Vol.2020-MBL-94 No.49
Vol.2020-UBI-65 No.49
2020/3/3

Synerex: 超スマート社会を支える需給交換プラットフォームの設計コンセプトと機能

河口信夫 [1,2]　米澤拓郎 [2]　廣井 慧 [2]

概要：「超スマート社会」の実現には多様な情報システムの高度な連携が必須である．しかし，現在の情報システム間の連携手法には，設計時に想定された機能やデータ構造を前提としており，新しい機能やデータの導入といった「変化」には，システム全体の改修が前提となっている．社会基盤として利用されるシステムは，全体を入れ替えることなく継続的に稼働し，部分的な更新による機能向上が図られるべきであり，システムの一部が「変化」することを前提とすべきである．我々は，需給交換の概念を導入し，全体のシステム改修を行うことなく，一部の更新により機能を追加して継続的なイノベーションを可能にする需給交換プラットフォーム Synerex を構築している．本論文では，その設計コンセプトと機能について説明する．

キーワード：機器間連携手法，連携プロトコル，需給交換，サービスプラットフォーム，システム間連携

Synerex: Design Concept and Function of Demand-Supply Exchange Platform for Supporting Super Smart Society(Society5.0)

NOBUO KAWAGUCHI[†1,2]　TAKURO YONEZAWA[†2]
KEI HIROI[†2]

Abstract: Society 5.0 could be enabled through the integration of various information systems. However, current method of integration for information systems are based on fixed data structures at the time of design. For the platform system, flexible and robust inter-system integration method is essential for long-term sustainable system. So, it is required to support continuous change of subsystems. In this paper, we propose and describe an new demand-supply exchange system named "Synerex", which enables sustainable innovation for Society 5.0.

Keywords: System Integration Protocol, Demand-Supply Exchange, Service Platform, System of Systems

1. はじめに

情報技術の発展により，計算能力や通信能力の大幅な向上が実現し，これまでは容易ではなかった大量のリソースを用いた機械学習などの大規模な計算が可能になった．一方，「超スマート社会」は，そのような単一目的の情報システムではなく，多様な情報システムが連携した分散システム，いわゆる System of Systems[1]による実現が必須である．しかし，現在の情報システム間の連携手法は，設計時に想定された機能やデータ構造を前提としており，新しい機能やデータの導入といった「変化」には，システム全体の改修が前提となっているという大きな課題がある．社会基盤として利用されるシステムは，全体を入れ替えることなく継続的に稼働し，部分的な更新による機能向上が図られるべきであり，システムの一部が「変化」することを前提とすべきである．本研究では，様々な変化を前提とし，全体のシステム改修を行うことなく，一部の更新により機能を追加して継続的なイノベーションを可能にする新しい情報システム間連携手法の構築を目指す．

本稿では，まず，既存のシステム間連携や通信手法についてその課題を挙げる．さらに，System of Systems において必要なシステム間連携機能について検討する．また超スマート社会において求められる機能についても検討する．

2. システム間連携における課題

大規模なシステムを効率的かつロバスト・スケーラブルに実現するため，多様な情報システム間連携手法が活用されている．現在の情報システム間連携は，API(Application Program Interface)や, RPC(Remote Procedure Call)といった概念が主に使われ，単一の計算機内の連携から，異なる情報システム間での連携まで，様々なスケールで利用されている．これらの仕様として CORBA[2]や Java-RMI[3]といった

1 名古屋大学 未来社会創造機構
Institutes of Innovations for Future Society, Nagoya University.

2 名古屋大学工学研究科
Graduate School of Engineering, Nagoya University

Vol.2020-MBL-94 No.49
Vol.2020-UBI-65 No.49
2020/3/3

仕様が利用されてきたが，2000 年代からはコンポーネント技術である EJB や Servlet が広く使われ，REST[4]などのシンプルな手法が主流になり，それ以降は広く普及するような新たな基盤技術は登場していない．

2006 年にはクラウドの雄である AWS(Amazon Web Services)が登場し，仮想化技術の進展とともに，大規模分散システムの構築が容易になり，2013 年にはコンテナ技術の Docker[5]が発表された．コンテナ技術により，ライブラリや実行環境までを含めたアプリケーションを安全に提供できる枠組みが実現され，FaaS(サーバレスの関数サービス)と共に一段と大規模サービスが構築されるようになりつつある．現在は，コンテナを用いてマイクロサービス間連携を行うのが大規模システムにおける定番の開発手法となってきている．

一方，複数システム間の連携は，Publish/Subscribe[6]系の仕組みでの連携が利用されていることが多い．IoT やセンサデバイスなどを対象にした軽量プロトコルである MQTT[7]や，Java におけるメッセージング基盤である JMS，より高度な機能を導入した AMQP[8]などが存在する．これらのメッセージングプロトコルは多機能であり，多様な目的で利用できるが，一方，特定の目的を達成するためには，定まっていない部分が多く，アプリケーション毎に個別にメッセージ内容を策定する必要がある．その結果，システム間の結合が強くなってしまう，という問題が生じている．

また，超スマート社会では，サイバー世界上だけでなく実世界上の様々なリソースも管理する必要がある．実世界リソースには，物理的・空間的な制約が存在しているため，電子リソースのようにコピーすることができない．また，リソースがある目的で利用されている間は，他の目的には利用できないといった制約が存在する．このようなリソースを選択する，といったコンセプトも，従来のシステム間連携では考慮されてこなかった．結果的に，特定のサーバがリソースを管理する仕組みとなり，クライアントとの強い結合が生じてしまう結果となっている．

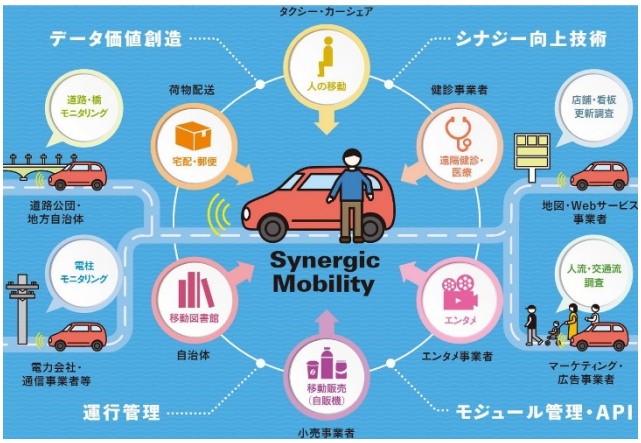

図 1　Synergic Mobility のコンセプト図

Figure 1　The concept of Synergic Mobility.

3.　変化への対応の必要性

我々は，2 節で挙げたような課題の存在を実際のアプリケーションの構築を通じて実感してきた．具体的には，JST の未来社会創造事業で実施した「Synergic Mobility の創出」というプロジェクトである．このプロジェクトでは，自動運転技術が普及した先の社会を想定し，様々なサービスが自動運転を通じて融合する超高効率な社会を構想した．例えば，自動運転車両が普及すれば，各車両が搭載するレーザスキャナ(LiDAR)や車載カメラにより，安価に大量の実世界データを集めることが可能になる．これらを用いて周囲の物体認識を行えば，道路やトンネル・電柱などの社会インフラの確認，店舗や看板・道路標識などの実世界更新情報の取得，人や車両の流動状況や混雑度などが獲得できる．さらに，自動運転車両は人に加え，郵便や貨物・飲食品の運搬・販売も可能であり，様々なサービスが相乗りできる．また，自動運転車両は複数台の運用管理が重要であり，需要に応じた最適な配車技術が求められる．

こういった背景のもと，このプロジェクトでは，多様なサービスを融合し，走れば走るほどデータが集まり，価値を生み出すモビリティ基盤「Synergic Mobility」を構想した．コンセプトを図 1 に示す．このような複数のサービスを単一の車両が行うためには，多数のシステム間連携が必要となる．しかし，REST などによる従来型のサーバ・クライアントモデルや，MQTT などに代表されるシンプルな Publish/Subscribe モデルを使うだけでは，課題がある．これについて簡単に宿泊業界を例にとって説明したい．図 2 は，現在の宿泊業界のシステム間連携の模式図である．宿泊したいエンドユーザと，部屋を提供するホテル事業者の間に，メタサイト・予約サイト・サイトコントローラ・クラウド PMS などの多数のシステムが入って無駄が生じている．一方，民泊においては，AirBnB が独占しており，他事業者の入る余地がない．これは，従来型のアーキテクチャが生んだ弊害であり，このような無駄や停滞を生まない新しいアーキテクチャが必要である．

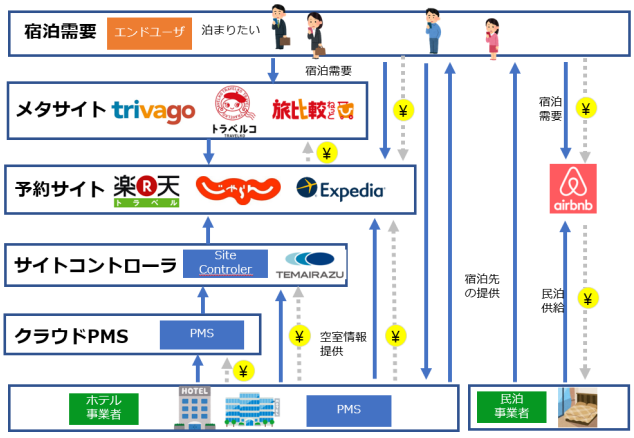

図 2　宿泊業界におけるシステム間連携

Figure 2　Status of accommodation system integration.

Vol.2020-MBL-94 No.49
Vol.2020-UBI-65 No.49
2020/3/3

4. Synerex の設計コンセプト

我々は，現在の宿泊業界のような混沌とした状況を生まないためにも，システム間連携に新しい枠組みを導入する必要があると考えた．宿泊業界で，最も大きな課題は，予約サイトが保持している部屋の情報が，サイトによって異なるため，希望者が複数のサイトを見に行く必要がある点である．これは，ホテルの部屋の情報や，宿泊者の需要の状況が全体で共有されていないことに要因がある．

この状況を改善するためには，宿泊に関する需要や供給を交換する場が必要であると考え，これをシナジー効果を生み出すための交換所として当初は「シナジック・エクスチェンジ」と呼ぶことにした．長いので省略して，現在では Synerex と呼んでいる．Synerex では，ユーザの宿泊需要や，ホテル側の部屋の供給といった情報をそれぞ需要や供給，といった形で標準化し，その情報を交換する仕組みを提供している．需要と供給という形式に整理したため，宿泊事業だけでなく，当初目的としていたモビリティサービスや，様々な事業・サービスで利用できる．

図 3 に宿泊事業において，Synerex を導入した場合のシステム間連携の様子を示す．Synerex が宿泊需要・部屋の供給をすべて交換するため，エンドユーザやホテル事業者にとっては，より良い環境になることが期待できる．一方，ユーザ向けプロバイダ（従来の宿泊サイト）にとっては，それぞれの工夫によって他事業者との差別化を行う必要がある．また，新規参入もしやすいため，サービス向上が期待できる．証券市場においても，株や債券の市場は共通であっても，複数の証券会社が様々なサービスを競いあっているように，本システムのような枠組みでも，同様の仕組みが実現できると期待している．

また，モビリティサービスにおいても，図 4 のように，様々な事業者を巻き込んだサービスの実現が可能になる．一方，複数の事業者の間で，同じリソース（ここでは宿泊やモビリティ）を共有することになるため，事業者間でのプロトコルを定める必要がある．

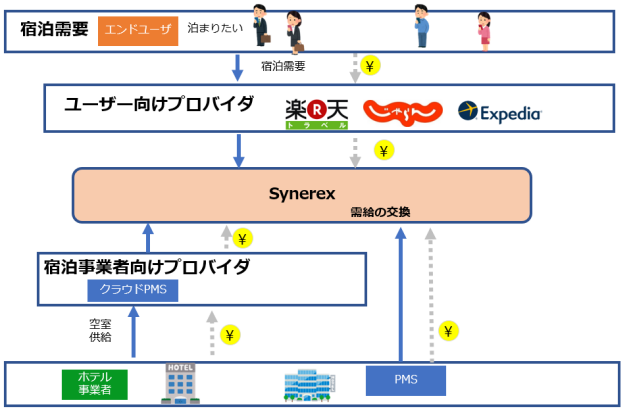

図 3　宿泊事業における Synerex の導入例

Figure 3　Sample deployment of Synerex in hotel industry.

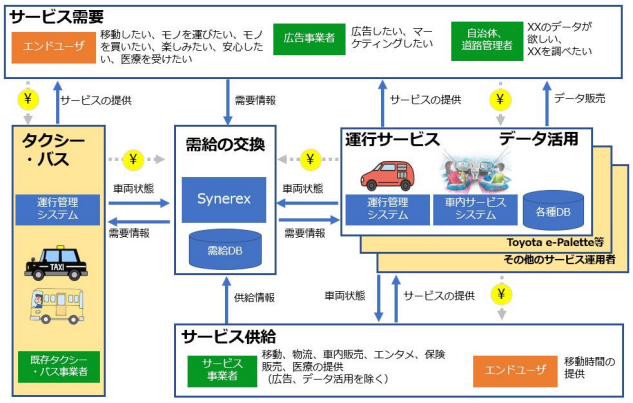

図 4　モビリティサービスにおける Synerex の導入

Figure 4　Synerex deployment in mobility services

4.1 需要と要求によるサービスの抽象化

Synerex では，従来のサーバ・クライアントモデルでは，エンドユーザやサービサーの間で直接やりとりされていた情報を，需要と供給という形で分離し，エンドユーザが直接サービスと結びつかない形とした点に特徴がある．これにより，同じ需要であっても，複数のサービサーが異なる供給を提供できる．つまり，ある種のサービスの抽象化が実現できていると言える．また，ユーザの需要を受け取るプロバイダと，ユーザに供給を提供するプロバイダを Synerex を挟んで分割した形になっている．これにより，ユーザは直接サービス指定せず，自分の需要だけをプロバイダに登録する形になる．以下では，具体例を通じて Synerex が実現する需給交換の例を示す．

サービスシナリオ例：

「地方の高齢者が駅前まで行って買い物をしたい」という需要があるとする．利用者は，自分の地域のプロバイダ（地域プロバイダ）を通じて，目的地と行きたい時刻を移動需要として登録する．タクシー，ワンマイルモビリティ（近距離専用），コミュニティバス，経路サービスといったサービスプロバイダが存在する場合，往路には次の選択肢が提案される．

- タクシーで直接駅に行く（1000 円）
- バス停までタクシー（500 円）＋コミュニティバス（無償）で駅に行く
- ワンマイルモビリティ（300 円）＋コミュニティバス（無償）で駅に置く

経路サービスプロバイダは，経路の組合せサービスを提供しており，異なる交通機関の組合せを提案する．さらに，荷物配送プロバイダにより，復路では買い物した荷物だけ別便で送る（＋300 円），といったオプションも可能である．利用者は，これらの選択肢からその時の気分や予算・状況に応じてサービスを選択できる．図 5 にシナリオを実装するプロバイダの実装図，図 6 にシーケンス図を示す．

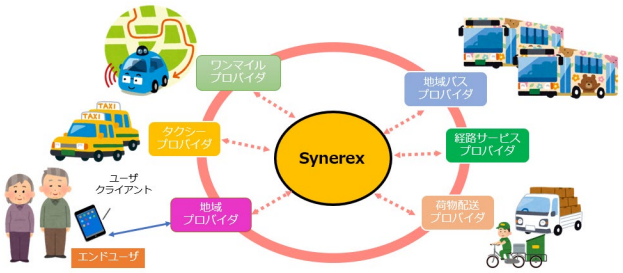

図 5　サービスシナリオ例の実装

Figure 5　Implementation of Service Scenario.

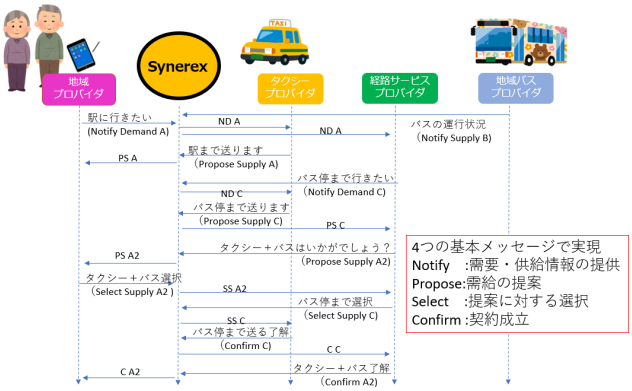

図 6　タクシーとバスの組合せのシーケンス例

Figure 6　Example combination sequence of Taxi and Bus.

Synerex では，図 6 に示すように，4 つの基本メッセージを用いて機器間連携を行う．

・Notify : 自分の需要や供給を他のプロバイダに通知する．

・Propose: 他のプロバイダの需要や供給に対し，自身が提供可能な供給や需要を手案する．

・Select:　提案された需給を選択したことを通知する．

・Confirm: 選択されたことを確認する．

なお，サービス選択がなされた後は，プロバイダ間に Mbus と呼ばれる 1 対 1 のメッセージ通信路が構成され，この間でサービスに関わる情報が交換される．上記のシーケンス図も一見，複雑に見えるが，実際には Notify→Propose→Select→Confirm の 4 ステップが複数のプロバイダで行われているだけであり，双方向のサービスが Synerex を通じて実現できている点に特徴がある．このシーケンスの場合，自分では移動サービスを実装していない経路プロバイダが，タクシーとバスの組み合わせを提案し，それによってユーザがより安い交通手段を選択できている点に特徴がある．なお，現在の Synerex では，費用の支払いについては考慮されておらず，サービス利用時にどのようにコストや費用の分担をするのかといった点については，機器間連携とは独立して考慮する必要がある．

この仕組みを用いれば，後から新しいプロバイダが異なるサービス提供を行った際にも，これまでつながっていたプロバイダを変更する必要が無い．

また，移動サービスを受けたいという需要が，具体的にどのような移動サービスを受けるか，という実装と分離されているため，様々な提案を受けることが可能になっている．例えば，自動運転によるワンマイルプロバイダや，荷物配送プロバイダ，場合によっては，シェア自転車プロバイダが，この Synerex に参加しても，ユーザ側の需要を変更しなくても，様々なサービスの組み合わせの提案を受けることが可能になる．つまり，Synerex の利用により，利用者インタフェース（地域プロバイダ）と移動サービス間の結合が弱まり，「変化」に強いシステムができているとも言える．

4.2　Synerex による複数事業者間連携の容易化

事業者間で連携を行う場合，通常は事業者毎に異なるシステムを用いているため，その連携のためには，相互の API を参照するような枠組みが必要となり，相互接続するためには，個別に連携のプロトコルの協議が必要であった．この協議にはコストがかかり，さらに，プロバイダ毎に通信手順が異なると，その度に構築コストも必要になるという課題がある（図 7）．一方，Synerex では，需給の形で通信プロトコルを統一しているため，図 8 に示すような 3 階層のサービス階層モデルが実現できる．これにより，事業者が増えても他の事業者の変更が不要になり，容易にシステム間連携ができる．また，プロトコル間の変換を行う枠組みも Synerex に追加できるため，異なるプロトコルであっても相互接続が容易に実現できる．

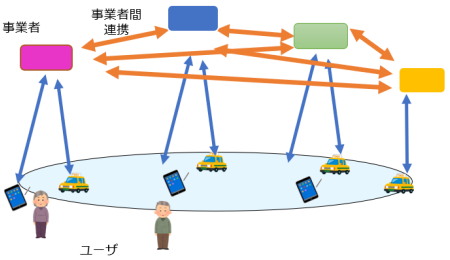

図 7　複数事業者間のシステム連携の課題

Figure 7　Issue on the multi-vendor system integration.

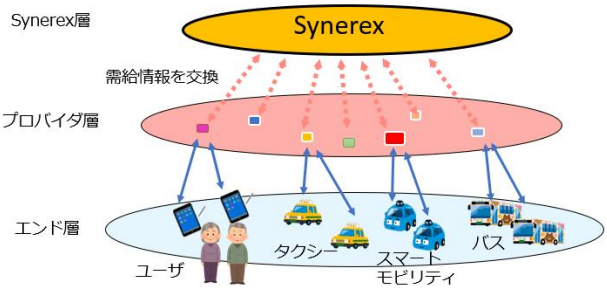

図 8　Synerex におけるサービス階層モデル

Figure 8　Service layer model of Synerex.

Vol.2020-MBL-94 No.49
Vol.2020-UBI-65 No.49
2020/3/3

4.3 Synerex 間接続による柔軟性と分散化

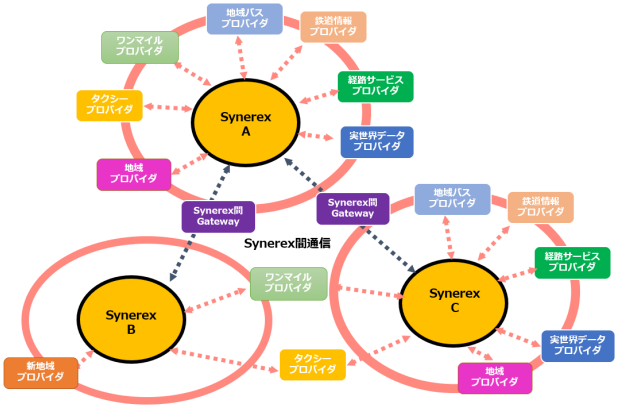

図 9　Gateway による Synerex 間接続

Figure 9　Inter Synerex connection using Gateway.

Synerex では，様々な需給情報が Synerex-Server(以下では，需給交換を担当するソフトウェアコンポーネントを，混乱避けるため Synerex-Server と呼ぶ）に集まるため，そのスケーラビリティが問題となる．我々はこの問題に対象するために，複数の Synerex-Server 間を Gateway を通じて接続し，連携できる枠組み(図 9)を構築した．また，プロバイダも特定の Synerex-Server に縛られることなく，複数の Synerex-Server に接続することが可能である．また，この Gateway にセキュリティやプライバシ分離の機能を導入することにより，様々なトラスト[9]への対応も可能になる．

5. 現時点での Synerex の機能

我々は Synerex の開発を 2018 年の開発当初から Github 上で完全オープンソースで進めてきた．2018 年 12 月に実施した実証実験では，synerex_alpha[10]を用い，コミュニティバスとタクシーとの連携を実証した．これらのコンセプトは[11]にて議論されている．一方，この実証を通じていくつかの課題も新たに認識された．特に，プロトコルがモノシリックであったため，一部の変更が全体へ影響を与えることが問題となった．そこで，プロトコル構成を変更し，2019 年 9 月からは，synerex_beta[12] としての開発を開始した．Synerex Beta 版は，Alpha 版と異なり複数の git submodule リポジトリから構成されている．これにより，特定のドメイン向けのプロトコルを開発しても，全体に大きな影響を与えることが無くなり，開発の独立性が向上した．以下では，現時点での主要機能について説明する．

5.1 Node Server

Synerex では，各プロバイダや Synerex-Server に個別に Node-ID が振られる．これをプロバイダ間でユニークにするため，Node Server を導入している．Node Server は，複数の Synerex-Server を管理することが可能であり，各プロバイダに適切な接続先の Synerex-Server の情報を通知する．さらに，各プロバイダは接続状態を保持するため Keep-Alive メッセージを定期的に Node Server に送る必要がある．これにより，Node Server, Synerex-Server, 各プロバイダが相互にモニタリングするロバストなシステムが構築できている．現在の Synerex では，これらのどれか 1 つが問題を生じて shutdown しても再起動により回復できる仕組みを持っている．さらに，Keep-Alive メッセージ が来なくなったプロバイダを自動的に Synerex-Server から切断する機能も有している．また，Synerex-Server を再起動したい場合，接続しているプロバイダ側に問題が生じないように行う必要がある．そこで，現時点での Synerex では，Node-Server から各プロバイダに，別の Synerex-Server に接続を変更するような依頼を行う機能を有している．これにより，サーバの負荷分散や，サーバの更新，ハードウェアの更新などの作業が，プロバイダ間のデータ欠落無しに行うことが可能になる．

5.2 Gateway

Synerex-Server 間を特定のチャネルで接続する機能として Gateway が導入された．これにより，複数の Synerex-Server 間でクラスタのような動作を実現できる．現時点では，メッセージはマルチホップする仕組みはなく，Gateway は 1 回通過するだけの制約を加えている．これは環状に接続した場合に，メッセージがループすることが無いようにするためである．

5.3 Forward Provider

Gateway は，Synerex-Server 間の双方向のデータのやり取りを実現するが，時には 1 方向だけで十分な場合もある．さらに，Notify などの特定のメッセージのみを対象としたい場合がある．Forward Provider は，異なる Node Server 間を接続可能な Provider で，特定のチャネルの Notify をフォワードできる機能を有している．これにより，簡単にパブリッシュドメインの拡張が可能になっている．

5.4 RPA Provider

Synerex 上で会議室予約システム間の需給交換を行うシステムをデモできるシステムとして構築されている．

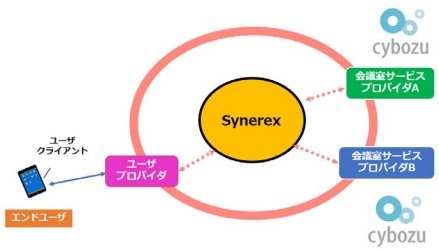

図 10　RPA Provider による会議サービス需給交換デモ

Figure 10　Meeting facility booking demo system using RPA.

5.5 HarmoVIS Provider

河口研究室が中心となって開発してきた 3 次元可視化ライブラリ Harmoware-VIS[13]と Synerex 上でのデータ連携を用いて，様々な 3 次元可視化機能を導入したプロバイダを構築している．このプロバイダを用いることにより，実世界データの可視化や，人流・交通流シミュレータの分析・活用が可能になる．

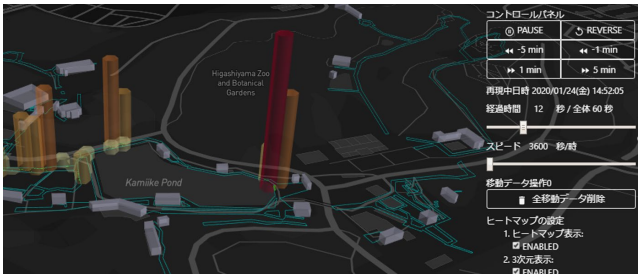

図 11 HarmVIS Provider による空間情報可視化例
Figure 11 Visualization Example of HarmVIS Provider.

6. おわりに

本稿では，我々が開発している需給交換サービスプラットフォーム Synerex の設計コンセプトと，現時点での機能について説明した．需給交換という枠組みにより，変化に対応可能で，継続的イノベーションを実現可能なシステム連携手法が実現できている．今後は，Synerex をベースに様々な社会システムの実装を進め，実社会で長期に利用され，超スマート社会に資するシステムを目指して開発を推進していきたい．

謝辞 本研究の一部は，JST MIRAI, JST OPERA (JPMJOP1612)，総務省 SCOPE，NICT 委託研究により支援されています．

参考文献

[1] Alex Gorod, Brian Sauser, John Boardman, "System-of-Systems Engineering Management: A Review of Modern History and a Path Forward", IEEE Systems Journal, 2008.
[2] Ron Ben-Tatan, "CORBA: A Guide to Common Object Request Broker Architecture", McGraw-Hill, 1995.
[3] William Grosso, "Java RMI: Designing & Building Distributed Applications (JAVA SERIES)", O'REILLY, 2001.
[4] Roy Thomas Fielding, "Architectural Styles and the Design of Network-based Software Architectures" (PhD Thesis), UC Irvine, Information & Computer Science, 2000.
[5] Dirk Merkel, "Docker: lightweight linux containers for consistent development and deployment", Linux Journal, 2014.
[6] Patrick, E.,et.al, "The Many Faces of Publish/Subscribe", ACM Computing Surveys, 2003.
[7] A Banks, R Gupta, MQTT Version 3.1.1, OASIS Standard, 2014.
[8] O'Hara, J.,"Toward a commodity enterprise middleware" , ACM Queue. 5 (4): 48-55 (2007).
[9] 平野流，廣井慧，米澤拓郎，河口信夫, "異なるサービス提供者間におけるトラストモデル とその実装に関する研究", マルチメディア，分散協調とモバイルシンポジウム 2019 論文集(DICOMO2019)(2019).
[10] https://github.com/synerex/synerex_alpha, 2018
[11] 河口信夫, "自動運転社会における Synergic Mobility の創出", 電子情報通信学会総合大会, BP-3-1, pp.1-2(2018).
[12] https://github.com/synerex/synerex_beta, 2020.
[13] https://github.com/harmoware/Harmoware-VIS, 2020