバとの通信を Publish/Subscribe Model[23] による実装を行うために，MQTT による実装を行った．MQTT Broker として，VerneMQ[*7]を利用し，クライアントの実装として Paho[*8]により提供されるライブラリ群を利用する．

インタラクティブなテーブルとボードで動作するマルチタッチアプリケーションは，Java の MT4J ライブラリで開発を行う．2つの遠隔地間で2つのシステムを同期するために，それらは WebRTC プロトコルを介して通信を行った．JADE プラットフォームは FIPA 準拠であるため，HTTP サーバをホストでき．別の JADE プラットフォームにあるエージェントにメッセージを送信することができる．その際，同じプラットフォーム上のエージェントのようにメッセージを通信することが可能である．しかしながら，HTTP プロトコルの通信ではビデオチャネルのパフォーマンス要件を満たすことができないため，WebRTC により通信を仲介するプログラムを C#により実装する．

7.3.2　ビデオチャネルの実装

ビデオチャネルの各コンポーネントは，他のエージェントからの制御を可能とするためにリソースコネクタプラットフォームのライブラリを利用し，Visual C#用いてダイナミックリンクライブラリ形式で実装を行った [89]．リソースコネクタプラットフォームで動作するエージェントは，JADE エージェントプラットフォーム [13] で動作するエージェントとエージェントメッセージの送受信を行うことにより，共同作業を行うことが可能である．

また，Video Switcher として ATEM Television Studio を利用し，SDI ケーブルによりカメラデバイスとして4台の LUMIX DMC-GH4 を接続した．Video Switcher を制御するためのプログラムは C#により実装

[*7] https://vernemq.com/
[*8] http://www.eclipse.org/paho/

し，Video Switcher からの映像は Blackmagic Intensity Pro 経由でデータとして取り込んだ．

ビデオチャネルが実現するビデオチャット機能は WebRTC（Web Real-Time Communication）に基づいて実装を行った．図 6.12 の Sender, Viewer，そして Tag-Interface は Visual C#を用いて実装を行い，各拠点に WebRTC のクライアントとしてインストールを行った．Video-Channel-Server は，Signaling サーバとして PeerServer[*9]と STUN/TURN サーバとして coturn[*10]及び Janus[*11]，そして Python 及び MySQL を用いてサーバプログラムの実装を行った．

7.3.3　チャネル間アノテーション機能の実装

チャネル間アノテーション機能を実現するために，Python により各モジュールを実装し，データベースサーバとして MySQL[*12]を利用した．各チャネル要素との通信には MQTT プロトコルを利用し，実装及び利用したサーバやプログラムは OpenStack のクラウドサービスである ConoHa[*13]上にデプロイを行った．

7.3.4　エージェントプラットフォームの実装

リソースコネコネクタプラットフォームと各種エージェントとして Edge Resource は，C# (.NET Framework 4.5.1)，Cloud Resource は Python 3.6 を使用して実装を行った．エージェントの空間を形成するために，ConoHa のサーバ上には MQTT の Broker の Topic としてコテリが形成され，サーバ上のエージェントによりアクティビティの保存や永続化の処理を実装した．エージェント群はサーバの同一の空間（コテリ）において接続可能とされるため，どの場所においても同一のメッセージ空間において接続可能となり，各エージェントがインタラクション可能な形となる．サーバとの通信は TLS により暗号化され，クライアント証明書を持たない場合，接続自体ができない構造として実装を行った．

7.4　動的に垂直連携可能なマルチメディアチャネルを持つブレインストーミング支援システム及びエージェントプラットフォームの評価実験

7.4.1　全体システムの動作確認実験

提案システムにおいて同期型ブレインストーミングと非同期型ブレインストーミングを行うことができるか検証するために動作確認実験を行った．このとき，サイト A を千葉工業大学（日本），サイト B をコンピエーニュ工科大学（フランス）とした．この動作確認実験のために，2 つの大学間で数年の間にいくつかの実験が行われてきた．ネットワーク通信による障害を克服し，大学に設置されたセキュリティ機能を考慮に入れて，ここに提示されたコンセプトを実現することを可能とした．さらに，このプロジェクトはリモートコラボレーションプロジェクトであるため，良いベンチマークとしての役割も果たしている．

本実験では，まず，マルチメディアチャネルを実現できているかどうかを確認するために，はじめにチャネルランチャを起動し，続いてノートチャネルを 2 つとビデオチャネルを 3 つ起動することでマルチメディアチャネルを構築できることを確認した．2 番目の実験項目として，リソースの資産化ができているか確認するために，チャネル間アノテーション機能におけるアノテーションの生成の確認を行った．具体的には，チャネ

[*9] https://github.com/peers/peerjs-server
[*10] https://code.google.com/p/coturn/
[*11] https://janus.conf.meetecho.com/
[*12] https://www.mysql.com/
[*13] https://www.conoha.jp

ルランチャ機能の起動時に同時に表示されるチャネル間アノテーション機能のインタフェースにおいてチャネル要素で起こったイベントがアノテーションとして表示されているか確認を行った．

図 7.3 に実験中の画像を示す．図 7.4a に，CIT での実験の状況を示す．情報ネットワーク学部では，会議の閲覧者を表示するために大きなタッチボードを使用し，データエディタを表示するために別のタッチスクリーンを使用した．また，場に存在する視点の異なる 3 台のカメラで部屋の状況を取得する環境であった．図 7.4b に，UTC での実験の状況を示す．そのイノベーションセンター[*14]では，大きなタッチボードと大きなタッチテーブルを備えた部屋を利用して実験を行った[*15]．部屋内では，複数のカメラにより，異なる方向の視点を取得可能な環境であった．また，マイクはカメラに接続されている状態であった．そして，クラウドサーバは CIT チームによって管理されていた．

実験により，ノートチャネル及びビデオチャネルがすべて起動でき，すべてのチャネル要素が動作できており，状況が共有できていることが確認できた．また，チャネル間アノテーション機能のインタフェースにより，生成されたアノテーションが表示できていることが確認できた．

最後に，実装を行ったシステムの性能面の確認を行うために，動作時のビデオチャネルにおけるレイテンシとメモリ使用量の測定を行った．

まず，ビデオチャネルとデータチャネルに関する CIT のサイトと UTC のサイトとの間の遅延を測定した．図 7.5 に，これらの遅延を測定するために使用された実験的システムを示す．両方のサイトのローカルプログラムとデバイスは，クラウドサーバにアクセスして 2 つのチャネルを構築する．各カメラの解像度は 640 × 480，フレームレートは 30fps とした．

まず，正確なターンアラウンドタイムを測定するために，Boyaci らの手法 [18] を参考に測定するアプリケーションを実装した．このアプリケーションは，現在の時刻を QR コードに変換し，仮想カメラを使用してビデオチャネルにビデオを出力する機能を持っている．これらの機能は，OpenCV[*16]と ZXing.NET[*17]によって実装され，XSplit Broadcaster[*18]の仮想カメラを用いた．

次に，ping コマンドを使用して遅延を測定し，測定されたターンアラウンドタイムからシステムのレイテンシを計算した．図 7.5 に，この測定の結果を示す．このシステムで実験を行った結果，ターンアラウンドタイムは 694.32ms（フランスと日本のサイトごとにクラウドサーバまでのインターネット回線遅延を含む）であり，システム全体の遅延は 126.12ms であった（クラウドサーバでの処理時間を含む）．さらに，一方向のチャネル遅延は，63.06ms と測定された．

また，この実験では，同じ条件下で Skype を測定したところ，ターンアラウンドタイムは 1109.46ms であり，測定時のシステム遅延は 541.26ms であった．また，一方向の Skype の遅延は 270.63ms であった．

さらに，ビデオチャネルのメモリ利用量を測定を行った．結果としてメモリ利用量は CIT 側では平均 115MB，UTC 側では平均 139MB であった．また，同一の解像度と回線状況において Skype(バージョン：7.33.0.105) のメモリ利用量を測定したところ，結果は平均 163MB であった．

7.4.2　チャネル間アノテーション機能の動作確認実験

アノテーションは，ノートチャネルの「ノート生成」，「ノートの移動」，「ノートの削除」，「ノートのクラスタへの登録（分類）」などのイベントを抽出し，このイベントをビデオストリームデータのタイムスタンプへの

[*14] UTC Innovation Center: https://www.utc.fr/en/innovation.html
[*15] これらのデバイスは，Ubikey 社によって設計され，開発された: http://www.ubikey.fr/
[*16] OpenCV site: http://opencv.org/
[*17] ZXing.NET site: https://zxingnet.codeplex.com/
[*18] XSplit Broadcaster site: https://www.xsplit.com/

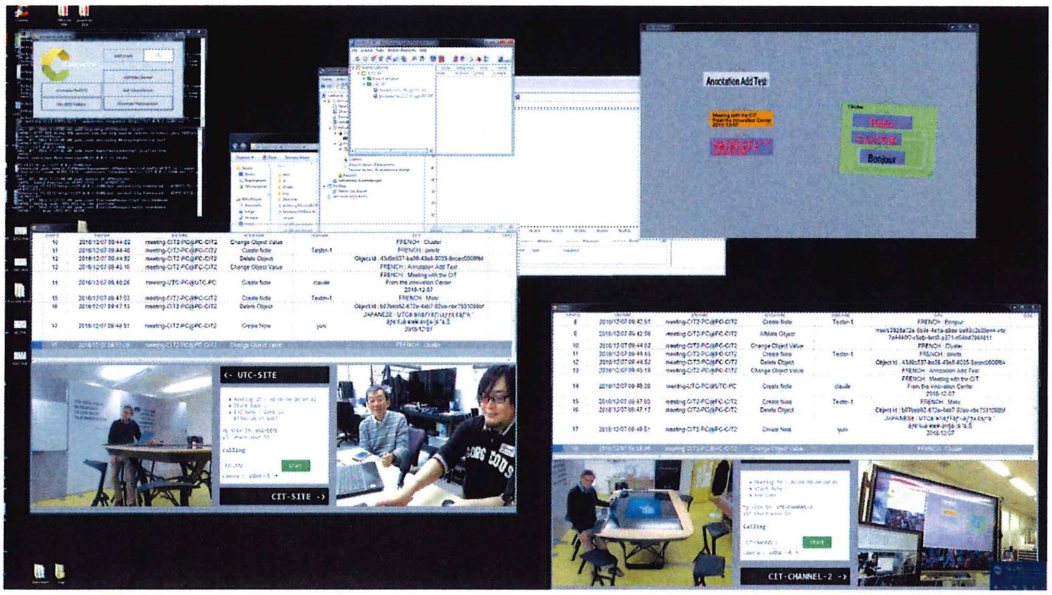
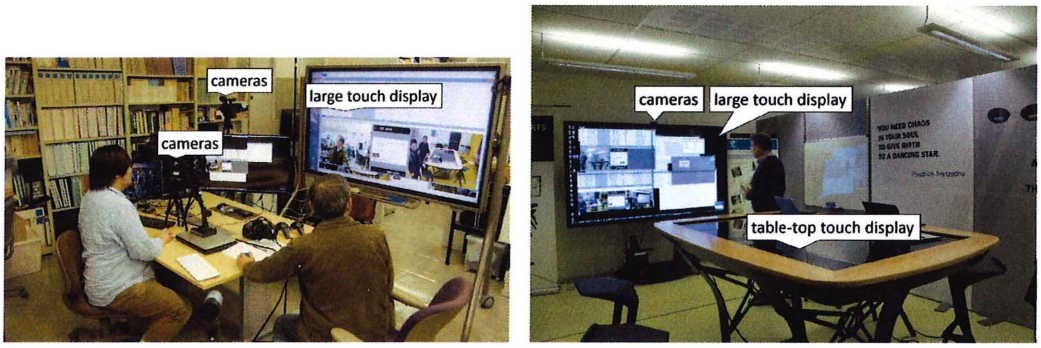


図 7.3　評価実験中のスクリーンショット



(a) CIT の観点から見た実験状況.　(b) UTC の観点から見た実験状況.

図 7.4　リモートコラボレーション支援システムの実験状況.

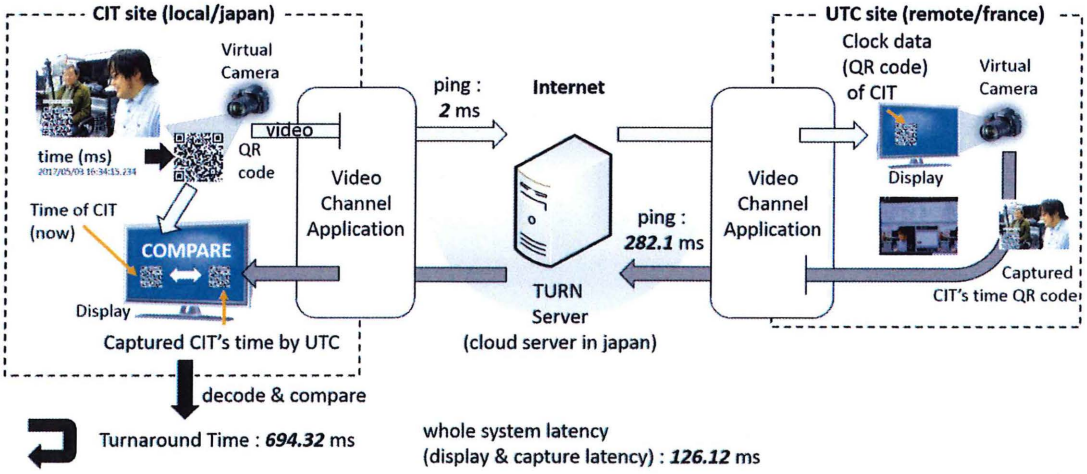


図 7.5　ビデオチャネルアプリケーションの通信レイテンシの測定.

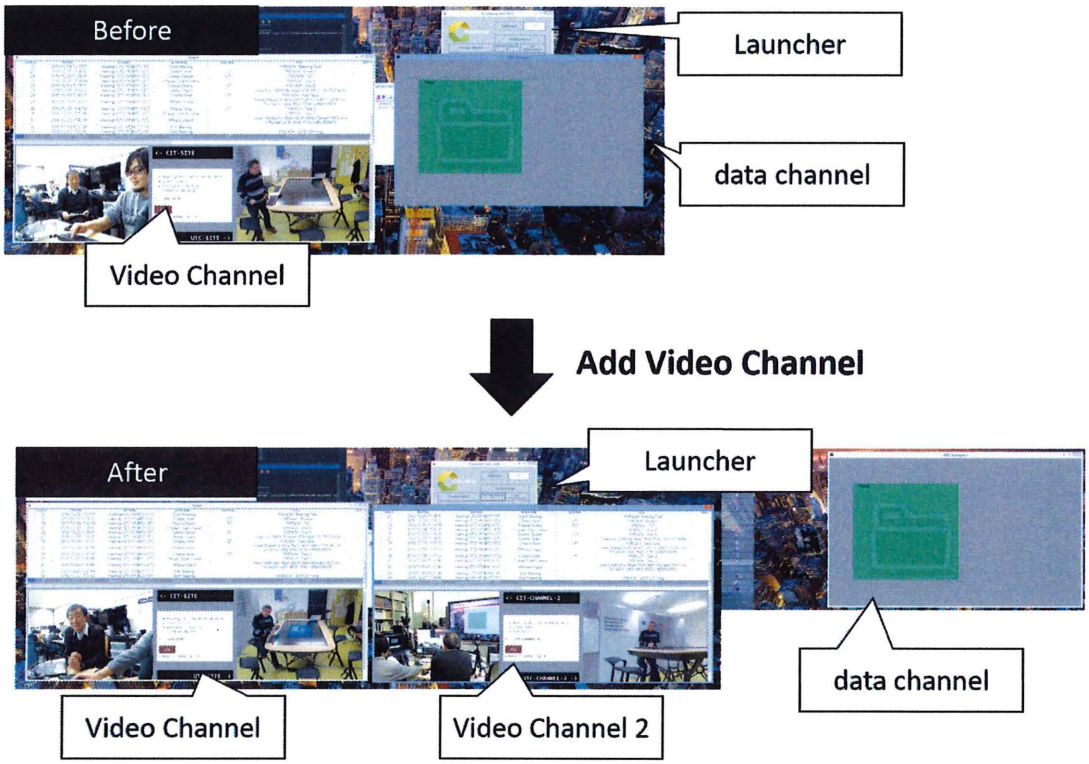


図 7.6　リモートコラボレーション支援システム上のチャネル追加操作の実験

リンクを生成する機能を作成し，図 7.3 に示したアノテーションテーブル上に表示する実験を行った．このアノテーションテーブルは，非同期ブレインストーミングの支援機能として，利用者に提示できた．これを利用して，あるノートを生成した時のディスカッションの画面を参照できるため，ブレインストーミングの記録として資産化可能となる．また，ビデオスイッチャの切り替えをイベントとして，切り替え時のノート表示画面にリンクするアノテーション機能を確認した．さらにタイムスタンプ自体がイベントとして利用できるため，ビデオ映像やノートの表示画面が，過去の任意の時点に遡れるため，ブレインストーミングの支援機能として有効であることが分かった．

ブレインストーミング支援システムのエージェントによる動的な垂直連携機能の確認実験

また，遠隔会議を管理していたユーザによって開かれた会議に利用しているシステムにビデオチャネルとデータチャネルを追加する実験を行った．この実験は，ブレインストーミング支援システムの動作時に 5 において提案を行った IoMT アーキテクチャにより，動的な垂直連携可能な IoMT アプリケーションとしての機能を実現できているか確認するために行った．これらの実験では，図 5.1 に示されている典型的な要件をテストした．システムの初期構成は，各サイトのカメラとディスプレイからなる 2 つのビデオチャネルで構成とした．

数分後，CIT サイトのユーザは，初期設定にデータチャネルを追加するために，この実験用に作成したローカル PC のコントロールパネル（Launcher）でシステムの操作を行った．操作により，両方のサイトの 2 つのエージェントが，CIT サイトと UTC サイトのリソースコネクタにメッセージを送信した．この実験では，2 つのエージェントがローカルのリソースコネクタとリモートのリソースコネクタの識別子を知っており，コラボレーション用のマルチタッチディスプレイを制御するデータチャネルコンポーネントを起動し，データチャ

ネルストリーミングを送受信するチャネルを認識していると仮定した．図 7.6 に，CIT サイトの 2 つのビデオチャネルと，ユーザの操作でディスプレイに追加されたデータチャネルのビューアを示す．

数分後，ユーザはランチャを再び操作することで，2 つのビデオチャネルを追加し，もう 1 人のビューアをディスプレイに追加し，さらに異なるカメラからの異なる画像を図 7.6 に示すように閲覧可能とした．次に，新しいビューアがデータチャネル上に表示されたため，ユーザは図 7.6 の画面上の別の場所にデータチャネルビューアの移動行った．この時，図 7.6 の状況の後に，図 5.1 に示された要件が正常に実行されたことが確認できた．

7.4.3　実験に関する考察

本システムの利点は，ノートチャネルとビデオチャネルの複数のチャネル要素がシステムに統合されているため，支援システムのユーザがリモートブレインストーミングを行う時に，チャネルランチャにより簡単にセットアップできることである．7.4.2 項及び 7.4.1 項で述べた実験では，ビデオチャネルの遅延がフランスと日本の間であっても 400 ミリ秒未満であったため，6.5.2 項で述べたリアルタイム性の基準の観点から，本システムは実用的なアプリケーションであると考えている．マルチメディアチャネルには，両方のチームで生成されたリソースを保存するためのリポジトリが含まれている．参加者は，ビデオチャネルとノートチャネルのリソースの変化をほぼリアルタイムで見ることが可能である．また，リモートブレインストーミング中およびリモートブレインストーミング後のリソースの記録をすべてのチャネルで閲覧することができる．

また，チャネル間アノテーション機能により，チャネル要素間の連携関係をモデル化し，アノテーションとしての機能を開発する事が出来た．例えば，メンバが共有ディスプレイ上でポストイットノートを送信した際，両チームのビデオストリームを見なければならない場合，操作をタイムスタンプをベースとしたイベントとして関係付けることによって，ビデオストリームを検索可能とし，ノートチャネルとの連携を可能とした．逆に，ビデオチャネルのリポジトリに格納されたビデオストリームをノートチャネルのイベントにより関係付けることにより，ビデオストリームからノートチャネルのリポジトリに格納されたノートの検索するを可能とした．チャネル間アノテーションの機能は，別のリソースをイベントにより関係付けることによってリソースを検索及び再利用可能にする点が重要であると考える．

以上により，本システムにより，同期/非同期ブレインストーミングの支援機能を同時に支援するマルチメディアチャネルの実現ができたと考える．

次に，7.4.2 項において，5.5.2 項で提案されたアーキテクチャに基づいて開発した動的な垂直連携可能な IoMT アプリケーションとしての機能を検証した．IoMT アーキテクチャの最上位第 2 層のエージェントがリソースコネクタに関する知識を持っている場合，提案された IoMT アーキテクチャに基づいて開発されたシステムの動的追加機能の動作を確認できた．この情報は，アーキテクチャのリソースコネクタのリソースプロファイルに格納され，Agents レイヤのエージェントは，Resource Connectors レイヤで使用可能なリソースコネクタの種類がわかっている場合に情報を要求できる．本実験では，エージェントはそれについて知識を持っていると仮定した．この実装による検証と動作の確認により，提案を行った IoMT によるアーキテクチャを持つエージェントプラットフォームの実現を確認することができた．

本実装では，各サイトとクラウドサーバ上に Resource Layer と Agent Layer があり，エージェントレイヤからリソースレイヤの指示が動的に変化することが可能であるため，動的な垂直連携を 5 章の IoMT アーキテクチャを持つエージェントシステムにより実現できたと考える．これまでは，人間の操作が必要であり，迅速なリソースの入れ替えができず，システム運用者の負担となっていた部分をエージェント機能により軽減できた．

7.5　まとめ

本章では，まず，6 章で示したブレインストーミング支援のためのコラボレーションのためのツール群と資産化のための機能を持つブレインストーミング支援システムをリソースコネクタプラットフォームにより実現するために実装と評価実験を行った．実験の結果，ブレインストーミング支援のために必要な要素であるノートチャネル及びビデオチャネルの実装と動作が確認できたことで，コラボレーションシステムの実現を確認できた．

次に，遠隔の 2 つの拠点間で，同期型ブレインストーミングと非同期型ブレインストーミングを同時に支援するための遠隔ブレインストーミング支援システムの実装と評価を行った．データ入力のデバイス，データ出力（表示）のデバイス，通信回線の組み合わせをチャネル要素と呼び，この要素を複数組み合わせた機構をマルチメディアチャネルと呼ぶ．チャネル要素は，通信の対象とするデータをリアルタイムで送信する機能と，クラウドのストレージに蓄積する機能，およびそのデータを参照する機能を一つの要素で実現する．これにより，マルチメディアチャネルは，同期型ブレインストーミングと非同期型ブレインストーミングを同時に支援する機能を実現した．ブレインストーミング時に生成された各チャネル要素のデータは，クラウドのサーバ内でアノテーションデータとして抽出され，タイムスタンプにより各チャネル要素が蓄積したデータにリンクが張られることにより，非同期に蓄積したデータを参照することが促進される．

実験により，提案システムを用いて千葉工業大学（日本）とコンピエーニュ工科大学（フランス）の間でブレインストーミングを行い，同期型ブレインストーミングと非同期型ブレインストーミングを同時に支援することが可能であることを確認した．実験結果において計測されたレイテンシは実用的なシステムとしての必要十分な性能を示し，それぞれの要素がリアルタイムに同期していることが確認できた．

以上の 2 つの実装と実験による評価が達成できたことにより，本章で 5.5.2 節で提案されたアーキテクチャに基づいて開発した動的な垂直連携可能な IoMT アプリケーションとしての機能を検証した．実験の結果，チャネルの概念に基づき，同期的・非同期的なチャネルを状況や環境に応じて適応的に追加・削除することを可能とし，各チャネル要素を実現している IoMT アプリケーションにおける動的な垂直連携が可能なアーキテクチャの実現が確認できた．そして，チャネル間の連携を行うことで資産化したブレインストーミングの成果の再利用が可能である事を確認した．

よって，提案を行った IoMT によるアーキテクチャを持つエージェントプラットフォームの実現を確認することができた．

第 8 章

結論

8.1 まとめ

本研究では，3 章及び 5 章において設計方法論を提案し，その検証を行うことで，本エージェントプラットフォームにより IoT アプリケーションを設計し実装するために有効であることを確認した．

まず，3 章では，IoT アプリケーションのエージェント指向開発方法論の検討を行った．IoT アプリケーションは 3 階層のアーキテクチャに従う．これらの各層は，機能や特性が異なる要素から構成される．したがって，エージェント指向開発方法論は，各層の特性に基づいたエージェントのモデルを必要とする．ただし，それらのエージェントは，共通のプロトコルで，協調可能であることが必要である．しかしながら，これらの条件を満足するエージェントモデルが存在しない．なぜなら，現在のユビキタス及び IoT におけるエージェント型ミドルウェアによる開発方法によれば，下層レイヤを扱うことのみに終始しており，且つ，下層レイヤの IoT デバイスはその内部に持つモノのセンシング及び制御のためのアプリケーションプログラムが強固にラッピングアプローチを用いて結びついてしまっており，切り替えることが難しい構造を成しているからである．3 章では，ラッピングアプローチに対するエージェント型アーキテクチャの構成方法としてリソースコネクタと呼ぶ構造の提案を行った．リソースコネクタは，ラッピングアプローチと違い，上述の IoT デバイス内のアプリケーションプログラムを構造上分離し，プラグインプログラムとして扱うことで再利用と切り替えを可能とする．また，リソースコネクタは扱うことが可能な対象としてデバイス・データ・プログラムである情報リソースをエージェント化することが可能であり，これにより IoT アーキテクチャにおける各層のコンポーネントをエージェント化し，エージェントプロトコルによる通信を可能とすることで，それぞれのコンポーネントをエージェントにより扱うことが可能となった．このリソースコネクタにより，各層のコンポーネントはエージェント空間において共通のプロトコルにより協調可能となった．

次に，4 章では，IoT アプリケーションのエージェント指向開発方法論に基づくプラットフォームの検討を行った．IoT アプリケーションのアーキテクチャの各階層のエージェントは，異なるインフラストラクチャの上で動作することが必要である．たとえば，クラウドのエージェントプラットフォームは，アプリケーションの論理動作を高速に処理し，クラウド内の通信プロトコルで効率的にメッセージ送信を行う．ネットワークを構成する要素を実現するエージェントは，大量のストリームを送受信したり，多数のエージェント間の安定したメッセージの総配信を行う機能を持つ必要がある．デバイスを制御するエージェントは，直接物理デバイスを制御するための仕組みを必要とする．このように，IoT アプリケーションの各階層に対応して異なる特性を持つエージェントを実装するためのプラットフォームが存在しない．4 章では，3 章で実現したエージェント指向開発方法であるリソースコネクタを，実際の開発において利用可能なプラットフォームとして実現するために，その開発方法におけるモデルを実現する為のエージェントプラットフォームであるリソースコネクタプ

ラットフォームの設計と実装を行った．リソースコネクタプラットフォームは，IoT アプリケーションの各階層に対応して異なる特性を持つエージェントをリソースコネクタの概念に基づいて実装を行うことで，物理デバイスを制御することが可能な仕組みを提供する．そして，プラットフォームの支援機構により，開発方法論を実現する為の様々な支援ツールの開発を行った．さらに，プラットフォームの実現を確認するために，複数の特性を持つ情報リソースを対象としてプラットフォームによる IoT アプリケーションの実装を行う事で検証を行った．この実装とその確認により，エージェント指向開発方法論に基づく，異なるインフラストラクチャ上で動作可能なプラットフォームの実現できた．

そして，5 章では，IoT アプリケーションの垂直連携型アーキテクチャの検討を行った．IoT アプリケーションシステムは，IoT アプリケーションの 3 層構造を参考として，クラウドアプリケーション，ネットワーク，デバイスの 3 階層のアーキテクチャと考えられる．ネットワークは，クラウドのインフラストラクチャ，インターネット，デバイスとインターネットを接続する無線ネットワークから構成される．デバイスは，カメラなどのセンサ，ディスプレイなどのアクチュエータおよび移動端末などから構成される．IoT アプリケーションの各階層は，多数の要素から構成される複雑なシステムであり，これらが連携して目的とするサービスを実現する．このような階層間の要素の連携を垂直連携と呼ぶ．IoT アプリケーションにおける垂直連携は，サービス利用者の位置や状況で要素が変化するため，的確なサービスを実現するためには，動的に垂直連携を構成することが必要であるが，このための効率的な仕組みが存在しないことが，IoT アプリケーション開発の方法論の課題である．5 章では．3 章と 4 章において提案，設計，実装を行い，開発したエージェントプラットフォームであるリソースコネクタプラットフォームを用いて垂直連携が可能な IoT アプリケーションを開発するための設計方法について述べた．本エージェントプラットフォームを用いて開発を行う場合，その対象の IoT アプリケーションはエージェント型設計を行う必要がある．エージェント型設計を行わず，IoT アプリケーションの開発を行う場合，本エージェントプラットフォームが適応できない．そのため，まず，IoT アプリケーションを設計するために，本エージェントプラットフォームが持つ特性であるリソースの概念とチャネルの概念を示し，エージェント型設計を行うために必要な項目と考え方，及び概念を示した．5 章で示した，エージェント型設計におけるチャネルの概念は，IoT アプリケーション開発における課題である動的な垂直連携を実現する為に有効であり，各階層間の要素の連携を適応的にエージェントによって構成する支援を行うことが可能である．そのため，この概念を用いた拡張をリソースコネクタプラットフォームに施すことで，エージェントプラットフォームにより，IoT アプリケーション開発方法論の課題の解決を行った．IoT に関する技術の適応対象として，オフィス支援の分野があり様々な研究が行われており，会議支援に関する研究が存在する．以降の章では，会議の一形態である遠隔地間のブレインストーミングを支援するための IoT アプリケーションに着目し，本エージェントプラットフォームを利用して開発を行う．遠隔地間のブレインストーミングを支援するためのシステムには，Edge と Cloud のどちらの要素も必要であり，また，ネットワークに接続されたアプリケーションと共に様々なデバイス（カメラ，マルチタッチディスプレイ等）が必要となる（これらは IoT デバイスと同義）．よって，IoT アプリケーションとして必要な要素を備えているため，本設計方法を用いてエージェントプラットフォームによる IoT アプリケーションが実現できるか検証を行った．

最後に，6 章及び 7 章では，上述のエージェントプラットフォームのテストベッドによる実現可能性の検証を行うために，スマートオフィスの実現に向けたテストベッドによる検討を行った．オフィス業務は，書類や資料の作成，会議などを繰り返し行い，それぞれの作業の成果を共有したりするワークフローにより実現されている．これらを支援するためには，高度に連携するマルチメディアシステムを実現することが必要である．しかしながらオフィス業務支援ツールは，個別に動作するツールの寄せ集めであり，これをワークフローの各段階に合わせて利用するためには，各ツールの調整が利用者の大きな負担になる．この負担を軽減するための IoT アプリケーションの構成法が課題である．本研究では，オフィス業務における会議支援に焦点を当て，中

でもマルチメディアの要素がより重要視される課題として，遠隔地でのブレインストーミング会議の支援を行うシステムにおける構成をテストベッドとして扱うこととした．

6 章では，5 章で示した IoT アプリケーションの垂直連携型アーキテクチャを用いて設計・実装されたエージェントプラットフォームであるリソースコネクタプラットフォームを用いて IoT アプリケーションの構成を行うことで，オフィス業務における会議支援での負担の軽減を図った．リソースコネクタプラットフォームによる IoT アプリケーションの実現により，ブレインストーミング会議における各ツールはエージェントにより動的に構成可能とされ，同期的・非同期的に連携が可能となった．また，各ツールにおけるそれぞれの要素がクラウドサーバに蓄積されることで再利用を可能とし，アノテーションにより意味づけを行う事でセマンティクスを付与し資産化（Capitalization）を可能とした．

さらに，7 章では，6 章で示されたテストベッドにおける IoT アプリケーションの提案と設計を用いて，5 章で示したマルチメディアチャネルの概念の提案を用いて，より具体化した実装を行う事で，会議支援システムとしての機能の拡張と評価を行った．機能の拡張として，同期型および非同期型のブレインストーミングの成果を表現するマルチメディアデータを遠隔拠点間で共有するためのチャネル要素を提案し，チャネル要素を利用者の要求に対応して組み合わせたマルチメディアチャネルにより，ブレインストーミングの型にあった支援を可能とした．さらに，チャネル要素間のデータをリンクするアノテーション機能により，これまで資産化したブレインストーミングの成果を再利用することが可能とした．7 章では，評価実験として，実装されたアプリケーションを利用して，日本とフランスの二つのチーム間で行われたリモートブレインストーミング実験により，目的とする支援機能が実現されていることを示すために，実装による検証と，レイテンシ等の測定による実装した全体システムの実験による検証を行った．実験の結果，チャネルの概念に基づき，同期的・非同期的なチャネルを状況や環境に応じて適応的に追加・削除することを可能とし，スケーラブルなアーキテクチャの実現が確認できた．そして，チャネル間の連携を行うことで資産化したブレインストーミングの成果の再利用が可能である事を確認した．

以上により，各章において，序論において示した課題の解決を行うことができたと考えられる．よって，本研究により，IoT アプリケーションを開発するためのエージェントプラットフォームを提案した．これに基づいて，エージェントを開発し，運用するためのプラットフォームを設計及び実装を行った．このエージェントプラットフォームを用いて，IoT アプリケーションを開発するための垂直連携型モデルを提案した．最後に，垂直連携型モデルに基づいてスマートオフィスの IoT アプリケーションであるリモートブレインストーミング支援システムの開発と評価実験を行い，提案したエージェントプラットフォームにより IoT アプリケーションが開発可能であることを示した．

IoT アプリケーションは物理空間や人の社会と密接に関係するため，環境や状況に合わせて動作条件が変化する．これまでのオブジェクト指向ではその変化に対処することが難しいが，本章により，適応的に処理を行う IoT アプリケーションの開発方法論が明確になった．

8.2　本研究の貢献

また，本研究で実現したエージェントプラットフォームを用いることで，様々な IoT アプリケーションを容易に実現可能なると考えらえる．例えば，多種多様なセンサやアクチュエータを用いることが可能でより柔軟なスマートオフィスの実現が可能となる．さらに，近年，様々な場所に IoT のインフラが存在する社会が実現しつつある．それらの IoT インフラを用いることができる社会において，本エージェントプラットフォームは，ユーザの望むサービスを提供できる可能性が高いと考えられる．

2020 年には東京オリンピックが行われる予定であるが，進化をする IT インフラにより大量の人々が集ま

るそれらのイベントにおいて多量のデータが発生し，それを用いたサービスの提供によりイベントの運営コストやイベント時の事故の軽減，効率的な人の誘導，参加者に対する様々な言語への対応等，ユーザのニーズに応じて適応的にサービスを行う必要がある．この時，大規模なシステムを一つ構成し対応することだけでは，ユーザのニーズに対して網羅的にサービスを提供することは難しい．この時，本研究において提案を行ったミドルウェアやチャネルの概念により実現されるネットワークの適応的な構成法を用いることで，インフラまでを含むサービスの適応的な構成とそれを用いたシステムを，エージェントの利用により様々なベンダによって実現することで，ユーザのニーズに合わせたサービスが提供できる環境が構築できると考えられる．

参考文献

[1] Hafiz Farooq Ahmad, Hiroki Suguri, Arshad Ali, Sarmad Malik, Muazzam Mugal, M Omair Shafiq, Amina Tariq, and Amna Basharat. Scalable fault tolerant agent grooming environment: Sage. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pp. 125–126. ACM, 2005.

[2] Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, and Moussa Ayyash. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Communications Surveys & Tutorials*, Vol. 17, No. 4, pp. 2347–2376, 2015.

[3] Sheeraz A. Alvi, Bilal Afzal, Ghalib A. Shah, Luigi Atzori, and Waqar Mahmood. Internet of multimedia things: Vision and challenges. *Ad Hoc Networks*, Vol. 33, pp. 87 – 111, 2015.

[4] Kevin Ashton. That 'internet of things' thing. *RFiD Journal*, Vol. 22, No. 7, 2011.

[5] Luigi Atzori, Antonio Iera, Giacomo Morabito, and Michele Nitti. The Social Internet of Things (SIoT) When social networks meet the Internet of Things: Concept, architecture and network characterization. *Computer Networks*, Vol. 56, No. 16, pp. 3594–3608, nov 2012.

[6] Ronald M. Baecker. *Readings in human-computer interaction : toward the year 2000.* Morgan Kaufmann Publishers, 1995.

[7] Rafael Ballagas, Michael Rohs, and Jennifer G. Sheridan. Sweep and point and shoot: Phonecam-based interactions for large public displays. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '05, pp. 1200–1203, New York, NY, USA, 2005. ACM.

[8] Jakob E Bardram and Thomas R Hansen. Context-based workplace awareness. *Computer Supported Cooperative Work (CSCW)*, Vol. 19, No. 2, pp. 105–138, 2010.

[9] Jean-Paul Barthès. Omas—a flexible multi-agent environment for cscwd. *Future Generation Computer Systems*, Vol. 27, No. 1, pp. 78–87, 2011.

[10] Jean-Paul Barthès. Improving human-agent communication using linguistic and ontological cues. *International Journal of Electronic Business*, Vol. 10, No. 3, pp. 207–231, 2013.

[11] Jean-Paul Barthès, Milton Ramos, Omar González, and Kenji Sugawara. Twa: An experimental approach for studying knowledge sharing in multicultural cooperation. In *Computer Supported Cooperative Work in Design (CSCWD), 2011 15th International Conference on*, pp. 858–861. IEEE, 2011.

[12] Ivano Bartoli, Giovanni Iacovoni, and Fabio Ubaldi. A synchronization control scheme for video-conferencing services. *Journal of multimedia*, Vol. 2, No. 4, pp. 1–9, 2007.

[13] Fabio Bellifemine, Agostino Poggi, and Giovanni Rimassa. Jade–a fipa-compliant agent framework. In *Proceedings of PAAM*, Vol. 99, p. 33. London, 1999.

[14] Yazid Benazzouz, Christophe Munilla, Ozan Gunalp, Mathieu Gallissot, and Levent Gurgen. Shar-

ing user iot devices in the cloud. In *Internet of Things (WF-IoT), 2014 IEEE World Forum on*, pp. 373–374. IEEE, 2014.

[15] Rafael H. Bordini, Jomi F Hübner, and Renata Vieira. Jason and the golden fleece of agent-oriented programming. In *Multi-Agent Programming: Languages, Platforms and Applications*, pp. 3–37. Springer, 2005.

[16] Rafael H. Bordini, Jomi Fred Hübner, and Michael Wooldridge. *Programming multi-agent systems in AgentSpeak using Jason*, Vol. 8. John Wiley & Sons, 2007.

[17] Alessio Botta, Walter de Donato, Valerio Persico, and Antonio Pescap. Integration of cloud computing and internet of things: A survey. *Future Generation Computer Systems*, Vol. 56, pp. 684 – 700, 2016.

[18] Omer Boyaci, Andrea Forte, Salman Abdul Baset, and Henning Schulzrinne. vdelay: A tool to measure capture-to-display latency and frame rate. In *Multimedia, 2009. ISM'09. 11th IEEE International Symposium on*, pp. 194–200. IEEE, 2009.

[19] Danah M. Boyd and Nicole B. Ellison. Social network sites: Denition, history, and scholarship. In *Journal of Computer-Mediated Communication*, Vol. 13(1), pp. 210–230, 2008.

[20] Andrew Clayphan, Anthony Collins, Christopher Ackad, Bob Kummerfeld, and Judy Kay. Firestorm: a brainstorming application for collaborative group work at tabletops. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces - ITS '11*, p. 162. ACM Press, 2011.

[21] Andrew Clayphan, Judy Kay, and Armin Weinberger. ScriptStorm: scripting to enhance tabletop brainstorming. *Personal and Ubiquitous Computing*, Vol. 18, No. 6, pp. 1433–1453, aug 2014.

[22] Étienne Deparis, Marie-Hélène Abel, Gaëlle Lortal, and Juliette Mattioli. Designing a system to capitalize both social and documentary resources. In *Proceedings of the 2013 IEEE 17th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pp. 581–586, Canada, June 2013.

[23] Patrick Th. Eugster, Pascal A. Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. The many faces of publish/subscribe. *ACM Computing Surveys*, Vol. 35, No. 2, pp. 114–131, jun 2003.

[24] Dave Evans. The internet of things: How the next evolution of the internet is changing everything. 2011. *URL: http://www.cisco.com/web/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf*, 2015.

[25] Florian Forster. Improving creative thinking abilities using a generic collaborative creativity support system. *Research, reflections and innovations in integrating ICT in education*, Vol. 1, pp. 539–543, 2009.

[26] Giancarlo Fortino, Antonio Guerrieri, and Wilma Russo. Agent-oriented smart objects development. In *Proceedings of the 2012 IEEE 16th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pp. 907–912, May 2012.

[27] Adele Goldberg and David Robson. *Smalltalk-80: the language and its implementation.* Addison-Wesley Longman Publishing Co., Inc., 1983.

[28] Anandasivam Gopal, Tridas Mukhopadhyay, and Mayuram S Krishnan. The role of software processes and communication in offshore software development. *Communications of the ACM*, Vol. 45, No. 4, pp. 193–200, 2002.

[29] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of

Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, Vol. 29, No. 7, pp. 1645–1660, sep 2013.

[30] Robert Hardy and Enrico Rukzio. Touch & interact: Touch-based interaction of mobile phones with displays. In *Proceedings of the 10th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '08, pp. 245–254, New York, NY, USA, 2008. ACM.

[31] Mario Hermann, Tobias Pentek, and Boris Otto. Design principles for industrie 4.0 scenarios. In *System Sciences (HICSS), 2016 49th Hawaii International Conference on*, pp. 3928–3937. IEEE, 2016.

[32] Luis Iribarne, Nicolas Padilla, Javier Criado, and Cristina Vicente-Chicote. Metamodeling the structure and interaction behavior of cooperative component-based user interfaces. *Journal of Universal Computer Science*, Vol. 18, No. 19, pp. 2669–2685, november 2012.

[33] ITU. G.114 (05/2003). *Networks*, 2003.

[34] Robert Johansen. *Groupware : computer support for business teams.* Free Press, 1988.

[35] Alistair Jones, Atman Kendira, Thierry Gidel, Claude Moulin, Dominique Lenne, and Jean-Paul Barthès. Personal assistant agents and multi-agent middleware for cscw. In *16th IEEE International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pp. 72–79, Wuhan, China, 2012.

[36] Alistair Jones, Atman Kendira, Dominique Lenne, Thierry Gidel, and Claude Moulin. The TATIN-PIC Project, A Multi-modal Collaborative Work Environment for Preliminary Design. In *15th International Conference on Computer Supported Cooperative Work in Design, CSCWD 2011*, pp. 154–161, June 8-10, 2011.

[37] Andrew Jones, Claude Moulin, Jean-Paul Barthès, Dominique Lenne, Atman Kendira, and Thierry Gidel. Personal assistant agents and multi-agent middleware for cscw. In *Computer Supported Cooperative Work in Design (CSCWD), 2012 IEEE 16th International Conference on*, pp. 72–79. IEEE, 2012.

[38] Yuki Kaeri, Yusuke Manabe, and Kenji Sugawara. A platform for prototyping an agent space interacting with real space and digital space. In *In Proceedings of the 2nd International Workshop on Smart Technologies for Energy, Information and Communication (IW-STEIC 2013)*, pp. 113–120, 2013.

[39] Yuki Kaeri, Yusuke Manabe, Kenji Sugawara, and Claude Moulin. An iot application connecting edge resources and cloud resources using agents. *International Journal of Energy, Information and Communications (IJEIC)*, Vol. 8, No. 15, pp. 1 – 14, 2017.

[40] Yuki Kaeri, Yusuke Manabe, Kenji Sugawara, and Claude Moulin. Agent-based System Architecture Supporting Remote Collaboration via an Internet of Multimedia Things Approach. *IEEE Access*, 2018.

[41] Yuki Kaeri, Kenji Sugawara, Yusuke Manabe, and Claude Moulin. Prototyping a Meeting Support System using Ubiquitous Agents. In *19th IEEE International Conference on Computer Supported Cooperative Work in Design*, pp. 18–23, May, 2015.

[42] Hak-Man Kim, Yujin Lim, and Tetsuo Kinoshita. An intelligent multiagent system for autonomous microgrid operation. *Energies*, Vol. 5, No. 9, pp. 3347–3362, 2012.

[43] Srdjan Krco, Boris Pokric, and Francois Carrez. Designing IoT architecture(s): A European per-

spective. In *2014 IEEE World Forum on Internet of Things (WF-IoT)*, pp. 79–84. IEEE, mar 2014.

[44] Yue Lu, Yong Zhao, Fernando Kuipers, and Piet Van Mieghem. Measurement study of multi-party video conferencing. In *Proceedings of the 9th IFIP TC 6 International Conference on Networking*, NETWORKING'10, pp. 96–108, Berlin, Heidelberg, 2010. Springer-Verlag.

[45] Kalle Lyytinen and Youngjin Yoo. Issue and challenges in ubiquitous computing. *Communications of the ACM*, Vol. 45, No. 12, pp. 63–96, 2002.

[46] Jennifer Marlow, Scott A Carter, Nathaniel Good, and Jung-Wei Chen. Beyond Talking Heads: Multimedia Artifact Creation, Use, and Sharing in Distributed Meetings. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing - CSCW '16*, pp. 1701–1713. ACM Press, 2016.

[47] Wojciech Mazurczyk, Maciej Karaś, Krzysztof Szczypiorski, and Artur Janicki. Youskyde: information hiding for skype video traffic. *Multimedia Tools and Applications*, Vol. 75, No. 21, pp. 13521–13540, 2016.

[48] Julien Mineraud, Oleksiy Mazhelis, Xiang Su, and Sasu Tarkoma. A gap analysis of Internet-of-Things platforms. *Computer Communications*, Vol. 89-90, pp. 5–16, sep 2016.

[49] Alok Mishra and Deepti Mishra. Cultural issues in distributed software development: A review. In Robert Meersman, Hervé Panetto, Alok Mishra, Rafael Valencia-García, António Lucas Soares, Ioana Ciuciu, Fernando Ferri, Georg Weichhart, Thomas Moser, Michele Bezzi, and Henry Chan, editors, *On the Move to Meaningful Internet Systems: OTM 2014, ISDE Workshop*, Vol. 8842 of *Lecture Notes in Computer Science*, pp. 448–456. Springer, 2014.

[50] Motoki Miura, Taro Sugihara, and Susumu Kunifuji. GKJ: Group KJ Method Support System Utilizing Digital Pens. *IEICE Transactions on Information and Systems*, Vol. E94-D, No. 3, pp. 456–464, 2011.

[51] Claude Moulin, Alistair Jones, Jean-Paul Barthès, and Dominique Lenne. Preliminary Design on Multi-touch surfaces Managed by Multi-agent Systems. *International Journal of Energy, Information and Communications*, Vol. 2, No. 4, pp. 195–210, November 2011.

[52] Claude Moulin, Yuki Kaeri, Kenji Sugawara, and Marie-Hélène Abel. Capitalization of Remote Collaborative Brainstorming Activities. *Computer Standards & Interfaces (Elsevier)*, Vol. 48, No. C, pp. 217–224, November 2016.

[53] Claude Moulin, Kenji Sugawara, Yuki Kaeri, and Shigeru Fujita. Synchronous interaction for supporting remote multilingual brainstorming. In *Computer Supported Cooperative Work in Design (CSCWD), Proceedings of the 2014 IEEE 18th International Conference on*, pp. 135–139. IEEE, 2014.

[54] Katashi Nagao, Katsuhiko Kaji, Daisuke Yamamoto, and Hironori Tomobe. Discussion mining: Annotation-based knowledge discovery from real world activities. In *Pacific-Rim Conference on Multimedia*, pp. 522–531. Springer, 2004.

[55] Hyacinth S. Nwana, Divine T. Ndumu, and Lyndon C. Lee. Zeus: An advanced tool-kit for engineering distributed multi-agent systems. In *Proceedings of PAAM*, Vol. 98, pp. 377–391, 1998.

[56] Barry Andrew Piorkowski, Richard David Evans, and James Xiaoyu Gao. Development of a face-to-face meeting capture and indexing process. In *CSCWD 2011, 15th International Conference on*

Computer Supported Cooperative Work in Design, pp. 4–8, Lausanne, Switzerland, June 2011.

[57] Stefan Poslad, Phil Buckle, and Rob Hadingham. The fipa-os agent platform: Open source for open standards. In *proceedings of the 5th international conference and exhibition on the practical application of intelligent agents and multi-agents*, Vol. 355, p. 368, 2000.

[58] Dave Raggett. The web of things: Challenges and opportunities. *Computer*, Vol. 48, No. 5, pp. 26–32, 2015.

[59] Wolfgang Raschke, Massimiliano Zilli, Johannes Loinig, Reinhold Weiss, Christian Steger, and Christian Kreiner. Patterns of software modeling. In Robert Meersman, Hervé Panetto, Alok Mishra, Rafael Valencia-García, António Lucas Soares, Ioana Ciuciu, Fernando Ferri, Georg Weichhart, Thomas Moser, Michele Bezzi, and Henry Chan, editors, *On the Move to Meaningful Internet Systems: OTM 2014, ISDE Workshop*, Vol. 8842 of *Lecture Notes in Computer Science*, pp. 428–437. Springer, 2014.

[60] Mohammad Abdur Razzaque, Marija Milojevic-Jevric, Andrei Palade, and Siobhan Clarke. Middleware for Internet of Things: A Survey. *IEEE Internet of Things Journal*, Vol. 3, No. 1, pp. 70–95, feb 2016.

[61] Stuart Russell and Peter Norvig. Artificial intelligence: A modern approach. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, Vol. 25, p. 27, 1995.

[62] Faouzi Sebbak, Aicha Mokhtari, Abdelghani Chibani, and Yacine Amirat. Context-aware ubiquitous framework services using jade-osgi integration framework. In *Machine and Web Intelligence (ICMWI), 2010 International Conference on*, pp. 48–53. IEEE, 2010.

[63] Shanzhi Chen, Hui Xu, Dake Liu, Bo Hu, and Hucheng Wang. A Vision of IoT: Applications, Challenges, and Opportunities With China Perspective. *IEEE Internet of Things Journal*, Vol. 1, No. 4, pp. 349–359, aug 2014.

[64] Norio Shiratori, Kenji Sugawara, Yusuke Manabe, Shigeru Fujita, and Basabi Chakraborty. Symbiotic computing based approach towards reducing user's burden due to information explosion. *Information and Media Technologies*, Vol. 7, No. 1, pp. 12–19, 2012.

[65] John A Stankovic. Research directions for the internet of things. *IEEE Internet of Things Journal*, Vol. 1, No. 1, pp. 3–9, 2014.

[66] Takuo Suganuma, Kenji Sugawara, Tetsuo Kinoshita, Fumio Hattori, and Norio Shiratori. Concept of symbiotic computing and its agent-based application to a ubiquitous care-support service. *International Journal of Cognitive Informatics and Natural Intelligence (IJCINI)*, Vol. 3, No. 1, pp. 34–56, 2009.

[67] Kenji Sugawara and Jean-Paul Barthès. An approach to developing an agent space to support users' activities. In *Proc. of The Fifth International Conference on Advances in Human-oriented and Personalized Mechanisms (CENTRIC), Lisbon, Portugal, IARIA*, pp. 84–90, 2012.

[68] Kenji Sugawara, Yusuke Manabe, Claude Moulin, and Jean-Paul Barthès. Design assistant agents for supporting requirement specification definition in a distributed design team. In *15th Int. Conference on Computer-Supported Cooperative Work in Design (CSCWD)*, pp. 329–334, Lausanne, Switzerland, June 8-11 2011. IEEE.

[69] Kenji Sugawara, Yusuke Manabe, Claude Moulin, and Jean-Paul Barthès. Design assistant agents for supporting requirement specification definition in a distributed design team. In *Computer*

Supported Cooperative Work in Design (CSCWD), 2011 15th International Conference on, pp. 329–334. IEEE, 2011.

[70] Kenji Sugawara, Yusuke Manabe, Norio Shiratori, Salem Ben Yaala, Claude Moulin, and Jean-Paul Barthès. Conversation-based support for requirement definition by a personal design assistant. In *Cognitive Informatics & Cognitive Computing (ICCI* CC), 2011 10th IEEE International Conference on*, pp. 262–267. IEEE, 2011.

[71] Sarah Tausch, Doris Hausen, Ismail Kosan, Andrey Raltchev, and Heinrich Hussmann. Groupgarden: supporting brainstorming through a metaphorical group mirror on table or wall. In *Proceedings of the 8th Nordic Conference on Human-Computer Interaction Fun, Fast, Foundational - NordiCHI '14*, pp. 541–550. ACM Press, 2014.

[72] Takahiro Uchiya, Takahide Maemura, Hideki Hara, and Tetsuo Kinoshita. Interactive design method of agent system for symbiotic computing. In *Cognitive Informatics, 6th IEEE International Conference on*, pp. 312–320. IEEE, 2007.

[73] Dieter Uckelmann, Mark Harrison, and Florian Michahelles. An Architectural Approach Towards the Future Internet of Things. In *Architecting the Internet of Things*, pp. 1–24. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[74] Jesús Vallecillos, Javier Criado, Luis Iribarne, and Nicolás Padilla. Dynamic mashup interfaces for information systems using widgets-as-a-service. In Robert Meersman, Hervé Panetto, Alok Mishra, Rafael Valencia-García, António Lucas Soares, Ioana Ciuciu, Fernando Ferri, Georg Weichhart, Thomas Moser, Michele Bezzi, and Henry Chan, editors, *On the Move to Meaningful Internet Systems: OTM 2014, ISDE Workshop*, Vol. 8842 of *Lecture Notes in Computer Science*, pp. 438–447. Springer, 2014.

[75] James R. Wallace, Stacey D. Scott, and Carolyn G. MacGregor. Collaborative sensemaking on a digital tabletop and personal tablets: Prioritization, comparisons, and tableaux. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pp. 3345–3354, New York, NY, USA, 2013. ACM.

[76] Mark Weiser. The computer for the 21 st century. *Scientific american*, Vol. 265, No. 3, pp. 94–105, 1991.

[77] Daniel Wigdor, Joe Fletcher, and Gerald Morrison. Designing user interfaces for multi-touch and gesture devices. CHI' 09 Extended Abstracts on Human Factors in Computing Systems, pp. 2755–2758, 2009.

[78] Chauncey Wilson. *Brainstorming and Beyond: A User-Centered Design Method.* Morgan Kaufmann, 2013.

[79] Michael Wooldridge. An introduction to multi-agent systems, department of computer science, university of liverpool, uk, 2002.

[80] Michael Wooldridge and Nicholas R Jennings. Intelligent agents: Theory and practice. *The knowledge engineering review*, Vol. 10, No. 2, pp. 115–152, 1995.

[81] Qihui Wu, Guoru Ding, Yuhua Xu, Shuo Feng, Zhiyong Du, Jinlong Wang, and Keping Long. Cognitive internet of things: a new paradigm beyond connection. *IEEE Internet of Things Journal*, Vol. 1, No. 2, pp. 129–143, 2014.

[82] Susan Yee and Kat S. Park. StudioBRIDGE. In *Proceedings of the SIGCHI conference on Human*

factors in computing systems - CHI '05, p. 551, New York, New York, USA, April 2005. ACM Press.

[83] Guoqiang Zhong, Kenichi Takahashi, Tarek Helmy, Kouichirou Takaki, Tsunenori Mine, Shigeru Kusakabe, and Makoto Amamiya. Kodama: As a distributed multi-agent system. In *Parallel and Distributed Systems: Workshops, Seventh International Conference on, 2000*, pp. 435–440. IEEE, 2000.

[84] 岡留剛, 前川卓也, 服部正嗣, 柳沢豊. センサネットワーク環境における実世界イベント検索システム. 情報処理学会論文誌, Vol. 48, No. 7, pp. 2351–2361, 2007.

[85] 屋代智之, ThomasF.LaPorta. Nomadic agent system : インフラに依存しない位置情報サービス提供システム. 情報処理学会論文誌, Vol. 46, No. 12, pp. 2952–2962, dec 2005.

[86] 加藤丈和, 松山隆司. i-energy profile: スマートタップネットワークによるエネルギーの情報化プロファイル. 電子情報通信学会論文誌 B, Vol. 94, No. 10, pp. 1232–1245, 2011.

[87] 喜連川優. 情報爆発のこれまでとこれから. 電子情報通信学会誌, Vol. 94, No. 8, pp. 662–666, 2011.

[88] 原英樹, 藤田茂, 菅原研次, 木下哲男, 白鳥則郎. Adips フレームワークのためのエージェント開発支援環境. 情報処理学会論文誌, Vol. 40, No. 11, pp. 4030–4040, 1999.

[89] 顧優輝, 真部雄介, 菅原研次. ユビキタスデバイスとデジタルリソースを論理エージェントに接続するためのエージェントプラットフォームの開発. 情報処理学会論文誌, Vol. 56, No. 1, pp. 284–294, jan 2015.

[90] 幸島明男, 和泉憲明, 車谷浩一, 中島秀之. ユビキタス計算環境におけるコンテンツ流通のためのマルチエージェントアーキテクチャ: Consorts. 人工知能学会論文誌, Vol. 19, No. 4, pp. 322–333, 2004.

[91] 高橋健一, 雨宮聡史, 鍾国強, 松野大輔, 峯恒憲, 雨宮真人. エージェント・プラットフォームを相互運用するためのメッセージ通信プロトコル. 電気学会論文誌 C (電子・情報・システム部門誌), Vol. 123, No. 8, pp. 1503–1511, 2003.

[92] 今野将, 羽鳥秀明, 吉村智志, 岩谷幸雄, 阿部亨, 木下哲男. 能動的情報資源を用いた知識型ネットワーク管理支援機構の設計と試作. 合同エージェントワークショップ&シンポジウム JAWS2004 講演論文集, pp. 165–172, 2004.

[93] 宗森純, 堀切一郎, 長澤庸二. 発想支援システム郡元の分散協調型 KJ 法実験への適用と評価. 情報処理学会論文誌, Vol. 35, No. 1, pp. 143–153, 1994.

[94] 松山隆司. 分散協調視覚-研究成果と今後の展望. 情報処理学会研究報告コンピュータビジョンとイメージメディア (CVIM), Vol. 2000, No. 33 (1999-CVIM-121), pp. 41–48, 2000.

[95] 松平正樹, 星川恭子, 瀧浩平. コンテキストアウェアネスとその応用 (金融ソリューション・サービス特集). Oki テクニカルレビュー, Vol. 75, No. 1, pp. 94–97, 2008.

[96] 川村隆浩, 田原康之, 長谷川哲夫, 大須賀昭彦, 本位田真一. Bee-gent: 移動型仲介エージェントによる既存システムの柔軟な活用を目的としたマルチエージェントフレームワーク. 電子情報通信学会論文誌 D, Vol. 82, No. 9, pp. 1165–1180, 1999.

[97] 土田貴裕, 大平茂輝, 長尾確. ゼミコンテンツの再利用に基づく研究活動支援. 情報処理学会論文誌, Vol. 51, No. 6, pp. 1357–1370, 2010.

[98] 浮田宗伯, 松山隆司. 能動視覚エージェント群による複数対象の実時間協調追跡. 情報処理学会論文誌コンピュータビジョンとイメージメディア (CVIM), Vol. 43, No. SIG11 (CVIM5), pp. 64–79, 2002.

[99] 由井薗隆也, 宗森純. 発想支援グループウェア KUSANAGI を用いた集合知型会議の検討. 情報処理学会論文誌, Vol. 53, No. 11, pp. 2635–2648, 2013.

[100] 爰川知宏, 前田裕二, 郷葉月, 伊藤淳子, 宗森純. Web ベース発想支援システム GUNGEN-SPIRAL II の複数タブレット端末による拡張. 情報処理学会論文誌, Vol. 54, No. 2, pp. 639–646, 2013.

謝辞

本研究はもちろんのこと，学部からの懇切なるご指導を賜りました菅原研次教授に謹んで深謝の意を表します．菅原先生のおかげで，様々な障害を乗り越え，博士への道をそして研究を進めることができました．また，悩み立ち止まってしまったときに暖かく手を差し伸べて下さり私を導いてくださいました．ひとえに菅原先生の導きがあったからこそ，研究者としての道をあきらめずに来られたと思っております．まだまだ至らぬ点が多い未熟者ですが，今後共，ご指導の程何卒宜しくお願いいたします．

本研究に際し，多大なるご指導，ご支援をいただきました真部雄介准教授に心より感謝の意を表します．また，真部先生から頂いた日々の意見のおかげで，研究者としての問題意識とあり方を学ぶことができ，次へ踏み出す一歩や気づきを与えて下さりました．さらなる研究の深化に向けて先生方と共に研究を進められるように，今後も努力していこうと思っております．

そして，共同研究に際し，ご指導を頂きましたコンピエーニュ工科大学の Claude Moulin 教授，Jean-Paul Barthès 教授に深く感謝いたします．

本研究は千葉工業大学情報科学研究科で行われたものであり，大学における研究生活の中で時に躓き，乗り越える際に様々な場面で手を差し伸べて頂いた諸先生方のご支援があってこそ遂行することが出来ました．厚く御礼申し上げます．

また，大変ご多忙であるにもかかわらず，数多くのご意見やご協力をいただきました菅原・真部研究室の各氏，実験にご協力いただいた各氏にも深く感謝いたします．

最後に，これまで暖かく見守り，心の拠り所になってくれていた家族に感謝いたします．

研究業績

論文誌(査読有)

1. **Kaeri, Y.**, Moulin, C., Sugawara, K., & Manabe, Y. (2018).
   Agent-based System Architecture Supporting Remote Collaboration via an Internet of Multimedia Things Approach, *IEEE ACCESS,* doi : 10.1109/ACCESS.2018.2796307.
2. **Kaeri, Y.**, Manabe, Y., Sugawara, K., & Moulin, C. (2017).
   An IoT Application Connecting Edge Resources and Cloud Resources using Agents, *International Journal of Energy, Information and Communications (IJEIC), Vol.8, Issue.1* (pp. 1-14), http://dx.doi.org/10.14257/ijeic.2017.8.1.01.
3. Moulin, C., **Kaeri, Y.**, Sugawara, K., & Abel, M.-H. (2016).
   Capitalization of remote collaborative brainstorming activities, *Computer Standards & Interfaces, 48* (pp. 217-224), http://doi.org/10.1016/j.csi.2015.11.006.
4. **顧優輝**, 真部雄介, 菅原研次. (2015).
   ユビキタスデバイスとデジタルリソースを論理エージェントに接続するためのエージェントプラットフォームの開発. *情報処理学会論文誌, 56*(1) (pp. 284-294).

国際会議(査読有)

1. **Kaeri, Y.**, Sugawara, K., Manabe, Y., & Moulin, C. (2015).
   A support system for remote brainstorming sessions. In *2015 IEEE 14th International Conference on Cognitive Informatics & Cognitive Computing (ICCI*CC)* (pp. 303-308). IEEE. http://doi.org/10.1109/ICCI-CC.2015.7259402.
2. **Kaeri, Y.**, Sugawara, K., Manabe, Y., & Moulin, C. (2015).
   Prototyping a meeting support system using ubiquitous agents. In *2015 IEEE 19th International Conference on Computer Supported Cooperative Work in Design (CSCWD)* (pp. 18-23). IEEE. http://doi.org/10.1109/CSCWD.2015.7230927.

3. Moulin, C., Sugawara, K., **Kaeri, Y.**, & Abel, M.-H. (2015).
Distributed Architecture for Supporting Collaboration (pp. 391-400). *OTM Workshops 2015*. Springer. http://doi.org/10.1007/978-3-319-26138-6_42.

4. **Kaeri, Y.**, Manabe, Y., Sugawara, K., & Moulin, C. (2014).
A Prototypical Support System for Remote Multilingual Brainstorming and its Experiment. In *International Conference on Smart Technologies for Energy, Information and Communication (IC-STEIC)*.

5. **Kaeri, Y.**, Manabe, Y., Sugawara, K., & Moulin, C. (2014).
Integration of two Agent Platforms to bridge Real Space and Digital Space. In *International Conference on Smart Technologies for Energy, Information and Communication (IC-STEIC)*.

6. Ohnuki, Y., **Kaeri, Y.**, Manabe, Y., & Sugawara, K. (2014).
Awareness of human activities by detecting state change of room equipment. In *2014 IEEE 6th International Conference on Awareness Science and Technology (iCAST)* (pp. 1-7). IEEE. http://doi.org/10.1109/ICAwST.2014.6981825.

7. Sugawara, K., **Kaeri, Y.**, Manabe, Y., Moulin, C., & Barthes, J.-P. (2014).
An application of an agent space connecting real space and digital space. In *2014 IEEE 13th International Conference on Cognitive Informatics and Cognitive Computing* (pp. 230-235). IEEE. http://doi.org/10.1109/ICCI-CC.2014.6921464.

8. Moulin, C., Sugawara, **K., Kaeri**, Y., & Fujita, S. (2014).
Synchronous interaction for supporting remote multilingual brainstorming. In *Proceedings of the 2014 IEEE 18th International Conference on Computer Supported Cooperative Work in Design (CSCWD)* (pp. 135-139). IEEE.
http://doi.org/10.1109/CSCWD.2014.6846830.

9. Moulin, C., Sugawara, **K., Kaeri**, Y., Fujita, S., & Abel, M.-H. (2014).
Collaborative Brainstorming Activity Results and Information Systems (pp. 399-407). *OTM Workshops 2015*. Springer. http://doi.org/10.1007/978-3-662-45550-0_40.

10. **Kaeri, Y.**, Manabe, Y., & Sugawara, K. (2013).
A Platform for Prototyping an Agent Space Interacting with Real Space and Digital Space. In *Proceedings of the 2nd International Workshop on Smart Technologies for Energy, Information and Communication (IW-STEIC 2013)* (pp. 113-120).

11. Wouters, L., **Kaeri, Y.**, & Sugawara, K. (2013).
Multi-domain multi-lingual collaborative design. In *Proceedings of the 2013 IEEE 17th International Conference on Computer Supported Cooperative Work in Design (CSCWD)* (pp. 269-274). IEEE. http://doi.org/10.1109/CSCWD.2013.6580974.

12. **Kaeri, Y.**, Manabe, Y., Moulin, C., Sugawara, K., & Barthes, J.-P. A. (2010).
A Cloud-Based Support System for Offshore Software Development. In *2010 International Conference on Broadband, Wireless Computing, Communication and Applications* (pp. 315-319). IEEE. http://doi.org/10.1109/BWCCA.2010.89.

研究会

1. **願優輝**, 真部雄介, & 菅原研次. (2010).
オフショア開発支援のためのクラウド型 Web サービスの開発(ユビキタス・クラウドへの知的アプローチ,「Web サービス・クラウド」及び一般). *電子情報通信学会技術研究報告. AI, 人工知能と知識処理, 110*(172), 13-18.

2. 石橋智幸, **願優輝**, 脇屋達, 真部雄介, & 菅原研次. (2009).
ソーシャルブックマークを用いた Web ページ推薦システムの開発(「社会的インタラクションにおける知」及び一般). *電子情報通信学会技術研究報告. AI, 人工知能と知識処理, 109*(211), 7-12.

3. 石橋智幸, 脇屋達, **願優輝**, 真部雄介, 今野将, & 菅原研次. (2008).
ソーシャルブックマークを用いた Web ページ推薦方式の提案(「Web インテリジェンス」及び一般). *電子情報通信学会技術研究報告. AI, 人工知能と知識処理, 108*(119), 69-74.