論文 / 著書情報
Article / Book Information

題目(和文)	IoTを用いた製造工程における状態管理フレームワークの研究
Title(English)	
著者(和文)	石塚康成
Author(English)	Yasunari Ishizuka
出典(和文)	学位:博士(工学), 学位授与機関:東京工業大学, 報告番号:甲第11647号, 授与年月日:2020年9月25日, 学位の種別:課程博士, 審査員:出口 弘,三宅 美博,山村 雅幸,小野 功,石井 秀明
Citation(English)	Degree:Doctor (Engineering), Conferring organization: Tokyo Institute of Technology, Report number:甲第11647号, Conferred date:2020/9/25, Degree Type:Course doctor, Examiner:,,,,
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

IoT を用いた製造工程における
状態管理フレームワークの研究

2020 年 8 月

情報理工学院

情報工学系　知能情報コース

石塚　康成

目次

第 1 章 序論 1
1.1 はじめに 1
1.2 研究の背景 1
1.3 既存技術の課題 1
1.4 マネジメントの IoT 化 4
1.5 ドメイン概念 5
1.6 研究の目的と意義 6
1.7 本論文の構成 7

第 2 章 状態管理手法 8
2.1 タスクの状態とその捕捉 8
2.2 工作機械による加工タスクの状態 10
2.3 状態遷移によるタスクの状態の捕捉 12
2.4 工作機械の状態 13
2.4.1 工作機械の振る舞い 13
2.4.2 工作機械への指令に基づく状態管理の課題 14
2.4.3 稼働状況のみの状態管理の課題 15
2.4.4 ドメイン概念に基づく状態管理の優位性 16
2.5 工作機械稼働状況の状態変換による状態の捕捉 16
2.6 状態管理フレームワークとしての状態管理手法 17

第 3 章 状態管理フレームワーク 18
3.1 マネジメントの IoT 化で必要とされる 9 項目 18
3.2 状態管理フレームワークとしての要件 19
3.3 状態管理フレームワークとしての CPS モデル 20

第 4 章 実証モデルの構築と結果の検証 25
4.1 想定する実証環境 25
4.2 作業者タスク状態管理実証モデル構築と検証 25
4.2.1 作業者の活動に付加される情報の検討 25
4.2.2 状態遷移図の記述 26
4.2.3 状態遷移コントローラー用 DSL の設計と実装 28
4.2.4 状態遷移コントローラーの検証 30
4.3 機械稼働タスク状態管理実証モデルの構築と検証 34
4.3.1 実証において使用した機器－BCD-MQTT ボード 34
4.3.2 実証モデル基本構成 35
4.3.3 ロボドリル専用状態定義の設計 36
4.3.4 状態コンバーター用変換定義ファイルの設計 37
4.3.5 機械稼働タスク状態管理の実証構成 38
4.3.6 リアルタイム状態提示画面の実装 39
4.3.7 日次状態集計チャートの実装 39
4.3.8 稼働機械タスク状態管理の実証結果の検証 39

4.3.9　不明状態の解消に伴う状態定義更新事例.....41

第5章 評価と考察.....45
5.1　機械稼働タスク状態管理の評価と考察.....45
5.2　日次状態集計チャートの効果.....50
5.3　状態管理フレームワーク実現のための要件に関する評価と考察.....53
5.4　状態管理フレームワークの導入・運用に関する考察.....55

第6章 結論.....56
6.1　本研究の成果.....56
6.2　今後の課題.....57

謝辞.....58

参考文献.....59

本研究に係る業績.....64

第1章 序論

1.1 はじめに

本研究では，出口(2018b)の提唱する「マネジメントの IoT 化」という概念のもと，IoT を用いた新たな状態管理手法を提案する．この状態管理では，工作機械等の稼働状況と，作業遂行状況を把握するための状態を区別し，機械内部的な稼働状況を，作業現場の人間が理解可能な作業遂行状態として収集・可視化する．製造現場の変化に追随し進化可能な状態管理の枠組みとして，提案する状態管理手法を実現する状態管理フレームワークを構築し，実在工場での実証を行い，有用性と課題を明らかとした．

1.2 研究の背景

さまざまな技術の進展に伴い，市場の多様化やグローバル化が進む中，企業は激しい競争にさらされている．製造業においても，業務改善を促進し競争力向上を図る上で，企業独自の考え方に基づく業務の見える化や分析といった業務管理（マネジメント）は不可欠なものとなっている．IoT（Internet of Things）(Ashton, 2009; Wortmann & Flüchter, 2015; Yang et al., 2018; 池澤ら, 2016)や CPS（Cyber Physical Systems）(Lee, 2008; 相原, 2019; 岩野・高島, 2015; 加藤, 2014)といった技術を活用し，今何が行われているかを把握することは重要である．このような技術を利活用し，業務の中で生産されるさまざまなデータを収集し，分析し，業務上の課題を明らかにしたうえで，独自の考え方に基づき改善を行っていく．製造企業が活動する領域において，垂直統合型の情報システムや，広くつながるための標準化を目的とした概念や枠組みもさまざまに提案されている．しかしながら，さまざまな製造現場において，それらが導入され活用されているとは言い難い．導入され活用されているとしても，企業の変化に追随してシステムを進化させているような事例は見当たらない．近年では，DevOps（Develop and Operations）(Kim et al., 2016)と呼ばれる概念も提案されている．開発チームと運用チームが互いに協調しあい，迅速に継続的に開発を進めるためのソフトウェア開発手法の一つである．開発チームと運用チームの意思疎通を図り，フィードバックループを継続させ，製品価値やユーザーの利便性を高め，早く確実なデリバリーを継続するという概念である．しかしながら，このような手法をすぐに取り入れ実践できたとしても，即座に成果が出るとは言い難い．

1.3 既存技術の課題

製造業において，工作機械などの製造設備からデータを取得し，それを利活用するための情報システムとして MES（Manufacturing Execution System：製造実行システム）(Jürgen, 2007; 貝原, 2006; 児玉, 2012; 高橋ら, 2010; 中村, 2000)があり，生産管理システムの一つ

として知られている．MES は ERP（Enterprise Resource Planning）などの経営情報システムと，PLC（Programmable Logic Controller）などの制御システムをつなぐ役割を担うものとして位置づけられている．国際標準 ANSI/ISA-95（IEC62264）の定義では，企業内の生産活動における計画と実際を管理し，製造計画やスケジューリングなどを最適化して生産能率の最大化を目指す製造管理システムの規格となっている(He et al., 2012; 児玉, 2012).

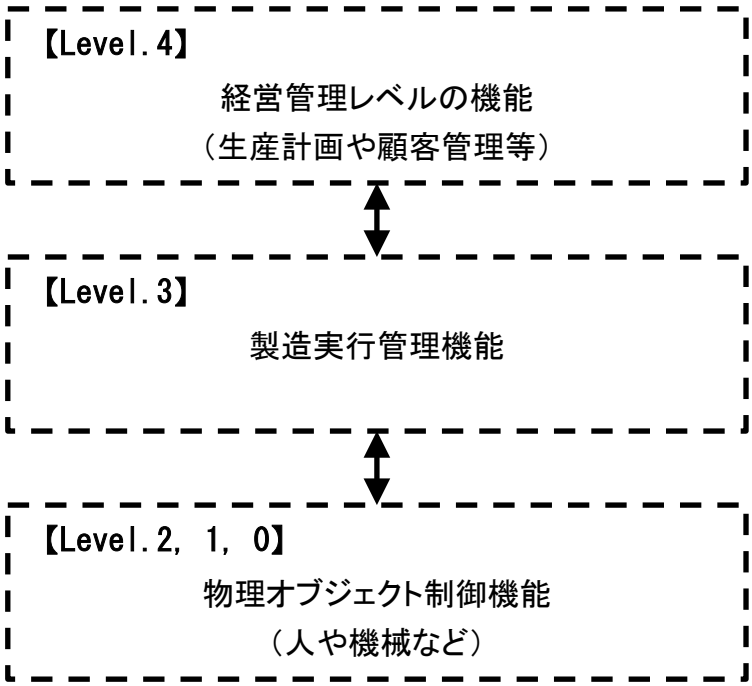


図 1-1　製造管理システムにおける機能階層

製造管理システムにおける機能は Level.0～4 までの機能の階層として定義されており（図 1-1），Level.4 が最上位に位置づけられている．Level.0～2 は人や機械などの物理オブジェクトを対象とした制御機能，Level.3 は製造実行管理機能，Level.4 が経営管理レベルの機能となっている．MES は製造における実行管理を行うシステムとして位置づけられているが，この規格はリファレンスモデルとしても定義されている．MES の機能は，生産資源の配分と監視，作業のスケジューリング，差立て・製造指示，仕様・文書管理，データ収集，作業者管理，製品品質管理，プロセス管理，設備の保守・保全管理，製品の追跡と製品体系の管理，実績分析となっている．データの取得や実行の指示などは，物理オブジェクト制御機能と行い，取得したデータは経営管理レベルに渡され，経営管理レベルにおいて分析等を行う．実際に情報システムを構築するためには，リファレンスモデルを用いて利用環境に合わせたモデルを作成し，それに基づくソフトウェアを設計・実装する必要がある．もちろん，このような標準化された定義に基づくことで，さまざまな機器からより多くの情報を取得し活用することができる．しかしながら，垂直統合された大規模なシステムが即座に導入で

きるとは限らず，実際 MES のような大きなシステムが導入されていない製造現場は，中小企業では多い．集中型 MES ソリューションは，その性質上柔軟性に欠けることが指摘されている(Bratukhin & Sauter, 2010)．このような課題に対し，分散型 MES も提案されてはいる(Bratukhin & Sauter, 2011)ものの，採用する分散型 MES のアーキテクチャに基づくソフトウェアを設計・実装する必要がある．さらに，対象とする工作機械の稼働状況としてさまざまな情報が取得できるものの，作業者の状態を扱う手法や，現場の変化に応じてシステムを進化させる方法などは示されていない．

より広くつながることを目的とした情報システムの規格として，Industrie 4.0 (JETRO, 2015; 岩本, 2015; 川野, 2015)がある．ドイツの製造業の競争力向上を目指した国家戦略として，スマート工場実現のための標準化の取り組みが進められている．考える工場の実現，製造コストの最小化などを目的とし，製造企業におけるバリューチェーン上の全機能のデジタル化を目指している．さまざまな研究も行われており，広範な研究のとりまとめなども見られる(Lu, 2017)．RAMI4.0 として，Industrie 4.0 のリファレンスアーキテクチャモデルの開発も進められている(撫中, 2016)．Industrie 4.0 コンポーネントなど，大まかな構造についてはガイドラインが存在するものの，開発途上ともいわれている(SmartFactoryKL, 2018; ロボット革命イニシアティブ, 2018)．

工作機械メーカーが提供するライブラリなどを用いれば，機械内部の詳細な情報は工作機械から直接取得できる．そのような情報は，稼働状況の監視やツールの予知保全などに用いられることが一般的である．Lee et al.(2010)は，コンピューター制御の数値制御工作機械（CNC）の I/O ポートから PLC（Programmable Logic Controller）を経由してデータを取得し，MES によって機械稼働状況を収集する仕組みを提案している．Oliveira et al.(2008)は，CNC から，オープン CNC という標準化されたインタフェースを介して取得した情報を用いて，生産数，加工時間，機械イベントなどを取得し可視化するアーキテクチャを提案している．これまで述べたように，製造工程で用いられる機械設備に着目した状態管理は，機械そのものの稼働状況を取得するものがほとんどであり，機械稼働状況を作業者の作業遂行状態として扱う手法などは示されていない．また，現場の変化に応じてシステムを進化させているような事例も見当たらない．

日本国内の製造業における「見える化」などの取り組み状況については，2019 年 12 月に実施された調査の結果が，2020 年版ものづくり白書にて報告されている．この報告によれば，およそ過半数の企業が生産プロセスに関する設備の稼働状況等のデータ収集を行っているという(経済産業省, 2020, p.66)．一方で，設備や人員などの見える化の実施状況については，2 割前後の企業が実施しているものの，4 割前後の企業が可能であれば実施したいという(経済産業省, 2020, pp.66-67)．単純に比較できないが，設備稼働状況等のデータ収集を実施しているにも関わらず，見える化などのデータ利活用には至っていない企業があると捉えることもできる．また，設備や人員の見える化に期待するものの，どのように見える化を行えばよいかがわからないという企業が少なくないとも考えられる．このような想定に

よるならば，設備稼働状況等のデータ収集と可視化をどのように行うかということも課題の一つといえる．さらに，日本企業の約 8 割が複雑化・老朽化・ブラックボックス化した基幹系システムを抱えているとの報告もある(経済産業省, 2020, p.68)．また，生産設備のうち，金属工作機械，第二次金属工作機械，鋳造設備といった機械機種では，5 割から 7 割近くが導入から 15 年以上経過しており，30 年以上導入から経過しているものも 2 割を超えているという(経済産業省, 2020, p.21)．このような既存の設備や情報システムも活用しつつ，マネジメントとしての見える化を実践するのであれば，マネジメントの必要な個所に小さく導入でき，現場にとって意味のある情報による見える化を実現可能とする情報システムが必要である．また，製造工程における生産活動には，作業者によるオペレーションを伴う．このオペレーションは作業者によって何らかの目的を伴い実施される．単に機械の稼働状況を収集し可視化するのではなく，作業者の意図が反映された状態として捕捉し提示できれば，マネジメントに有用な意味のある情報といえる．

1.4 マネジメントの IoT 化

出口(2018a)は，業務プロセスの遂行管理やデータ実測の困難さもあり，原価管理も製番などの単位ではなく月単位での大きなくくりで行われてきた点を指摘している．このような課題に対し，製番やロットなどで管理される単位で，製造プロセスのワークフローを一つのプロジェクトとして捉え，その遂行プロセスの計画と実際の差異を把握し修正する，もしくはそのための資料を提供するという目的で「マネジメントの IoT 化」というコンセプトが提唱されている(出口, 2018b)．「マネジメントの IoT 化」というコンセプトは，業務遂行の正しさ，製造計画と実際作業の差異，原価の把握等を目的とし，工作機械制御や作業自動化ではない，マネジメントのための IoT 利活用を主題としている．製造現場におけるさまざまな工程そのものは，早ければ月単位という早いサイクルで変化するのに対し，従来のソフトウェアの設計・導入・運用・改修というライフサイクルは通常年単位であり，その改修コストや期間も製造現場の変化によるものと比べて大きいという．さらに，情報システムの設計側と製造現場のライン設計側との概念ギャップが大きい点も挙げ，“多様性が大きいものづくりの現場では，現場と概念を共有しそれに基づき設計と実装と導入後の改修・組換えを行っていく必要がある”という(出口, 2018b)．本研究でも，同じ立場をとる．

マネジメントの IoT 化では，2 ラインアーキテクチャという概念も提唱されている(出口, 2018b)．工作機械等の情報を取得する際は，機械固有のインタフェースを利用することが一般的である．しかしながら，旧式の設備など，すべての設備にそのようなインタフェースが搭載されているとは限らない．また，機械固有のインタフェースは，制御用として固有のハンドシェイクが必要となる．そのようなインタフェースによらず，工作機械等の設備からの出力信号を，第 2 の情報ラインとして活用するという概念である．この 2 ラインアーキテクチャの事例として，24V 端子の信号を 5V へ変換する 1ch，16ch，32ch の基板を設計・

実装し，データ取得用のコンテナも作成されている．この 2 ラインアーキテクチャによる，IoT 化の実践事例として，製造現場で稼働する工作機械の稼働状況の提示を目的とした，Web 表示灯システムが報告されている(出口 2018b)．日本の製造現場では，その稼働状況を示す目的で工作機械に 3 色の表示灯が搭載されているのが一般的であり，そのうちの稼働中を示す緑点灯を，1ch 変換基板を用いて 2 ラインアーキテクチャにより IoT 化した実践事例となっている．Web 表示灯システムは，工作機械の緑点灯に相当する稼働状況として，稼働中かそうでないかの 2 種類の情報のみを，Web ブラウザを通して提示する（図 1-2）．16ch の変換基板など，より多くの信号を取得する仕組みを用いれば，より多くの機械信号が取得可能という．このような機械直接的な稼働状況を，作業者の意図が反映された状態として取得できれば，マネジメントに有用な意味のある情報といえる．しかしながら，機械稼働状況から製造現場で活動する人間の作業遂行状態として取得するような手法は確立されていない．機械稼働状況をマネジメントに有用な状態として管理可能となったとき，そのような情報システムは製造現場の変化に追随可能か，という点での実証も行われていない．

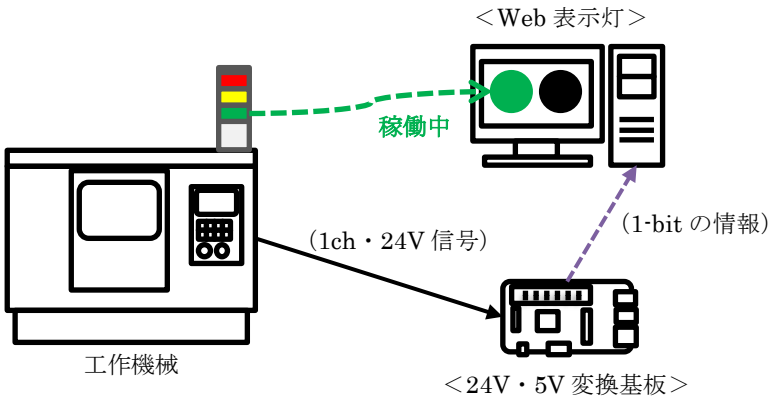


図 1-2　Web 表示灯システムの動作イメージ

1.5 ドメイン概念

現場にとって意味のある情報とは何か，という点も明らかとすべき課題である．工場などの製造現場では，さまざまな活動が行われている．1 製造企業内でも複数の工場が稼働しているものもあり，各工場が自律的に活動を行っている．工場のような作業現場において，それぞれ業務として達成すべき目標があり，その現場が培ってきた理念や流儀に基づき，日々の業務が行われていると考えられる．ここで，製造現場を一つのマネジメント領域（ドメイン）とするならば，そのドメインにおいて業務遂行に用いられる業務知識を「ドメイン概念」と呼ぶこととする（図 1-3）．マネジメントを導入する箇所を中心に，マネジメントのフィードバックを受け取る範囲がドメインであり，そのドメイン概念に基づいてマネジメント

に有用な情報として状態を規定していくことが必要である.

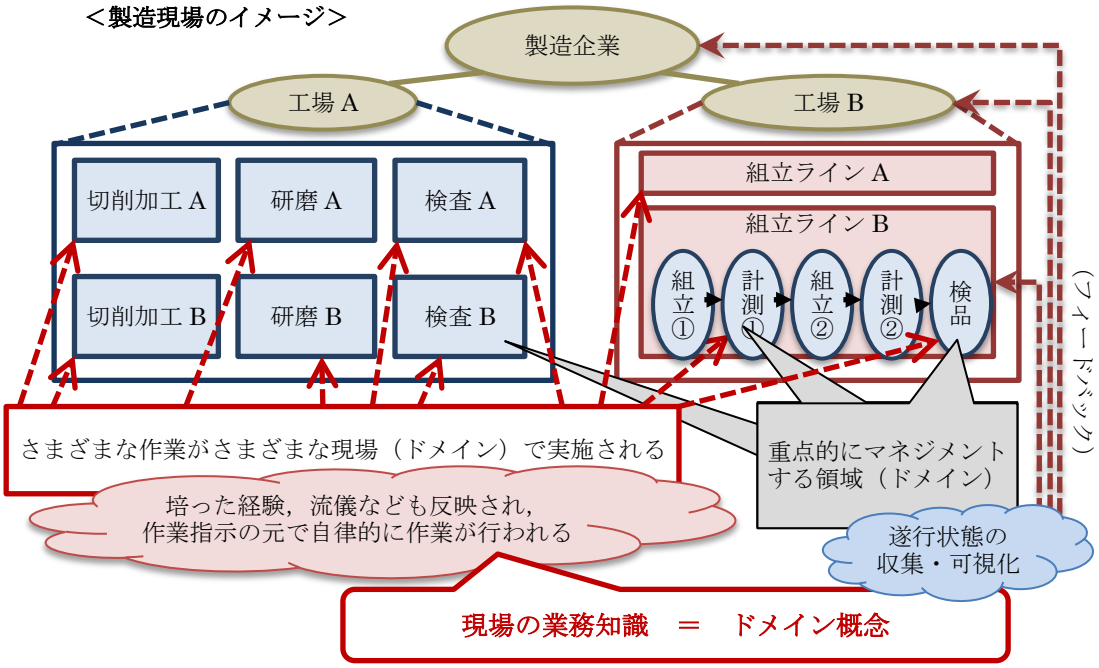


図 1-3 ドメイン概念

1.6 研究の目的と意義

これまでに述べたような課題に対し，本研究では，製造工程で行われる工作機械による加工タスクの状態収集と可視化に着目し，製造現場で意味のある状態によるマネジメント支援を目的として，IoT を用いた新たな状態管理手法を提案する．具体的には，工作機械等の稼働状況と，作業遂行状態を把握するための状態を区別し，機械内部的な稼働状況を，作業現場の人間が理解可能な作業遂行状態として収集・可視化する．マネジメントの IoT 化というコンセプト(出口, 2018b)のもと，製造現場の変化に追随し進化可能な状態管理の枠組みとして，提案する状態管理手法を実現する状態管理フレームワークを構築し，実在工場での実証によって，状態管理フレームワークとして実践される状態管理の有用性を示す.

本研究で提案する状態管理フレームワークは，重点的にマネジメントが必要な個所に小さく導入し，徐々に改良や組換えを行いながら，現場とともに進化することを可能とする．本研究の成果によって，そこで活動する作業者や技術者，経営者の意識共有が促進され，自律的なイノベーションの促進に寄与できる．小さく導入し組み替え容易な状態管理フレームワークが実現できれば，早いサイクルで組み替えが行われる現場の進化に追随可能といえる．これらが達成されれば，状態管理フレームワークを導入し利活用する製造企業のみならず，そのような製造企業と取引のある他の製造企業やサプライヤーが自社の新たなサー

ビスとして展開し，企業価値を高めていくことも期待できる．このようなイノベーションが伝搬され促進されていくことを，本研究は目指している．

1.7 本論文の構成

本論文の構成について説明する．第 2 章では，作業タスクの状態について考察し，提案する状態管理手法について述べる．第 3 章では，マネジメントの IoT 化の要件に基づき，状態管理フレームワークの CPS モデルについて述べる．第 4 章では，実証環境と実証モデルの導入と実証結果について述べる．第 5 章では，実証結果に基づく評価と考察について述べる．第 6 章では，結論として本研究の成果と課題について述べる．

第2章 状態管理手法

本章では，まず，タスクの状態と基本的な捕捉方法について論じる．次に，工作機械を用いた加工タスクに着目し，作業者側と工作機械のそれぞれの状態空間が存在することを示した上で，状態管理フレームワークにおける作業者側と工作機械側の状態管理手法について説明する．

2.1 タスクの状態とその捕捉

マネジメントの IoT 化の基礎となるマネジメント概念では，マネジメントの最小単位を「タスク」とし，さまざまな仕事に対応するタスクが業務のワークフローに従い半順序で結合された「プロジェクト」とともにマネジメントを行うこととしている(出口, 2015, 2018a, 2018b)．タスクは，その内部的な遂行において手戻りがあっても，当該タスク終了後はそのタスク自体に戻らないという仕事の単位で切り出されたものとしている．プロジェクトを構成するタスク間では，前のタスクに戻らないことを原則としている．工場のようなものづくりにおいて，タスクは加工や組み立てといった仕事に対応し，プロジェクトは製番やロットなどで管理されるワークフローに対応する．このタスクやプロジェクトといった単位で進捗を正しく把握することが，より正確な原価管理を行う上で重要とされる．例えば，ある製品の製造作業として，2 種類の形状の切削加工を行う「切削 A タスク」と「切削 B タスク」があるとする．「切削 A タスク」と「切削 B タスク」の両方が完了したとき，「組み立てタスク」が実行されるものとし，2 ロットの製品製造を行うと仮定した場合，プロジェクトとタスクの構成は図 2-1 のようになる．本研究では，このタスクの状態管理に着目する．

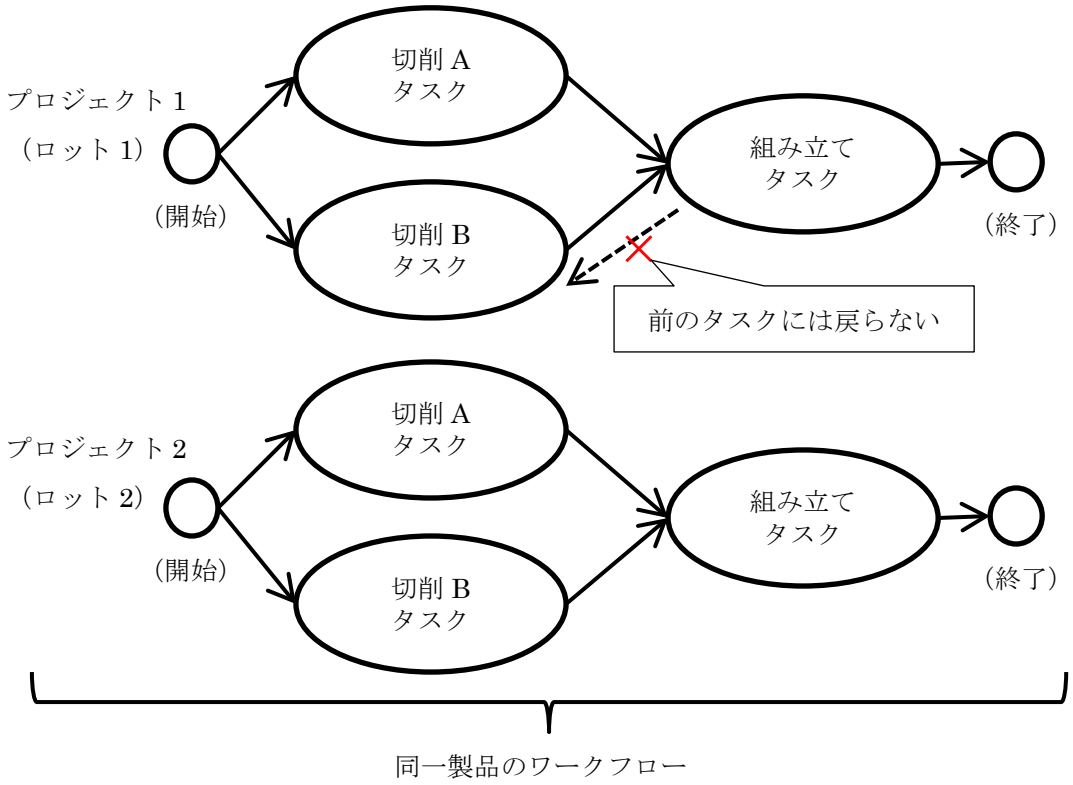


図 2-1　ワークフロー／プロジェクト／タスクの例

このタスクの最もシンプルなモデルは，図 2-2 のようなモデルで表現されている（出口, 2018a, 2018b）．このモデルでは，タスクが開始されると作業準備中となり，主作業のための段取り替えなどが行われる．作業準備が完了すれば，主作業が開始され，主作業中となる．主作業中には，さまざまなトラブルが発生することもあり，トラブルが発生すればトラブル対応中となる．トラブル対応中には，発生した問題に対処し，復旧を行い，トラブルが解消されれば主作業に戻る．主作業がすべて終了すれば，終了作業中となる．終了作業中には，後片付けや作業報告などが行われる．終了作業もすべて完了し，このタスクは終了する．

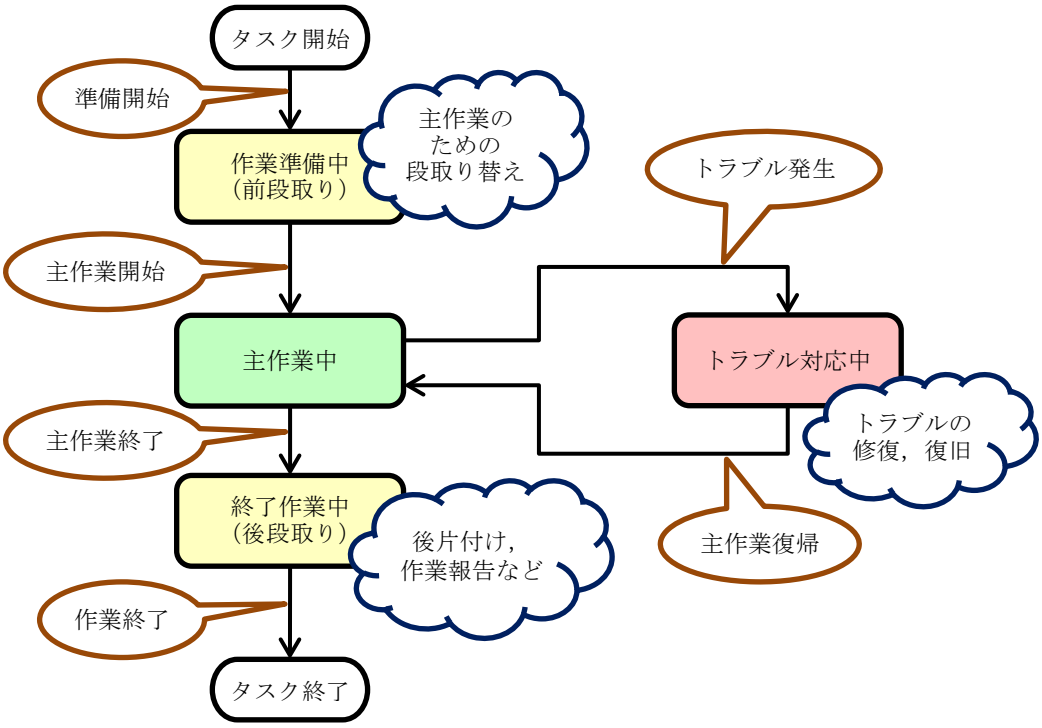


図 2-2　シンプルなタスクモデル

マネジメントの IoT 化では，製造工程における機械加工や組み立て作業といったものがタスクに相当するとしている．機械加工タスクにおいても，段取り替えや加工対象物の脱着など，人の活動に関連する工程も含まれる．また，多くの作業工程で測定工具や計測機器を用いて測定が行われる．計測機器等からのデータ出力や，人の活動のみで行われる作業のアウトプットなどが，タスクの状態を進める要因となる．この作業遂行に対応するタスクの状態を適切にマネジメントすることが，マネジメントの IoT 化では必要とされる．

2.2 工作機械による加工タスクの状態

次に，工作機械による加工タスクの状態について検討する．ここでは，切削加工を行う作業を，次のように仮定する．まず，何らかの作業を行う上で，何を，どのように，何個作るかという作業内容（製造指示）が作業者に与えられる．作業者はその指示に従い，作業準備として，加工対象物の準備，加工対象に応じたツール（切削工具）の入れ替え，加工プログラムの投入，治具の調整を行う．作業準備が完了した後は主作業として，加工対象物投入，加工開始，加工終了後の加工対象物取り出し，測定を行う．測定結果によっては，加工プログラムの調整や，消耗ツール（切削工具）の交換などもある．また，加工実行中には何らかのトラブルにより機械が停止することもある．この場合，そのトラブルを調査し，復旧作業を行い，加工を再開する．測定に問題がなければ終了作業として，後片付け（清掃），作業

報告を行い，この切削加工作業は終了する．このような想定においての各作業は，作業準備中，主作業中，トラブル発生中，作業終了中というシンプルなタスクモデルの状態に，図 2-3 のように対応する．

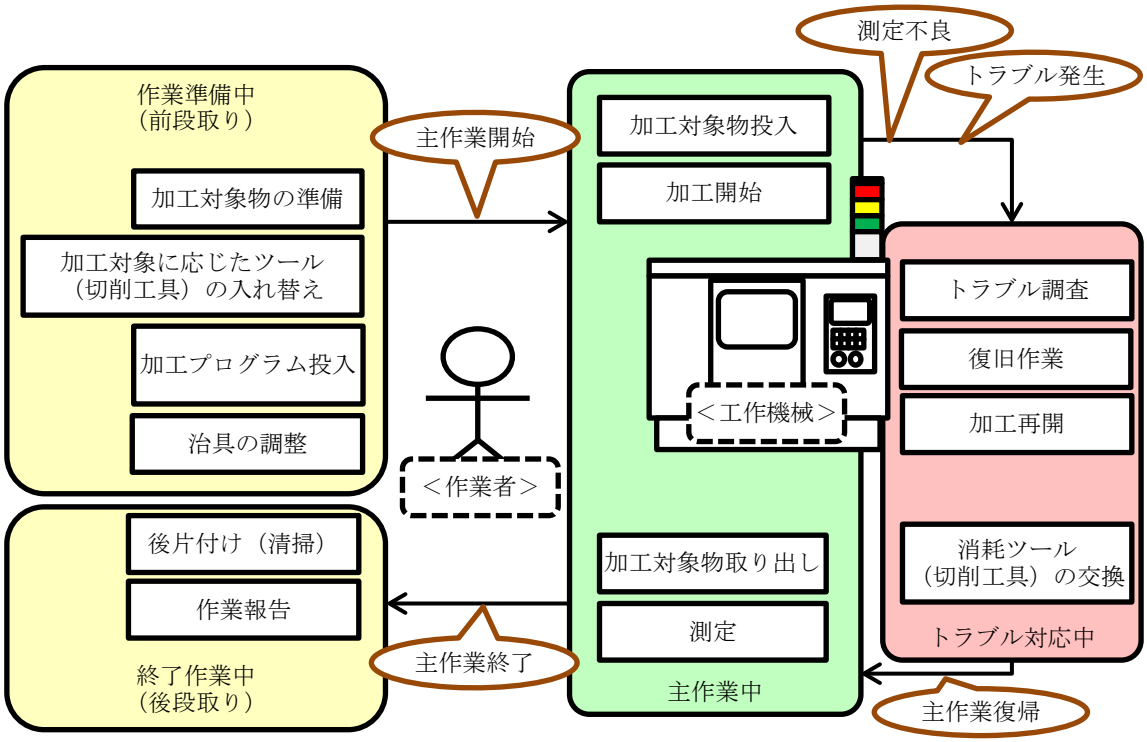


図 2-3　切削加工タスクにおける作業者の状態例

このような想定によれば，工作機械による加工タスクはシンプルなタスクモデルで表現することができ，作業者である人間を主体とした作業は，タスクとしてその状態を管理可能といえる．しかしながら，実際の製造作業では，加工方法や使用する工作機械によって異なることは十分に考えられる．さらに，その企業独自の作業方法や手順もある．ここで，このタスクが実行される現場をマネジメントのドメインと捉えると，作業遂行状況を表す状態は，現場固有の作業手順ともいえる．図 2-3 で示した作業のうち，主作業中に相当する作業は，加工対象物投入，加工開始，加工対象物取り出し，測定の順に行われることは明らかである．より詳細な粒度の状態として捉えるならば，加工対象物投入と加工対象物取り出しを脱着中，加工を開始して加工プログラムが実行されている状態を加工実行中，測定を測定中，トラブル発生後のトラブル対応中と，測定不良発生後の測定不良対応中とみれば，図 2-4 のような状態遷移も想定できる．

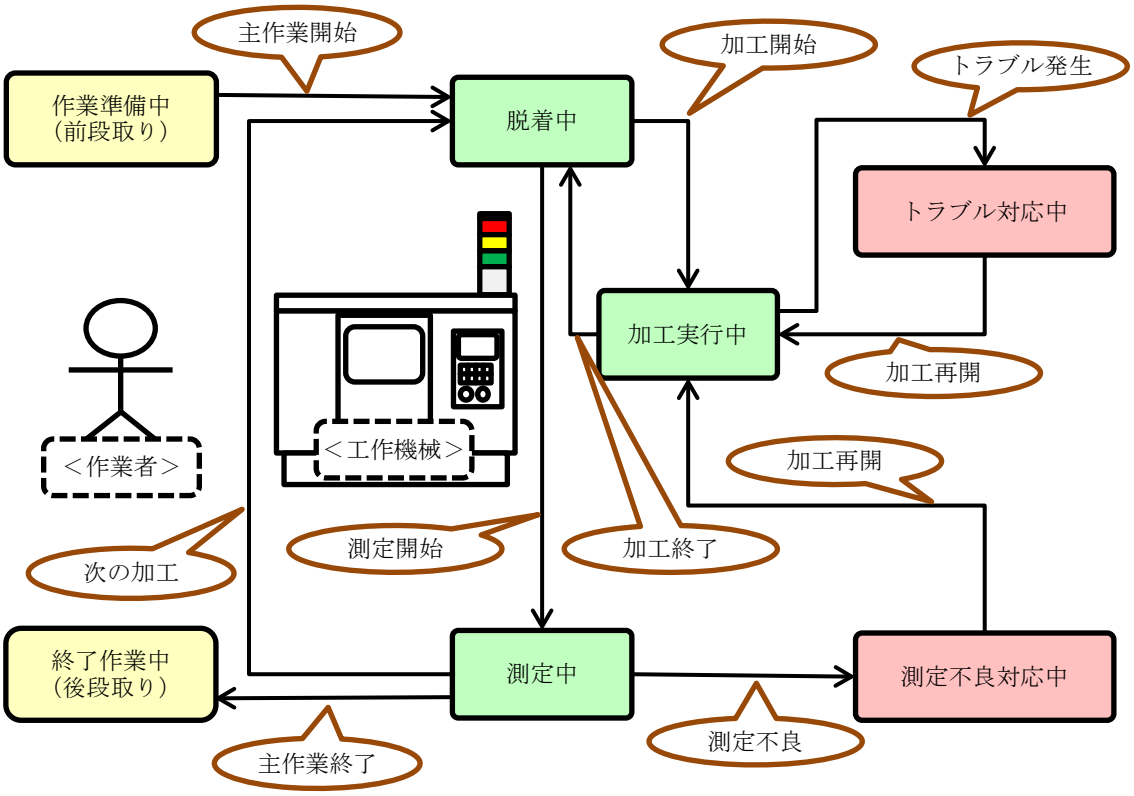


図 2-4　切削加工タスクにおけるより詳細な作業者の状態例

このような想定に基づけば，作業者の作業進行状況を表すタスクの状態とその遷移は，ドメイン概念に基づいて決定可能といえる．したがって，マネジメント主体が状態管理を行うには，マネジメント対象のタスクにおける作業手順から，状態とその遷移を規定し，規定された状態を収集・可視化すれば良い．

2.3 状態遷移によるタスクの状態の捕捉

タスクの取りうるすべての状態が，タスクとして切り出された作業固有の手順から決定されるとするならば，その状態数は有限である．これを状態遷移として捉えるとするならば，何らかの入力によって，次の状態が一意に定まるものとする．このようなモデルは，決定性有限オートマトンとして知られている．また，状態遷移をマネジメントにおいて必要とされるタスクの遂行状況として捉えるには，何らかの出力を伴うものとすれば良い．出力を伴う状態遷移モデルは，順序機械（Sequential Machine）として知られている．状態の有限集合をQ，入力記号の有限集合をΣ，出力記号の有限集合をΔ，状態遷移関数をδ，出力関数をλ，初期状態をq_0とすると，順序機械Mは形式的に次のように定義されている．

$$M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$$

ここで，現在の状態をp，それへの入力をa，次の時点の状態（遷移先状態）をq，出力をbとおくと，状態遷移関数δは次の規則に従って与えられたものである．

$$\exists p \in Q, \exists a \in \Sigma, \exists q \in Q \implies \delta(p,a) = q$$

また，出力関数λは次の規則に従って与えられたものである．

$$\exists p \in Q, \exists a \in \Sigma, \exists b \in \Delta \implies \lambda(p,a) = b$$

順序機械Mのパラメーターが次のものであるとき，

$$Q = \{p, q\}, \quad \Sigma = \{0,1\}, \quad \Delta = \{a, b, c, d\}$$

$$\delta(p,0) = q, \quad \delta(p,1) = p, \quad \delta(q,0) = q, \quad \delta(q,1) = p$$

$$\lambda(p,0) = b, \quad \lambda(p,1) = a, \quad \lambda(q,0) = c, \quad \lambda(q,1) = d$$

$$q_0 = p$$

この例における順序機械Mの状態遷移図を図 2-5 に示す．

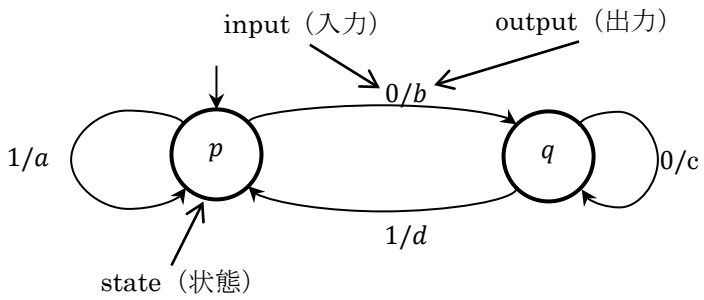


図 2-5　順序機械Mの状態遷移図例

タスクの状態遷移において，作業進捗に応じた入力に対し，その遷移先状態を出力とすれば，その出力を収集・可視化することで，作業の実際を把握することができる．作業者の作業進行状況を表すタスクの状態とそれへの入力，状態の入力の組み合わせによる遷移先状態が，ドメイン概念に基づいて定義可能とするならば，その定義によって構築された状態遷移モデルは，現場の変化にも追随可能といえる．

2.4 工作機械の状態

2.4.1 工作機械の振る舞い

次に工作機械の稼働状況を示す状態について検討する．まず，工作機械を用いた切削加工作業において，作業者と工作機械は物理的に独立している．これは，作業者と工作機械が並列に動作することを意味する．作業者は，作業指示に応じて必要な作業を自律的に行う．工作機械を操作することによって，工作機械を稼働させ，加工終了や異常発生などの工作機械からのフィードバックによって，その都度必要な作業を行う．工作機械が取りうる稼働状況は，待機や一時停止，プログラム実行，ドア解放などが想定できる．しかしながら，実際に

取りうる稼働状況は工作機械の機種やメーカーによってさまざまである．

ここで，マネジメント対象とするタスクも並列なもの，つまり独立したタスクと想定する．これらタスクを区別するために，ここでは便宜上，作業者側のタスクを「作業者タスク」，工作機械側のタスクを「機械稼働タスク」，と呼ぶこととする（図 2-6）．

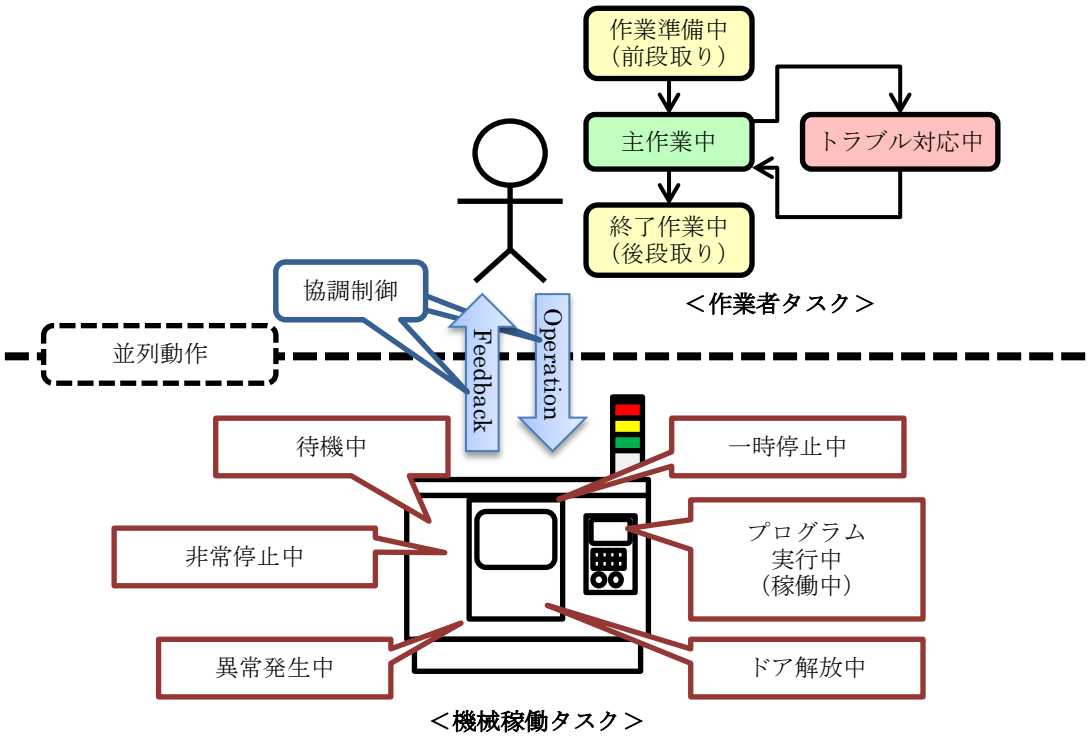


図 2-6　切削加工作業の作業者タスクと機械稼働タスクの動作イメージ

このような想定において，工作機械の直接的（機械内部的）な稼働状況を，機械稼働タスクの意味のある状態として管理可能か，そしてその状態は，作業手順や内容の変化に追随可能か，という点が課題となる．

2.4.2 工作機械への指令に基づく状態管理の課題

工作機械のうち，位置や速度などの数値情報によって制御し，一連の動作をプログラムした指令によって実行する工作機械を「数値制御工作機械」という（JIS B 0105:2012）．一般的には「NC 工作機械」と呼ばれており，コンピューターによって制御されるものは「CNC 工作機械」とも呼ばれている．NC 工作機械を制御する数値情報は，指令となるコードと，座標や速度といったパラメーターで NC プログラムを構成し，その NC プログラムを NC 工作機械で実行することで，工作機械の振舞いを制御する．指令コードは，準備機能である G コードと，補助機能である M コードというコード体系が，JIS 規格化されている（JIS B 6315-1:2013）．G コードは G00 から G99 の範囲で，一部未指定が含まれているものの，位

置決め(G00), 直線補完(G01), 時計回りの円弧補完(G02), 反時計回りの円弧補完(G03), タッピング (G63), 原点復帰 (G74) といったものが規定されている. 一方 M コードは M00 から M99 の範囲で, 一部未指定が含まれているものの, プログラムストップ (M00), エンドオブプログラム (M02), 主軸停止 (M05), 工具交換 (M06), 工作物クランプ (M10), 工作物アンクランプ (M11) といったものが規定されている. しかしながら, 工作機械メーカーや機種によって, 独自のコードや意味が異なるコードが含まれている場合もある.

NC 工作機械で実行中の指令コードとパラメーターが取得できると仮定した場合, それが実行され正しく動作しているならば, 稼働状況を表しているといえる. しかしながら, あくまでも指令であるため, NC プログラムが実行されていない状態や, 異常状態などの稼働状況を表すものは含まれていない. したがって, この NC プログラムのみでは, 機械の稼働状況を表現するには不十分といえる.

2.4.3 稼働状況のみの状態管理の課題

日本の製造現場で稼働する工作機械には, その稼働状況を知らせる目的で, 3 色 (赤・黄・緑) の表示灯 (積層表示灯とも呼ばれる) が接続されていることが一般的である. この 3 色表示灯に着目すると, 単に稼働状況を見るのであれば, 表示灯の点灯状況を, 赤点灯は「非常停止中」, 黄点灯は「待機中」, 緑点灯は「稼働中」, 全消灯は「停止中」という意味で捉えるとする (図 2-7). この場合, 直接的な機械の稼働状況を示すのみであり, 人間が理解可能ではあるが, 工作機械が稼働しているか否か, といった理解でしかない. ここで, 工作機械に投入した加工対象物の加工状況を逐次確認しながら行う「テスト加工」が実施されたとする. これは, 通常の量産加工とは異なる目的で実施された作業といえる. この想定において, テスト加工がドメイン概念で「段取り」と認知されているとするならば, この例の状態には「段取り」という目的は含まれていない. ここで, 段取りか量産かといった目的がマネジメント上必要であるとするならば, この例のような工作機械の稼働状況のみでは不十分といえる.

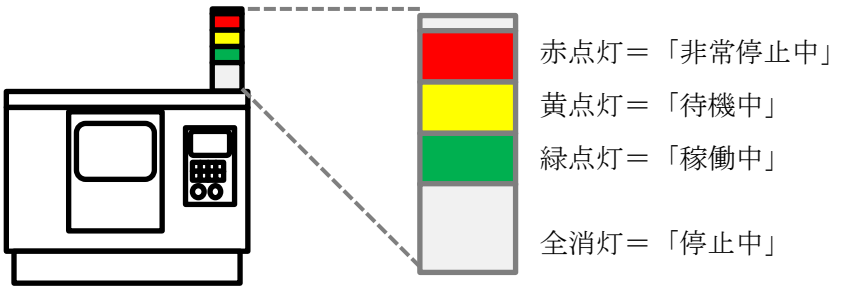


図 2-7　工作機械の 3 色表示灯点灯状況に基づく稼働状況の想定

2.4.4 ドメイン概念に基づく状態管理の優位性

前述のテスト加工のような逐次実行が，工作機械のモード切り替えによって実現していると仮定する．通常の実行時を「通常モード」，逐次実行時を「逐次モード」とし，このモードも工作機械の稼働状況として取得可能であるとする．ここで，ドメイン概念に基づく状態を次のように定義する．まず，通常モードでのドメイン概念に基づく状態として，緑点灯なら「量産：加工」，黄点灯なら「量産：脱着」，赤点灯なら「量産：トラブル」，すべて消灯なら「量産:一時停止」とする．さらに逐次モードでのドメイン概念に基づく状態として，緑点灯なら「段取り：加工」，黄点灯なら「段取り：脱着」，赤点灯なら「段取り：トラブル」，すべて消灯なら「段取り：一時停止」とする．これらをまとめたものを表 2-1 に示す．

表 2-1　工作機械の 3 色表示灯とモードの組み合わせによる状態

	通常モード	逐次モード
赤点灯	量産：トラブル	段取り：トラブル
黄点灯	量産：脱着	段取り：脱着
緑点灯	量産：加工	段取り：加工
全消灯	量産：一時停止	段取り：一時停止

このような前提においても，工作機械の稼働状況としては，3 色表示灯の点灯状況と通常モードか逐次モードのどちらかの状態のみである．ドメイン概念において，工作機械の稼働状況の組み合わせが何を意味するかが認知されていれば，この組み合わせを見て人間が判断することは可能である．ここで，業務改善として「段取り」は「試作」に定義変更されたとする．これはドメイン概念上の変更であり，工作機械に変更点はない．しかしながら，作業の目的がマネジメント上必要な情報であるとするならば，ドメイン概念に基づいて定義された状態は有用であり，工作機械稼働状況のみでは得られない情報が得られるという点において，ドメイン概念に基づくタスクの状態管理は直接的な機械稼働状況よりも優位であるといえる．

2.5 工作機械稼働状況の状態変換による状態の捕捉

2.4.4 項で示した 3 色表示灯の点灯状況とモードの組み合わせによる状態の想定において，3 色表示灯の点灯状況を赤・黄・緑のそれぞれが点灯中か否かという単純な ON/OFF 信号と捉え，「通常モード」と「逐次モード」の 2 通りのモードを「通常モード」か否かという単純な ON/OFF 信号と捉えれば，これら 4 つの信号の組み合わせと状態との対応は表 2-2 のように表すことができる．

表 2-2　機械稼働状況の組み合わせと状態の対応例

状態	赤点灯	黄点灯	緑点灯	通常モード
量産：トラブル	ON	OFF	OFF	ON
量産：脱着	OFF	ON	OFF	ON
量産：加工	OFF	OFF	ON	ON
量産：一時停止	OFF	OFF	OFF	ON
段取り：トラブル	ON	OFF	OFF	OFF
段取り：脱着	OFF	ON	OFF	OFF
段取り：加工	OFF	OFF	ON	OFF
段取り：一時停止	OFF	OFF	OFF	OFF

この想定によれば，工作機械の稼働状況に応じた情報（信号）を入力とし，その入力から対応表に従い導出された状態を出力とすれば，その出力を収集・可視化することで，工作機械によって遂行されている作業の実際を把握することができる．この対応表は，状態変換定義ともいえる．この状態変換定義が，ドメイン概念に基づいて定義可能とするならば，その定義によって構築された状態変換モデルは，現場の変化にも追随可能といえる．

2.6 状態管理フレームワークとしての状態管理手法

これまでに論じたように，工作機械による加工タスクに着目し，作業者側のタスク（作業者タスク）の状態と，工作機械側のタスク（機械稼働タスク）の状態があることを示した．工作機械は作業者の操作によって作業が進行するという意味で協調的に作動する．NC プログラム等で自律的に作動するものの，そこには作業者の目的が反映されている．異常状態の発生や NC プログラム実行終了などの工作機械からのフィードバックを作業者が受け取り，それに応じて次の作業が行われる．このような想定によれば，作業者タスクは状態遷移によってその遂行状況を把握するための「状態遷移型タスク」であり，機械稼働タスクは工作機械等の稼働状況を示す複数の信号等の組み合わせから状態を決定する「状態変換型タスク」である．これら 2 つのアプローチによって，製造現場における状態管理が実現可能となる．状態管理を行うためには，作業者タスクと機械稼働タスクのそれぞれにおいて，状態定義を設計し，その状態定義を状態管理フレームワークに反映し，状態管理フレームワークを導入・運用すれば良い．状態管理フレームワークによって，マネジメント対象のタスクの状態が収集され可視化される．この収集・可視化で得られた情報に基づき，日々の業務の課題について検討し，業務改善を行っていく．業務の改善などで作業手順が変更されても，その変更を状態定義にフィードバックし，状態管理フレームワークに反映すれば，マネジメント対象の変化にも追随可能といえる．

第3章 状態管理フレームワーク

本章では，状態管理手法を実践する状態管理フレームワークの設計概念について説明する．マネジメントに用いるための状態管理フレークワークとしての要件を定義し，局所的な導入と組み替え容易な構成とする上での疎結合型モデルを示す．

3.1 マネジメントの IoT 化で必要とされる 9 項目

「マネジメントのための IoT 化」(出口, 2018b)では，製品製造工程の作業をワークフロー，プロジェクト，タスクという概念を用いて，その遂行状態を管理するための基本アーキテクチャを定義している．複数のタスクを半順序で結合したものがワークフローであり，製番やロットなどの単位で管理するワークフローをプロジェクトと呼ぶ．プロジェクト型のワークフローを管理する目的のため,マネジメントの仕組みとして必要とされる 9 項目が，IoT ベースのマネジメントのための基本アーキテクチャとして次のように規定されている．

① 人間や機械装置や測定工具・計測機器（ゲージ）からマネジメントに必要なデータを得るには，データを取得するための装置を IoT コンポーネントとしてラッピングし利活用するための仕組みと，データ取得のためのハードウェアと連携するソフトウェアを開発する枠組みが必要となる．

② IoT コンポーネントを組み替えの容易な疎結合コンポーネントとして扱うために，ネットワーク上でのノード（IoT コンポーネント）間の相互通信のための疎結合型アーキテクチャを必要とする．この組み替えの容易な通信枠組みとして，MQTT プロトコルと呼ばれる通信方式を用いる．

③ MQTT プロトコルでやりとりされるデータは JSON や CSV のような浅い構造化で，それぞれのコンポーネントで局所的に定式化される．コンポーネント間を横断した大域的なスキーマの統合は敢えておこなわず，データは必要に応じて必要な個所で変換を行うというデータフロー型の設計を基本とする．

④ タスクの内部管理を行うための枠組みとして，タスクの内部状態とその遷移を把握し，さらにさまざまな計測データを把握する枠組みが必要となる．

⑤ タスクが結合したプロジェクトでは，現在どのタスクが着手中であるか，着手可能であるか，終了済みであるかといった進捗を管理するための枠組みが必要となる．

⑥ 複数のプロジェクトが並列に実行される環境では，それぞれのプロジェクトの各タスクに対して人や機械などの資源を一定のルールで割り当てるスケジューリングの枠組みが必要となる．

⑦ タスク単位で，原料（仕掛品）と人的資本サービス，物的資本サービスをどれだけ投入すると製品（仕掛品）が生成されるかについての原価管理と，それをプロジェクト単位でまとめた原価管理の枠組みも必要となる．

⑧ タスクの遂行に関して取得したさまざまなデータに対して，表示や分析のためのデータ処理を行う枠組みも必要となる．

⑨ 企業の壁を超えて部品やサービスに関する情報を不改竄証明付きで追跡可能とするトレーサビリティの枠組みも必要となる．

3.2 状態管理フレームワークとしての要件

マネジメントの IoT 化を基礎とする状態管理フレームワークとするならば，重点的にマネジメントを行う領域に小さく導入し，作業の遂行状況である状態を収集・可視化できることが必要とされる．また，工作機械の作業状況を取得する IoT 機能をコンポーネント化し，疎結合によってコンポーネントが結合できることが必要とされる．さらに，コンポーネント間で交換されるデータは浅い構造のデータフォーマットでコンポーネントごとに局所化できることが必要とされる．また，タスクの状態の収集・可視化が行えることも必要とされる．以上のことから，状態管理フレームワークの満たすべき要件を，次の 6 項目として設定する．

(1) 局所的に導入できる

作業タスク等のマネジメントにおいて，重点的にマネジメントを行う対象に局所的導入が可能となれば，複数の箇所に順次追加していくことも可能となる．

(2) 疎結合型コンポーネントである

データの保存や可視化を行うコンポーネントと機械等のセンサリングを行う IoT コンポーネントの間には，交換されるデータの型式のみで結合されることを原則とする．工作機械や測定装置などの情報取得対象の IoT 化は，その装置固有の IoT コンポーネントとして開発すればよい．この原則に従う限り，機能追加や変更は，その対象となるコンポーネントに対してのみとなり，追加や変更の影響が最小化できる．コンポーネント間で交換されるデータは，JSON を原則とする．

(3) ドメイン概念に基づいた状態を規定できる

マネジメント主体が自らのドメイン概念に基づいて状態を規定できることは，状態管理フレームワークの導入容易性を高める上で必要であり，製造現場の変化に追随するという点においても必要となる．

(4) 規定した状態をシステムへ反映できる

ドメイン概念に基づいて規定された状態は，状態管理フレームワークに反映され，その規定に基づいた状態管理が行われることが必要となる．ドメイン概念に基づいて規定された状態を用いている限り，ドメイン概念に基づいた変化にも追随可能となる．

(5) 収集した状態データを可視化できる

マネジメント主体への情報のフィードバックであり，状態管理において必須である．

(6) 収集した状態データを抽出できる

収集したデータは，マネジメント主体が任意に取得できることが必要となる．データが自由に取得できれば，マネジメント主体はそのデータを用いてさまざまに分析を行うことが可能となる．

3.3 状態管理フレームワークとしての CPS モデル

3.2 節で設定した状態管理フレームワークの要件を満たすため，状態管理フレームワークとしての CPS モデルの基本構成を，次のように想定する．まず，状態管理フレームワークの通信基盤として，MQTT プロトコルを利用する．通信によって交換されるデータは，基本的に JSON とする．データ取得装置が，物理世界（Physical 空間）のデータを取得するモジュールとなる．状態遷移や状態変換，状態表示画面などの機能（Cyber 空間）は，ソフトウェアコンポーネントとして構成する．基本的に，モジュールやソフトウェアコンポーネントは，データフロー型のコンポーネントとする．これらがマイクロサービスとなり，疎結合型コンポーネントとして独立して動作する．独立して実行可能なコンポーネントである限り，変更の影響も局所的で，組み換えや変更も容易となる．データストレージにはドキュメント型データベースを想定する．スキーマレスで扱えれば，データ構造変更の影響も極小化できる．

MQTT(MQTT, 2020; OASIS, 2014; Yassein et al., 2017)は，1999 年に IBM の Andy Stanford-Clark と Arcom（現 Eurotech）の Arlen Nipper によって発明された，パブリッシュ／サブスクライブ方式に基づく TCP/IP 上の非常に軽量な通信プロトコルである．サーバーに相当する MQTT ブローカーを介し，トピックという宛先を用いてメッセージが交換される．さまざまなプログラミング言語用の MQTT クライアントライブラリがオープンソースソフトウェアとして公開されている．また，クラウドサービスやオープンソースソフトウェアとして公開されている MQTT ブローカーも利用できる．MQTT の通信例を図 3-1 に示す．MQTT の通信は，MQTT ブローカーを介してメッセージ単位で行われる．パブリッシャーがメッセージの送信者であり，サブスクライバーがメッセージの利用者である．この例ではまず，サブスクライバー1 が “topicA” というトピックをサブスクライブしているものとする．そのとき，パブリッシャーが “topicA” というトピックに対し data1 という内容のメッセージをパブリッシュすると，そのトピックをサブスクライブしているサブスクライバー1 にメッセージ（data1）が配信され，サブスクライバー1 がそのメッセージによって処理を行う．次に，サブスクライバー2 が新たに “topicA” をサブスクライブしたとする．その後，パブリッシャーが “topicA” に対し data2 という内容のメッセージをパブリッシュすると，そのトピックをサブスクライブしているサブスクライバー1 とサブスクライバー2 にメッセージ（data2）が配信され，それぞれのサブスクライバーがそのメッセージによって処理を行う．このように，パブリッシャーがあるトピックに対しメッセージをパブリッシ

ュすると，そのトピックをサブスクライブしているすべてのサブスクライバーに，メッセージが配信される．このようなパブリッシュ／サブスクライブ方式の特性により，メッセージをサブスクライブしてその内容を可視化するビューアーのようなコンポーネントは，複製も容易となる．また，入力された値を読み取りメッセージとしてパブリッシュするようなコンポーネントも，異なるトピックを割り当てることで追加することも容易となる．

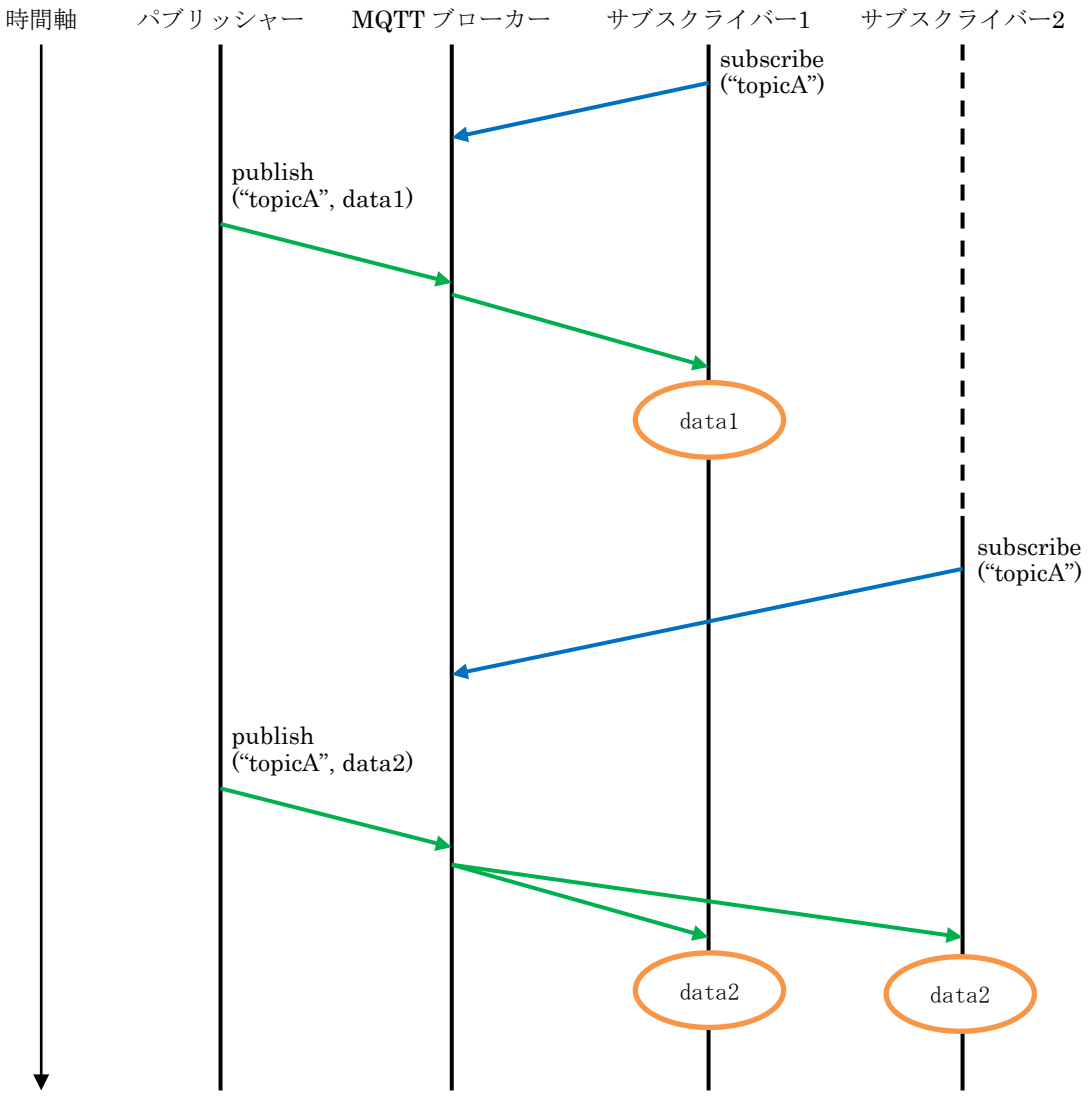


図 3-1　MQTT 通信例

MQTT メッセージで交換されるデータフォーマットとして利用する JSON(RFC8259, 2017)は，極めてシンプルなデータ構造のテキストフォーマットの一つである．JSON は，文字列，真偽値，数値，null 値，オブジェクト（名前と値のペア），配列を値として記述できる．JSON のオブジェクトの記述例を図 3-2 に示す．この例のようにデータを構成することで，その内容は極めて説明的なドキュメントとなる．

```
{
    “time”：123456789,
    “state”：“状態 1”,
    “start_time”：123455560,
    “end_time”：123456789,
    “duration”：1229
}
```

図 3-2　JSON オブジェクトの記述例

マイクロサービスアーキテクチャ(Balalaie et al., 2016; Lewis & Fowler, 2014; Richardson, 2019)は，ソフトウェア設計技法の一つとして知られている．マイクロサービスの基本的な概念は，小規模で独立した複数のサービスを結合し，ソフトウェアアプリケーションを構成する手法である．マイクロサービスは，何らかの有用な機能を実装し，スタンドアロンの個別にデプロイできるソフトウェアコンポーネントである．コンポーネントは，外部と連携するための API を持ち，その API のみで連携する．API のみでの連携とすることで疎結合となり，非同期通信によってコンポーネント間の依存を小さくする．データイベントも API の一種とみなされている．マイクロサービスアーキテクチャを用いてソフトウェアを構成する際の通信基盤には，より軽量なものを利用することが前提とされている．マイクロサービスアーキテクチャの設計概念では，データベースもサービス内でローカルに持つこととされている．より大きなアプリケーションをマイクロサービスで構成する場合，分散型のアーキテクチャであるが故の複雑さなどが課題になるとの指摘もある．

データ取得装置は，工作機械や HID（Human Interface Device），センサー等から直接データを取得し，その情報を JSON 形式でパブリッシュする装置として構成する（図 3-3）．基本的に，取得したデータをパブリッシュするものとし，モジュールとして独立して実行可能なものとする．この想定においては，疎結合型のコンポーネントであり，故障しても同一の機能を実装する装置でリプレースすればよく，機能変更の場合も，他のコンポーネントへの影響は，データの内容のみとなる．

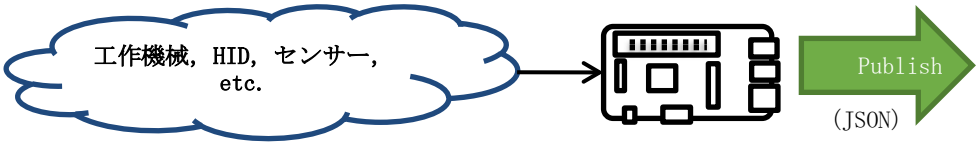
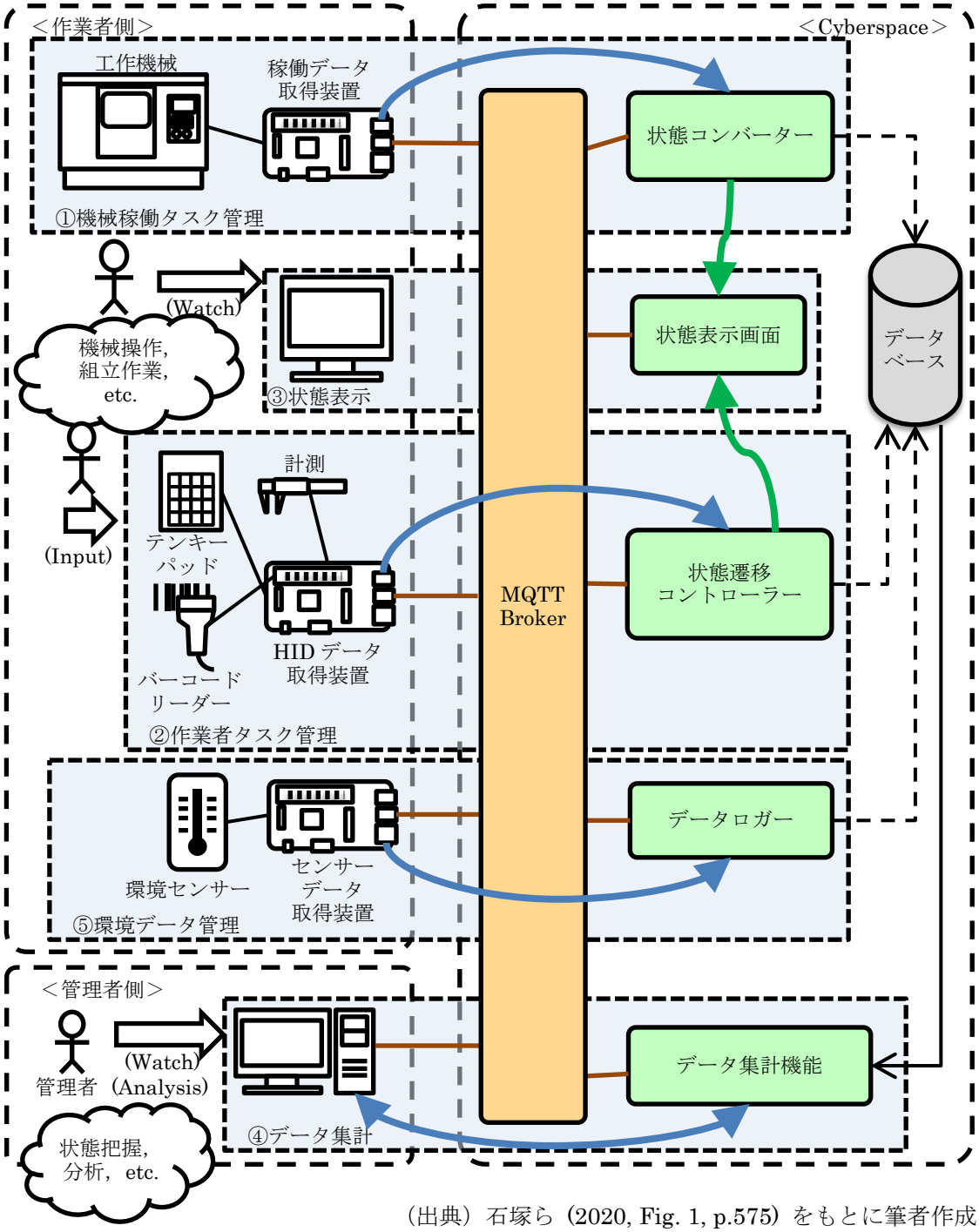


図 3-3　データ取得装置（モジュール）の基本構成

状態遷移や状態変換，状態表示画面などの機能は，ソフトウェアコンポーネントとして構成する．基本的に，サブスクライブしたデータを，機能に応じて処理し，その結果をパブリッシュするものとする．マイクロサービスとしてデータイベントのみで他のコンポーネン

トと連携する想定とする．この想定においては，独立して実行可能であり，疎結合型のコンポーネントであるといえる．機能変更の場合も，他コンポーネントへの影響は，データの内容のみとなる．データ交換を行うコンポーネント間のみでデータの内容を取り決めれば良く，自身ともう一方との対応のみで影響範囲を局所化できる．データベースへの書き込み，または読み込みの機能を含むものも想定するが，トランザクションの必要がないデータベースの利用である限り，マイクロサービスの要件を満たすといえる．

これまで論じた基本構成に基づき，工場内での状態管理を前提とした状態管理フレームワークの CPS モデルとして，図 3-4 のような構成を想定する．この CPS モデルでは，機械稼働タスク管理 (図 3-4 の①)，作業タスク管理 (図 3-4 の②)，状態表示 (図 3-4 の③)，データ集計（図 3-4 の④)，環境データ管理（図 3-4 の⑤）という 5 の機能を想定した．このうち，状態コンバーター，状態遷移コントローラー，状態表示画面，データ集計機能，データロガーは，ソフトウェアコンポーネントである．機械稼働タスク管理（図 3-4 の①）では，工作機械専用の稼働データ取得装置によって稼働状況データが取得され，そのデータによって状態コンバーターがドメイン概念に基づく状態に変換する．変換された状態データは，データベースに保存され，状態表示画面にも配信される．作業タスク管理（図 3-4 の②）では，作業の進捗状況に応じてバーコードリーダーやテンキーパッド，計測装置などの HID を用いて作業進捗を入力する．入力されたデータを状態遷移コントローラーが受け取り，そのデータによって状態遷移が行われる．状態遷移によって出力された状態は，データベースに保存され，状態表示画面にも配信される．状態表示（図 3-4 の③）では，配信された状態データを可視化する．データ集計（図 3-4 の④）では，データベースに蓄積されたデータを取得し，分析を行う．さらに，環境データ管理（図 3-4 の⑤）では，環境センサーなどの計測データを取得し，そのデータはデータロガーによってデータベースに保存される．状態管理フレームワークは，パブリッシュ／サブスクライブ型の軽量な通信プロトコルである MQTT を用いる．MQTT を介して交換されるメッセージは，JSON フォーマットのテキストドキュメントとする．次に，稼働データ取得装置や HID データを取得するデータ取得装置は，データ取得というサービスを提供するモジュールとする．基本的に，データ取得のみを行い，取得したデータをイベントとしてパブリッシュする．このような設計とすることで，データ取得モジュールは独立であり，単独でデプロイできる．変更や交換も独立して行える．従って，データ取得装置はマイクロサービスであり，局所的に導入でき疎結合型であるといえる．



（出典）石塚ら（2020, Fig. 1, p.575）をもとに筆者作成

図 3-4　状態管理フレームワークの CPS モデル

このような CPS モデルとすることで，モジュール単位や機能単位で独立に実行できるといえる．この想定によれば，局所的導入も可能で，マイクロサービス単位での機能変更やサービスの追加といった進化も可能であるといえる．

第4章 実証モデルの構築と結果の検証

本章では，第 3 章で示した状態管理フレームワークの CPS モデルに準じた実証モデルを構築し，実在工場での実証内容とその結果について説明する．

4.1 想定する実証環境

実証は，IoT 利活用に関する研究に協力頂いている，自社製品を製造する中小規模工場において，当該工場の承諾のもと実施した．作業者タスクの状態管理として，組立工程や工作機械による加工などの人間が行う作業における利用を想定した．また，機械稼働タスクの状態管理として，当該工場で稼働中のファナック製ロボドリル（プログラムにより自動工具交換可能な CNC 工作機械）を対象とした．

4.2 作業者タスク状態管理実証モデル構築と検証

4.2.1 作業者の活動に付加される情報の検討

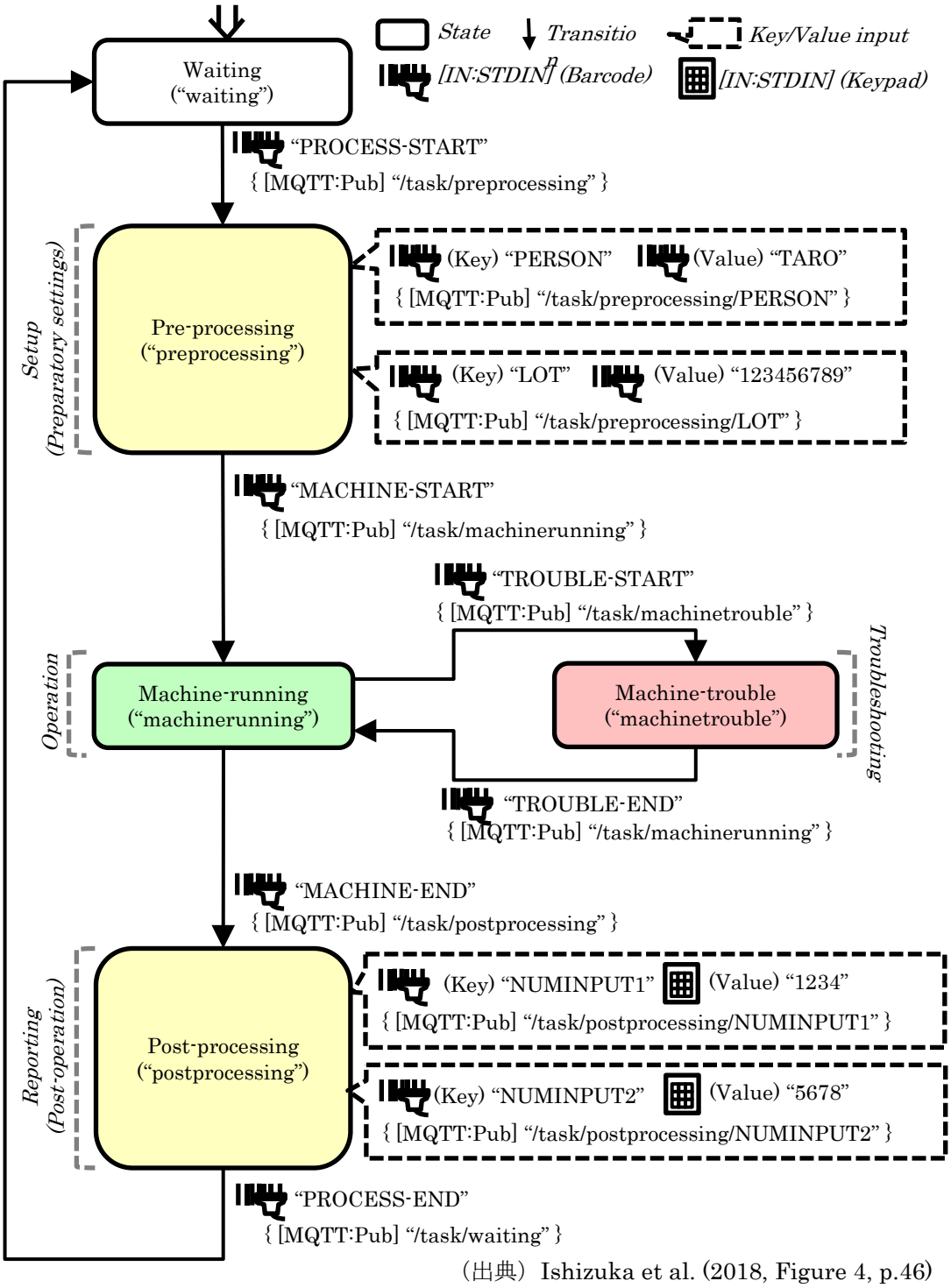
作業者タスクの作業遂行状態を取得するための状態遷移コントローラーは，主に人間の活動を捕捉する目的で用いられることを想定している．2.3 節で論じた通り，作業者の作業状況を捕捉する状態遷移モデルとしては，規定された状態遷移の定義によって，タスクを構成する状態の遂行を正しく把握することが主眼にある．状態遷移を行うためには，状態を遷移させる入力が必要となる．この入力を行う装置として，バーコードやテンキーパッドなどの HID を想定している．HID は，通常 USB で PC に接続され，キーボードの一種として扱われる．同一の PC に異なる種類の HID を接続しても，PC が認識する限りにおいて，キー入力として取り出すことができる．バーコードであれば，コード化された文字列であり，テンキーパッドであれば Enter キーが押されるまでに入力された一連の数字である．このような一組の文字列を，状態を遷移させる 1 入力記号として扱う．以降，この入力記号をトリガーと呼ぶこととする．トリガーと現在の状態により，遷移先状態が決定する．一方，作業者の作業状況を管理する上では，作業者の情報や，製番やロットなどの管理番号，作業完了時の良品数や測定データなど，作業者の作業状況に応じて発生する情報もある．製造現場では，一つの作業現場で常に同一の作業者が作業を行うとは限らない．HID のようなデバイスとそのデータ取得モジュール等を，作業者が異なる現場へ持ち運ぶということも現実的ではない．もし持ち運びが可能だとしても，どの作業現場かという情報が付加情報として必要になる．重点的にマネジメントを行う現場に局所的に導入することを考慮しても，HID とそのデータ取得モジュールは常設型として想定することが最も適当といえる．複数の作業者が交代で作業を行う現場であれば，待機中，作業準備中，主作業中，トラブル対応中，終了作業中という作業状態において，次のようなシナリオが想定できる．

- 待機中は初期状態
- 作業者の作業開始
- 待機中に，作業準備中へ遷移するトリガーを入力し，作業準備中へ移行
- 作業準備中は，オペレーターが自身の ID とロット番号を入力
- 作業準備が完了したとき，主作業中へ遷移するトリガーを入力し，主作業中へ移行
- 主作業中に何らかのトラブルが発生した場合は，トラブル対応中へ遷移するトリガーを入力し，トラブル対応中へ移行
- トラブル対応中にトラブルが解消されれば，主作業中へ遷移するトリガーを入力し，主作業中へ移行
- 主作業中にその作業が終了した後，終了作業中へ遷移するトリガーを入力し，終了作業中へ移行
- 終了作業中に，完成数などの作業結果を入力
- 作業結果入力完了後，待機中へ遷移するトリガーを入力し，待機中へ移行
- 作業者の作業は終了

ここで，オペレーターID，ロット番号，状態を遷移させるトリガーといった入力は予め決められた値であり，このような固定値の入力にはバーコードを用いる．また，完成数などの可変値は数値としてテンキーで入力する．このような要件実現のため，キーと値のペアを入力する可変値入力開始のトリガーを指定可能とする．可変値入力が開始されると，そのあとの入力と直前の入力を組み合わせ，キーと値のペアとして扱う．これは，ラベル付けされた可変値として入力を扱えることを意味する．

4.2.2 状態遷移図の記述

前項で示したシナリオを記述した状態遷移図の記述例を図 4-1 に示す．この状態遷移図では，記号を用いてバーコード入力とテンキー入力であることを示している．また，矢印が状態遷移の辺であり，中括弧｛｝で囲まれたものが出力に相当するアクションである．｛[MQTT:Pub] *topic* ｝は，アクションとして MQTT メッセージを *topic* に出力することを意味する．さらに，「待機中（Waiting)」という状態を設け，状態遷移が循環するようにしている．現状態が「待機中（Waiting)」のときに“PROCESS-START”という文字列をバーコードで入力すると，遷移先の状態をトピック“/task/preprocessing”にパブリッシュし，「作業準備（Pre-processing)」状態に遷移する．現状態が「作業準備（Pre-processing)」のとき“MACHINE-START”という文字列をバーコードから入力すると，遷移先の状態をトピック“/task/machinerunning”にパブリッシュし，「実作業（Machine-running)」状態に遷移する．現状態が「作業準備（Pre-processing)」のとき“PERSON”もしくは“LOT”という文字列をバーコードから入力すると，そのあとに入力された値と合わせ，キーと値のペアとして，アクションに指定されているトピックにパブリッシュする．



（出典）Ishizuka et al. (2018, Figure 4, p.46)

図 4-1　作業者の活動に付加される情報取得も含めたシナリオの状態遷移図例

4.2.3 状態遷移コントローラー用 DSL の設計と実装

マネジメント対象の作業タスクにおいて，状態遷移が規定され状態遷移図として作成したならば，その状態遷移定義を状態遷移コントローラーに反映する方法が必要となる．状態遷移モデルは順序機械の定義に従うので，状態の有限集合，入力記号の有限集合，出力の有限集合，状態遷移関数，出力関数，初期状態をパラメーターとして与えれば良い．さらに，状態遷移関数は状態と入力記号のすべての組み合わせに対する遷移先状態の定義であり，出力関数は状態と入力記号のすべての組み合わせに対する出力の定義である．図 4-1 に示した状態遷移図の定義を転写可能とする上で，作業者の活動に付加される情報を取得するためには，入力記号の有限集合に含まれない入力記号をある状態において受け入れ，後続する入力値と合わせ，キーと値のペアとして出力できることが必要となる．また，状態と入力に応じた遷移先状態も，遷移が発生するもののみ記述すれば良い．このような記述を可能とし，その内容を状態遷移コントローラーに反映可能な領域固有言語（Domain Specific Language：DSL）を設計し実装する．

DSL における基本的な状態記述方法として，取りうるすべての状態を列挙し，各状態において，遷移条件に相当する入力値と遷移先，遷移発生時に行うアクションを記述可能とする．また，キーと値のペアを入力するための可変値入力開始のトリガーを記述可能とする．このようなパラメーターの関係は，階層構造として記述可能である．そこで，DSL の基本構造を JSON フォーマットとし，出力に相当するアクションを記述可能とする．このような記述様式によれば，DSL は設定ファイルのような極めて宣言的なスクリプトといえる．

これまで論じたような設計で，実証用の DSL を実装した．図 4-1 の状態遷移図の通りに振る舞う DSL スクリプトの一部を，DSL 記述例として図 4-2 に示す．この DSL 記述例には，状態遷移の制御に関するほとんどの記述型式が含まれている．DSL 記述例において，“States”ブロックが，状態遷移の定義となっている．これは名前と値のペアを要素とする JSON オブジェクトである．ここでは，名前と値のペアの名前を JSON 名と呼び，名前と値のペアの値を JSON 値と呼ぶ．“States”ブロックには，タスクの取りうるすべての状態を JSON 名として記述する．JSON 値は，状態の定義として JSON オブジェクトを指定するブロックの要素であり，状態定義ブロックと呼ぶ．状態定義ブロックでは，“type”，“key_patterns”，“Transitions”を記述する．“type”は“keyvalue”または“trans_only”を記述する．“keyvalue”は，当該状態でのキーと値のペアの入力を許可することを意味し，“trans_only”は，キーと値のペアの入力がないことを意味する．“keyvalue”が指定されている場合，“key_patterns”は可変値入力開始のトリガーとする文字列の配列である．“Transitions”ブロックは，遷移ブロックの配列であり，遷移ブロックは図 4-2 のように記述する．遷移ブロック内の“apply_pattern”には，遷移のトリガーとする文字列を記述する．また，“next”には，遷移先状態名を記述する．

```
"Task" : {
  "KeyValueStateMachine" : {
    "topic_prefix" : "/task",
      "state_payload" : {
        "type" : "json-string", "charset" : "UTF-8",
        "value" : "{¥"time¥":¥"${datetime}¥"}"
      },
      "keyvalue_payload" : {
        "type" : "json-string", "charset" : "UTF-8",
        "value"                                                                                                    :
"{¥"time¥":¥"${datetime}¥",¥"key¥":¥"${key}¥",¥"value¥":¥"${value}¥"}"
      },
      "start_state" : "waiting",
      "States" : {
        "postprocessing" : {
          "type" : "keyvalue",
          "key_patterns" : ["NUMINPUT1", "NUMINPUT2"],
          "Transitions" : [
            {
              "trigger" : { "stdin.line" : {
                "apply_pattern" : "PROCESS-END"
              }},
              "next" : "waiting"
            }
          ]
        },
        "machinerunning" : {
          "type" : "trans_only",
          "Transitions" : [
            {
              "trigger" : { "stdin.line" : {
                "apply_pattern" : "TROUBLE-START"
              }},
              "next" : "machinetrouble"
            },
            {
              "trigger" : { "stdin.line" : {
                "apply_pattern" : "MACHINE-END"
              }},
              "next" : "postprocessing"
            }
          ]
        },
      }}}}
```

（出典）Ishizuka et al. (2018, Figure 6, p.47)

図 4-2　DSL 記述例

4.2.4 状態遷移コントローラーの検証

状態遷移コントローラーの検証として，4.2.1 項で示したシナリオを用いて，状態の取得と可視化が可能かを検証した．状態遷移コントローラーの検証環境として，「トランクキット」を利用した．トランクキットは，東京工業大学出口研究室で開発された，IoT の導入キットである（図 4-3）．

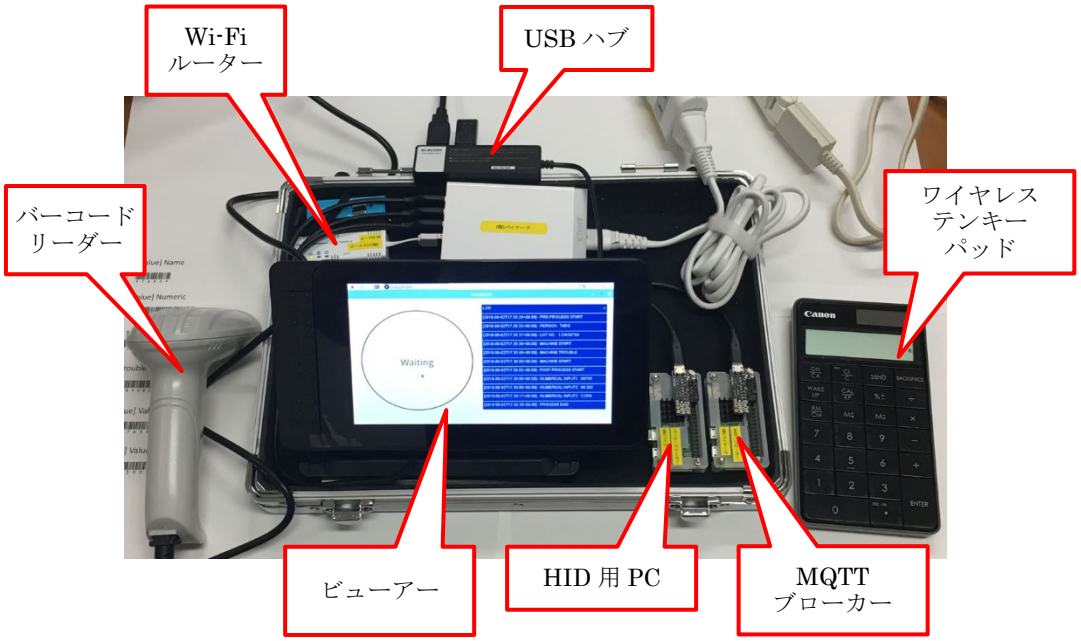


図 4-3　トランクキット

トランクキットには，バーコードリーダーやテンキーパッドの 2 つの HID 対応機器があり，HID 用 PC に接続されている．Wi-Fi ルーター，MQTT ブローカー，液晶タッチパネルを装備したビューアーPC とあわせ，一つのケース内にパッキングされている．この HID 用 PC 上で，状態遷移コントローラーを実行する．トランクキットの HID 用 PC に対し，図 4-1 に示す状態遷移遂行のための DSL スクリプトを実行する状態遷移コントローラーを組み込み，実行した．トランクキットのビューアーは，状態遷移コントローラーが出力する MQTT メッセージに対応して表示が切り替わるものとなっている．なお，DSL の設計・実装，および DSL 実行のための状態遷移コントローラーの実装・導入は筆者が行い，ビューアーに表示される状態表示は MQTT メッセージの内容によって表示内容が変化するよう，IT ベンチャー企業により実装されたものである．シナリオの実行において，次に示す順序でバーコードとテンキーパッドの入力を行った．

(1)　“PROCESS-START” をバーコードで入力
(2)　“PERSON” をバーコードで入力
(3)　“TARO” をバーコードで入力

(4) “LOT”をバーコードで入力
(5) “123456789”をバーコードで入力
(6) “MACHINE-START”をバーコードで入力
(7) “TROUBLE-START”をバーコードで入力
(8) “TROUBLE-END”をバーコードで入力
(9) “MACHINE-END”をバーコードで入力
(10) “NUMINPUT1”をバーコードで入力
(11) “1234”をテンキーパッドで入力
(12) “NUMINPUT2”をバーコードで入力
(13) “5678”をテンキーパッドで入力
(14) “PROCESS-END”をバーコードで入力

この入力によって得られる MQTT メッセージを別の PC 上でサブスクライブしその内容を確認した．シナリオの実行によって収集した MQTT メッセージの内容を表 4-1 に示す．この表の［状態］を現在の状態とし，そのときの［入力］によって得られた結果が［取得した MQTT メッセージ］である．また，［取得した MQTT メッセージ］の“T”は Topic，“P”は Payload の値であることを意味する．

表 4-1　状態遷移コントローラーのシナリオ実行結果としての MQTT メッセージ

状態	入力	取得した MQTT メッセージ		
Waiting	"PROCESS-START"	(a)	T	/task/preprocessing
			P	{"time":"2018-06-02T23:02:46+09:00"}
Pre-processing	"PERSON", "TARO"	(b)	T	/task/preprocessing/PERSON
			P	{"time":"2018-06-02T23:03:29+09:00", "key":"PERSON","value":"TARO"}
	"LOT", "123456789"	(c)	T	/task/preprocessing/LOT
			P	{"time":"2018-06-02T23:04:14+09:00", "key":"LOT","value":"123456789"}
	"MACHINE-START"	(d)	T	/task/machinerunning
			P	{"time":"2018-06-02T23:05:00+09:00"}
Machine-running	"TROUBLE-START"	(e)	T	/task/machinetrouble
			P	{"time":"2018-06-02T23:05:40+09:00"}
Machine-trouble	"TROUBLE-END"	(f)	T	/task/machinerunning
			P	{"time":"2018-06-02T23:07:51+09:00"}
Machine-running	"MACHINE-END"	(g)	T	/task/postprocessing
			P	{"time":"2018-06-02T23:08:27+09:00"}
Post-processing	"NUMINPUT1", "1234"	(h)	T	/task/postprocessing/NUMINPUT1
			P	{"time":"2018-06-02T23:09:17+09:00", "key":"NUMINPUT1","value":"1234"}
	"NUMINPUT2", "5678"	(i)	T	/task/postprocessing/NUMINPUT2
			P	{"time":"2018-06-02T23:10:07+09:00", "key":"NUMINPUT2","value":"5678"}
	"PROCESS-END"	(j)	T	/task/waiting
			P	{"time":"2018-06-02T23:10:45+09:00"}

（出典）Ishizuka et al. (2018, Table Ⅰ, p.49)

表 4-1 の MQTT メッセージ (a～j) に対応するビューアーの表示内容は，図 4-4 の a～j の通りとなった．なお，ビューアーの表示においては，状態「Pre-processing」と状態「Post-processing」の表示名を“Process in progress”としている．これらの結果から，状態遷移図から DSL を記述し，状態遷移コントローラーによってその DSL を実行することで，状態遷移によるタスクの遂行状況としての状態収集と可視化が実現可能であることを確認できた．

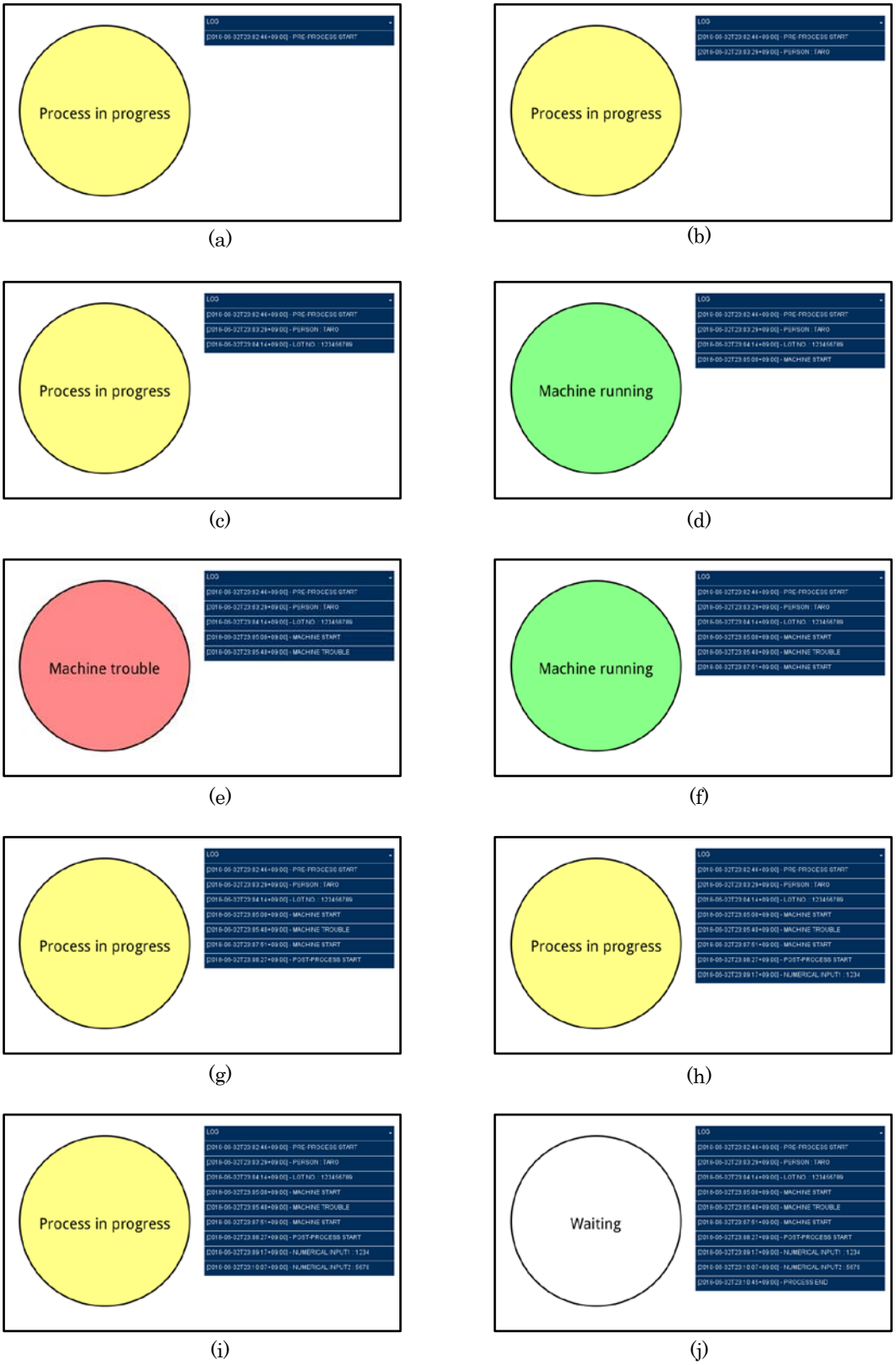
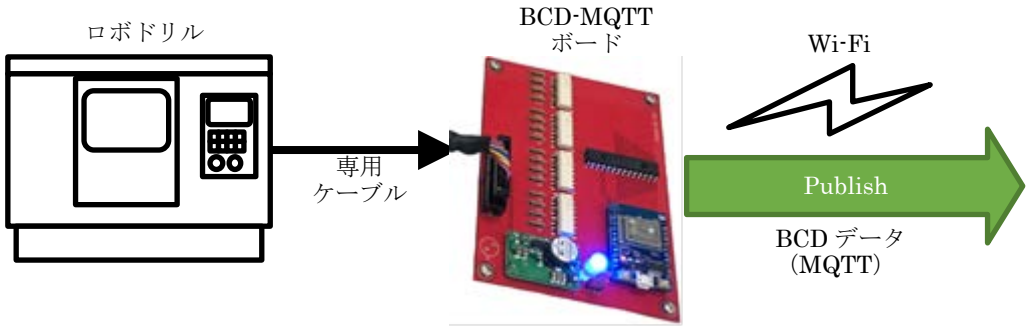


図 4-4　状態遷移コントローラーのシナリオ実行結果としての状態表示

4.3 機械稼働タスク状態管理実証モデルの構築と検証

4.3.1 実証において使用した機器－BCD-MQTT ボード

ロボドリルの稼働状況を取得するための専用データ取得装置として，東京工業大学出口研究室と IT ベンチャー企業によって開発された 16ch の 24V・5V 信号変換基板（以降，「BCD-MQTT ボード」と呼ぶ）を利用した（図 4-5）.



（出典）石塚ら（2020, Fig. 2, p.576）

図 4-5　ロボドリル専用の稼働データ取得装置

もともと工作機械などの装置に組み込まれている接点から BCD（Binary Coded Decimal：2 進化 10 進数）を取得する目的で開発されたものが基礎となっており，最大 16ch の 24V 信号を，16-bit のデジタルデータ（以降，「BCD 値」と呼ぶ）に変換した後，タイムスタンプと BCD 値を JSON 形式のテキストデータ（以降，「BCD データ」と呼ぶ）として MQTT ブローカーへパブリッシュすることが可能なデバイスである. ロボドリルと BCD-MQTT ボードは，専用のケーブルによって接続される．ロボドリルから BCD-MQTT ボードに出力される 16ch の信号は，ロボドリルのフロントパネルから任意に設定可能であり，当該工場の技術者によって設定されている.

ロボドリルの信号を取得し，BCD-MQTT ボードが出力（パブリッシュ）する BCD データ（JSON フォーマット）を図 4-6 に示す.

図 4-6　BCD データ（JSON）の構造

なお，BCD-MQTT ボードは，ロボドリルから供給される電力により稼働する．BCD-MQTT ボード稼働中は，定期的に現在の BCD 値を含む BCD データをパブリッシュする. ロボドリルの電源が OFF のとき，BCD-MQTT ボードは稼働していないため，その期間は

BCD データがパブリッシュされないことになる．このような振る舞いによって，BCD データを利用するソフトウェアコンポーネントは，一定時間以上 BCD データを受け取れなかった場合にロボドリルの電源が OFF であると判断することも可能である．ただし，通信環境のトラブルによって通信が行えないときなど，電源 OFF と判断できない状況もあるため，電源 OFF か通信トラブルかの判断を可能とするような方法については，今後の検討課題となっている．

4.3.2 実証モデル基本構成

機械稼働タスク状態管理の実証モデルでは，BCD-MQTT ボードが出力（パブリッシュ）した BCD データを，状態コンバーターが取得（サブスクライブ）する．状態コンバーターは，変換定義に従い BCD データを状態に変換する．状態への変換後，状態コンバーターはリアルタイム状態提示用データを出力（パブリッシュ）し，状態データを MongoDB に保存する．リアルタイム状態提示用データと保存用状態データは，ともに JSON フォーマットとしている．MongoDB は，JSON 形式のデータをドキュメントとしてそのまま保存可能な，NoSQL 型のデータベースである(MongoDB, 2020)．リアルタイム状態提示画面が提示用データに基づき，現在の状態を表示する．また，日次状態集計チャートは，MongoDB に蓄積された状態データから，1 日分のデータでチャートを生成し表示する．これは一種の分析機能といえる．また，オープンソースソフトウェアの Mongo-Express を用いて，MongoDB の蓄積データを閲覧・取得する機能を提供する．この基本構成を図 4-7 に示す．

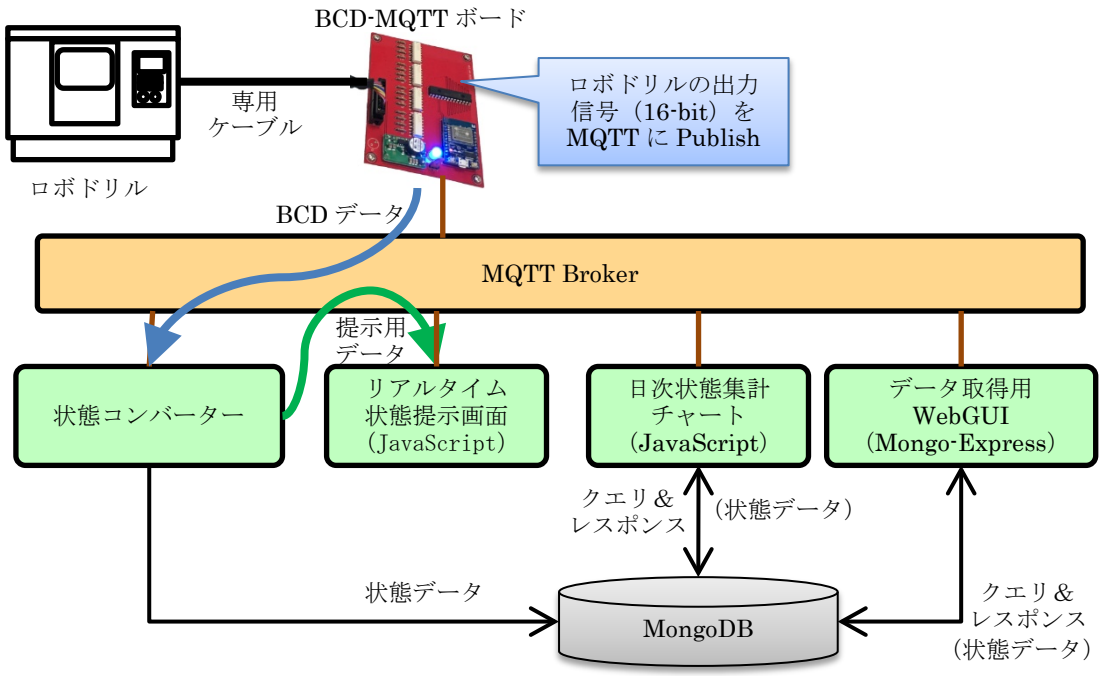
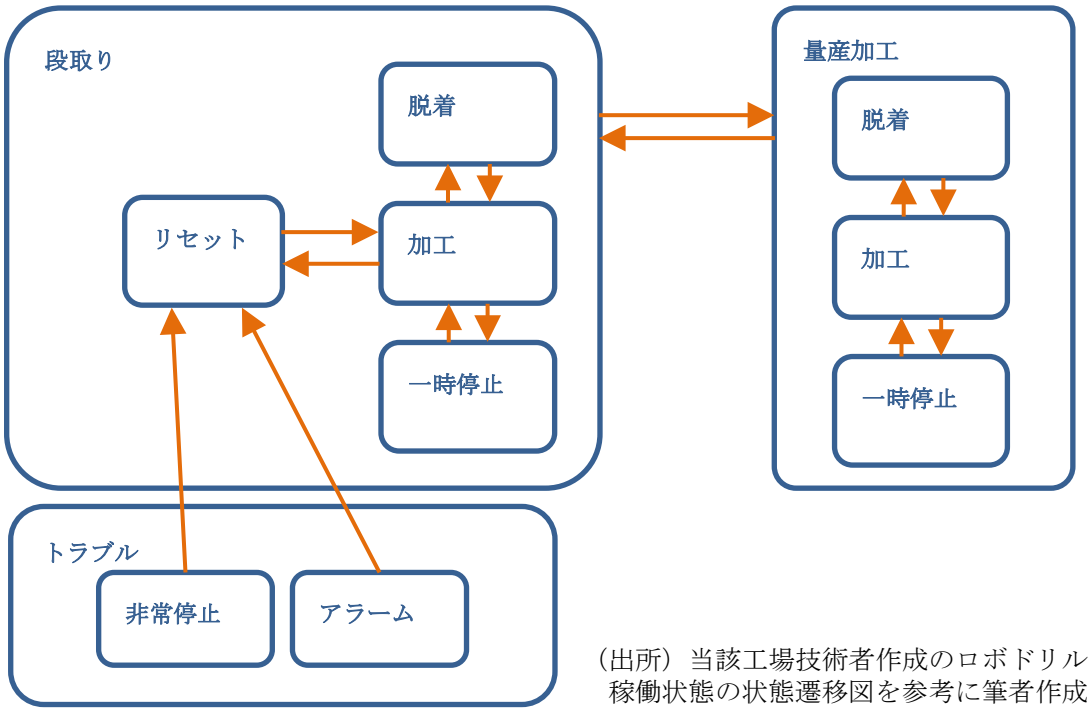


図 4-7　機械稼働タスク状態管理実証モデルの基本構成

4.3.3 ロボドリル専用状態定義の設計

当該工場技術者によってロボドリルから出力される 16ch の信号が決定され，その信号の組み合わせから，ロボドリルの状態が図 4-8 のような状態遷移図として当該工場技術者によって設計された．実証モデルを構築するにあたり，当該工場技術者が設計した各状態の信号組み合わせをもとに，信号と状態の対応表としてまとめたものを図 4-10 に示す．この対応表は，当該工場技術者によって確認・承認され，ロボドリルの状態定義（初版）となった．



（出所）当該工場技術者作成のロボドリル稼働状態の状態遷移図を参考に筆者作成

図 4-8　工場技術者により設計されたロボドリルの稼働状態を示す状態遷移図

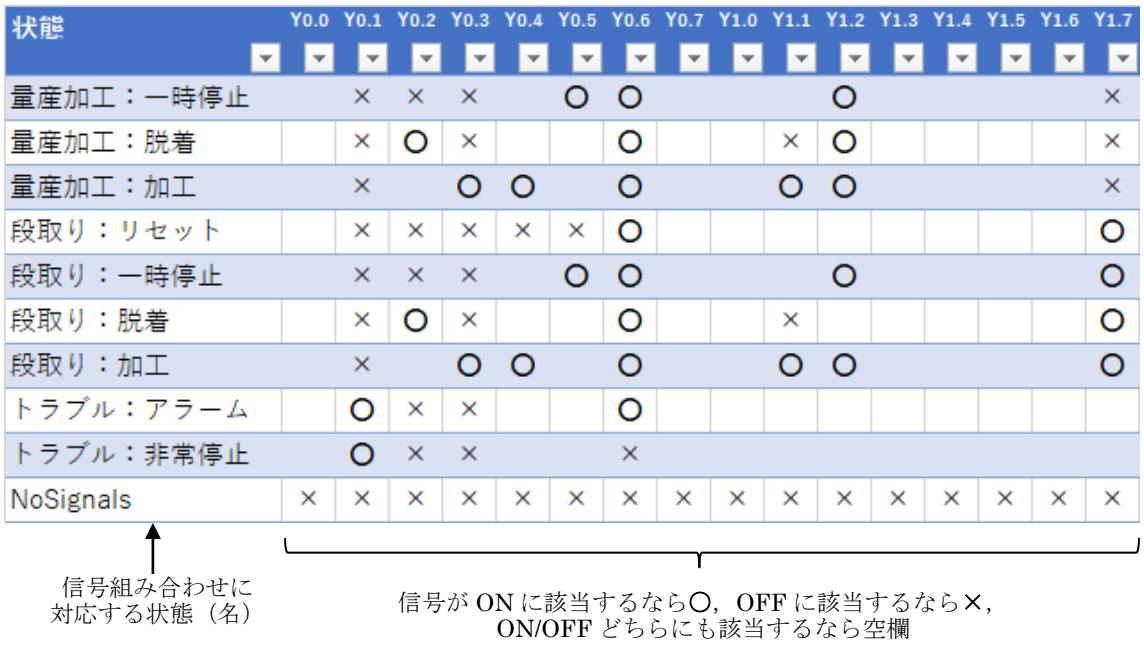
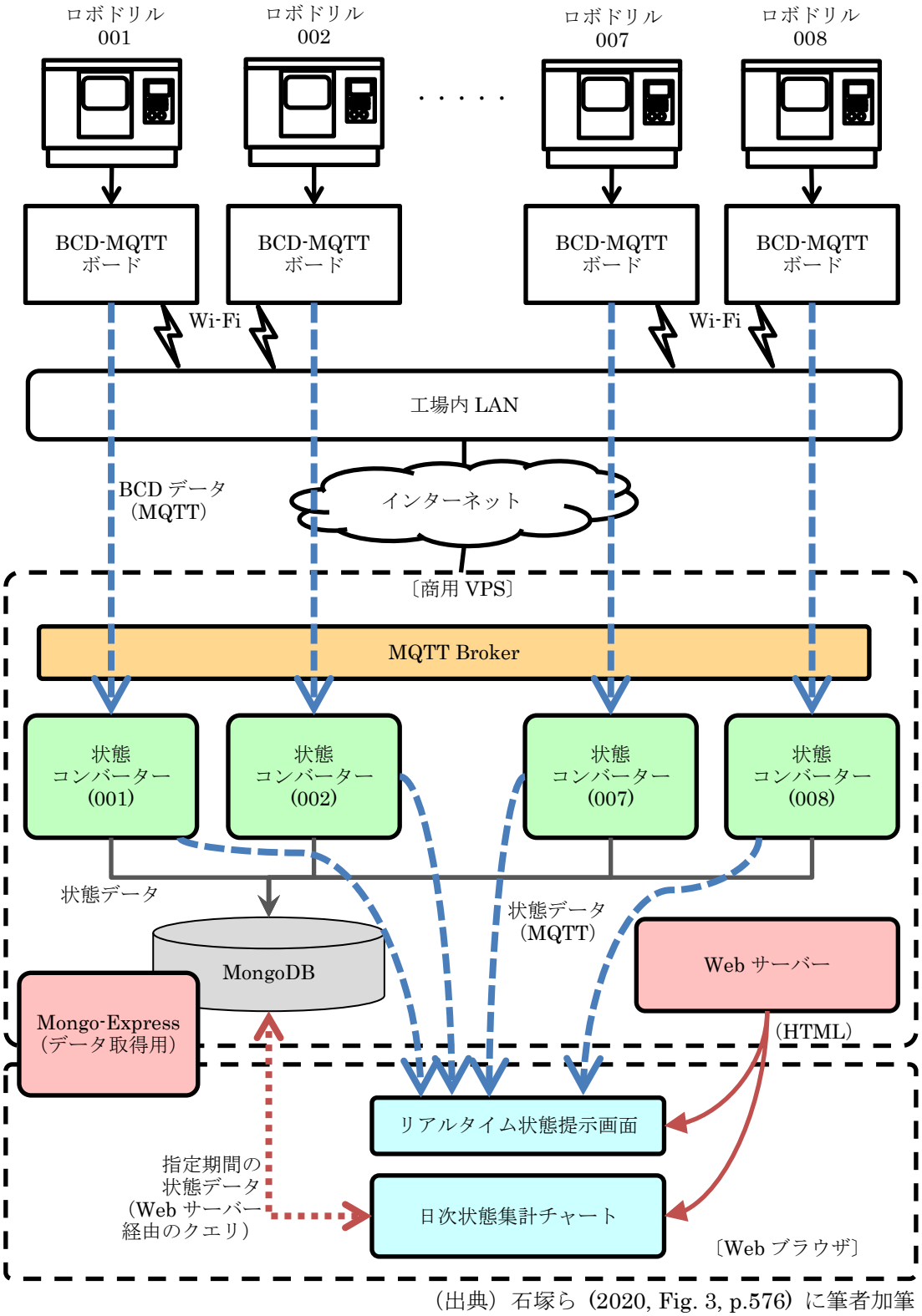


図 4-9　ロボドリルの出力信号組み合わせによる状態定義（初版）

4.3.4 状態コンバーター用変換定義ファイルの設計

状態定義（初版）で規定された通り，ロボドリル稼働状況をドメイン概念に基づく状態に変換することを前提として，状態コンバーターを構築する．状態コンバーターは，BCD-MQTT ボードから MQTT 経由で BCD データを受け取り，当該工場技術者作成の定義に基づいて機械稼働タスクの状態に変換する機能を提供するソフトウェアコンポーネントである．状態への変換後，その状態をリアルタイム状態提示画面がサブスクライブするトピックへパブリッシュし，MongoDB への保存も行う．MongoDB へ保存されたデータは，日次状態集計チャートに利用される．この状態コンバーターは，機械稼働タスクの状態とそれに対応する BCD 値のビットパターンを，JSON フォーマットの状態定義ファイルとして指定可能となるよう設計・実装した．ビットパターンにはワイルドカードによって（0 もしくは 1 どちらにも合致する）不定ビットも記述できるようにした．この状態定義ファイルには，リアルタイム状態提示画面へ表示するデータの宛先や，データ保存先のデータベースも指定可能とした．状態コンバーターでは，BCD データが状態定義に規定されたどの状態にも該当しない場合，「不明状態（@UnknownBCD）」という状態を規定している．また，BCD データが一定時間以上受け取れなかったとき，「タイムアウト（@TimedOut）」という状態を規定している．リアルタイム状態提示画面と日次状態集計チャートでは，このタイムアウト状態を電源 OFF 状態として利用している．ただし，4.3.1 項で説明した通り，厳密に電源 OFF と通信トラブルを区別する方法については今後の課題となっている．

4.3.5 機械稼働タスク状態管理の実証構成



（出典）石塚ら（2020, Fig. 3, p.576）に筆者加筆

図 4-10　機械稼働タスク状態管理の実証構成

当該工場における機械稼働タスクの状態管理実証環境を，図 4-10 に示す構成で構築した．当該工場で稼働するロボドリル 8 台に BCD-MQTT ボードを取り付け，無線 LAN によって当該工場内 LAN に接続した．状態コンバーターなどのソフトウェアコンポーネントは，商用 VPS（Virtual Private Server）上に導入した．VPS には，MQTT ブローカーとデータベース，さらにリアルタイム状態提示画面と日次状態集計チャートを Web ブラウザで閲覧可能とするための Web サーバーも導入した．MQTT ブローカーには，オープンソースソフトウェアとして公開されている Mosquitto(2018)を利用した．また，データベースには，JSON フォーマットのデータをそのまま保存できる，オープンソースライセンスで公開されている MongoDB(2020)を利用した．Web サーバーには，オープンソースソフトウェアである Apache(1997)を利用した．この実証環境のうち，状態コンバーターの実装と導入，MongoDB の導入，日次状態集計チャートの実装と導入は，筆者が行った．それ以外の環境は，IT ベンチャー企業が実装・構築・導入を行った．

4.3.6 リアルタイム状態提示画面の実装

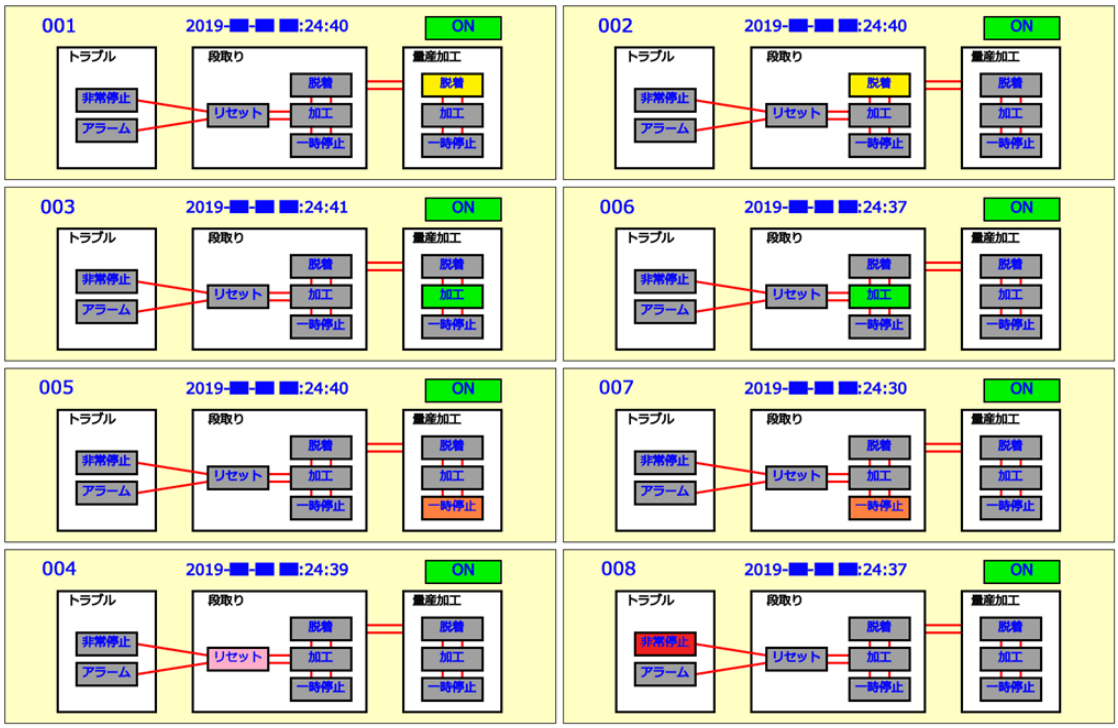
リアルタイム状態提示画面の作成には，東京工業大学出口研究室主導のもとで IT ベンチャー企業によって開発された「サーバルキャット」という Web 画面作成ツールを用い，IT ベンチャー企業によって実装・導入された．このツールにより，MQTT 経由で配信される JSON フォーマットのデータに応じて表示内容が書き換えられる Web 画面を作成できる．リアルタイム状態提示画面は，Web サーバーにより Web ブラウザに配信され，Web ブラウザ上で現在の機械稼働タスクの状態を確認できる．

4.3.7 日次状態集計チャートの実装

日次状態集計チャートは，集計対象のロボドリルと日付を指定し，当該日の機械稼働タスクの状態を集計し表示するよう実装した．状態のタイムラインと，状態別に持続時間を集計した円グラフなどが表示される．日次状態集計チャートも，Web サーバーにより Web ブラウザに配信され，Web ブラウザ上で指定日の機械稼働タスクの状態を確認できる．この画面は，オープンソースの JavaScript 用チャートライブラリを用いて，JavaScript のみで実装している．日次状態集計チャートの実装・導入は，筆者が行った．

4.3.8 稼働機械タスク状態管理の実証結果の検証

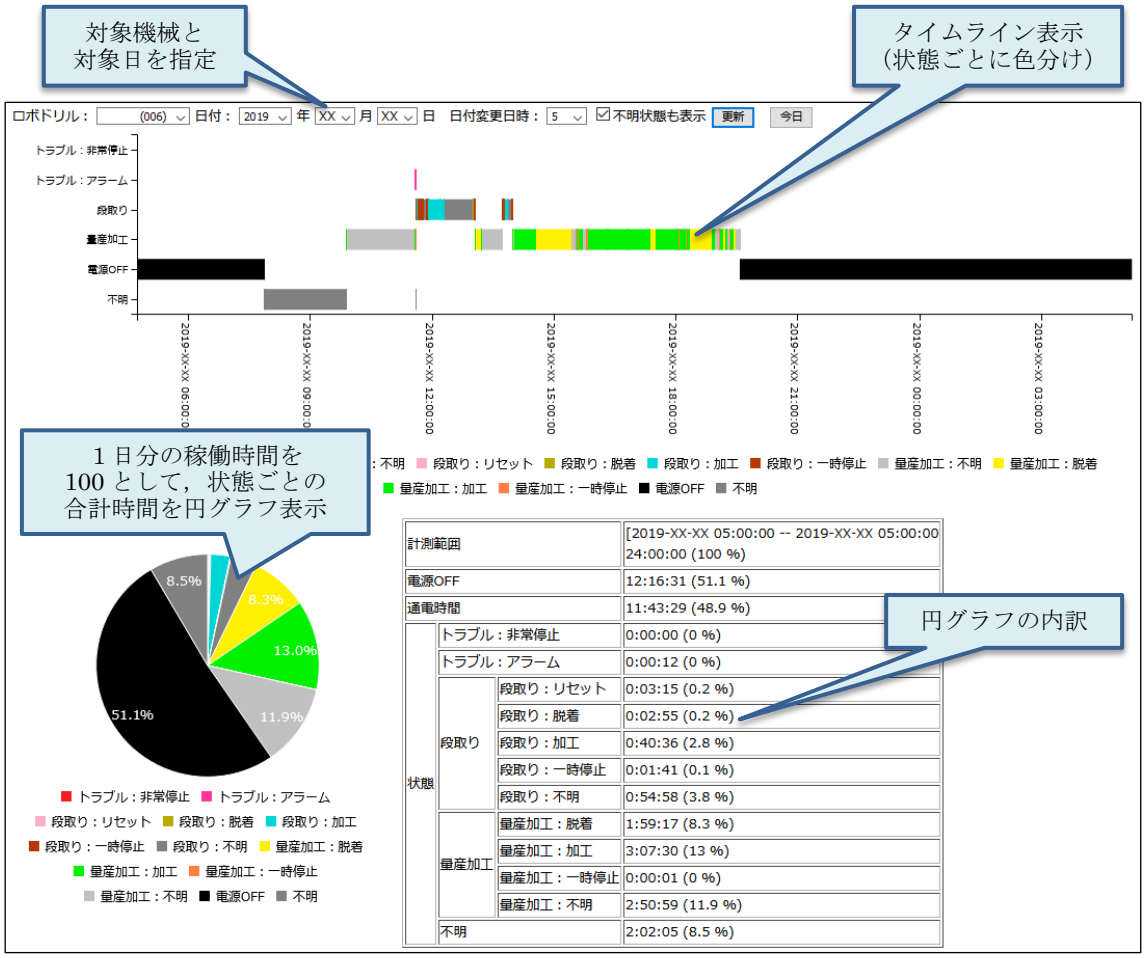
実証モデルとして構築した機械稼働タスク状態管理機能は，当該工場のロボドリルからの信号を BCD-MQTT ボードを介して状態コンバーターが状態に変換するものである．状態コンバーターは，変換後の状態を逐次 MQTT にて配信している．さらに，変換後の状態はデータベースにも記録されている．当該工場技術者が作成した変換定義の通り，BCD データが状態に変換されていることは，データベースに記録されたデータから確認した．



（出典）石塚ら（2020, Fig. 5, p.578）をもとに筆者作成

図 4-11　リアルタイム状態提示画面のロボドリルの状態

機械稼働タスクの状態の可視化となるリアルタイム状態提示画面の内容を図 4-11 に示す．この図においては，実際の表示内容からロボドリルの通称を番号に変更し，日付と時刻の一部を伏せている．このリアルタイム状態提示画面により，ドメイン概念に基づいた状態による現在のロボドリル稼働状況が得られることが確認できた．



（出典）石塚ら（2020, Fig. 6, p.578）をもとに筆者作成

図 4-12　日次状態集計チャートの表示

次に，日次状態集計チャートの内容を図 4-12 に示す．これはロボドリル（006）でのある 1 日の状態別持続時間を示している．タイムラインチャートにより，いつ，何が行われたかが一目で判別できる．また，円グラフにより，対象日の各状態の合計持続時間比率が判別可能となっている．さらに，状態別集計表に，各状態の合計持続時間が示されている．この提示画面により，ドメイン概念に基づいた状態による，過去のロボドリル稼働状況が選択的に得られることも確認できた．

4.3.9 不明状態の解消に伴う状態定義更新事例

実証開始時の状態定義（初版）によって変換され収集された機械稼働タスクの状態を確認すると，さまざまな個所で不明状態が確認できた．これは，当該工場の技術者も想定していなかった．その後，状態定義を 2 回改訂し，状態定義（第三版）にて不明状態が解消され，機械の稼働状況に応じて一意に決定した状態が得られるようになった．ここで行った手順は，次の通りである．

(1) 状態データの不明箇所の分析と状態定義の改定（当該工場技術者）
(2) 改定後の状態定義（第二版）の実証環境への反映（筆者）
(3) 状態重複の発生にともなう問題の分析と改善版定義の提案（筆者）
(4) 当該工場技術者承認を経て実証環境へ反映（当該工場技術者，筆者）

この事例は，ドメイン概念に基づいて当該工場技術者により作成された状態定義が，当該工場技術者の手で改変されたことを示している．さらに，初版から第二版への改定では，信号の組み合わせに対応する状態が増加した．これは，作業現場の進化に状態管理フレームワークの機能が追随したともいえる．実証環境への反映は筆者が行っているため，製造現場のみで完結していないが，この貴重な事例についての詳細を，次に述べる．

(1) 状態データの不明箇所の分析と状態定義の改定（当該工場技術者）

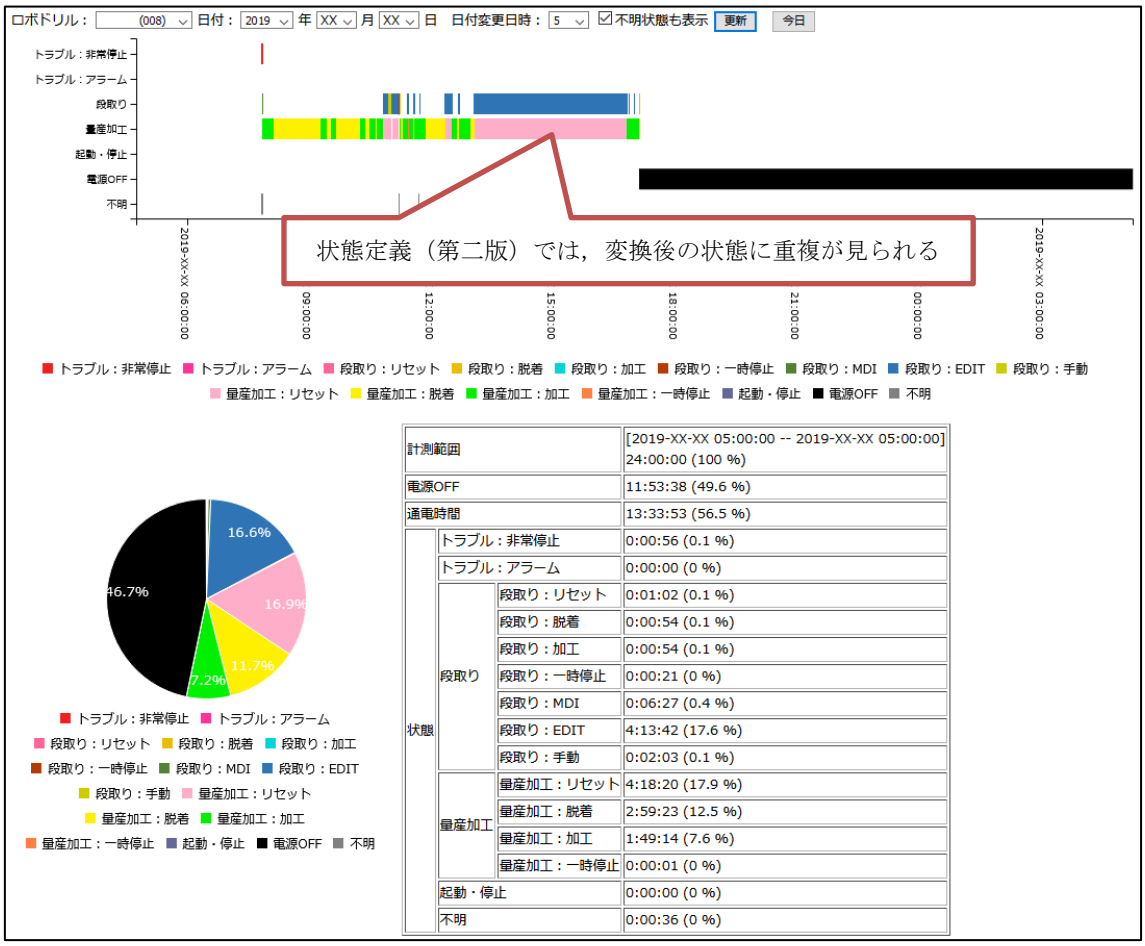
データベースには，状態コンバーターによって変換された状態の他，変換元の BCD データも記録している．このデータを利用して当該工場技術者が分析したところ，プログラムによって行われる加工作業のほかに，加工対象物の加工状況を確認するため，計測実施やステップ実行などを行っている可能性が指摘された．当該工場のドメイン概念によれば，このような作業は段取りの一種だとされた．そして，不明状態を解消するための新しい状態定義（第二版）が規定された．

(2) 改定後の状態定義（第二版）の実証環境への反映（筆者）

状態コンバーターの変更は，状態定義ファイルを新たに規定された状態定義（第二版）に差し替えるのみで完了した．

(3) 状態重複の発生にともなう問題の分析と改善版定義の提案（筆者）

状態定義（第二版）では，BCD 値に複数の状態が該当してしまうという新たな問題（図 4-13）が確認された．この問題を確認するため，状態定義を分析し，当該工場技術者によって規定された状態において，重複なく，不明状態も発生しない新しい状態定義（最新版）を作成した．この分析により，ロボドリルの内部的な振る舞いにおいて信号が切り替わるタイミングにずれがあることが判明した．これが問題をより複雑にしていた．例えば，黄ランプが点灯中に処理が切り替わり，黄ランプが消灯して緑ランプが点灯するといったような振る舞いにおいて，黄ランプと緑ランプが同時に点灯する状態は発生しない想定であったが，実際にはそのような状況もあった．このような問題について対処した状態定義（第三版）を作成し，工場技術者へ提案した．



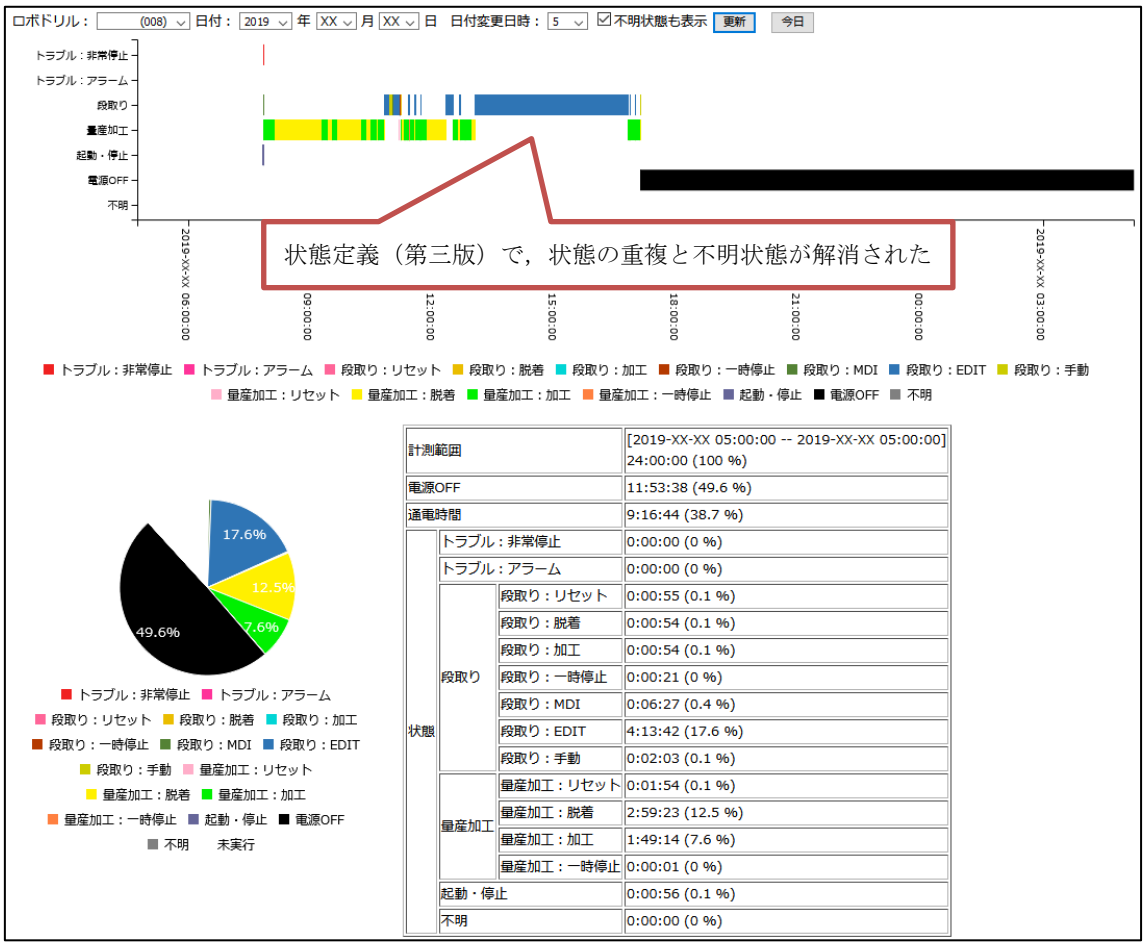
計測範囲			[2019-XX-XX 05:00:00 -- 2019-XX-XX 05:00:00] 24:00:00 (100 %)
電源OFF			11:53:38 (49.6 %)
通電時間			13:33:53 (56.5 %)
状態	トラブル：非常停止		0:00:56 (0.1 %)
	トラブル：アラーム		0:00:00 (0 %)
	段取り	段取り：リセット	0:01:02 (0.1 %)
		段取り：脱着	0:00:54 (0.1 %)
		段取り：加工	0:00:54 (0.1 %)
		段取り：一時停止	0:00:21 (0 %)
		段取り：MDI	0:06:27 (0.4 %)
		段取り：EDIT	4:13:42 (17.6 %)
		段取り：手動	0:02:03 (0.1 %)
	量産加工	量産加工：リセット	4:18:20 (17.9 %)
		量産加工：脱着	2:59:23 (12.5 %)
		量産加工：加工	1:49:14 (7.6 %)
		量産加工：一時停止	0:00:01 (0 %)
	起動・停止		0:00:00 (0 %)
	不明		0:00:36 (0 %)

（出典）石塚ら（2020, Fig. 7, p.579）をもとに筆者作成

図 4-13　状態定義（第二版）による状態の重複

(4) 当該工場技術者承認を経て実証環境へ反映（当該工場技術者，筆者）

状態定義（第三版）は，当該工場技術者の確認を経て，承認された．図 4-13 で使用した BCD データに状態定義（第三版）を適用し，状態の重複や不明状態が解消された結果を図 4-14 に示す．



計測範囲			[2019-XX-XX 05:00:00 -- 2019-XX-XX 05:00:00] 24:00:00 (100 %)
電源OFF			11:53:38 (49.6 %)
通電時間			9:16:44 (38.7 %)
状態	トラブル：非常停止		0:00:00 (0 %)
	トラブル：アラーム		0:00:00 (0 %)
	段取り	段取り：リセット	0:00:55 (0.1 %)
		段取り：脱着	0:00:54 (0.1 %)
		段取り：加工	0:00:54 (0.1 %)
		段取り：一時停止	0:00:21 (0 %)
		段取り：MDI	0:06:27 (0.4 %)
		段取り：EDIT	4:13:42 (17.6 %)
		段取り：手動	0:02:03 (0.1 %)
	量産加工	量産加工：リセット	0:01:54 (0.1 %)
		量産加工：脱着	2:59:23 (12.5 %)
		量産加工：加工	1:49:14 (7.6 %)
		量産加工：一時停止	0:00:01 (0 %)
	起動・停止		0:00:56 (0.1 %)
	不明		0:00:00 (0 %)

（出典）石塚ら（2020, Fig. 8, p.579）をもとに筆者作成

図 4-14　状態定義（第三版）による状態重複の解消

第5章 評価と考察

本章では，実在工場での実証によって収集された機械稼働タスクの状態情報をもとに，機械内部的な稼働状況との比較によって，提案する状態管理手法によって得られる状態が有用であることを論じる．さらに，実証によって可視化されたいくつかの工作機械の作業状況をもって多様な状態を含んでいることや，第 3 章で定義した状態管理フレームワークとして実現すべき要件の達成度についての考察を示す．

5.1 機械稼働タスク状態管理の評価と考察

機械稼働タスクの状態は，ドメイン概念に基づき当該工場技術者によって規定された状態である．実証の結果に基づく本研究の評価として，収集した 1 日分の機械稼働タスクの状態データから，ロボドリルに装備されている 3 色表示灯（赤・黄・緑）の点灯状況ごとの合計持続時間（表 5-1），状態定義（初版）によって変換され収集された状態ごとの合計持続時間（表 5-2），状態定義（第三版）によって変換され収集された状態ごとの合計持続時間（表 5-3）を示し，それぞれについて比較する．

表 5-1　3 色表示灯点灯色別合計持続時間

Lamp Status G:GREEN Y:YELLOW R:RED	ロボドリル別合計持続時間（秒）							
	001	002	003	004	005	006	007	008
G=ON, Y=off, R=off	59001	26880	27033	21016	52977	61502	46567	6688
G=off, Y=ON, R=off	24292	19725	21222	11081	11011	24842	12064	11078
G=off, Y=off, R=ON	0	220	0	601	0	0	0	0
G=ON, Y=ON, R=off	6	3	4	4	13	7	1	2
G=ON, Y=off, R=ON	0	0	0	0	0	0	0	0
G=off, Y=ON, R=ON	0	0	0	0	0	0	0	0
G=ON, Y=ON, R=ON	0	0	0	0	0	0	0	0
G=off, Y=off, R=off	2927	6243	7035	6899	13015	41	14170	15636
合計	86226	53071	55294	39601	77016	86392	72802	33404

（出典）石塚ら（2020, Table 1, p.580）をもとに筆者作成

表 5-2　状態定義（初版）の状態別合計時速時間

状態名（初版）	ロボドリル別合計持続時間（秒）							
	001	002	003	004	005	006	007	008
@TimedOut	0	0	4029	84	0	0	0	0
@UnknownBCD	3509	6922	24223	6867	13493	107	14778	15895
NoSignals	0	23	101	64	32	0	54	36
トラブル:非常停止	0	220	0	26	0	0	0	0
トラブル:アラーム	0	0	0	575	0	0	0	0
段取り：リセット	73	43	0	0	0	0	0	62
段取り：一時停止	0	0	0	0	0	0	0	17
段取り：脱着	0	26	0	0	0	0	0	53
段取り：加工	41	224	0	0	0	0	0	54
量産加工:一時停止	5	1	12	1	0	0	1	1
量産加工：脱着	23688	19006	0	10997	10576	24776	11448	10732
量産加工：加工	58910	26606	26929	20987	52915	61509	46521	6554
合計	86226	53071	55294	39601	77016	86392	72802	33404

（出典）石塚ら（2020, Table 2, p.580）をもとに筆者作成

表 5-3　状態定義（第三版）の状態別合計持続時間

状態名（第三版）	ロボドリル別合計持続時間（秒）							
	001	002	003	004	005	006	007	008
@TimedOut	0	0	4029	84	0	0	0	0
@UnknownBCD	0	0	0	0	0	0	0	0
起動・停止	0	39	135	115	60	0	84	56
トラブル:非常停止	0	220	0	26	0	0	0	0
トラブル:アラーム	0	0	0	575	0	0	0	0
段取り：リセット	32	4	0	0	0	0	0	55
段取り：一時停止	0	77	0	0	0	0	0	21
段取り：脱着	0	0	0	0	0	0	0	54
段取り：加工	41	224	0	0	0	0	0	54
段取り：MDI	2695	2128	2931	5985	985	0	1477	387
段取り：EDIT	126	671	247	182	345	0	233	15222
段取り：手動	182	112	586	125	265	0	141	123
量産加工:リセット	443	3468	248	495	11834	41	12743	114
量産加工:一時停止	22	860	13	1	0	0	127	1
量産加工：脱着	23775	18662	20177	11026	10612	24842	11476	10763
量産加工：加工	58910	26606	26928	20987	52915	61509	46521	6554
合計	86226	53071	55294	39601	77016	86392	72802	33404

（出典）石塚ら（2020, Table 3, p.580）をもとに筆者作成

なお，3 色表示灯の状態は，BCD-MQTT ボードが取得する信号として出力されるよう，当該工場技術者によって設定されており，BCD-MQTT ボードが出力する BCD 値から抽出し集計した．4.3.9 項で示した通り，ロボドリルの内部的な振る舞いによる信号切り替えタイミングのずれが原因で，3 色表示灯のうち重複して点灯するものもあった．表 5-1 には，各ランプの組み合わせごとに集計した結果を示す．この結果の通り，緑ランプと黄ランプのみが重複して点灯しており，これは当該工場技術者でも想定していなかった状態であるため，重複点灯は意図した結果ではないことを付け加えておく．

まず，各表の状態に着目する（表 5-4）．3 色表示灯の点灯状況の状態数（= 8），状態定義（初版）の状態数（= 12），状態定義（第三版）の状態数（= 16）のみで比較すると，3 色表示等の点灯状況が最も少ない状態数となっている．状態数の差が，そのまま情報量の差となっていることは明らかである．また，状態定義（初版）の状態と状態定義（第三版）の状態数にも差がある．同じ 16ch の信号数でも，その組み合わせ方によって差がある．これは実証実験の初期段階とその後の運用において，状態の捉え方の精度が高まったともいえる．さらに，状態の意味を考慮すれば，状態定義によって規定された状態には，作業の目的が含まれており，この目的はマネジメント主体である当該工場技術者によって規定されていることから，マネジメント主体がその意味を理解できることは明らかである．

表 5-4　各合計持続時間集計表の状態数の比較

状態数	Lamp Status G:GREEN, Y:YELLOW, R:RED （表 5-1）	状態名（初版） （表 5-2）	状態名（第三版） （表 5-3）
1	G=ON, Y=off, R=off	@TimedOut	@TimedOut
2	G=off, Y=ON, R=off	@UnknownBCD	@UnknownBCD
3	G=off, Y=off, R=ON	NoSignals	起動・停止
4	G=ON, Y=ON, R=off	トラブル：非常停止	トラブル：非常停止
5	G=ON, Y=off, R=ON	トラブル：アラーム	トラブル：アラーム
6	G=off, Y=ON, R=ON	段取り：リセット	段取り：リセット
7	G=ON, Y=ON, R=ON	段取り：一時停止	段取り：一時停止
8	G=off, Y=off, R=off	段取り：脱着	段取り：脱着
9		段取り：加工	段取り：加工
10		量産加工：一時停止	段取り：MDI
11		量産加工：脱着	段取り：EDIT
12		量産加工：加工	段取り：手動
13			量産加工：リセット
14			量産加工：一時停止
15			量産加工：脱着
16			量産加工：加工

続いて，3 色表示灯点灯色別合計持続時間と状態定義（第三版）の状態別合計持続時間を比較する．

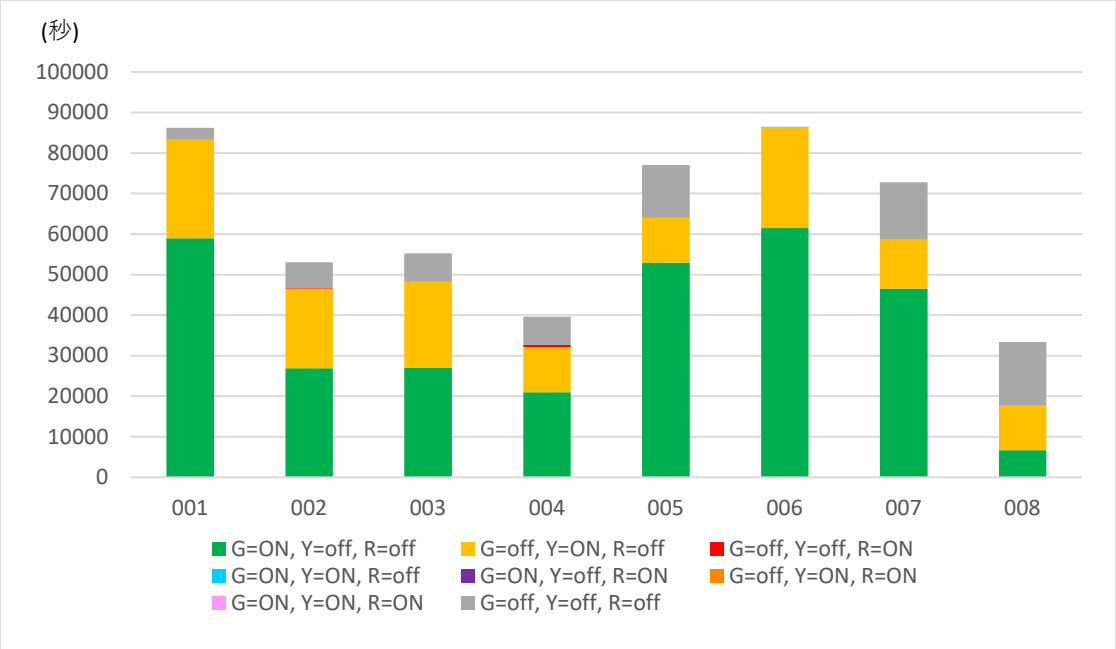


図 5-1　各ロボドリルの 3 色表示灯点灯色別合計持続時間

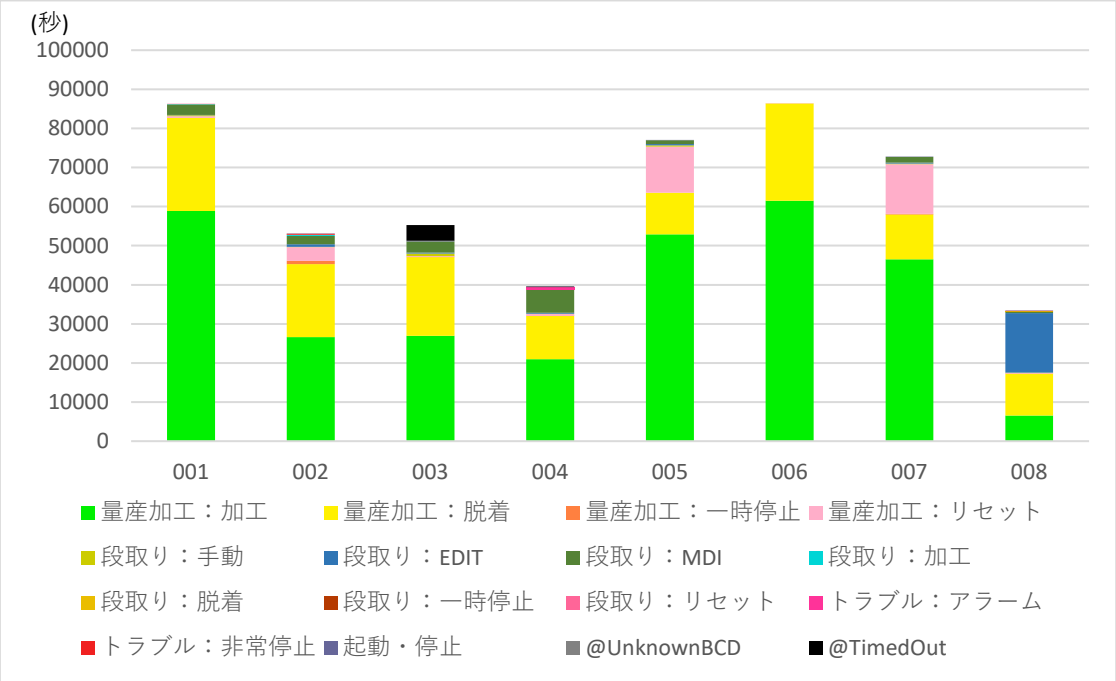


図 5-2　各ロボドリルの状態定義（第三版）の状態別合計持続時間

図 5-1 は表 5-1 の内容を各ロボドリルの積み上げ棒グラフとして表したものであり，図 5-2 は表 5-3 の内容を各ロボドリルの積み上げ棒グラフとして表したものである．図 5-1 が示すように，3 色表示灯で表現可能な状態を稼働状態としてみるのが一般的である．一方で，図 5-2 が示すドメイン概念に基づき規定された状態は，図 5-1 が示す状態よりも具体的で意味のある情報が取得可能であることを示している．以上のことから，ドメイン概念に基づく状態管理によって得られるこのような情報はマネジメントに資する情報であり，機械内部的な稼働状況よりも有用であるといえる．

次に，状態定義（第三版）（表 5-3）の“段取り：加工”および“量産加工：加工”（以後，“量産：加工”と表記する）と，3 色表示灯の緑点灯状態（表 5-1）に着目して比較する．実証に利用したロボドリルでは，“段取り：加工”と“量産加工：加工”は 3 色表示灯の緑ランプが点灯する状態となっている．表 5-5 に，緑点灯と“段取り：加工”ならびに“量産：加工”との合計持続時間比較表を示す．

表 5-5　緑点灯と段取り／量産加工（第三版）の合計持続時間の比較

	ロボドリル別合計持続時間（秒）							
	001	002	003	004	005	006	007	008
a) 緑点灯	59007	26883	27037	21020	52990	61509	46568	6690
b) 段取り：加工（第三版）	41	224	0	0	0	0	0	54
c) 量産：加工（第三版）	58910	26606	26928	20987	52915	61509	46521	6554
d) b+c	58951	26830	26928	20987	52915	61509	46521	6608
e) a-d	56	53	109	33	75	0	47	82

（出典）石塚ら（2020, Table 4, p.580）をもとに筆者作成

ここでは緑点灯のみに着目することとし，緑点灯の合計持続時間（表 5-5 の a）には黄ランプとの重複点灯持続時間も含めている．“段取り：加工”の合計持続時間（表 5-5 の b）と“量産：加工”の合計持続時間（表 5-5 の c）の和が表 5-5 の d である．緑点灯の合計持続時間（表 5-5 の a）と表 5-5 の d の差が表 5-5 の e である．表 5-5 の a と表 5-5 の d が一致しない要因は，3 色表示灯の点灯状態以外の信号も加味したドメイン概念による状態定義によって，“段取り：加工”と“量産：加工”以外の状態に分類されているものがあるためと考えられる．表 5-5 によれば，“段取り：加工”の合計持続時間はゼロもしくは非常に短く，“量産：加工”の合計持続時間と緑点灯の合計持続時間の差も非常に短い．これらのことから，緑点灯の合計持続時間を量産加工時間と見なすこともできる．ここで，表 5-5 においてロボドリル 003 とロボドリル 005 の“段取り：加工”時間が増大したと仮定する．“量産：加工”と“段取り：加工”はともに緑ランプが点灯する動作であるから，“段取り：加工”時間が増大したならば，その増大分だけ“量産：加工”時間が減少するが，緑点灯時

間は変化しないものと想定できる．この想定を図で表したものを，図 5-3 に示す．図 5-3 に示したように，段取りとして分類される作業時間が増大したとき，緑点灯の合計持続時間を単に量産加工時間とみなすことはできなくなる．

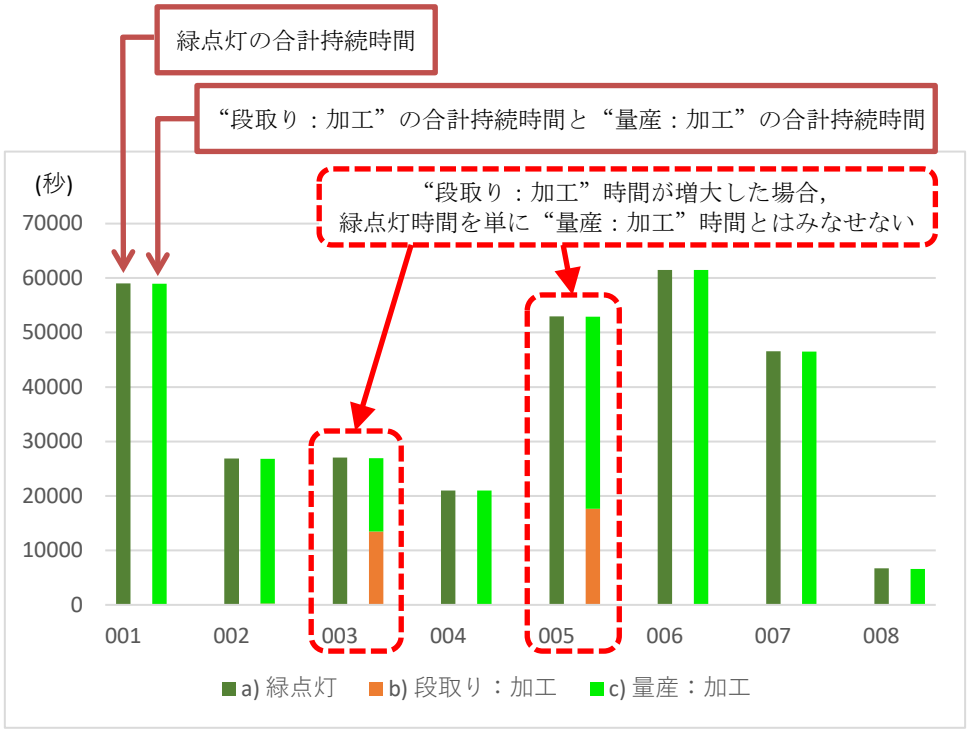


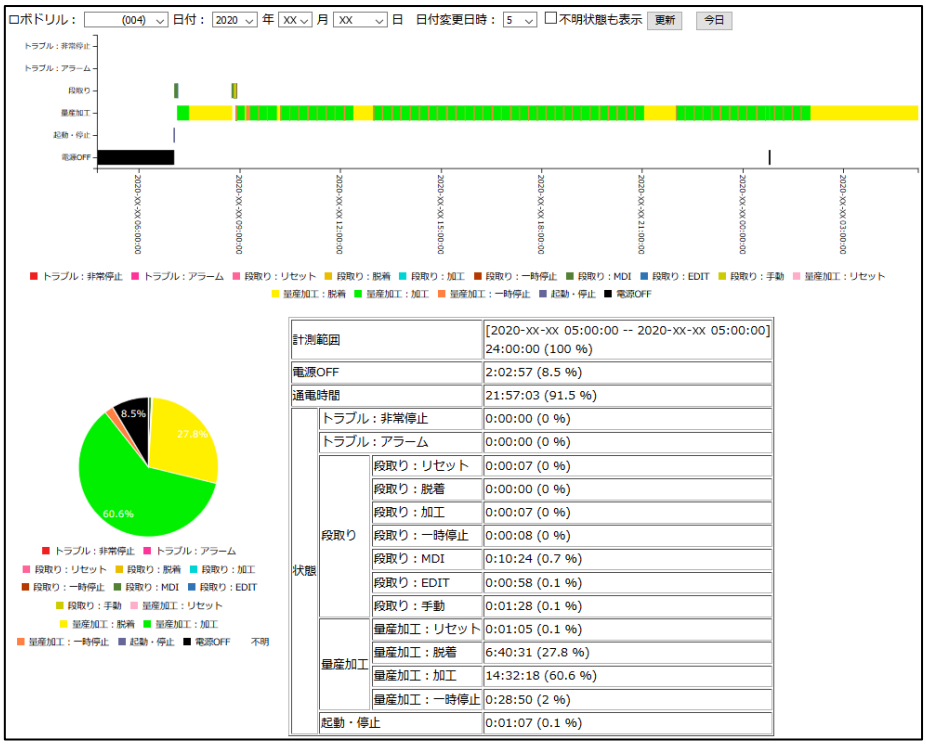
図 5-3　“段取り：加工”合計持続時間増大と仮定したときの緑点灯合計持続時間との比較

この“段取り：加工”の意味を，マネジメントにおいてどのように解釈するかは，マネジメント主体のドメイン概念による．定常的生産（量産）のみが行われているのであれば，“段取り：加工”時間の増大は生産性低下とみなせる．一方で，新製品の試作等で“段取り：加工”時間が増大したとするならば，それは製造企業にとって有用な時間ともいえる．現時点では，状態定義によって導出される状態から詳細な作業内容を判別することはできないが，少なくとも段取りか量産かといった緑点灯持続時間の計測のみでは得られないより有意な情報が，ドメイン概念に基づく状態管理によって得られることを意味する．以上のことから，ドメイン概念に基づく状態管理は有用であり，このような状態管理を可能とする情報システムもまた有用といえる．

5.2 日次状態集計チャートの効果

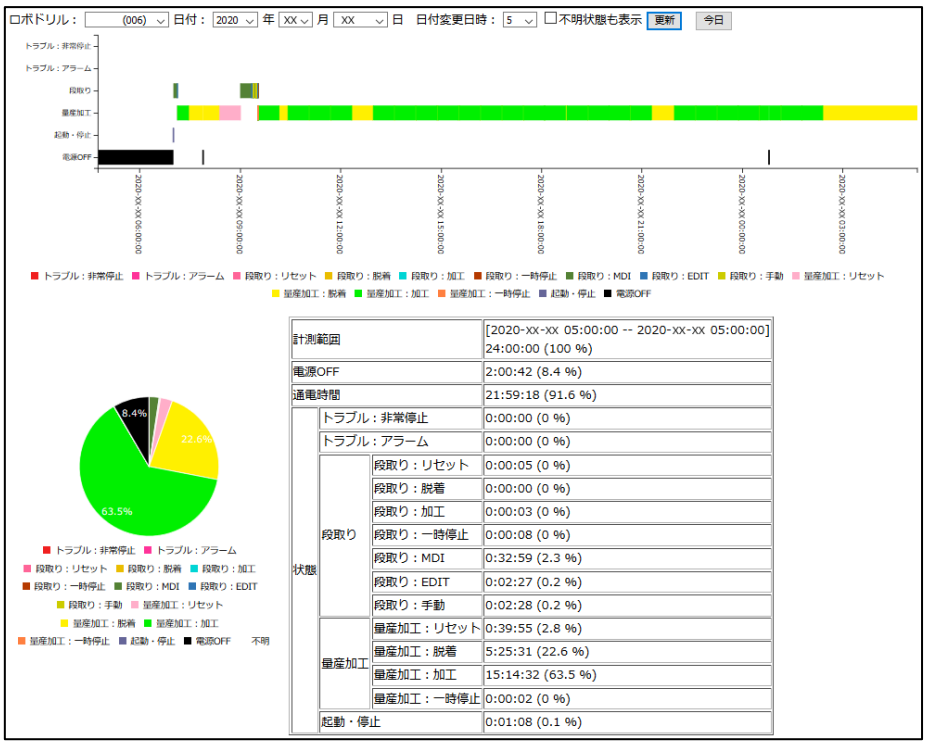
機械稼働タスク状態管理の実証において実装した日次状態集計チャートは，1 日分の機械稼働タスク状態の履歴を確認可能な可視化方法である．リアルタイムの状態提示では，閲覧時点での状態情報を提示するのみであるが，日次状態集計チャートは過去にどのような振

る舞いを工作機械が行っていたかを閲覧することができる．図 5-4 はロボドリル 004 のある 1 日（m 月 a 日）の日次状態集計チャートの内容を示す．図 5-5 はロボドリル 006 のある 1 日（m 月 a 日）の日次状態集計チャートの内容を示す．どちらも同日の作業状態を示すものであるが，ロボドリルが異なれば状態のあり様は異なる．一方，図 5-6 はロボドリル 008 のある 1 日（m 月 b 日）の日次状態集計チャートの内容，図 5-7 はロボドリル 008 のある 1 日（m 月 c 日）の日次状態集計チャートの内容を示す．どちらも同一のロボドリル（008）の作業状態を示しているが，作業日が異なれば状態のあり様も異なる．これは，同一の工作機械でも異なる作業が実施されていると考えられる．このように，実証工場で行われていた実際の作業は，作業日や工作機械の機種によらず多様であることが明らかとなった．ここで示したような結果が得られたことで，実際に作業を行う作業者に対しても，どのような作業を行っていたか，という作業現場における振り返りの取り組みにも寄与できる．



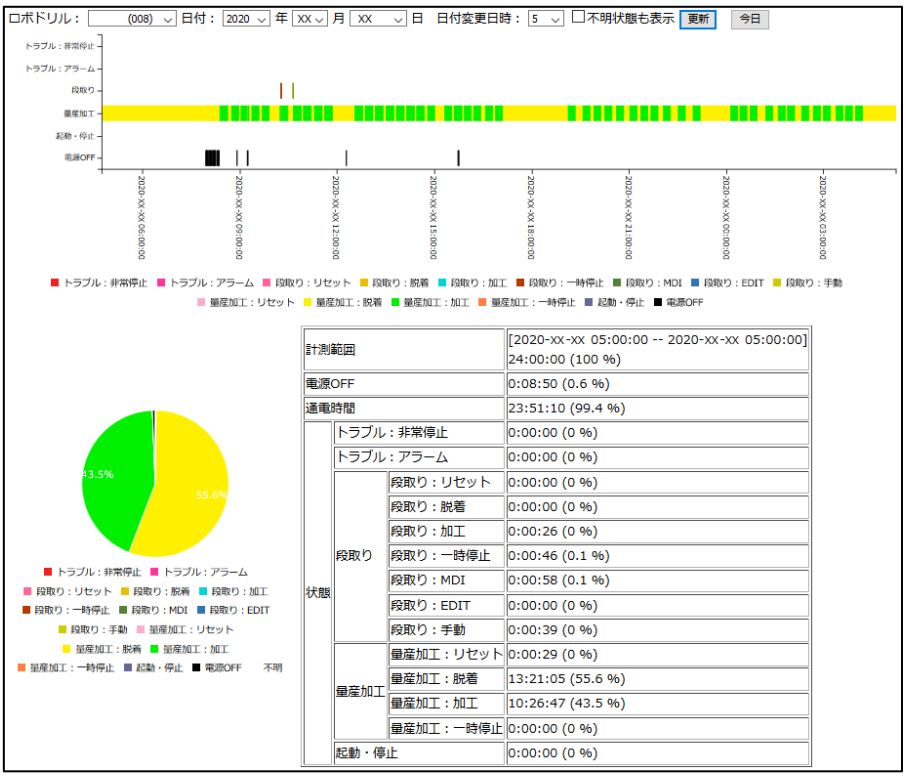
計測範囲			[2020-XX-XX 05:00:00 -- 2020-XX-XX 05:00:00] 24:00:00 (100 %)
電源OFF			2:02:57 (8.5 %)
通電時間			21:57:03 (91.5 %)
状態	トラブル：非常停止		0:00:00 (0 %)
	トラブル：アラーム		0:00:00 (0 %)
	段取り	段取り：リセット	0:00:07 (0 %)
		段取り：脱着	0:00:00 (0 %)
		段取り：加工	0:00:07 (0 %)
		段取り：一時停止	0:00:08 (0 %)
		段取り：MDI	0:10:24 (0.7 %)
		段取り：EDIT	0:00:58 (0.1 %)
		段取り：手動	0:01:28 (0.1 %)
	量産加工	量産加工：リセット	0:01:05 (0.1 %)
		量産加工：脱着	6:40:31 (27.8 %)
		量産加工：加工	14:32:18 (60.6 %)
		量産加工：一時停止	0:28:50 (2 %)
	起動・停止		0:01:07 (0.1 %)

図 5-4　ロボドリル 004（m 月 a 日）の日次状態集計チャート



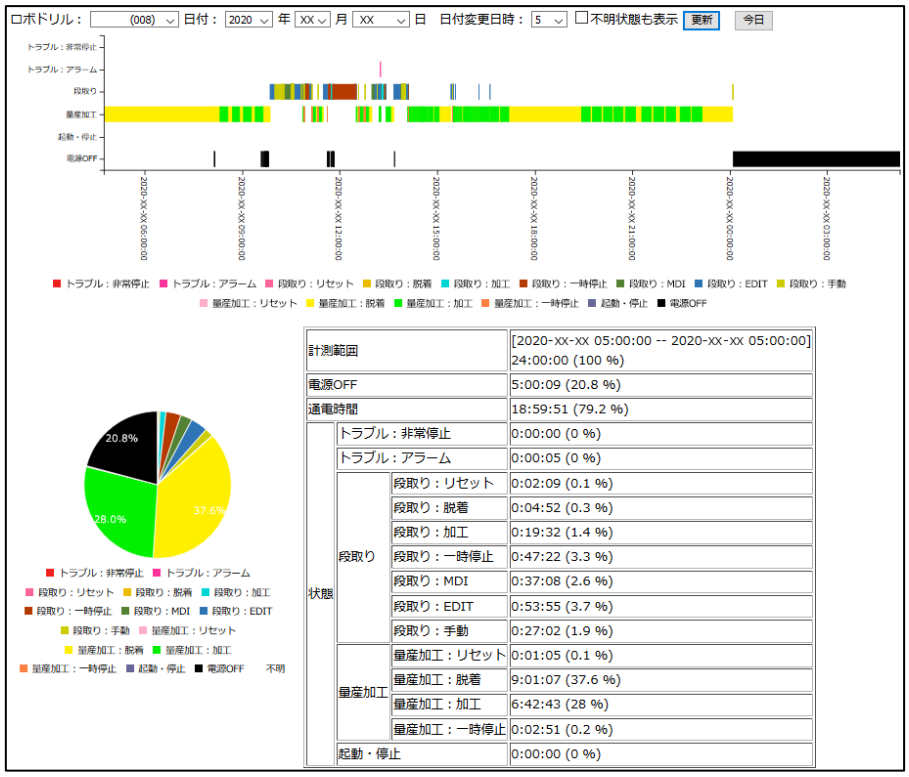
計測範囲			[2020-XX-XX 05:00:00 -- 2020-XX-XX 05:00:00] 24:00:00 (100 %)
電源OFF			2:00:42 (8.4 %)
通電時間			21:59:18 (91.6 %)
状態	トラブル：非常停止		0:00:00 (0 %)
	トラブル：アラーム		0:00:00 (0 %)
	段取り	段取り：リセット	0:00:05 (0 %)
		段取り：脱着	0:00:00 (0 %)
		段取り：加工	0:00:03 (0 %)
		段取り：一時停止	0:00:08 (0 %)
		段取り：MDI	0:32:59 (2.3 %)
		段取り：EDIT	0:02:27 (0.2 %)
		段取り：手動	0:02:28 (0.2 %)
	量産加工	量産加工：リセット	0:39:55 (2.8 %)
		量産加工：脱着	5:25:31 (22.6 %)
		量産加工：加工	15:14:32 (63.5 %)
		量産加工：一時停止	0:00:02 (0 %)
	起動・停止		0:01:08 (0.1 %)

図 5-5　ロボドリル 006（m 月 a 日）の日次状態集計チャート



計測範囲			[2020-XX-XX 05:00:00 -- 2020-XX-XX 05:00:00] 24:00:00 (100 %)
電源OFF			0:08:50 (0.6 %)
通電時間			23:51:10 (99.4 %)
状態	トラブル：非常停止		0:00:00 (0 %)
	トラブル：アラーム		0:00:00 (0 %)
	段取り	段取り：リセット	0:00:00 (0 %)
		段取り：脱着	0:00:00 (0 %)
		段取り：加工	0:00:26 (0 %)
		段取り：一時停止	0:00:46 (0.1 %)
		段取り：MDI	0:00:58 (0.1 %)
		段取り：EDIT	0:00:00 (0 %)
		段取り：手動	0:00:39 (0 %)
	量産加工	量産加工：リセット	0:00:29 (0 %)
		量産加工：脱着	13:21:05 (55.6 %)
		量産加工：加工	10:26:47 (43.5 %)
		量産加工：一時停止	0:00:00 (0 %)
	起動・停止		0:00:00 (0 %)

図 5-6　ロボドリル 008（m 月 b 日）の日次状態集計チャート



計測範囲			[2020-XX-XX 05:00:00 -- 2020-XX-XX 05:00:00] 24:00:00 (100 %)
電源OFF			5:00:09 (20.8 %)
通電時間			18:59:51 (79.2 %)
状態	トラブル：非常停止		0:00:00 (0 %)
	トラブル：アラーム		0:00:05 (0 %)
	段取り	段取り：リセット	0:02:09 (0.1 %)
		段取り：脱着	0:04:52 (0.3 %)
		段取り：加工	0:19:32 (1.4 %)
		段取り：一時停止	0:47:22 (3.3 %)
		段取り：MDI	0:37:08 (2.6 %)
		段取り：EDIT	0:53:55 (3.7 %)
		段取り：手動	0:27:02 (1.9 %)
	量産加工	量産加工：リセット	0:01:05 (0.1 %)
		量産加工：脱着	9:01:07 (37.6 %)
		量産加工：加工	6:42:43 (28 %)
		量産加工：一時停止	0:02:51 (0.2 %)
	起動・停止		0:00:00 (0 %)

図 5-7　ロボドリル 008（m 月 c 日）の日次状態集計チャート

5.3 状態管理フレームワーク実現のための要件に関する評価と考察

状態管理フレームワーク実現のために満たすべき要件として設定した項目（3.2 節）について，機械稼働タスク状態管理の実証結果に基づく評価と考察を示す．

(1) 局所的に導入できる

ロボドリルの状態管理という実証によって，局所的な導入は実証できた．機械稼働タスクとしては，ロボドリル 8 台への導入という形で，単一の工作機械のみでなく，複数台の工作機械に導入できることも実証できた．一方で，作業者タスクの状態管理については，検証用に作動させたモデルのみであり，実際の現場での実証は今後の課題である．しかしながら，ロボドリルで行う加工の製品単位での開始と終了を追跡したいという要望はあり，その方法については工場関係者と協議を進めていく．

(2) コンポーネントの関係が疎である

機械稼働タスク状態管理の実証モデルにおいて，BCD-MQTT ボードと状態コンバーター間に，BCD データ以外の関連性はない．リアルタイム状態提示画面や日次状態集計チャートは，状態コンバーターにより変換された後の状態データのみを利用している．この点において，BCD-MQTT ボードと状態提示画面との間に関係はない．よって，この実証モデルにおけるソフトウェアコンポーネントは，疎な結合であるといえる．疎結合であれば，機能改

修などの進化は，基本的に改修対象のソフトウェアコンポーネントに対してのみ行えばよく，BCD-MQTT ボードのような装置が故障しても，それを入れ替えるだけで良いといえる．ただし，状態コンバーターはデータベースへの書き込みを行っており，そのデータベースが他のコンポーネントからも利用されているという点は，今後も検討すべき課題とする．

(3) ドメイン概念に基づいた状態を規定できる

状態コンバーターの状態定義は，当該工場技術者によって作成されたものである．また，作業状況に応じた状態をより正確に得るために，改定も行われている．この点のみで評価するならば，ドメイン概念に基づいた状態を規定できるといえる．

(4) 規定した状態をシステムに反映できる

当該工場技術者によって規定された状態定義は，筆者によって状態管理フレームワークの実証環境へ反映している．規定した状態をシステムに反映できるとはいえるが，当該工場の関係者のみでシステムへの反映が完結する方が，より早いサイクルで改善が行えると想定できる．この点においては，状態定義の反映方法などを，さらなる検討が必要である．

(5) 収集した状態データを可視化できる

リアルタイム状態提示と日次状態集計チャートという 2 つのアプローチで可視化を実践した．リアルタイム状態提示は，出口研究室で開発が進められているサーバルキャットというツールを用いて，IT ベンチャー企業によって画面が作成されている．サーバルキャットを用いることで，本来は工場技術者などのマネジメント主体で，画面デザインと運用が行えると想定しており，今後の実証における課題としたい．日次状態集計チャートは，Web アプリケーションであり，データベースに保存された状態データを参照する．その点では，疎結合型のマイクロサービスとはみなせない．これは，より独立した機能として検討する必要がある．しかしながら，日次状態集計チャートにより可視化された意義は大きい．ドメイン概念に基づいて規定された状態によって，自らの作業を確認できるという点で，過去のデータを振り返ることができ，マネジメントを行う上での大きな前進であるといえる．

(6) 収集した状態データを抽出できる

収集した状態データはデータベースに記録されている．実証結果の評価のみならず，状態定義を行った当該工場技術者も，収取された状態データを用いて状態定義改定のための分析を行っている．データを分析するようなコンポーネントを新たに用意することで，収集したデータの利活用も可能となる．しかしながら，データの利活用方法について，さらなる検討の余地がある．

工作機械とそれを操作する作業者はそれぞれ独立であり，協調的に動作することはすでに説明した．作業者タスクと機械稼働タスクとの連携については，今後も議論が必要である．

5.4 状態管理フレームワークの導入・運用に関する考察

状態管理フレームワークの利用には，作業内容によって必要となる技術や知識は異なる．ここでは，状態管理フレームワーク利用のための作業項目を，①状態の定義，②状態定義の反映，③導入・運用，④保守，⑤機能追加・変更，の 5 項目について考察する．①状態の定義については，対象とする機械装置等の知識や，マネジメント対象の製造活動に関する業務知識が必要といえる．実証においても状態定義は当該工場技術者が実践しており，これには利用する工作機械の知識のみならず，どのような加工を行っているかなどの業務知識も密接に関連することからも明らかである．②状態定義の反映については，変更が必要な個所の機能はもちろんのこと，関連するソフトウェアコンポーネントの実行方法なども熟知している必要がある．実証を行った機械稼働タスクの状態管理でいえば，状態コンバーターに指定する状態定義ファイルの他，リアルタイム状態定義画面や日次状態集計チャートなど，状態の定義やその提示に関するものが対象となる．実証においては，状態定義の改変を当該工場技術者が行ったものの，実証環境への反映は筆者によるものである．しかしながら，筆者が行った反映手順を理解し実践することで，ドメインの関係者自らが反映可能となることも期待できる．③導入・運用については，商用 VPS への環境構築や必要なソフトウェアコンポーネント等の導入，保守などが必要となる．このレベルでは，情報システムの構築やプログラミングなどの高度な知識は必要ではないものの，商用 VPS で稼働する OS やアプリケーション等のセットアップ・保守といった IT 知識が必要となる．しかしながら，実証環境と同様の状態管理フレームワークを導入・運用するのであれば，その手順を確立することで，高度な IT 知識を必要とせずとも対応可能になると考えられる．④保守については，状態管理フレームワークを構成するシステムのトラブル等への対処も考慮すると，より柔軟で広範囲な技術・知識が必要といえる．⑤機能追加・変更に至っては，プログラミン等を含むさらに高度な技術・知識が必要となる．

実証環境へ導入した BCD-MQTT ボードそのものは，ロボドリル専用ケーブルと合わせて 1 台 2～3 万円ほどで IT ベンチャー企業により制作されている．BCD-MQTT ボードの設定・導入も IT ベンチャー企業により実施されている．IT 企業への委託を想定するならば，そのコストは IT 企業により異なる．さまざまな環境での実証を経ることで，導入・運用手順がより洗練され簡素化されれば，必要となる技術や知識も単純化され，工数短縮によるコストダウンも期待できる．

第6章 結論

本章では，本研究の成果について述べた後，今後の課題について述べる．

6.1 本研究の成果

本論文では，製造企業のドメイン概念に基づいた状態管理という点に着目し，工作機械を操作する作業者のタスクと，自律的に動作する工作機械の稼働タスクという 2 つのアプローチで状態管理を行う方法について説明した（第 2 章）．新たに提案した状態管理手法を実践するための状態管理フレームワークとして実証モデルを検討し（第 3 章），実在工場にて実証した（第 4 章）．実証結果から，工作機械の内部的な稼働状況は，ドメイン概念に基づき，機械稼働タスクの状態へ転換可能であることを示した．また，実際に取得した機械稼働タスクの状態データを用いて集計を行い，工作機械の内部的な稼働状況と比較して，ドメイン概念を用いて規定された状態のほうが，意味のある情報として有用であることを示した（第 5 章）．この実証を通して，マネジメント主体が，自らのドメイン概念に基づいて状態を決定できることを実証した．さらに，そのように決定された状態は，マネジメント主体にとって容易に理解できる情報であった．実際，筆者も含めた工場外の人間には，規定された状態の言葉の意味は推測できるが，どのような目的でそれが行われているかを想定することは難しい．しかしながら，機械稼働タスクの実証を通して，リアルタイム状態提示により今何が行われているかが判別可能となり，日次状態集計チャートにより過去の状態の振り返りも可能となった．このことで，過去に何が行われたか，という検討も進められつつある．また，自らの作業に関連する状態が確認できるという状況は，新たな変化に繋がった．本研究での実証後，さらに 4 台のロボドリルが新たな管理対象として追加された．本研究において掲げた「現場の変化に追随可能か」という点のみで評価するならば，現場の変化に追随できたといえる．一方で，現場の関係者が直接的に拡張を行えるかという点は，状態管理フレームワークの利便性を高める上でも重要な課題であり，今後も協議と検討を進めていきたい．

実在工場での実証を行った状態管理フレームワークは，マネジメントの IoT 化(出口 2018b)における 2 ラインアーキテクチャという概念を用いて，機械稼働状況を人間に意味のある状態として取得可能とした．これは，マネジメントの IoT 化の実践事例として，研究の進展への寄与となる．

実証に用いた状態管理フレームワークによる状態収集と可視化を通して，新たなイノベーションも展開されつつある．実証実験を行った企業とは異なる企業において，ロボドリルとは異なる，ロボットセルでの実証も進めている．本論で示した実証モデルの実在工場への導入とほぼ同様の手順にて，新たな実証環境への導入を進めており，さまざまな状態が取得できつつある．また，実証によって得られた成果から，特許取得の動きにも発展している．

6.2 今後の課題

機械稼働タスクの状態を規定する上での困難も明らかとなった．不明状態や重複の発生など，想定しない状況を事前に防ぐ方法なども引き続き検討する．マネジメント主体が状態を規定できるものの，その更新方法や，状態管理フレームワークそのものの運用・保守といった観点では未検証である．また，工作機械の加工タスクとしての想定において，作業者が行うタスクと，工作機械が行うタスクとに分離したが，工作機械を操作する側である作業者タスクの状態管理については，工場関係者による実証には至っていない．工作機械の稼働状況は，24V の信号として情報を取り出し，その信号の組み合わせから状態に変換している．これは，工作機械の作動を一切阻害しない．一方で，工作機械を操作する作業者は，日常的に業務を行っており，その手順は確立されている．そのような確立されている手順の間に，バーコード入力やテンキーパッド入力などの本来の作業とは異なる活動を，どのように組み入れられるかは，今後の課題の一つとなっている．機械から得られる情報のみではなく，日々の業務として行われている人間の活動こそ，正しくその状態を捉えることが必要といえる．この課題の克服方法として，ボタン等を用いて開始や終了といった作業遂行のトリガーを入力する案も検討しており，引き続き工場関係者と協議を進める．

状態管理フレームワークは，既存の環境に部分的に導入でき，拡張も可能であることは実証できた．しかしながら，既存のシステムとどのように連携するかという点については，今後も検討が必要である．例えば，すでに生産管理の情報システムが稼働しており，その情報と状態管理フレームワークによって得られた状態等のデータを連携させる方法などである．既存情報システムとデータ連携することで，より正確な作業状況を把握でき，これはマネジメントの IoT 化のコンセプトにおいても必要とされている原価管理のためのマイクロな遂行状況の把握といえる．このような観点でも検討を進めていきたい．

謝辞

本論文の作成にあたり，多くの方々からご指導とご支援を頂きましたことに，この場をお借りして厚く御礼申し上げます.

東京工業大学の出口弘先生には，指導教員として，本研究のご指導のみならず研究を行う上での姿勢に至るまで熱心にご指導ご鞭撻を頂いたこと，さらにはさまざまな事柄についても広く深くご教示頂いたことに，心より感謝致します.

また，東京工業大学の山村雅幸先生，三宅美博先生，小野功先生，石井秀明先生には，論文審査を通じ，貴重なご助言ならびに丁寧なご指導を頂いたことに深く感謝致します.

株式会社協和精工の皆様には，実証実験の環境をご提供頂くのみならず，業務の内容や貴重な資料等もご提供頂いた上，貴重なご助言も頂けたこと，改めて深く感謝申し上げます.

東京工業大学入学以来，出口研究室の皆様には大変お世話になりました．深く感謝致します.

最後に，筆者の本学における研究活動に対しご理解ご協力くださった勤務先の株式会社パイケークの木寺重樹氏，倉田正氏，助台良之氏，そして博士課程の研究活動において筆者を支えてくれた妻に心より感謝致します.

2020 年 8 月

参考文献

1. Apache. (1997). The Apache HTTP Server Project. https://httpd.apache.org/ (Accessed 2020-08-19).

2. Ashton, K. (2009). That 'Internet of Things' Thing. https://www.rfidjournal.com/that-internet-of-things-thing (Accessed 2020-08-19).

3. Balalaie, A., Heydarnoori, A., & Jamshidi, P. (2016). Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture. *IEEE Software*, 33(3), pp.42–52.

4. Bratukhin, A., & Sauter, T. (2010). Bridging the gap between centralized and distributed manufacturing execution planning. *2010 IEEE 15th Conference on Emerging Technologies Factory Automation (ETFA 2010)*, pp.1–8.

5. Bratukhin, A., & Sauter, T. (2011). Functional Analysis of Manufacturing Execution System Distribution. *IEEE Transactions on Industrial Informatics*, 7(4), pp.740–749.

6. He, D., Lobov, A., & Martinez Lastra, J. L. (2012). ISA-95 Tool for Enterprise Modeling. *ICONS 2012 : The Seventh International Conference on Systems*, pp.83–87.

7. JETRO (2015) 『インダストリー4.0 実現戦略 プラットフォーム・インダストリー4.0 調査報告』. https://www.jetro.go.jp/ext_images/_Reports/01/c982b4b54247ac1b/20150076.pdf (Accessed 2020-08-19).

8. Jürgen, K. (2007). *Manufacturing Execution Systems—MES*. Springer, Berlin, Heidelberg.

9. Kim, G., Humble, J., Debois, P., & Willis, J. (2016). *The Devops Handbook: How to Create World-Class Agility, Reliability, & Security in Technology Organizations*. IT Revolution Press. (長尾高弘(訳), 榊原彰(監修) (2017) 『The DevOps ハンドブック: 理論・原則・実践のすべて』, 日経 BP 社 , 日経 BP マーケティング(発売))

10. Lee, E. A. (2008). Cyber Physical Systems: Design Challenges. *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, pp.363–369.

11. Lee, S., Nam, S. J., Park, J. K., & Lee, J.-K. (2010). Monitoring system of equipment in shop floor for embodying MES. *The 40th International Conference on Computers Indutrial Engineering*, pp.1–6.

12. Lewis, J., & Fowler, M. (2014). Microservices. martinfowler.com. https://martinfowler.com/articles/microservices.html (Accessed 2020-08-19).

13. Lu, Y. (2017). Industry 4.0: A survey on technologies, applications and open research issues. *Journal of Industrial Information Integration*, 6, pp.1–10.

14. MongoDB. (2020). The most popular database for modern apps. MongoDB. https://www.mongodb.com (Accessed 2020-08-19).

15. Mosquitto. (2018). Eclipse Mosquitto. Eclipse Mosquitto. https://mosquitto.org/ (Accessed 2020-08-19).

16. MQTT. (2020). FAQ - Frequently Asked Questions | MQTT. http://mqtt.org/faq (Accessed 2020-08-19).

17. OASIS. (2014). MQTT Version 3.1.1. http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.pdf (Accessed 2020-08-19).

18. Oliveira, J. F. G., Júnior, F. F., Coelho, R. T., & Silva, E. J. (2008). Architecture for machining process and production monitoring based in open computer numerical control. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 222(12), pp.1605–1612.

19. RFC8259. (2017). The JavaScript Object Notation (JSON) Data Interchange Format. https://www.rfc-editor.org/rfc/rfc8259.html (Accessed 2020-08-19).

20. Richardson, C. (2019). *Microservices patterns: with examples in Java*. Manning Publications. (長尾高弘(訳), 樽澤広亨(監修) (2020) 『マイクロサービスパターン: 実践的システムデザインのためのコード解説』, インプレス).

21. SmartFactoryKL. (2018). SmartFactory^{KL} System Architecture for Industrie 4.0 Production Plants Whitepaper SF-1.2: 04/2018. SmartFactoryKL-System Architecture for Industrie 4.0 Production Plants Whitepaper SF-1.2: 04/2018. https://smartfactory.de/wp-content/uploads/2018/04/SF_WhitePaper_SystemArchitecture_1-2_EN_XS.pdf (Accessed 2020-08-19).

22. Wortmann, F., & Flüchter, K. (2015). Internet of Things. *Business & Information Systems Engineering*, 57(3), pp.221–224.

23. Yang, C., Shen, W., & Wang, X. (2018). The Internet of Things in Manufacturing: Key Issues and Potential Applications. *IEEE Systems, Man, and Cybernetics Magazine*, 4(1), pp.6–15.

24. Yassein, M. B., Shatnawi, M. Q., Aljwarneh, S., & Al-Hatmi, R. (2017). Internet of Things: Survey and open issues of MQTT protocol. *2017 International Conference on Engineering MIS (ICEMIS)*, pp.1–6.

25. 相原健郎 (2019) 「サイバーフィジカルシステムでの実世界データ収集」『電子情報通信学会論文誌 B』, J102-B(6), pp.387–398.

26. 池澤克哉, 鶴畑清臣, 小田信二, 林俊介 (2016)「産業界から見たシステム技術と IoT」『計測と制御』, 55(4), pp.295–299.

27. 岩野和生, 高島洋典 (2015) 「サイバーフィジカルシステムと IoT（モノのインターネット） 実世界と情報を結びつける」『情報管理』, 57(11), pp.826–834.

28. 岩本晃一 (2015) 『インダストリー4.0: ドイツ第 4 次産業革命が与えるインパクト』, 日刊工業新聞社.

29. 貝原俊也（2006）「生産管理ソフトウェア」『精密工学会誌』, 72(2), pp.171–175.

30. 加藤真平（2014）「サイバーフィジカルシステム：1. サイバーフィジカルシステムの概要と動向」『情報処理』, 55(9), pp.910–915.

31. 川野俊充（2015）「ドイツのモノづくり政策 Industrie 4.0 が狙う製造業の標準化戦略」『日本ロボット学会誌』, 33(5), pp.318–324.

32. 経済産業省（2020）『2020 年版ものづくり白書』, 経済産業省, https://www.meti.go.jp/report/whitepaper/mono/2020/honbun_pdf/pdf/all.pdf (Accessed 2020-08-19).

33. 児玉公信（2012）「MES（製造実行システム）のモデルに関する一考察」『研究報告情報システムと社会環境（IS）』, 2012-IS-122(4), pp.1–5.

34. 高橋達也, 武藤一夫, 児玉公信, 藤田一昭, 大竹洋介（2010）「標準技術の相互活用による工場内情報連携: Mesx プロトコルによる製販一体化(<小特集>生産システム部門研究発表講演会 2010)」『日本機械学会論文集 C 編』, 76(772), pp.3240–3245.

35. 出口弘（2015）「IOE 時代の P2M 支援環境としての実世界 OS」『国際 P2M 学会誌』, 9(2), pp.99–122.

36. 出口弘（2018a）「IoT 時代の人工物としての社会技術複合システムの拡張モデリングサイクルとそのプロジェクト&プログラムマネジメント」『国際 P2M 学会研究発表大会予稿集』, 2018.Spring, pp.243–262.

37. 出口弘（2018b）「製造業における IoT の導入支援」『化學工業』, 69(9), pp.646–653.

38. 中村実（2000）『MES 入門: ERP、SCM の世界と生産現場を結ぶ情報システム』, 工業調査会.

39. 撫中達司（2016）「IoT によりもたらされる新たな世界」『システム／制御／情報』, 60(10), pp.425–430.

40. ロボット革命イニシアティブ（2018）『Plattform Industrie 4.0 の管理シェルの概要調査報告書』. https://www.jmfrri.gr.jp/content/files/Open/2018/20180920_AdShell/Report_AdministrationShell.pdf (Accessed 2020-08-19).

以上

本研究に係る業績

査読付き国際会議論文

○Ishizuka, Y., Kurata, T., Chang, S., & Deguchi, H. (2018). Language Design for Task Management using State Transition Modeling. *Proceedings of 2018 Joint 10th International Conference on Soft Computing and Intelligent Systems (SCIS) and 19th International Symposium on Advanced Intelligent Systems (ISIS)*, Toyama, Japan, Dec. 2018, IEEE, pp.43–50.

 doi：10.1109/SCIS-ISIS.2018.00019

査読付き学会誌論文

○石塚康成, 倉田正, 橋場浩之, 成瀬慶亮, 出口弘 (2020)「IoT を用いた製造工程における状態管理システムの研究」『電気学会論文誌Ｃ（電子・情報・システム部門誌)』, 140(6), pp.573–582.

doi：10.1541/ieejeiss.140.573

ポスター発表等

○石塚康成, 出口弘 (2017)「状態遷移を用いたタスクプログラミングのための言語設計」『計測自動制御学会　第 14 回社会システム部会研究会』, 2017-09-01.

以上