生産管理のための疎結合アーキテクチャの研究

○竹林康太　浦口智貴　Chang Shuang　出口弘（東京工業大学）

A Study on Loosely-Coupled Architecture Model for Production Control

* K.Takebayashi, T.Uraguchi, Chang Shuang and H.Deguchi (Tokyo Institute of Technology)

Abstract - IoT technology is undergoing rapid development in recent years, especially the ones for enterprises. Although the industry is pursuing research on IoT technology in manufacturing actively, including Industry 4.0 in Europe, how to utilize IoT in particular factories is still remained unclear. Different from the top down approach, such as Industry 4.0, a bottom-up IoT architecture with a concrete structure, "DESMDCAT cycle", as a framework of production management was proposed, which enables onsite "Kaizen". In this work, we conduct a comparative study of loosely coupled architecture for production management based on the DESMDCAT cycle. In addition, we apply it to a concrete application by developing a process monitoring and production management system to visualize the factory production.

キーワード:　生産管理，IoT，マイクロサービス

1　研究背景

2015 年に話題を集めた IoT(Internet of Things)は，2016 年に入った今も各企業で盛んに研究が進められている．昨今においては，様々な企業が独自の IoT プラットフォームを提唱しており，産業，農業といった様々な業種への導入が推し進められている状況である．とりわけ IoT プラットフォームに用いられる技術として従来から用いられるクラウド技術があるが，日本におけるクラウド事業では様々なボトルネックが発生しているのが現状である．

本研究では，出口研究室で提唱される概念”DSEMDCAT サイクル”を基にした，疎結合の製造業向け IoT 情報システムアーキテクチャの設計・実装を行い，食品加工工場という実際の工場モデルを用いた導入検証を行い，従来のクラウド技術による導入との比較を行う．

1.1 IoT/IoE の普及

IoT とは Internet of Things の略語であり，しばしば「モノのインターネット」と訳される．人，機械，センサー，ソフトウェアエージェントなど様々なモノをインターネットに接続し，それらのモノが互いに情報を相互通信し合う仕組みのことをモノのインターネットとしている．モノの範囲をヒトに広げたものを，とりわけ IoE(Internet of Everything)と呼ぶ場合がある [1)]．

IoT というキーワードが世間に広まった背景は様々ある．その 1 つとして挙げられるのが，情報化社会のインフラを支える技術が進化してきたことである．以下の図 1 は，情報化社会インフラの基礎要素としてインターネットとモバイル端末，そして今後普及される IoT を用いて表したものである．

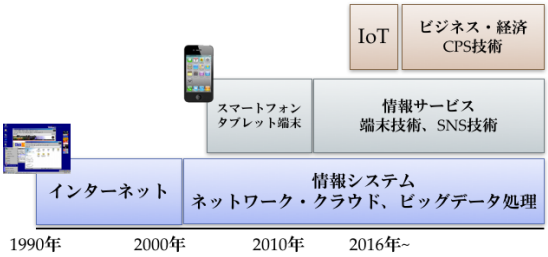

図 1 情報インフラの進化

その他の要因として挙げられるのが，通信機器やセンサーの単価自体が減少したことである．以下の図 2 は，総務省が平成 27 年度の情報通信白書で発表したグラフであり，2004 年から 2020 年までのセンサー単価の移り変わりを示している [2)]．

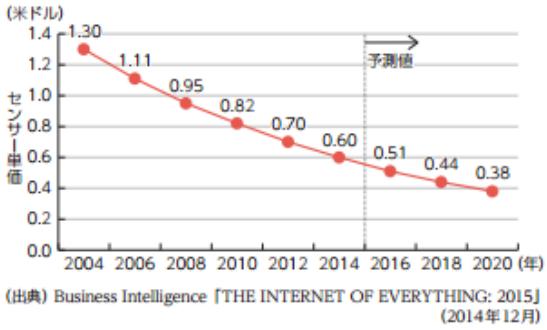

図 2 センサー単価の変遷

また Gartner は，2015 年から 2020 年にかけて，各種目別におけるネットワークに繋がる製品の数が急増するものと見込んでいる．以下の図 3 は Gartner が 2015 年に発表したものであり，一般消費者向け製品は 4.6 倍，産業分野の製品は 5.1 倍，自動車分野においては

9.4 倍もの製品が, 今後五年間において増える見込みであるとしている [3].

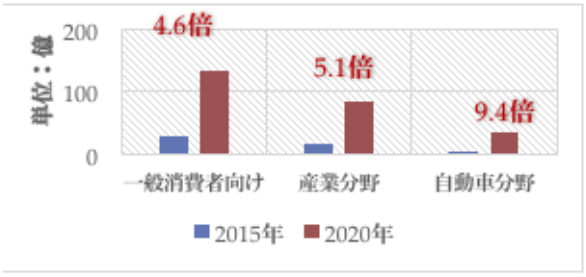

図 3 2015 年から 2020 年にかけてのネットワークに繋がる台数の増加

Cisco は，ネットワークに接続されたデバイスは 2020 年までに 500 億台に到達すると見込んでおり [4], クラウド型プラットフォームなどに代表される従来のような集中管理型(1 対多型のモデル)のアーキテクチャでは, このような膨大な数の通信プロセスは処理しきれないことが危惧される.

このような背景の中で，2015 年をピークに様々な企業から独自の IoT プラットフォームの提唱が盛んに行われた.

1.2 我が国における製造業向け IoT プラットフォームの普及動向

日本国内におけるクラウド型プラットフォームの例として，富士通株式会社の「FUJITSU Cloud Service IoT Platform」，株式会社日立製作所の「Lumada」が挙げられる．FUJITSU Cloud Service IoT では，島根工場における IoT 技術導入を行い，製造コストの 30%カットを実現している Lumada では，日立製作所の技術基盤を活かした幅広いソリューションを展開している.

また, Cisco が提唱する「フォグコンピューティング」技術を用いた IoT プラットフォームとして，ファナック株式会社の FIELD SYSTEM が挙げられる．フォグコンピューティングとは, データセンターで大部分の処理を集中管理的に行うクラウドコンピューティングに対し, 工場内のイントラネットに属するデバイス間の通信に焦点を当てたことが特徴である.

これらの IoT プラットフォームは，専ら中央管理サーバにデータが集められるため, データ分析などといったAI プラットフォームへの活用という側面のほうが大きいものとなっている．生産量の多い大企業であれば，これらの IoT プラットフォームから得られる分析データは多い．しかしながら，工場や設備の規模が大きくない中小企業にとって，これらの IoT プラットフォームを用いたとしても, 元となるデータ量が少ないため有用な手がかりが得られるとは限らない.

製造の現場では, 工場ごとにそれぞれ異なる悩みを抱えており, その課題を解決できるのは中央に集められたデータを分析して得られるものとは限らない. 従来の中央管理のアプローチとは異なる, 新たな手法による IoT システムの構築手法を模索する必要がある.

2 先行研究

本章では, 本研究手法のもととなる二つの開発および管理手法を紹介する. 一つはマイクロサービスアーキテクチャと呼ばれるもので，AWS IoT[5]などのクラウドサービス向けのアーキテクチャである.

二つ目は出口が提唱する DSEMDCAT と呼ばれるもので，生産ライフサイクル管理手法の PDCA サイクルを基に IoT 技術の導入を前提としたアーキテクチャである.

2.1 マイクロサービスアーキテクチャ

マイクロサービスとは，James Lewis によって提唱された開発手法である. マイクロサービスは，従来のようなモノリシック(一体型)のシステムではなく，最小単位から成るサービスの集合としてシステムを組み上げていく手法である [6]. ボブ・ファミリアによると，マイクロサービスの特性として，「自立性」と「分離性」が挙げられる [7][8]. 自立性は「独立して存在する，または存在することが可能. その他の部分から独立して応答する，反応する，または開発すること」と述べており，分離性は「他の部分から切り離され，別の場所や別の時間に発生すること」と述べている. マイクロサービスを取り入れることで，より多くの機能を，より短い時間で，より少ないリソースで提供することができる.

ボブ・ファミリアはまた，マイクロサービスアーキテクチャの一例として, マイクロサービスとクラウド技術を組み合わせたアーキテクチャモデルを以下の図 4 のように定義している.

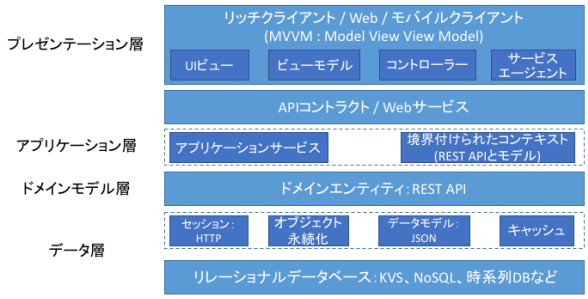

図 4 ボブ・ファミリアによるクラウドサービス向けアーキテクチャモデル

このアーキテクチャモデルは, 複数のレイヤから構成される階層モデルと呼ばれるものを使って示している. 階層は, 下から順にデータ層, ドメインモデル層, アプリケーション層, プレゼンテーション層となる. アプリケーション層は, ビジネスケイパビリティを定義するサービスとコントラクトを提供する. ドメインモデル層は, データ層で定義されたビジネスモデルに関わる操作のインメモリキャッシュ, 永続化, トランザクションの機能をカプセル化する役割を担う.

Bob familiar が提唱するクラウドサービス向けマイクロサービスアーキテクチャは, 従来のモノリシックな開発手法に替わるモデルではあるものの, 用途が情報サービスに限定されてしまい, 現実世界のヒトやモノとのコミュニケーションをカバーすることができない.

2.2 DSEMDCAT サイクル

DSEMDCAT サイクルとは, IoT 時代における生産ライフサイクル管理手法として出口が提唱した概念である. 出口研究室は, DSEMDCAT サイクル分析に関する技術の確立を目的とした, IoT カイゼン会と呼ばれるコンソーシアムを運営し, 中小企業の支援を行っている [9].

DSEMDCAT サイクルは, PDCA サイクルを基に考案された概念である. PDCA サイクルとは, Plan(計画), Do(実行), Check(評価), Act(改善)の頭文字を取ったもので, 品質の維持・向上及び継続的な業務改善活動を推進するマネジメント手法として, 多くの企業で採用されているものである [10]. しかし, PDCA サイクルは具体的な数値指標を決めるのが難しいこと, レポート管理の難しさなどによる様々な問題を抱えている. DSEMDCAT サイクルでは, 従来の PDCA サイクルに IoT の技術を適応することにより, リアルタイムによる具体的な数値を基にした生産計画や生産管理を継続していくことができる.

DSEMDCAT は, それぞれ Design(生産計画), Scheduling(スケジューリング), Execution and Monitoring(生産活動の可視化), Data Collection(生産実績のデータ化), Analysis(分析) and Traceability(証拠保全) の頭文字をとったものである. DSEMDCAT サイクルは生産活動における, プランニング, スケジューリング, モニタリング, 生産実績の収集, 改善活動という一連の流れを, IoT 技術の使用を前提としてカバーする. 以下の図 5 は, DSEMDCAT サイクルの概要を表したものである.

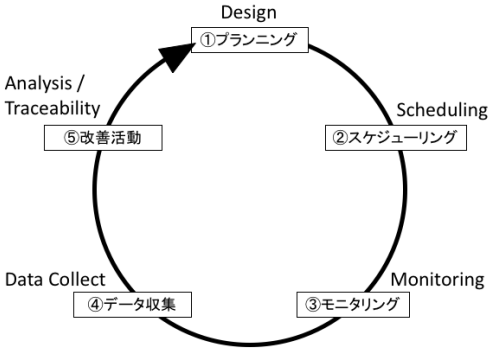

図 5 DSEMDCAT の概念図

生産計画であるプランニングとスケジューリングを行うのが Design および Scheduling フェーズである. Design フェーズでは, 生産プロセスをタスクと呼ばれる順序付けられた集まりとして定義する. タスクとは, ヒト一人ひとりに与えられた仕事のうちの一定の範囲のことをいう. Scheduling フェーズでは, Design フェーズにおいて設計したプロジェクトタイプに基いて, 原料の調達計画と設備や人員の割当の計画を立てる.

生産活動の記録, 見える化を行うのが Execution and Monitoring および Data Collection フェーズである. Execution and Monitoring フェーズでは, タスクの開始と終了の際にデータ取得を行い, それぞれのタスクの状態を一元的に可視化する. Data Collection フェーズでは, タスクの開始と終了を単位とし, 製品やサービスの生産に関する機械の稼働状況や生産報告の情報をタスク単位で収集保存する.

生産活動の改善, 製品のトラブル解決を行うのが Analysis および Traceability フェーズである. Analysis フェーズでは, Data Collection フェーズで収集したデータを様々な関数を用いて解析を行う. これにより, 生産活動の最適化などが可能となる. 最後に, Traceability フェーズでは, 製品に問題が生じた際, 原因究明のために生産時のデータにアクセスする. ここでは, 生産データを改竄防止のために暗号化し, サプライブロックチェーン技術を用いてアーカイブとして保存する.

DSEMDCAT サイクルはコンセプトがあるものの, 各フェーズの詳細な実装は未だ定まっていない. また, 工場における導入事例も限られており, その正当性を評価できていないという問題がある.

3 本研究の位置づけ

本研究では, DSEMDCAT において生産活動の記録と見える化を行う Execution Monitoring フェーズと Data Collection フェーズの実装を検討する. 更に, それらのアーキテクチャを踏まえた上で, 生産管理のための疎結合アーキテクチャの設計と提案を行う.

また, 情報システムのエンドポイントをネットワークエッジとすることで, 先行研究であるクラウド技術を用いた SaaS 型システムアーキテクチャとの差別化を図る.

さらに, 本研究のアーキテクチャについて, 食品加工工場をモデルとして実際のシステムを製作および導入し, 評価を行う.

4 疎結合アーキテクチャの設計

ここからは, 2 章で説明した DSEMDCAT サイクルとマイクロサービスを基にした製造業向け IoT システムアーキテクチャの設計を行う. まずは, 本研究手法である疎結合モデルについて先行事例であるマイクロサービスとの比較を行い, 次いで DSEMDCAT サイクルにおける Execution Monitoring フェーズと Data Collect フェーズに対する情報システムアーキテクチャのための実装を行う.

4.1 疎結合モデルの定義

疎結合モデルとは, タスクと呼ばれるある一定の範囲の仕事を与えられたヒトやモノ, ソフトウェアエージェントを構成要素とする集合のことである.

疎結合モデルは先行事例であるマイクロサービスとよく似ている概念であるが, マイクロサービスが専らソフトウェアエージェントのみを構成要素とするのに対して, 疎結合モデルでは現実世界のヒトやモノ自体もモデルの構成要素に含める. このような現実世界を対象とした IoT プラットフォームとして, 出口研究室で開発が進められているリアルワールド OS が挙げられる [11].

疎結合モデルを利用することで, 従来の情報システム開発でよく見られたトップダウン型の開発モデルではなく, ボトムアップ型の開発手法をとることが可能となる. これにより, SIer など第三者指導のもとパッケージ製品を都度カスタマイズして納品していた従来手法と比べ, 汎用性や低コスト性を活かして時間や経費などのコストを大幅に減らすことが期待される.

4.2 Execution and Monitoring アーキテクチャ

本節では, DSEMDCAT サイクルにおける Execution and Monitoring フェーズの詳細設計を行う.

Execution and Monitoring フェーズでは, 作業者は対話可能なクライアントデバイスを介して「データ記入システム」にアクセスし, 自身のタスクの開始時間及び終了時間を記入する. 管理者は, 作業者によって更新されたタスクの状態を「アンドンシステム」によって見ることができる.

以上のシステムおよびユーザの関係図を以下の図 6 に示す.

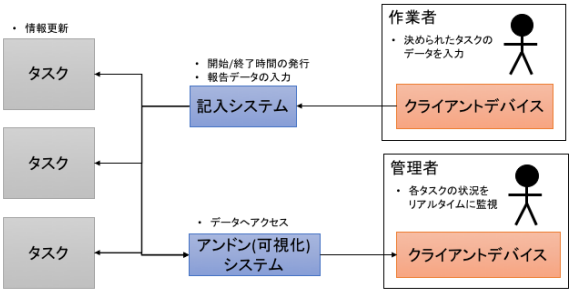

図 6 Execution Monitoring フェーズの関係図

Execution Monitoring フェーズでは, 各システム同士の通信は MQTT プロトコルを用いた Publish/Subscribe により成り立つ. MQTT は, TCP/IP ネットワークで利用できる通信プロトコルの一つで, 多数の主体の間で短いメッセージを頻繁に送受信する用途に向いた軽量なプロトコルである. MQTT では,Publish/Subscribe モデルと呼ばれる,一対多による非同期通信を行うための仕組みがある.MQTT メッセージングモデルを図 7 に示す.

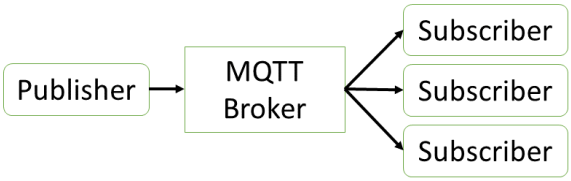

図 7 MQTT メッセージングモデル

メッセージの送り側を”Publisher(配信者)”,受け取り側を”Subscriber(購読者)”, メッセージの仲介サーバを”Broker”と呼ぶ.メッセージには Topic と呼ばれるタグがセットとなっており,受け取るメッセージの選別を行うことが出来るワイルドカードを使用することでブロードキャストも可能となる.従来の TCP/IP 通信と大きく違い,情報の送り先の IP アドレスや同期の必要性がないため,Publisher と Subscriber のプログラムはお互いに依存することなく書くことができる.

MQTT の問題点として,キューイングバッファを持たないということが挙げられる.この問題に対しては,一般的には外側にバッファ機能を持たせることで解決を図る.本研究では, 次節で述べる Data Collect アーキテクチャにおいて「データ保存システム」がその機能を担うことになる.

Pub/Sub メッセージを仲介するためのサーバ(従来の TCP/IP 通信におけるホストサーバにあたる)をブローカーと呼ぶ. 後述する工場モデルに対する導入システムでは, Mosquitto と呼ばれるオープンソースのソフトウェアを用いる.

MQTT を利用したアプリケーションの一例として, 著者は最小単位から成るモジュールにより組み上げら

れた「Go!Go!Duck!」と呼ばれる自律移動可能なデバイスを開発している [12)13)].

4.3 Data Collection アーキテクチャ

引き続き，本節では DSEMDCAT サイクルにおける Data Collection フェーズの詳細設計を行う．

Data Collection フェーズでは，管理者は「データ保存システム」により生成された，生産が完了したタスクの製造実績データにアクセスし，生産状況を分析する．このシステムにおける関係図を以下の図 8 に示す．

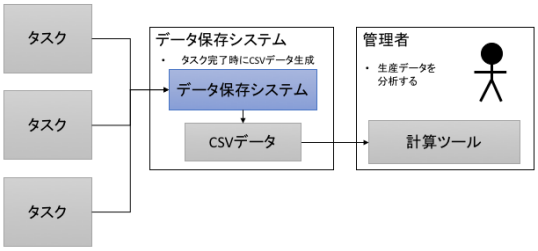

図 8 Data Collection フェーズの関係図

データ保存システムは，完了状態となったタスクの要素を CSV ファイルとして順次書き出されていく．ここでは CSV ファイルの構成について，タスクの要素をそれぞれ arg1,arg2,arg3,とおいたとき，

$$< arg1, arg2, arg3, \ldots argn >$$

と定義することができる．

4.4 製造業向け IoT アーキテクチャ

本節では，4.2 節および 4.3 節で述べた DSEMDCAT の各アーキテクチャを踏まえた上で，製造業向け IoT アーキテクチャの設計を行う．ベースとなるアーキテクチャとして，2 章で紹介したマイクロサービスのアーキテクチャの階層モデルを参考にすることとする．

2 章のアーキテクチャの例では，各階層を下から順に「データ層」「ドメインモデル層」「アプリケーション層」「プレゼンテーション層」として定義している．

本研究と先行事例の大きな違いとして，対象とする開発領域が挙げられる．先行事例は開発対象領域を情報システムのみに定めており，クラウドをベースとした SaaS 型モデルを採用している．また，マイクロサービス間のメッセージングモデルとして REST over HTTP を採用している．

本研究では，開発対象領域を情報システムから実世界タスクの疎結合モデルに拡大する．そのため，モノや機械との接点であるネットワークエッジという，データ層よりも低い階層のレイヤを考慮する必要がある．また，IoT のセンシングを行うにあたっては，一対一型であり集中管理型の REST over HTTP プロトコルよりも，多対多型通信が可能な MQTT を通信プロトコルとして採用するほうが好ましい．以下の図 9 は，REST over HTTP と MQTT の特徴を比較したものになる．

以上の検討により，製造業向け IoT のアーキテクチャモデルは以下の図 9 のようになる．

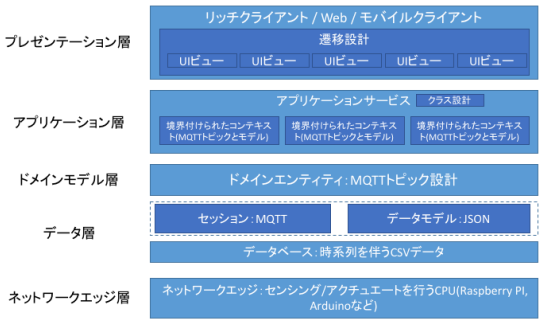

図 9 DSEMDCAT モデルを考慮した製造業向け IoT アーキテクチャモデル

プレゼンテーション層では，4.2 節におけるデータ記入システムより受信した MQTT メッセージに対してユーザが PC やモバイル，Web ブラウザなどの端末からアクセスおよび表示可能なユーザインタフェース(以下，UI とする)の画面設計を行う．たとえば，Processing ライブラリで書かれた Java プログラムや，Swift で書かれた iOS アプリなどが挙げられる．

プレゼンテーション層の UI は大きく分けて Publish 側と Subscribe 側の 2 つにすることができる．Publisher 側のクライアントはデータ記入システムに当該し，ユーザは送信するトピックとメッセージの仕組みを意識することなくキーワードとパラメータのセットのみを情報として与えられることが望ましい．Subscriber 側のクライアントは 4.2 節におけるアンドンシステムに当該し，ユーザは送信されたトピックを常に最新の状態で見られるようになることが望ましい．

5 評価方法

本章では，4 章で説明した本研究のモデルである製造業向け IoT アーキテクチャの評価を行う．評価方法としては，本研究とモデルとなる食品加工工場にて実際に導入が可能かどうかを現地にて確かめる．

まず，対象となる工場をモデル化し，導入先担当者とのヒアリングを通したシステム用件をまとめる．次にモデルとシステム用件を基に開発したシステムについて解説をし，最後に導入過程と結果について述べる．

5.1 工場モデルの概要

本研究のモデルとなる工場は，静岡県にある「静パック有限会社」という食品加工工場である[14]．この工場では，茶葉を原料として充填梱包とパッキングの 2 大工程を通した上で，製品として出荷することとなっている．この 2 大工程を，それぞれ「梱包工程」「パッキング工程」と呼ぶ．

梱包工程で用いる加工機械は全部で 40 台あり，それぞれ 1 部屋に 1 台ずつ導入されている．また，パッキング工程で用いる機械は全部で 10 台ある．ここでは，機械 1 台を 1 つのタスクとして定義し，1 つのタスクにつき以下の表 1 のような属性を付与する．

表 1 タスクに与えられた属性

名称	説明
工場 ID	工場固有の ID を示す
タスク ID	本タスク固有の ID を示す
状態	タスク(機械)の現在の状態を示す．値は，未着手，仕掛中，トラブル発生中，完了の 4 つ
ロット番号	加工した製品のロット番号を示す
原料 / 製品	使用する原料/製品の種類を示す
作業者名	作業者の名前を示す
開始時間	本タスクの状態が開始となった時間を示す
終了時間	本タスクの状態が終了となった状態を示す
レポート	加工後の付加情報を示す

以上をまとめた，本工場のモデルを以下の図 10 に示す．

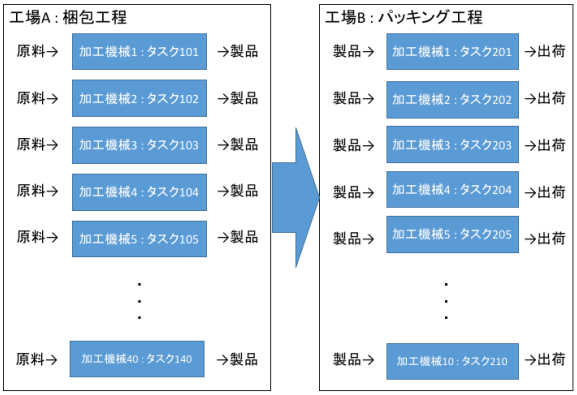

図 10 本工場のモデル

本タスクの状態は，最初は必ず「未着手」の状態となっており，一つのタスクを終えたら「完了」状態にならなければならない．本タスクモデルにおける状態遷移図を以下の図 11 として表す．

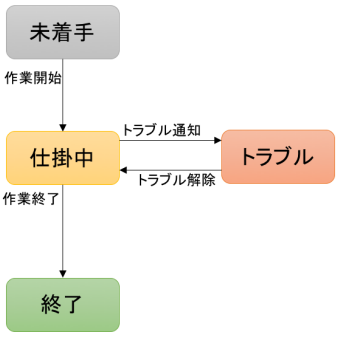

図 11 タスクの状態遷移図

5.2 導入目的

5.2.1 導入背景

この工場では従来，各タスクにおける加工実績を紙に記入していた．各ロットの生産が終了する度に記入を行い，1 日の作業が終了すると責任者のもとへ回収されるようになっている．

ここでの問題点は，リアルタイムでの各タスクの製造活動状況が把握できないことである．すなわち，どのタスクが手空きの状態か分からないため，スケジューリングをする上でも最適化ができないことが大きな問題となっていた．

そこで本研究の IoT アーキテクチャモデルをこの工場に適応し，DSEMDCAT ベースによる工場ラインの見える化を目的とし，本システムを導入する．

以下の図 12 は，存在するタスクの単位量について，システム導入前と導入後で比較したものである．図中の丸は，左側がシステム導入前のタスク，右側がシステム導入後のタスクを示し，青色は管理者が行うタスクを，橙色は作業者が行うタスクをそれぞれ表している．

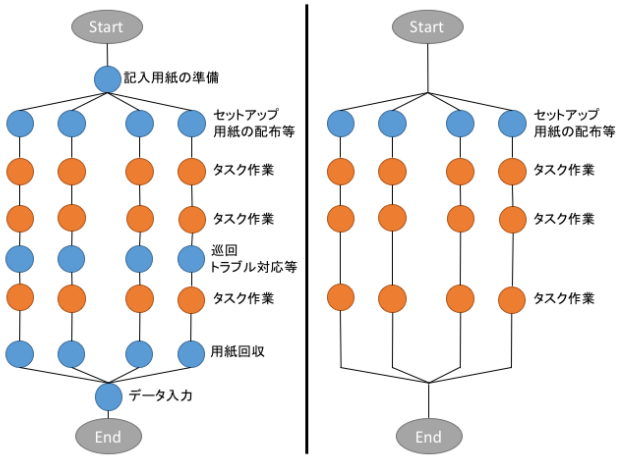

図 12 システム導入前後の比較

導入前後を比較すると，作業者が担うタスクの量が大幅に減っていることが分かる．また，作業者が行うタスクについても，開始/終了時間の自動記入などにより時間短縮が見込まれる．

本システムの導入により，DSEMDCAT サイクルにおける Execution Monitoring, Data Collection をカバーすることができる．更に，収集データを管理者が分析にすることにより，Analysis フェーズもカバーすることが可能となる．

5.2.2 導入前ヒアリング

また，本研究モデルの導入に当たって，導入先である静パック有限会社の担当者とヒアリングを行った．このヒアリングの内容を「導入前ヒアリング」として以下の表 2 にまとめて記す．

表 2 導入前ヒアリング

ヒアリング内容	対策
衛生面のため，密封容器に入れたものでないと作業場に持ち込めない．	密封袋に入れた状態でも使用可能なタブレット端末を用いる．
後工程(梱包工程)の従業員が前工程(充填工程)の状態をリアルタイムに見えるようにしたい．	IoT システムを用いて「見える化」が行えるようにする．
作業部屋ごとの生産状況を把握できるようにしたい．	作業部屋(1 タスク)ごとの生産状況を CSV データとして保存するシステムを導入する．
トラブル発生状況について，発生頻度・発生時間などを把握できるようにしたい．	各タスクのトラブル発生状態を CSV データとして保存するシステムを導入する．

「見える化」を対策として行う理由として，従業員の手待ち時間による機会ロスを減らせることが理由としてあげられる．例えば前工程である充填工程で遅延や機械停止などの異常が起きた際，その情報が後工程である梱包工程に伝えられれば，手待ち時間発生の回避行動をとることができる．

以下の図 13 は，横軸を一日の業務の時間経過とし，前工程･後工程における従業員の行動内容の関係を色分けで示したものである．画面上部は本システムを導入する前の状態であり，画面下部は本システムにより回避行動をとった場合の理想的な状態を表したものである．

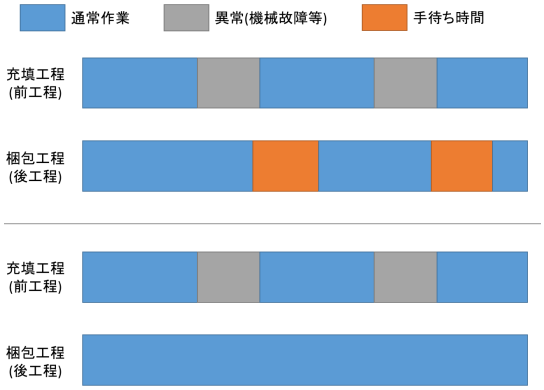

図 13 本システム導入による機会ロスの削減

5.3 導入システムの概要

本節では，0 節および 0 節で述べた工場モデルの用件および導入条件をもとにして設計･開発を行った導入システムについて，工場ボタン，アンドンシステム，報告データ作成システムについて説明する．

5.3.1 MQTT トピック設計

従来の TCP/IP 通信では，通信の送信先アドレスの判別方法として主に IP アドレスを利用していた．MQTT では，IP アドレスのかわりにトピックを用いることで多対多型の通信を実現している．

MQTT トピックは階層構造を成しており，階層ごとに意味づけをしてトピックの共通化を図る必要がある．MQTT トピックの設計方法はまだ確立されてはいないが，本研究ではタスクの属性に合わせて以下のように定義する．

$$工場ID/タスクID/Command$$

また，MQTT ではメッセージを送信することができるが，このメッセージの種類はトピックに対応付ける必要がある．次の表 3 および表 4 は，本システムにおける MQTT トピックとメッセージの組み合わせ，およびメッセージタイプの例を表したものである．

表 3 MQTT トピックとメッセージの組み合わせ

Command	メッセージタイプ	タスク属性との対応
start	Time	開始時間
end	Time	終了時間
error	Time	対応なし
recover	Time	対応なし
update	JSON	ロット番号，原料/種類，作業者名
report	JSON	レポート

cancel	String	対応なし

表 4 メッセージタイプの例

メッセージタイプ	メッセージの例
start	2016-12-01 10:00:00
end	2016-12-01 17:00:00
error	2016-12-01 14:00:00
recover	2016-12-01 15:00:00
update	{ロット番号:"ABC-001",原料:"茶葉A",作業者名:"東工太郎"}
report	{生産個数:100,軽量:3,過量:2,異物:1}
cancel	deleteAll

たとえば，タスクの開始を記録する際は，トピックは F001/T101/start, メッセージは 2016/12/01 10:00:00 となる．

5.3.2 工場ボタン

本システムでは，4.2 節における「データ記入システム」に相当するアプリケーションとして,「工場ボタン」を開発した．このアプリケーションは iPad 向けに作成されたもので，作業者は各々のタスクに関する開始/終了時間, 生産実績の記入に本アプリケーションを用いることとなる．以下の図 14 に，工場ボタンの状態遷移図を示す．

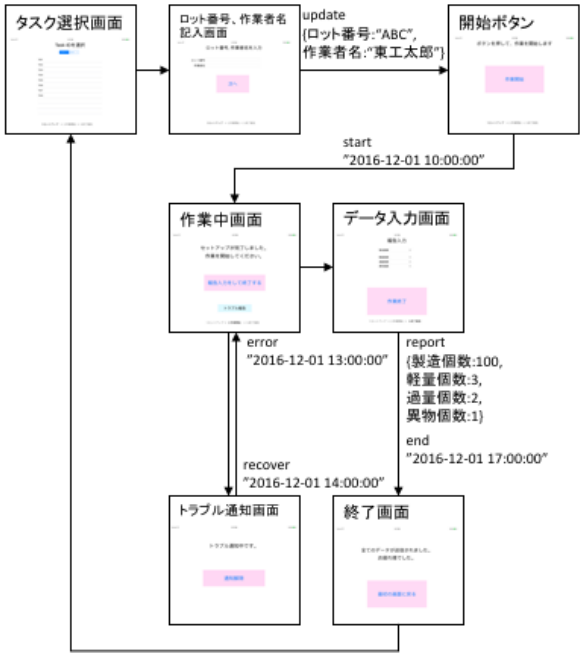

図 14 工場ボタンの状態遷移図

画面遷移の大きな流れとしては, 最初にセットアップを行い，次に作業時間の記録を行う．最後に，タスクが完了したことを知らせるために終了ボタンを押し, 加工実績を記入する．

セットアップ作業では，最初に「タスク選択画面」が表示され，次いで「ロット番号，作業者名記入画面」が表示される. 作業者はここでロット番号と作業者名をタブレット端末のキーボード入力機能を用いて入力する．次の「開始ボタン画面」への遷移の際，MQTT トピックを送信しタスクの状態を更新する．

作業時間の記録作業は，開始ボタン画面で作業者が Start ボタンを押したタイミングで開始される．作業中は本アプリケーションで行う操作は基本的にはないが, タスクにトラブルが発生した場合, 作業者はトラブル発生状態を通知することができる．

タスクが終了すると, 作業者は報告記入ボタンを押すことで「データ入力画面」に遷移し，加工実績をタブレット端末のキーボードを用いて記入する．記入後，タスク終了ボタンをおすことで終了時間が自動的に記録される．

5.3.3 アンドンシステム

4.2 節ではまた，各タスクの状態を一元的に管理するための「アンドンシステム」とよばれるシステムが存在していたが，こちらは Processing と呼ばれる Java のライブラリ及び開発環境を用いて開発されたアプリケーションとして実装する．管理者は，タスクに関する状態について, 本アプリケーションを通してリアルタイムに知ることができる．

以下の図 15 に，タスク状態可視化システムの UI を図として示す．

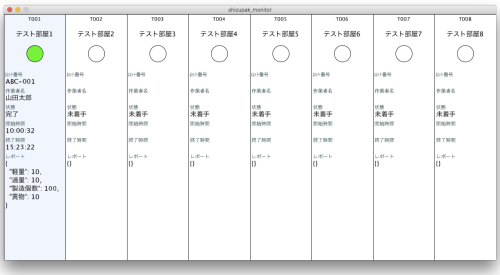

図 15 タスク状態可視化システムの UI

このアプリケーションでは，最大 8 つのタスクのリアルタイムの状態を見ることができる. タスクとして表示される情報は，上から順にタスク番号，タスク名称，タスクの状態(色で表示)，ロット番号，作業者名，タスクの状態(文字で表示)，開始時間，終了時間，報告データである．タスクの状態は，白は未着手を，黄色は着手中を，緑は完了を, 赤はトラブル発生を, それぞれ表す．

管理者は, このアプリケーションを見ることで各タス

クの現在の状況を，作業部屋から離れた環境下でも知ることができる．本アプリケーションは，必要に応じて複数台立ち上げることもでき，それぞれ異なるタスクの状態を出力することができる．

5.3.4 報告データ作成システム

本システムでは，4.3 節における「データ保存システム」について，node.js による CSV 作成アプリケーションにより実装する．管理者は，本アプリケーションにより生成された CSV ファイルを用いて加工実績の分析を行う．

以下に，本アプリケーションが出力する CSV データの一例を表 5 として示す．このデータは，タスクが完了した順に並んでおり，各タスクの要素が対応するセルに出力される．

5.3.5 システム配置図

以下の図 16 は，本システムの配置図を表したものである．

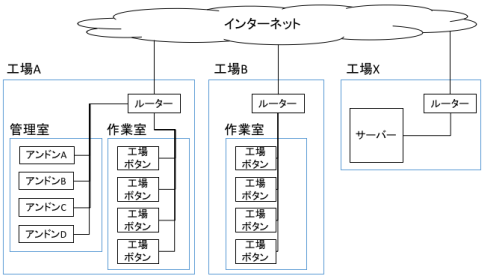

図 16 システム配置図

ここでは，充填工程を持つ工場を工場 A，パッキング工程を持つ工場を工場 B とし，ブローカーなどを設置するためのサーバを持つ工場を工場 X として定義する．各工場のアプリケーションはルータによりローカルネットとして繋がっており，各工場間の通信はインターネットを介して行われる．

5.4 本システムの導入

本節では，5.3 節で説明したシステムを食品加工工場で実際に導入した際の導入方法及び結果について述べる．

本システムを導入するにあたって，アタッシュケースを用いた持ち運び可能な一式となるキットを作成した．キットは以下の図 17 のようになっており，それぞれの構成部品の機能をまとめたものを表 6 に記す．

図 17 当システムの持ち運び用キット

表 6 持ち運び用キットの構成部品

部品名	説明
Raspberry Pi	タスク状態可視化システムに使用
Raspberry Pi	MQTT ブローカーとして使用
LCD モニタ	タスク状態可視化システムの UI を表示するために使用
キーボード・マウス	RaspberryPi の入力デバイスとして使用

当キットを導入先である静パック有限会社に持ち込み，接続及び導入テストを行った．テスト項目と結果を次の表 7 に記す．また，導入時の様子を図 18 に示す．

表 5 CSV データの例

工場 ID	タスク ID	ロット番号	作業者名	開始時間	終了時間	レポート
F001	T101	ABC-001	東工太郎	2016-12-01 10:00:00	2016-12-01 14:00:00	省略
F001	T102	DEF-001	東工花子	2016-12-01 11:00:00	2016-12-01 16:00:00	省略
F001	T101	ABC-002	東工太郎	2016-12-01 15:00:00	2016-12-01 17:00:00	省略

表 7 テスト項目と結果

テスト項目	説明	結果
接続テスト	現地環境の LAN で MQTT サーバが正常に動作することの確認.	動作した
動作テスト	作業室で工場ボタンを起動し，通常業務に合わせてタスク状態の更新や製造結果の送信を行うテスト．管理室でアンドンシステムを起動し，リアルタイムにタスク状態が反映されることの確認.	確認できた

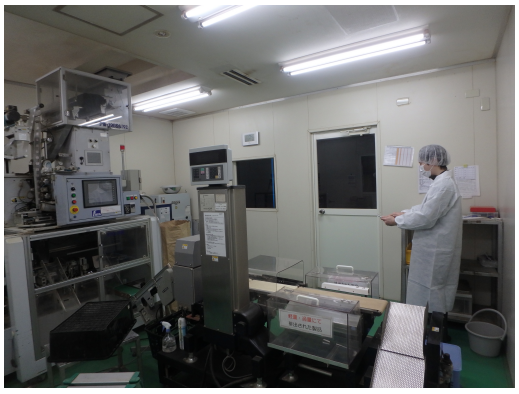

図 18 導入の様子

本システムの導入により，表 2 で挙げられていた導入前課題をすべてクリアできることができた．また，評価のため，導入後に再び静パック有限会社の担当者とのヒアリングを行った．ヒアリング内容を「導入後ヒアリング」としてまとめたものが以下の表 8 である.

表 8 導入後ヒアリング

ヒアリング内容	今後の対策案
従業員に前工程が遅延していることを知らせるシステムがほしい.	「遅延」状態を新たに設け，工場ボタンに遅延通知画面を追加する.
ロット番号をキーボードではなくバーコード読み取りで入力できるようにしたい.	タブレット端末のカメラデバイスより読み取りと自動入力ができるようにする.

遅延状態を本システムに導入した場合，状態遷移図は以下の図 19 のように変化する.

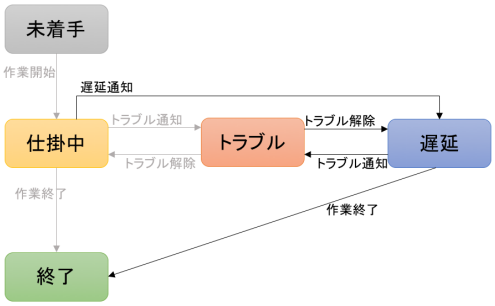

図 19 遅延状態を加えた状態遷移図

6 考察

本システムのベースとなる MQTT を用いたプログラムを用いることで，極めて短時間で IoT システムを導入できることがわかった.

以下の表 9 は，従来手法である Cisco によるフォグコンピューティングアーキテクチャを用いた導入手法と本手法を比べたものである.

表 9 IoT サービスの導入形態ごとの速度比較

導入形態	導入方法	導入速度	導入コスト	柔軟性
疎結合システム(本手法)	導入 Kit を導入先 LAN に接続する．あらゆる機械や作業場に導入可能	導入 Kit を持ち込むだけ	約 50 万円 (10 部屋の場合)	ネットワーク環境さえあれば適応可能
フォグコンピューティング	パッケージ製品などからデプロイを行う. IoT デバイスと工作機械の通信はパッケージ製品が採る規格に合わせなければならない	対応製品への買い替えが必要	工作機械の数によって，100 万〜	専用の工作機械 / ルータが必要

フォグコンピューティングは本番環境へのデプロイが素早く行えるが，IoT システムの場合，エンドポイントに当たる IoT デバイスの導入には工作機械との相性問題などに別途時間を要することになる.

本手法は，それぞれの工場の現場ごとの課題点を解決し，カイゼンしていくことのできる組み換え自由な IoT システムを提供することができる.

7　今後の展望

本研究では，製造業向け IoT システムアーキテクチャの設計と，そのモデルを用いたシステムの導入検証を行った．

本モデルは DSEMDCAT における一部分をカバーしたものである．今後の検討課題として，生産プロジェクトのプランニングやタスクスケジューリングの機能をアーキテクチャに取り入れていく必要がある．また，代数的実物簿記として記述可能な ADDL に対応することで，原価計算によるコスト改善を行うことが望まれる．

将来は，出口研究室で開発されている IoT アーキテクチャ「リアルワールド OS」により，タスクの管理・実行制御がスムーズに行え，誰でも簡単にプロジェクトプログラミンが可能なシステムの設計・開発を行う予定である．

8　謝辞

研究を進めるにあたって，多くの有益な助言を下さった出口弘教授，私の研究を常にサポートしてくださった助教の唱爽先生，本研究の様々なアイデアやアドバイスの提供を惜しみなくしてくださった株式会社パイケークの倉田様とインテル株式会社の浦口様にお礼を申し上げます．また，本研究の導入実験先として全面的に協力していただいた静パック有限会社の佐野様にお礼を申し上げます．

参考文献

1) IoT とは?｜IoT：Internet of Things（モノのインターネット）の意味，モノワイヤレス株式会社，http://mono-wireless.com/jp/tech/Internet_of_Things.html

2) 平成 27 年度情報通信白書，http://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h27/pdf/n5400000.pdf

3) Gartner Says 4.9 Billion Connected "Things" Will Be in Use in 2015, gartner, http://www.gartner.com/newsroom/id/2905717

4) シスコシステムズ合同会社 IoT インキュベーションラボ:(2013)Internet of Everything の衝撃

5) AWS IoT, Amazon Web Services, https://aws.amazon.com/jp/iot/

6) James Lewis, Microservices, http://martinfowler.com/articles/microservices.html

7) ボブ・ファミリア: (2016) Microservices on Azure

8) Sam Newman: (2016) マイクロサービスアーキテクチャ

9) 諏訪の中小企業と東工大が目指す“10 万円の工場 IoT”，ニュースイッチ, http://newswitch.jp/p/6952

10) PDCA サイクル，情報システム用語辞典，http://www.itmedia.co.jp/im/articles/1001/01/news028.html

11) 竹林康太:IoT アーキテクチャとしてのリアルワールド OS アーキテクチャデザイン，第十回社会システム部会，http://journals.socsys.org/symposium010/pdf/010-018.pdf

12) Kota Takebayashi, Kaya Akagi:(2016) Stage Model and Project Programming Architecture on Real World Operating System (RWOS), SICE Annual Conference 2016

13) Kaya Akagi, Kota Takebayashi:(2016) IoE data flow management with algebraic multi-dimensional bookkeeping system(AMBS) on real world operating system(RWOS) and its simulation, 10.1109/SICE.2016.7749259

14) 静パック有限会社, http://shizu-pack.com

付録

表 10 開発環境

種類	説明
PC	MacBook Air Early2014 CPU:1.5GHz Intel Core5 RAM:4GB 1600MHz DDR3 OS: macOS Sierra
MQTT サーバ，アンドンシステム	RaspberryPi 3 OS:Raspbian JESSIE
工場ボタン	iPad mini OS: iOS 10

表 11 開発に使用した言語とプログラムの対応表

該当プログラム	言語	使用したエディタ
工場ボタン	Swift	Xcode
CSV 保存ツール	Node.js	Atom
アンドンシステム	Processing	Processing IDE

表 12 CSV データにおけるレポートの一例

{“製造個数”:10000,”軽量個数”:23,”過量個数”:10,”異物個数”:8}