JOP-13-3

平成１２年度

生産システム情報統合専門委員会
成果報告書

平成１３年４月

ＦＡオープン推進協議会

財団法人　製造科学技術センター

目　次

１．背景と経緯

生産システムの情報統合におけるその範囲の拡大、精度の向上を目指す流れは、1980 年代の CIM システムの提案以来連綿と受け継がれてきているが、そのシステムの構築は、専用システムであったり、独自のアーキテクチャによるシステムであったりしてきて、オープン化には程遠いものであった。その結果、汎用ソフトウェアによるシステム構築やベンダの異なる機器の導入が十分に行われないことが生じるようになってきている。特に、生産現場と生産管理の情報統合はその傾向が顕著であり、その統合は、困難であった。そこで、本専門委員会では、生産現場の情報と生産管理情報の統合とオープン化を目指して、生産システムのオブジェクト指向によるモデル化、現場で用いる情報のコンテンツの標準化、稼動実績情報収集の標準化を行おうとしている。

本専門員会では、これらの検討の基本として、生産システムモデル専門委員会の OpenMES 仕様書（2000 年発行)、オープンコントローラ専門員会の工作機械管理データフォーマット 1.00J（1999 年発行)および FA イントラネット推進協会ネット推進協会のコンテンツガイドライン 0.9 版（2000 年発行）ならびに FA システムへの SNMP 適用に関する概説書（2000 年発行）を考慮して行おうとしている。

２．目的

近年、企業内では、ERP／MRP、SCADA、MES などに象徴される情報システムは、分散環境における処理を基本にサプライチェーンマネージメント構想などとあいまって着実に進展している。一方、FA 機器メーカは各社独自の異なるデータ構造とデータ転送手段を有しているため、異なるメーカの生産機械を同じネットワーク上に結合することは困難であり、実際の生産現場では制御装置と情報システムがうまくつながっていないのが現実である。 ERP などの情報システムは、現場の情報を的確に、タイムリーに把握して、処理する必要性が増大してきている。そこで、生産システム情報統合専門委員会は、オブジェクト指向モデルによる生産システムのモデル化、そのモデルを用いた情報統合フレームワークの提案、上位情報系コンピュータと FA 現場間の情報の標準化と通信プロトコルの標準化を進めることにより、FA 現場のネットワーク化を促進することを目的とする。

具体的には、従来の生産システムモデル専門委員会が開発してきた OpenMES と、オープンコントローラ専門委員会／ワーキンググループ５（WG5、管理データ WG）が作成してきた工作機械管理データフォーマットおよび FA イントラネット推進協会ネット推進協会の開発した SNMP を活用した工作機械管理手法を結合してより広範囲な情報統合を目指すものである。これにより、我が国の多くの工作機械メーカが提供しようとする機器の現場情報を上位の情報処理に提供することが可能であると考えている。

３．委員会の進め方

生産システムの情報を統合するために以下の活動を行うことにする予定である。

１） 生産システムの統合を目指すための概念構成図の作成。

２） OpenMES フレームワークにおけるコンテンツの確認。

３） OpenMES の普及と標準化活動。

４） SNMP 仕様のシステムと CORBA 環境の接続性の検討。

５） 現実的なブラウザを持つ FA 端末のガイドライン規定を検証すること。

６） OA 機器と同様に FA 機器をモニタできるようにするために、FA 機器の MIB 定義を工作機械のみならず FA 機器全般を見通して行うこと。

７） コンテンツの流通を促進するため、流通するコンテンツのガイドラインを検証・追加策定すること。

生産システム情報統合専門委員会においては、生産システムモデル WG（WG1）と生産システム情報管理 WG（WG2）の二つの WG を設置し、連携を取りながら活動を行う。WG1 では、上記の２)、３）の項目を、WG2 では、４)、５)、６)、７）を行う。１）については、WG1 および WG2 のコアメンバにより Adhoc-WG を作成して検討を行う。OpenMES と情報管理ネットワークの統合を計る場合は、そのインタフェースの検討なども Adhoc-WG により適宜行う。

4. 体制

4.1 委員名簿

（1） 生産システム情報統合専門委員会

● 委員長

福田　好朗　　法政大学　　工学部経営工学科　教授

● 委　員

松田　三知子　　神奈川工科大学　　工学部情報工学科　教授
光石　衛　　東京大学大学院　　工学系研究科産業機械工学専攻　教授
武藤　一夫　　職業能力開発総合大学校　　福祉工学科　講師
稲鶴　勇　　ヤマザキマザック(株)　　開発設計事業部
制御開発第２部プロダクションセンタグループ　主事補
岩渕　修　　(株)東芝　府中情報・社会システム工場
計測制御機器部エンジニアリングツール開発担当　主査
大石　重雄　　豊田工機(株)　　技術研究所　研究開発部情報技術開発室　主担当員
黒岩　惠　　トヨタ自動車(株)　　情報事業企画部　担当部長
高橋　秀夫　　富士電機(株)　　システム機器事業部統合コントローラ推進部
統合コントローラ部設計課　課長補佐
田口　方昭　　(株)ソフィックス　　システム開発部特定システム部　部長代理
鳥澤　由男　　オークマ(株)　　ＦＡシステム事業部ＩＴ製品部　ＩＴプラザ課
成谷　文明　　オムロン(株)　オープンコントロール事業開発室
第１ＳＥグループ　主事
橋向　博昭　　(株)山武　　制御機器事業部プロダクト事業統括部製品企画部　次長
藤井　達哉　　豊田工機(株)　　技術研究所　研究開発部情報技術開発室　担当員
藤田　悟　　三菱電機(株)　　名古屋製作所開発部　専任
三宅　慶幸　　オムロン　インダストリアルオートメーションビジネスカンパニー
システム機器統轄事業部　技術部
森田　尚起　　(株)森精機製作所　　技術情報センタ営業技術課　係長

（2） WG1

● 主査

福田　好朗　　法政大学

● 委員

松田　三知子　　神奈川工科大学

岩渕　修　　(株)東芝
大石　重雄　　豊田工機(株)
黒岩　惠　　トヨタ自動車(株)
成谷　文明　　オムロン(株)
藤田　悟　　三菱電機(株)

（3） WG2

- **主査**

武藤　一夫　　職業能力開発総合大学校

- **委員**

光石　衛　　東京大学
稲鶴　勇　　ヤマザキマザック(株)
高橋　秀夫　　富士電機(株)
田口　方昭　　(株)ソフィックス
鳥澤　由男　　オークマ(株)
橋向　博昭　　(株)山武
藤井　達哉　　豊田工機(株)
三宅　慶幸　　オムロン
森田　尚起　　(株)森精機製作所

（4） Adhoc-WG

- **主査**

松田　三知子　　神奈川工科大学

- **委員**

稲鶴　勇　　ヤマザキマザック(株)
大石　重雄　　豊田工機(株)
田口　方昭　　(株)ソフィックス
鳥澤　由男　　オークマ(株)
橋向　博昭　　(株)山武
藤田　悟　　三菱電機(株)
森田　尚起　　(株)森精機製作所

4.2　委員会および WG 開催状況

（1） 生産システム情報統合専門委員会

第１回　　平成１２年８月３１日（木）　　製造科学技術センター　会議室
第２回　　平成１３年１月１２日（金）　　製造科学技術センター　会議室

（２） WG1

第１回　　平成１２年９月２９日（金）　　製造科学技術センター　会議室
第２回　　平成１３年１月１２日（金）　　製造科学技術センター　会議室

（３） WG2

第１回　　平成１２年９月２８日（木）　　製造科学技術センター　会議室
第２回　　平成１２年１０月２０日（金）　製造科学技術センター　会議室
第３回　　平成１２年１１月２２日（水）　製造科学技術センター　会議室
第４回　　平成１２年１２月１４日（木）　製造科学技術センター　会議室
第５回　　平成１３年１月１２日（金）　　製造科学技術センター　会議室
第６回　　平成１３年２月２３日（金）　　製造科学技術センター　会議室
第５回　　平成１３年３月２３日（金）　　製造科学技術センター　会議室

（４） Adhoc-WG

第１回　　平成１２年９月２２日（金）　　製造科学技術センター　会議室
第２回　　平成１２年１０月１９日（木）　製造科学技術センター　会議室
第３回　　平成１２年１２月１１日（月）　製造科学技術センター　会議室

５． OpenMES の普及活動

OpenMES の普及活動は、ワーキンググループ１（WG1）において実施した。以下、その活動を報告する。

５.１ OpenMES の概要

５.１.１ MES とは

生産システムは、この数年の間に様々な変革を迫られている。代表的な問題点は、生産量の急激な変動に耐えられること、生産情報の利用範囲が拡大していてシステムの守備範囲が大きくなってきていること、生産情報の連携が重要視されてきていることなどがあげられる。このような問題点を解決する方法として、生産管理や販売情報と生産の進捗情報や制御情報と緊密に連携する方法が提案されてきている。その一つが、MES（Manufacturing Execution System）と呼ばれるものである。

MES システムは、MESA International から図 ５－１のような概念として提案されている。

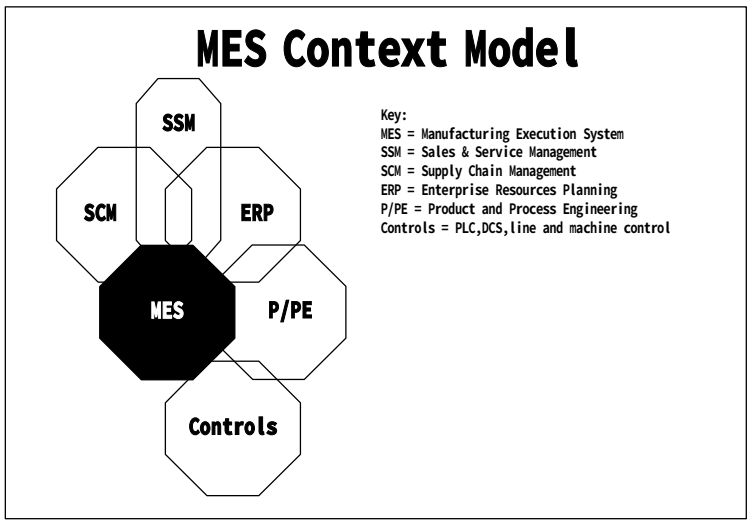

Figure 2: MES Context Model

MES provides an information hub that links to and sometimes between all of these systems. MES overlaps with other manufacturing system types, which also overlap with each other. For example, schduling may appear in both MES and SCM; labor management in MES, SSM, and the HR function of ERP; document control in MES and P/PE; and process management in both MES and Controls. Degrees of overlap vary by industry and implementation.

図 ５－１ MES と他の機能との関連

MES は、生産システムあるいは製造企業システム内の SCM（サプライチェーンマネージメント）、SSM（販売サービス管理）、ERP(企業資源管理)、P／PE（製品技術と生産技術）、Control（制御）の機能を結びつける位置に存在するものと考えられている。これは、製造企業のそれぞれのシステムをつなぐ機能であることを示している。

MESA International の白書において、MES に関する定義は、「MES は、発注から製品の生産

活動までを最適化にする情報を配信するものである。MES は、現在の状態を表す正確な情報を用いて、状態が発生する都度、応答する。変化への迅速な応答は、付加価値を生まない活動の削減と併せて、効率的な工場の運営につながる。MES は、配送、在庫、資産運用の改善に役に立つ。MES は、双方向のコミュニケーションを通して、企業全体、サプライチェーン全体にまたがった生産活動に関する情報を提供する。」である。

一般的に、製造企業のそれぞれのシステムは、それぞれのシステムの持つ特長に合せて開発されてきている。特に、生産システム内で重要な役割を果たしている生産管理情報、製品開発情報、製造情報は、その情報の取り扱われる時間の管理間隔、情報のボリュームなどの相異から、それぞれが独自の分野として開発してきている。たとえば、製造情報は、リアルタイムに情報が取り扱われ、ボリュームはそれほど多くないが、生産管理情報は、時間単位あるいは日単位で情報処理がなされ、その処理ボリュームは大きい。さらに製品開発情報は、日単位あるいは年単位で取り扱われ、ボリュームは非常に大きい。このことから、製造情報は、リアルタイムで処理されるような言語や設備など特殊な世界で開発されてきている。生産管理情報や製品開発情報は、バッチ処理に適した言語やシステムとして開発されてきている。このように、これらの生産システム内の情報を統合するのは、情報の性格、情報処理システムの性格などから非常に困難であるとされてきている。

しかしながら、前述したような生産システムの情報の統合は、現代の生活サイクルに合せるために生産リードタイムを減少しなければならなかったり、生産量の急激な変化に対応しなければならなかったり、品質の保証を行わなければならなかったり、生産の必要性だけでなく、市場の要求によってその重要性が強くなってきている。このことから、企業全体あるいはサプライチェーン全体に対して生産活動の情報を提供する必要が生じている。また、生産活動は、企業全体あるいはサプライチェーン全体からの情報を得て大規模なシステムに適したシステム運営を行わなければならない。つまり、生産活動の情報を他の機能に提供するとともに他の機能からの情報も入手する双方向のリアルタイム情報流通機構が必要となる。それが、MES である。

MESA　International では、図　５－１で示したような MES が実現できた場合の効果を、アメリカで MES に相当するシステムを導入している企業にアンケートを実施して、つぎのような回答を得ている。

- 生産サイクルの大幅な短縮（３５%)、
- 生産管理用データ入力時間の短縮（３６%）
- 製品品質の向上（２２%）
- 現場が必要な時に必要な情報を必要な時に受けられる
- 特急品などの外乱に対しても耐えられる
- 市場の変化に対応した迅速な生産体制が構築できる

実際に、このようなシステムを作成するためには、将来の市場や設備などのシステム変動を見越して設計を行わないと将来に対する柔軟性が欠落するなどの問題点が指摘されている。

このような問題を解決して、将来の技術変化や環境変化に対応するために、MES を標準化して広範囲なアプリケーションの結合が可能にするような努力をすべきであると MES 白書では指摘している。将来の問題を解決して、より広範囲な利用を意識した MES を開発するためには、対象となる生産システムの機能をモデル化すること、そのモデルでの情報の関連を明確にすること、共通的なプラットホームの定義などが必要であるともしてきている。このようなアプローチは、半導体の製造業が用いているＳＥＭＡＴＥＣのＣＩＭモデルのアプローチを参考にしている。

５.１.２　MES の必要な背景

（１） なぜフレームワークなのか

MES（Manufacturing Execution System）とは生産システムにおいて ERP 等からの生産計画および生産指示を受け、製造設備に作業指示を出し、実績収集、進捗管理を行う工程管理を中心とした統制システムである。MES の構築、調達において次の問題が存在する。

- 生産システムは、数多くのサブ・システムから構成されるが、標準化されたインタフェースが存在せず有機的に統合することが難しい。
- 生産システムは、他種類の製造設備から構成されるマルチベンダ環境が普通であるが、製造設備を管理する標準的なインタフェースが存在しないため、個別の作り込みによって製造設備をシステムに取り込む必要がある。
- 生産現場の環境の多様さに起因する様々な特殊要件が存在するため、システムのパッケージ化が難しく、生産現場ごとに個別に作り直す必要がある。

こうした理由から、生産システムの開発・保守には多大なコストがかかり、また真に統合された生産システムが存在しないために、システム全体にわたる生産の最適化を計ることが困難となっている。

しかし実は MES には共通なモデルが存在し、そのモデルに従ったソフトウェア部品を用意することにより、個々の生産現場にあった MES の構築を、必要なソフトウェア部品の組み合わせとそのカスタマイゼーションによって行うことが可能であると我々は考えている。このようなアプローチの実現のためにオブジェクト指向を採用した。オブジェクト指向によって、生産システムをモデル化しソフトウェア部品に分解することができる。また同時に、これらのソフトウェア部品間の連携を定義することにより、構築される生産システムのテンプレートを予め与えることができる。このテンプレートは MES アプリケーションフレームワークと呼ばれ、新たな機能部品の追加や、既存機能部品の振る舞いの変更などにより、必要なカスタマイゼーションを容易に行うことが可能である。

MES アプリケーションフレームワークには機能面以外にも次の要件が存在する。

- 小規模生産ラインから大規模生産ラインまで対応できるスケーラビリティを持つこと
- ハードウェア、OS 等のプラットホームに依存しないオープン性を持つこと

このため CORBA の分散オブジェクトを用い、複数のサーバにオブジェクトを分散させ、また実装言語として JAVA を採用することによりスケーラビリティとオープン性を確保する。

こういった MES アプリケーションフレームワークを導入することにより、現場の情報化が実現し、必要なデータにどこからでもアクセスできるようなシステムを短期間・低コストでユーザ自身が構築することが可能になる。さらにこのような情報化を推し進めることにより、ERP（Enterprise Resource Planning)、SCM（Supply Chain Management)、設備制御システムと協調して最適な生産が実現可能となる。

（2） なぜ CORBA なのか

MES は複数の設備、サーバ、クライアント端末から構成され、本質的に分散処理が要求される。また分散処理によって小規模な生産システムから複数の工場を含む生産システムまでサポートできるスケーラビリティを実現することができる。さらに拡張性、オープン性の観点からソフトウェアのコンポーネント化、プラグイン性が要求されている。このような環境を実現するプラットホームとして最近は

- DCOM
- CORBA

が注目されている。MES の場合、ハードウェアに関しては、計算機の種類が多岐に渡り、PC、ワークステーション、ビジネスコンピュータ、大型コンピュータから構成される。このような環境でシームレスなシステムを構築していくためにはプラットホームに依存しない CORBA が適している。

（3） フレームワークのメリット

(a) エンドユーザ

MES 導入によるメリットは次の通りである。

- Web Browser によって生産指示情報、生産実績情報にどこからでもアクセスすることが可能になり、意志決定が迅速・正確になる。例えば営業員は在庫状況、仕掛かり状況から納期を返答することが可能になり、また現場管理者は来期の生産スケジュールを知ることにより、負荷の集中や仕掛かり在庫を減少させることが可能になる。
- SCM/ERP との連携によって全社的な最適化が可能になる。現場の情報が SCM/ERP に反映されるため、販売、生産、調達、物流にわたる最適化が行えるようになる。

- 生産ライン改善のための Plan Do Check Action に必要なデータが得られる。Plan はスケジューリング、Do は作業指示、Check は進捗管理、Action は実績データであり、それぞれの局面において必要なデータが生成・加工される。また作業日報等をわざわざ作成することはない。
- MES による指示により複数の工場があたかも一つの工場として機能することができる。製品種別ごとに工場の組み合わせを柔軟に変更することができる。
- MES フレームワークのメリットとしては既存のソフトウェアコンポーネントを調達してシステムを構築できることがあげられる。

(b) システムインテグレータ

MES アプリケーションフレームワーク導入によるメリットは次の通りである。

- 生産装置のマルチベンダ環境を容易に構築できる。
- これまでに開発・使用されたコンポーネントを用いることで短期間で高品質な MES を構築できる。
- Web を用いた先進的なソリューションを提案、提供できる。

(c) コンポーネントプロバイダ

MES アプリケーションフレームワーク導入によるメリットは次の通りである。

- 自社のノウハウを組み込んだコンポーネントの提供によるビジネスが可能となる。

(d) 装置メーカ

MES アプリケーションフレームワーク導入によるメリットは次の通りである。

- 生産システムごとの、装置の繋ぎ込みが不要となる。

5.1.3 OpenMES とは

WG1 の前身である昨年度まで３年間活動していた JOP 生産システムモデル専門委員会では、前述の MES の要件を考慮して、かつ、さまざまなアプリケーションが自由に接続でき、MRP と現場の制御情報、実績情報が相互に連携できるようにすることを目的とした MES を OpenMES と名づけている。OpenMES は、フレームワークを理解するためのオブジェクトモデルとそれを用いて実装したフレームワークの両者が存在している。後者のフレームワークは、(財) 製造科学

技術センター（MSTC）が情報処理振興事業協会（IPA）の委託により開発した。WG1 で普及活動を行っている OpenMES は、このフレームワークを実装するときに用いたモデルに関するものである。

OpenMES のモデルでは、オブジェクト指向モデルを用いてそれぞれの機能と情報の関係を定義している。その対象は、組立、加工を中心としたディスクリート製造としている。当然のことながら、ディスクリート製造の範囲は広いのでモデル化の作業量が膨大になる。ここでは、具体的なイメージとして取り扱いやすい機械加工の FMS とした。FMS は、工作機械（CNC）、人の作業（段取りステーション）、倉庫、運搬、検査が含まれており、生産システムの基本的構成要素がすべて含まれているので、将来の適用性を考慮することができ、さらに、情報の流れが明確になっているので、整合性の検証がしやすいためである。対象としている範囲は、図 ５－２に示すような領域である。モデルの詳細は、添付資料の OpenMES 仕様書を参照のこと。

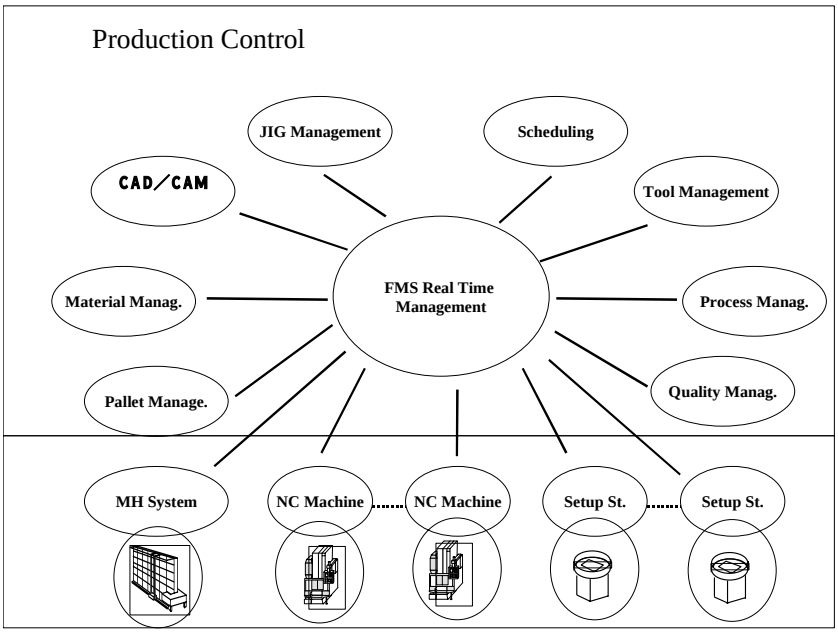

図 ５－２　OpenMES の対象範囲

OpenMES 仕様書は、このモデルを規定しているものであるが、このモデルをもとに、CORBA の環境において実装されることを念頭において作成されている。

この結果、OpenMES は、小規模な生産システムから大規模な生産システムまで、一つのフレームワークで構築できるので、段階的なシステムの成長に対応することが出来る。また、ハードウェア、OS 等に依存しないオープン性が確保されているので、システムの再構成、部分変更などに強いという特徴を有している。

OpenMES の仕様書は、現在開発されている機能は、工場管理、製造指示管理、製造仕様管理、工程仕様管理、工程管理、設備管理、搬送管理、資源管理、スケジュール管理の９つの機能である。これらの機能が行っている処理は、UML に準拠した表現によって記述されている。また、仕様書では、次の 2 つの理由から Java の API として仕様を定めている。通常 CORBA を採用した

フレームワークの仕様は IDL で記述されることが多いが、アプリケーションプログラマが CORBA の仕様に精通していなくとも開発できるようにする、フレームワークのパフォーマンスを高めるため、必要なオブジェクトだけを CORBA オブジェクトとするなどの理由から Java で記述してある。また、サーバとクライアントとで共有する必要がある非 CORBA オブジェクトは両者の間での値渡しとしている。この機能は現在 CORBA2.0 では規約化されていない。Visibroker のベンダ固有の機能として提供されている。したがって、この機能は標準的な IDL では記述できない。

提案している OpenMES のフレームワークのカスタマイズには次の 5 段階が考えられる。

- GUI 開発：Visual Age Java, Jbuilder, Visual Cafe 等の開発環境あるいは JDK を用いて、フレームワーク API を呼び出す Graphical User Interface の開発・変更を行う。
- アプリケーション開発：フレームワーク API を呼び出すアプリケーションのロジックをクライアント側あるいはサーバ側に実装する。サーバ側の場合、クライアントとの HTML の授受にはサーブレットや JSP を用いるのであれば、その設計・実装も行う。
- データのカスタマイズ：個々の MES ごとにデータをカスタマイズすることを想定しているクラスが用意されているので、必要なメンバ変数・メソッドを追加する。
- CORBA インタフェースのカスタマイズ：設備インタフェース、スケジューラインタフェースなど CORBA インタフェースにメンバ変数・メソッドを追加する場合は、CORBA インタフェースの導出インタフェースを定義し、そのインタフェースを実装する。それに伴い、新しいメソッドを呼び出すアプリケーションを開発することになる。また CORBA クラスに変数を追加する場合、実装クラスの導出クラスを定義・実装する。CORBA インタフェースを変更せず、ロジックを変更する場合は、実装クラスの導出クラスを定義し、メソッドを実装する。
- フレームワーク機能の追加：仕様書で規定されているクラス定義を呼び出すことにより、再利用を考慮した MES アプリケーションフレームワークの機能グループ・コンポーネントを追加する。分散処理が必要な場合、あるいは適切な場合は CORBA インタフェースを新規に定義し、実装する。

5.1.4 OpenMES フレームワーク

（1） フレームワークの範囲

MES が管理対象とする製造工程には、一般的に下記の三つの形態があると考えられる。

- 連続プロセス
- バッチ・プロセス

- ディスクリート・プロセス

オープン MES では、この内のディスクリート・プロセスをモデル化の対象として選択した。ディスクリート・プロセスとは、ロットなどの単位でまとめられたワークが、工程ごとに区切られた製造設備間を移動しながら、加工を施されるものである。機械加工や組立の工場がこれに相当する。

ディスクリート・プロセスには、さらにコンベヤ・ラインの様に固定された工程経路をとるフロー・ショップと、工程経路に自由度のあるジョブ・ショップに分かれる。オープン MES では、ジョブ・ショップに対応したモデルを用意し、その特殊形としてフロー・ショップに対応する。

フレームワークに含まれる機能は以下の通りである。

(a) 工場管理

本機能グループは工場および個々の設備スケジュールを管理する機能を提供するクラスから構成される。設備の保守時期も本機能グループを用いて表現される。またスケジュールに対する実績値も管理する。

(b) 製造指示管理

本機能グループは、生産計画システムとのインタフェースを提供する。製造指示を受けて必要な数のロットを作成し、工程に投入し、その製造指示の進捗を監視することができる。すなわち、製造指示と工程内を流れるロットの間の紐付けを管理する。

この紐付けは、受注生産か見込み生産かといった生産形態、MRP や製番管理といった管理方式、ロット分割など、様々な要因で変わってくる可能性がある。

(c) 製造仕様管理

本機能グループは、ロット編成や工程別のセットアップ・データ、加工データなど製品の仕様に関わるデータを管理する。

(d) 工程仕様管理

本機能グループは工程経路、工程資源を管理する機能を提供する。工程経路は工程ステップのシーケンスであり、この情報をもとに工程内作業指示が生成される。

(e) 工程管理

本機能グループは、ロットに対する製造工程の実行すなわち設備に対し作業指示を行い、作業進捗を追跡する。工程管理の中心になるのは、ロットに対する作業計画と作業実績を管理する LotJob と呼ばれるクラスと、作業計画に基づいて作業指示を各設備に差し立てる Dispatcher と呼ばれるクラスである。

ここでロットと呼ばれるのは、工程内を流れる物理的なロットと対応する製造ロットである。今回は、投入から完了までロットの統合や分割は行わないものとしてモデル化した。また、製造

指示との紐付けは、製造指示管理機能グループで取り扱うものとした。

以下に主なクラスの概要を示す。

- LotJobManager
  このクラスは、ロット・ジョブ全体を管理する。ロット・ジョブの登録や検索などの機能を提供し、また分散環境下では、ロット・ジョブの分散配置を行う。
- LotJob
  このクラスは、個別のロットにたいする作業計画と作業実績を管理する。ロットの一時停止・再開や作業実績の登録など、ロットに対するトランザクションは全て LotJob を通して行われる。
- ProcessJob
  このクラスは、個別のロットに対する工程別の作業計画と作業実績を管理する。
- Dispatcher
  このクラスは、作業指示を各製造設備に差し立てる機能を提供する。通常は Dispatcher は工程資源ごとに作成され、各資源に割り当てられている ProcessJob のリストを管理し、複数の代替え設備が同一資源に割り当てられている場合、設備間の競合を調停する役割を持つ。
- WorkOrder
  このクラスは、設備に差し立てられる作業指示を表す。ロットに関する情報や、着手予定時間や優先順位を属性として持つ。
- WorkInstruction
  このクラスは、作業指図を表す。作業指図の内容は基本的に実装依存である。、例えばセットアップ・データ、パート・プログラム、或いは作業者教示 (オペレータ・インストラクション) などが含まれる。

(f) 設備管理

本機能グループは、工場内の製造設備と MES 間の作業指示、作業実績、アラーム等のインタフェースとなる機能を提供する。この機能グループは、個々の製造設備に対応するクラスと、それらの設備をまとめて管理するクラスから構成されている。

ここで扱う設備は、物理的な実体を持たない抽象的な設備である。抽象設備は、下記の３つの性質を持つものとしてモデル化されている。

- 稼働状態の管理単位としての設備（MesEquipment）
- 工程の実行単位としての設備（MesProcessEquipment）

- 制御装置の単位としての設備（MesProcessEquipmentAdaptor）

以下に主なクラスの概要を示す。

- Equipment
  このクラスは、稼働状態の管理単位としての設備を表す。設備の起動・遮断などの操作と設備の稼働状態（遮断中、稼働中、非常停止中など）の取得、アラームの通知などの機能を提供する。
- ProcessEquipment
  このクラスは、工程の実行単位としての設備を表す。Equipment クラスを継承し、作業指示の差し立てや、作業の開始・終了などの状態を管理する機能を提供する。
- EquipmentManager
  このクラスは、設備全体の管理を行う。設備の登録情報、初期化データ、稼働状態などを一元的に管理する。各設備は始動時に EquipmentManager に初期化データを問い合わせ、また稼働中は、稼働状態の遷移を通知する。

（g） 搬送管理

本機能グループは、ロットを搬送する機能を提供する。

ただし搬送システムは、通常専門の搬送装置メーカによって一括してインストールされることが多い。従って、搬送システム内の AGV や自動倉庫の標準モデルが存在し得るかどうかは不明である。このためオープン MES では、搬送システムのインタフェースの抽象度に段階を設け、最も高い抽象度では、搬送システム全体をブラック・ボックスとして扱えるように考慮した。

（h） 資源管理

資材管理機能グループは、資材の利用実績を管理する。ただし、資材の在庫管理全体まではカバーしておらず、例えば資材の種別ごとのマスタ情報などは持っていない。これは、資材の在庫管理は ERP などの外部システムが行うものと仮定しており、ここで用意したクラスは、外部システムに対して利用状況を通知するためのインタフェースとして使われることを想定しているためである。

（i） スケジュール管理

本機能グループは、スケジュール作成機能を提供する。ただし、スケジューラは既に独立した製品としてパッケージ化が進んでいる。従ってオープン MES でもスケジューラは既存のパッケージの存在を前提として考えている。

スケジューラ本体はラッパーである Scheduler クラスを介して呼び出される。Scheduler クラスは、投入予定のロット・ジョブ、工程仕様、設備情報、製品仕様、投入済みのロット・ジョブ

の作業実績などに関するデータを取り込んでスケジューラに渡し、そのスケジューリング結果を受け取って、ロット・ジョブに対するプロセス・ジョブの登録を行う。

（2） カスタマイズ例

(a) 現場端末とバーコードリーダ

実証例は FMS を対象にしてきたが、生産システムには作業員も存在し、自動機械しか存在しないような生産システムはむしろ少ない。作業員がアクターとなる代表的なユースケースには2通り考えられる。

- 現場端末から自分が作業すべき作業指示を読み取り、作業が終われば端末の完了ボタンを押すことで実績を上げる。
- 現品についてくる現品票から自分の作業を認識し、作業が完了した時に現品票に印刷されているバーコードを読みこませ、実績を上げる。

前者の場合であれば、作業指示を送る設備を工作機械ではなく現場端末とする。送られてきた作業指示を画面に表示するように、設備管理機能グループで定義されている工程設備クラスの導出クラスを定義し、現場端末で実行させておく。作業の開始時点を把握する必要があるのであれば、現場端末に開始ボタンを設定する。開始ボタンや完了ボタンが押されると、工作機械が実績を上げる場合と同様に工程管理機能グループのメソッドを呼び出すことで実績を管理する。

後者の場合であれば、作業指示は現品票の印刷に相当しているので、プリンタが繋がっているPC で、この PC に送られてくる作業指示が現品票として印刷されるように工程設備クラスの導出クラスを定義して実行させておく。バーコードが繋がっている PC も同様に工程設備クラスの導出クラスを定義し、バーコードが入力されたら実績として工程管理機能グループのメソッドを呼び出すことで実績を管理する。

(b) フロー・ショップ

実証実験では FMS を対象にしたので、ジョブ・ショップを想定していた。コンベアで設備が直線的に結ばれているようなフロー・ショップも世の中に数多く存在する。生産ラインに対しMES が管理すべき対象をどのように考えるかによりカスタマイズ結果のモデルが異なる。

一番単純なのは、ライン全体を一つの工程設備とみなすことである。一つのロットに対し、一つの作業指示が対応づけられる。ラインの投入口に作業指示を出し、ラインから製品を出すときに実績を上げる。この場合、ラインに一旦入ったら出てくるまで進捗を把握できない。ライン内の設備に対する作業指示は SCADA 等が行うことになる。

ライン内の進捗を知るためには、ラインの途中にチェックポイントを数カ所に設定する。チェックポイントはバーコードリーダやリモート ID のライタ・リーダである。チェックポイントを製品が通過した時点で実績を上げる。この場合はチェックポイント間を一つの工程設備とみなして

MES を構成することができる。

さらに詳細な実績を取る必要があるのであれば、あるいは作業指示をライン中の設備、作業者に対して MES が出す必要があるのであれば、あたかも搬送系が存在しないジョブ・ショップとして構成することもできる。ある設備での作業が終われば次の設備に対して作業指示が出される。

どういった構成を取るかは、MES と設備系との担当範囲の切り分け方や、要求されるパフォーマンスによる。

（3） 実用化にむけて

以上のように、OpenMES はディスクリート系の生産システムを対象に仕様が定義されており、柔軟にカスタマイズすることができる。また実証実験を通し、ロットが数 100 個までは十分なパフォーマンスがあることも確かめられている。パフォーマンスが低下するような大規模なシステムの場合は、工程管理機能を分散処理するように構成することができる。

5.1.5 実証例

OpenMES への設備接続実証例としてマシニングセンタ用 OSEC NCKit [注)] の接続方法を以下に示す。

（1） インタフェース

インタフェースを図 5－3に示す。

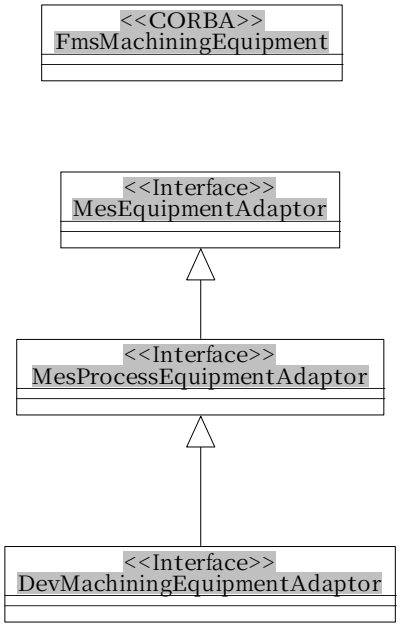

図 5－3 インタフェース

[注)] OSE協議会のNC装置評価用キット(http://www.sml.co.jp/osec/)

(a) FmsMachiningEquipment クラス

マシニングセンタ設備の CORBA インタフェース

CORBA サーバとして外部よりアクセスする為のインタフェース

マシニングセンタ設備を実装する時に使用するインタフェース

(b) MesEquipmentAdaptor クラス

抽象設備用の制御装置を実装する時に使用するインタフェース

(c) MesProcessEquipmentAdaptor クラス

抽象工程設備用の制御装置を実装する時に使用するインタフェース

(d) DevMachiningEquipmentAdaptor クラス

マシニングセンタ用の実装インタフェース

FmsMachiningEquipment クラスを元に作成する実装クラスから呼び出されるインタフェース

（2） 制御装置の実装

制御装置の実装クラス図を図 ５－４に示す。

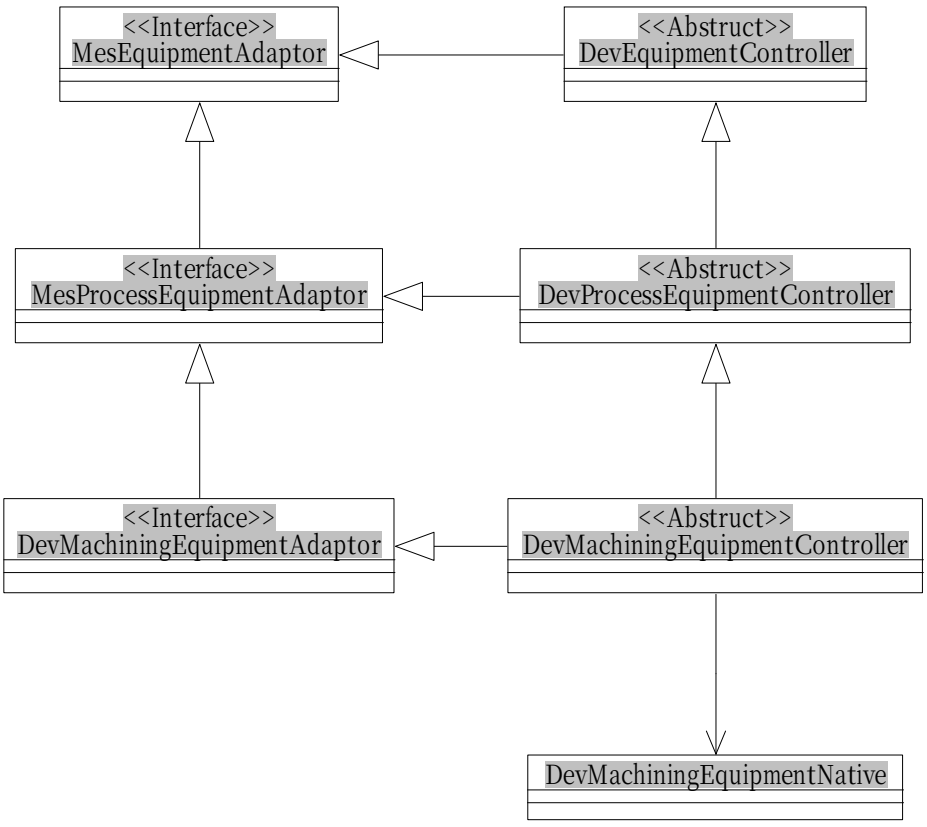

図 ５－４　制御装置の実装クラス図

（a） DevEquipmentController クラス

設備用の制御装置の実装

MesEquipmentAdaptor インタフェースクラスを実装した抽象設備制御装置クラス

（b） DevProcessEquipmentController クラス

工程設備用の制御装置の実装

MesProcessEquipmentAdaptor インタフェースクラスを実装し、DevEquipmentContller クラスから派生した抽象工程設備制御装置クラス

（c） DevMachiningEquipmentController クラス

マシニングセンタ用の制御装置の実装

DevProcessEquipmentAdaptor インタフェースクラスを実装し、
DevProcessEquipmentContller クラスから派生した抽象マシニングセンタ制御装置クラス

（d） DevMachiningEquipmentNative

マシニングセンタ用のネイティブインタフェース

DevMachiningEquipmentController より呼び出されるネイティブインタフェース

このクラスを通じてネイティブライブラリを呼び出す。

（3） OSEC NCKit の接続

OSEC NCKit の接続クラス図を図 ５－５に示す。

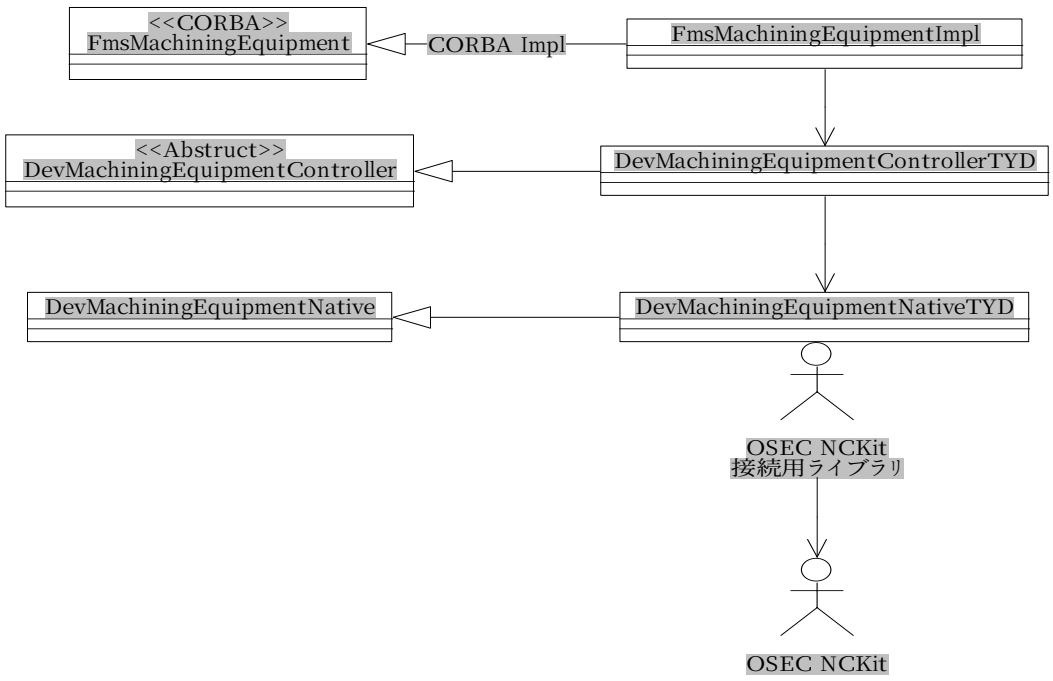

図 ５－５ OSEC NCKit の接続クラス図

（a） FmsMachiningEquipmentImpl クラス

マシニングセンタ設備用の CORBA サーバ実装クラス

tie メカニズムを使用し実装を行った。

（b） DevMachiningEquipmentControllerTYD

OSEC NCKit 用の制御装置クラス

（c） DevMachiningEquipmentNativeTYD

OSEC NCKit 接続ライブラリ用のネイティブインタフェース

（d） OSEC NCKit 接続用ライブラリ

OSEC NCKit 接続用の C ライブラリ

JNI 機能でヘッダファイルを作成し実装を行った。

（e） OSEC NCKit

OSEC NCKit 本体

（４） 機能の実装（半自動実行機能）

機能の実装例として半自動実行機能の処理シーケンス図を図 ５－６に、コーディングサンプルを図 ５－７および図 ５－８に示す。

（a） 半自動実行処理シーケンス図

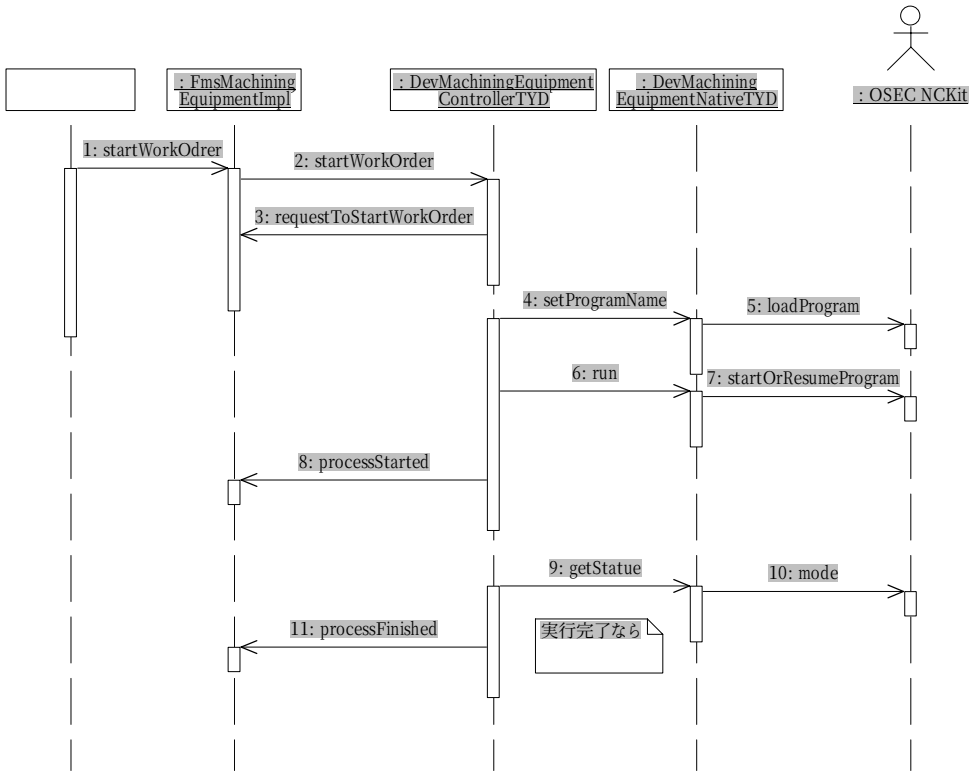

図 ５－６ 半自動実行機能の処理シーケンス図

(b) コーディングサンプル

- DevMachiningEquipmentControllerTYD クラス（Java）

```
/**
 * 工程内作業の開始が要求されたときに呼び出される
 * @param aWorkOrder 工程内作業指示
 */
public void startWorkOrder(com.ibm.mes.fwk.equipment.MesWorkOrder aWorkOrder)
throws MesEquipmentAdapterException {
      :
  // 工程内作業の開始
  getProcessEquipmentCallback().requestToStartWorkOrder(aWorkOrder);
      :
}

/**
 * 制御装置の作業を開始する
 */
public void startController()throws MesEquipmentAdapterException{
      :
  // プログラムをセットし起動
  mNative.setProgramName(mHandle,mNCProgramFile[mNCProgramFileCount].getAbsolutePath());

  // 開始
  System.out.println("run()");
  mNative.run(mHandle);
     :
  // 工程内作業が始まったことを通知
  getProcessEquipmentCallback().processStarted(mWorkOrderRequest.mWorkOrder);
      :
}

/**
 * スレッドから呼び出される定期実行メソッド
 * @return コントローラが実行可能な状態なら true を返す
 */
public boolean equipmentRun() {
      :
  // 作業が完了したら完了通知を行う
  if(mNative.getStatus(mHandle) == mNative.cReset) {
     :
    // 工程内作業が完了したことを通知
    getProcessEquipmentCallback().processFinished(mWorkOrderRequest.mWorkOrder);
      :
  }
}
```

２で呼び出されているメソッド

３の処理

４の処理

６の処理

８の処理

９の処理

１１の処理

注）図注の番号はシーケンス図の処理番号

図 ５－７　コーディングサンプル１

- OSEC NCKit **接続用ライブラリ**

```
switch(iPNCMachine->ncResource()->eiaDataDecoder()->mode()) {
case McEIADataDecoder::opmReset:
    pMroIF->status.status = cReset;
    break;
case McEIADataDecoder::opmStop:
    pMroIF->status.status = cStop;
    break;
case McEIADataDecoder::opmRunning:
    pMroIF->status.status = cRunning;
    break;
case McEIADataDecoder::opmHalt:
    pMroIF->status.status = cHalt;
    break;
}

    /* 起動コマンドイベント */
    if(pMroIF->command.data == RUN) {
        iPNCMachine->ncResource()->eiaDataDecoder()->startOrResumeProgram();
    }
    /* プログラム設定コマンドイベント */
    else if(pMroIF->command.data == SETPROGRAM) {
        iPNCMachine->ncResource()->eiaDataDecoder()->loadProgram(iNcProgram);
    }
```

５の処理の中身

６の処理の中身

４の処理の中身

注）図中の番号はシーケンス図の処理番号

図 ５－８　コーディングサンプル２

５.１.６　普及活動

（１） OMG（Object Management Group）との協業活動

1999 年 5 月に OMG TC ミーティングが東京で開催され、Manufacturing Domain Task Force（以下 Mfg Dtf）において、FA オープン推進協議会の各専門委員会が活動状況の紹介を行った。その結果、OMG と JOP では標準化対象範囲が共通しているものが多く、共同で標準化を進めたいと、Mfg Dtf の議長より提案があった。ただし専門委員会により態度が異なるため、JOP 全体

で対応するのではなく、専門委員会ごとで対応することとした。

その後 7 月に開催された JOP 運営委員会において、OMG との協業が承認され、OMG ミーティングへの出席費用は JOP が負担することになった。また、JOP は OMG の会員ではないため提案権がないので、リエゾン会員を介して提案することになった。Mfg Dtf からは、生産システムモデル専門委員会（当時）からのモデルの紹介を受け、標準化する部分、スケジュールを議論し、ロードマップに反映させたいという意向を受けた。

2000 年 1 月にアリゾナで OMG TC ミーティングが開催され、OpenMES の仕様を紹介し、ロードマップに関する議論、次回以降の Mfg Dtf ミーティングの議題について議論を行った。このミーティングでは OpenMES の仕様の英訳が完了しておらず、学会で発表した資料等をもとに OpenMES の説明を行った。MES モデルに関する議論では、MES モデルに関連する他の OMG モデルとの整合性が重要であり、関連モデルとのスコープ、機能グループの分け方、類似用語の比較が今後の課題として認識された。また Business Object Task Force で議論されている Work Flow Management モデルとの関係を整理することが必要であることが指摘された。

2000 年 9 月にサンフランシスコで OMG TC ミーティングが開催された。しかし、MES ベンダが ERP, SCM に注力しているため、MES モデルの策定が遅れていた。MES/MC WG の現在の議長が昨年で辞任する意向であったため、WG としての活動が終了することになる。議長としては、現在作成中の MES/MC Roadmap を、MES モデルと Mfg Dtf における他の WG における活動との関係、ならびに OMG における他の Dtf との関係を明確にした上で、将来への指針としてドキュメントを完成させることを予定している。OpenMES の英語版仕様は、OMG の Web 上で公開されている。OpenMES 関連の資料を公開している OMG の URL を参考資料に上げる。

- 2000 年 1 月議事録　ftp://ftp.omg.org/pub/docs/mfg/00-01-04.pdf
- 2000 年 9 月議事録　ftp://ftp.omg.org/pub/docs/mfg/00-09-06.pdf
- OpenMES 説明資料　ftp://ftp.omg.org/pub/docs/mfg/00-01-06.pdf
- OpenMES 英訳仕様　http://www.omg.org/cgi-bin/doc?mfg/2000-01-04

（2） OpenMES 仕様書

OpenMES 1.00J（日本語版）を添付資料とした。

６．製造装置からの情報発信

製造装置から情報発信を行うために仕組みについての検討を、ワーキンググループ２（WG2）において実施した。以下、その活動を報告する。

６.１　目的

近年のパーソナルコンピュータ、その OS（MS-Windows など）、そしてインターネットの普及により、TCP/IP という通信プロトコルと Web ブラウザという表示方法は、FA 分野、とくに生産システムの分野でもデファクトスタンダードになりつつある。しかしながら、FA 分野の制御装置（CNC）に求められる機能と位置づけは明確でなく、上位情報システムと生産現場の情報伝達は標準化を模索している段階である。

このような背景から、JOP オープンコントローラ専門委員会に管理系データモデルワーキンググループ（WG5）が 1996 年７月に設置され、４年間にわたる活動により、上位情報系コントローラと NC 装置間で実現される機能モデルが想定され、CNC から上位情報系に報告される実績情報の標準化案とそのファイルフォーマットとして CSV 形式を用いることを提案された。

一方、ＦＡイントラネット推進協会では SNMP を通信プロトコルとして採用して、上位情報系と CNC を繋ぐ標準化案を提案している。

そして、上記 WG5 では FA 機器からの管理情報を収集する手法（プロトコル）について、上記 FA イントラネット推進協会ですすめている SNMP を通信プロトコルの標準化をはかるために、JOP 生産システム情報統合専門委員会にワーキンググループ２（WG2）が設置された。

本 WG では、FA 用通信プロトコルに求められる、汎用性、リアルタイム性、通信負荷、容易な実装性を評価条件とし、さらに上記 WG5 が提案した実績情報の標準化案と、ＦＡイントラネット推進協会の提案するガイドラインを比較・評価・改定し標準化案（PIPI）として提案することを目的とする。

６.２　活動状況

管理系データモデルＷＧで提案された CSV 形式は、現存するアプリケーションへのデータ取り込みの容易性、生データの可読性を考慮し、さらに今後の HML への発展を見据えた結果であった。本 CSV 形式は、データ項目・データタイプの必要最小限の項目を必須項目として定義し、さらにさまざまなアプリケーションを考える上で必要とされるデータ項目を任意拡張項目として例示し、生産システムにおけるデータ授受の仕組みよりデータ表現方法の標準化に重点が置かれる内容となっている。実装の実現方法については、その必要性を認識しつつ、実現手段、またその組み合わせバリエーションの多さから特定のデータ授受手段に絞り込むことは適用範囲を狭め、本来の目的であるデータ項目・データタイプの標準化の普及を妨げる可能性があるとして、敢えてデータ授受手段について言及されていない。

一方、FA イントラネット推進協会（FAIPA）のコンテンツガイドライン、SNMP 適用概説書は、当該協会で３年間の活動により得られた成果物であり、ＦＡ環境、特に工作機械、周辺装置、工具、治具に至るまでデータが詳細に定義されている。そしてその通信プロトコルとして一般的にネットワーク機器の監視・管理用に使用されている SNMP をＦＡのネットワークに適用し、1999 年メカトロテックに JOP と共同実証展示された実績がある。ＦＡイントラネット推進協会の SNMP については、プロトコル自身は IETF で標準化されているもののプロトコル上で授受されるデータについては MIB（Management Information Base）にかかっており、ＦＡイントラネット推進協会で作成された MIB 定義が ERP/MRP・SCADA・MES・SCM など上位情報系情報システムからの観点で妥当性を確認することが必要とされた。

このような背景の中、上位情報系コンピュータとＦＡ現場間の情報の標準化と通信プロトコルの標準化を進めるにあたり、生産システムモデル専門委員会の OpenMES 仕様書、オープンコントローラ専門委員会の工作機械管理データフォーマット 1.00J、ＦＡイントラネット推進協会のコンテンツガイドライン 0.9 版、ＦＡシステムへの SNMP 適用に関する概説書を基本に検討を行った。

まず、OpenMES と工作機械管理データフォーマット、FAIPA MIB についてその位置付けを明確にする必要があったが、これは Adhoc-WG を立ち上げ、WG1、WG2 の活動と同時進行する形で進められた。

WG2 では、オープンコントローラ専門委員会の工作機械管理データフォーマット 1.00J について、データ項目を FAIPA MIB と比較し、工作機械管理データフォーマットを更新する方法も検討したが、FAIPA MIB の方がデータ項目に関して詳細であること、通信プロトコルとして SNMP はシンプルで実装容易性が高いこと、プロトコル自身がすでにデファクトスタンダードであることから FAIPA MIB 定義に工作機械管理データフォーマットの内容を適用し、矛盾点・不足点がないか検討を行い、追加修正を行うことで進めてきた。

また Adhoc-WG より統合モデル案の提示があり、システム上の矛盾がないか内容検討を行うと同時に、技術面から接続方法の検討も行った。さらに FAIPA MIB の汎用性を客観的に判断するため、Java コンソーシアム工業応用部会の JIM 仕様、日本 OPC 協議会の OPC（OLE for Process Control）も検討した。

６.３　PIPI 仕様案

６.３.１　用語

本仕様で使用される用語の規定を以下に行う。

- FA（Factory Automation）
  自動化工場

- FA **機器**（Factory Automation **機器**）

工作機械、搬送機械、PLC、管理コンピュータなど、FA で使用する装置、機器

- SNMP（Simple Network Management Protocol）

IETF で標準化された TCP/IP ネットワーク環境での管理プロトコル。管理する側の「SNMP マネージャ」と管理される側の「SNMP エージェント」の 2 つで MIB (Management Information Base)と呼ばれる管理情報を交換することで、機器の管理が行われる。

- MIB（Management Information Base）

SNMP によって管理される項目を定義したもので、自機の状態を保持する変数のこと。ハブなどのネットワーク機器が備える。各機器が基本的に備えるべき MIB は、RFC で定められており（MIB の基本的な要素は RFC1213 で定められているが、現在も拡張が進められている）、このほかにベンダ独自の MIB を備えていることがある

- IETF（The Internet Engineering Task Force）

Internet 上で開発されるさまざまな新しい技術の標準化を促進するために設立されたコンソーシアム。IETF が発行するドキュメントは RFC（Requests For Comment）として知られる。

公式ホームページ　http://www.ietf.org/

- RFC（Request For Comment）

インターネットで使用されている通信の各種規定の公開、あるいは紹介などの際に、広くコメントを求めることを意図する言葉である。そのために公開される書類は、例えば"RFC 1234"などのように"RFC・・"なる番号が付与されている。これから、単に RFC というと、インターネット関連の規定を総称する用語として使用される。RFC は、http://www.rfc-editor.org/など、インターネット上の多くのサイトから入手可能である。

- **イントラネット**（Intranet）

本書中では、インターネットの基本プロトコルである IP（Internet Protocol：RFC 791）を通信プロトコルとして使用している私設ネットワークを指す言葉として使用する。

- **プロトコル**（Network Protocol）

ネットワーク上でデータを受け渡しするための規約。プロトコルとは、通信のための手順を表わす。たとえば、２つのノードが通信したい場合、最初にどちらのノードが要求を出して、他方がそれにどう応え、送信するデータの形式やパケットの構造はどうするか、エラーが起こった場合はどうするかなど、事前に細かな取り決めが必要となる。これがプロトコルである。

- **プロトコルスタック**（Protocol Stack）

TCP/IP などの通信プロトコルを実装しているネットワークモジュール部のプログラムを指す言葉。

- **パケット**（Packet）

ネットワーク上を流れるひとかたまりのデータ。パケットには通常、先頭にプロトコルヘッダ（プロトコル階層毎に解釈・付加されるデータの内容を表わすためのデータ。宛先アドレス

や送信元アドレス、データの内容を表わすフラグなどが記録されている)、その次にデータ本体、そして最後にエラー検出コードなどが含まれている。

- **ノード（Node）**

  ネットワークに接続されているコンピュータやハブなどの機器のこと。

6.3.2 標準仕様の概要

各種の FA 機器がネットワークで相互接続された FA のシステムは、管理コンピュータなどの「管理する側の機器」と、工作機械や搬送装置などの「管理される側の機器」で構成される（場合によって単一の機器が、管理する側と管理される側を兼ねることもある)。

管理する側の機器は、ある管理される側の機器を指定した上で、これに対して必要に応じてステータス情報や、動作制御を要求し、管理される側の機器は要求に応じた情報を要求元に返したり、要求に応じた動作を実施したりする。

また、管理される側の機器は、管理する側の機器からの逐次の要求がなくても、事前の設定により、警報情報などを通知したりする。

ネットワークを介したこれらの要求・応答あるいは警報のメカニズムを達成するためには、管理する側とされる側において、主として、

(a) 通信の手続き（=プロトコル）を統一すること。

(b) プロトコルを介して送受信される情報が共通理解されること。

が必要となる。

本仕様では、機器の管理情報の収集、動作制御、警報通知などのために、イントラネットないしインターネットで広く用いられている SNMP（Simple Network Management Protocol）を上記(a)のプロトコルとして採用する。

SNMP では、一般に上記(b)の情報の共通理解を MIB（Management Information Base）と言う概念を用いて合理的に達成している。本仕様では、FA 機器の管理、制御のための MIB を規定する。

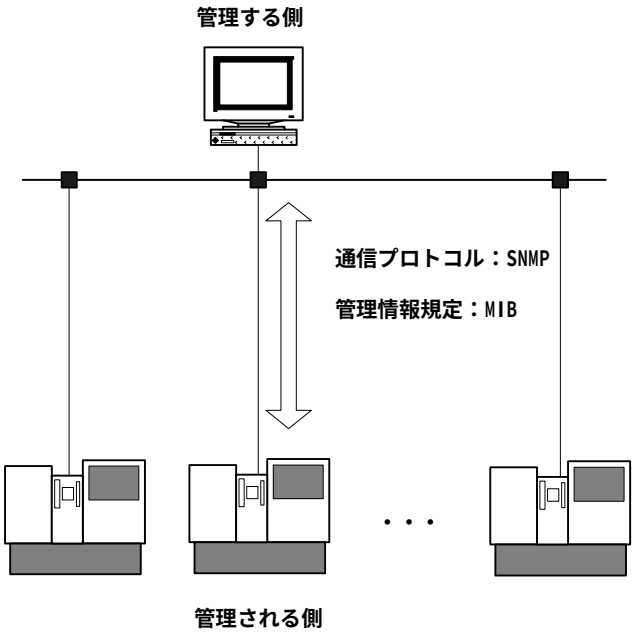

図 6－1　通信規定の概要

（1） SNMP の概要

SNMP（Simple Network Management Protocol）は、インターネットないしイントラネットのネットワーク機器を遠隔管理するために開発されたプロトコルであり、1990 年 5 月に制定された RFC 1157（用語 RFC は6.3.1 参照）に規定されている。

SNMP では、管理する側を「SNMP マネージャ」、管理される側を「SNMP エージェント」と呼ぶ。

SNMP は、その名の通り、大変シンプルなプロトコルであり、その手続きは、主として下記に上げる 3 種の動作で説明できる。

(a) [GetRequest／GetResponse]

SNMP マネージャは、所望のステータス情報、統計情報などを SNMP の手続きに従って SNMP エージェントに要求し、SNMP エージェントがこれに応答することによって指定された情報が SNMP マネージャに返される。

この場合、SNMP マネージャは「GetRequest」メッセージを含んだ SNMP パケットを生成し、所望の SNMP エージェントのアドレス（IP アドレス）に宛ててパケットを送信する。当該 GetRequest メッセージ中には獲得（Get）したい情報が特定されている（情報の特定方法は「（2）MIB の概要」を参照）。

ネットワークは当該 GetRequest メッセージを宛先 IP アドレスに従って配信する。このパケットを受信した SNMP エージェントは、パケットの送信元 IP アドレスから GetRequest を送ってきた SNMP マネージャを特定し、メッセージを解読して要求されている情報を知る。

SNMP エージェントは FA 機器に実装されている 1 つのソフトウェアタスクである。SNMP エ

ージェントは要求されている情報を知ると、その FA 機器の実装に従って要求されている情報の現状の値を得て、この値から「GetResponse」メッセージを生成し、要求元の SNMP マネージャの IP アドレスに宛てた SNMP パケットを送信する。

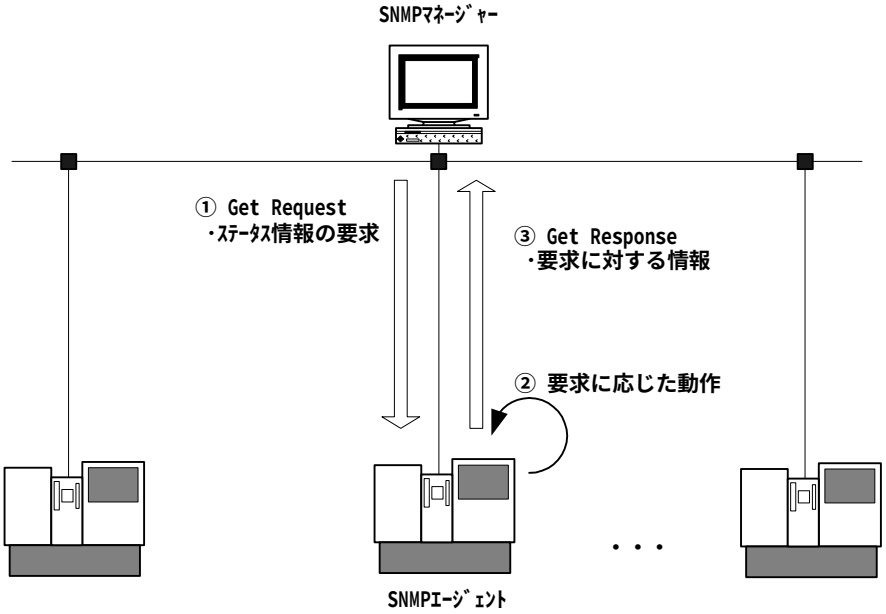

図 6－2　ステータス情報、統計情報の要求・応答

なお、SNMP では、例えばテーブル化されている情報（例えば、工作機械であれば、工具補正情報やパラメータ情報など）を連続して獲得するための「GetNextRequest」メッセージも定義されている。

(b)　[SetRequest／GetResponse]

SNMP マネージャが、SNMP エージェントの動作を制御しようとする場合は「SetRequest」メッセージを含んだ SNMP パケットを SNMP エージェントに送信する。SetRequest メッセージ中では制御したい情報ならびにその情報に新たにセットすべき値が特定される。

SetRequest を受信した SNMP エージェントは、メッセージを解読して SNMP マネージャの指定の情報を指定の値にセットし（その結果として動作のためのパラメータ値が書き換えられ、必要に応じて動作を変化させる）、セット後の値を含んだ GetResponse メッセージで（SetRequest に対しても GetResponse を用いて）応答する。

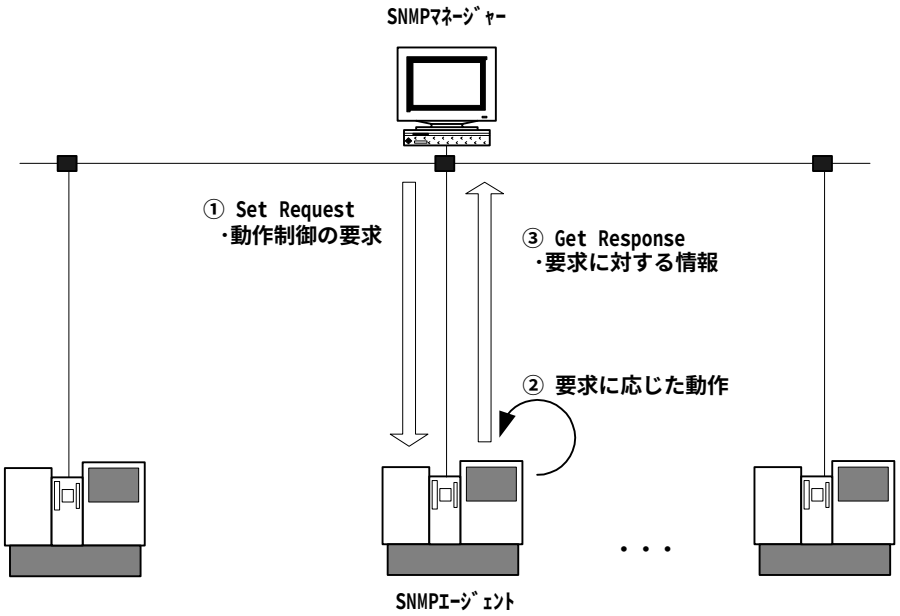

図 6－3　動作制御・応答

(c)　[Trap]

GetRequest／GetResponse および SetRequest／GetResponse は、いずれも SNMP マネージャ側から起動されるシーケンスである。SNMP では、警報情報の伝達のためなどのために SNMP エージェントが予め設定されているイベントの発生を検出した場合に、予め設定されている SNMP マネージャに対してこれを一方的に通知するためのメカニズムも用意されている。SNMP エージェントによるこの通知には「Trap」メッセージが使用される。

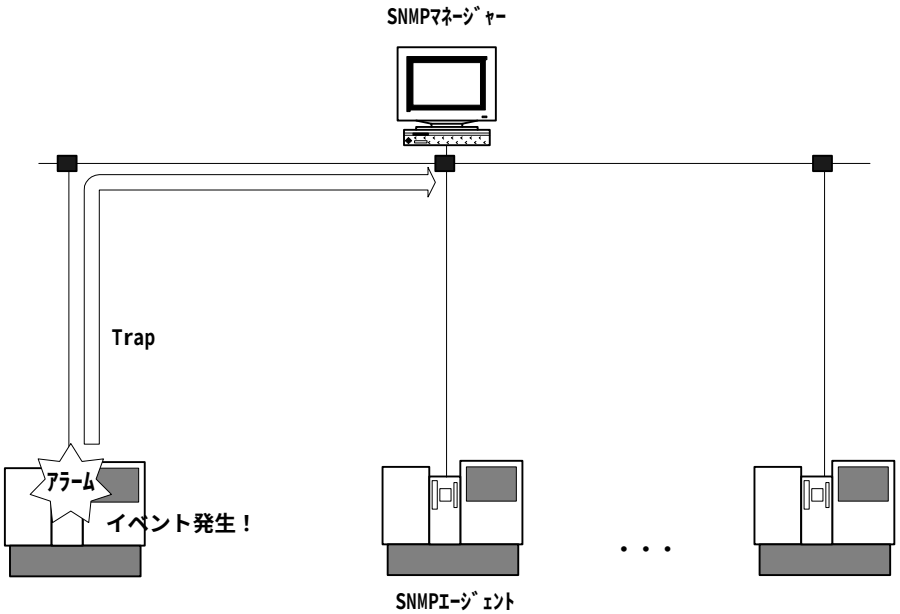

図 6－4　警報情報の通知

Trap は、SNMP エージェントより通知された警報発生を SNMP マネージャが認識し、Trap 通知を行った SNMP エージェントに対し警報情報を GetRequest により要求し､対応する SNMP エージェントが警報情報を GetResponse により送信するような手順を想定している。GetRequest／GetResponse による､警報情報の取得シーケンスは、「図 ６－２ ステータス情報、統計情報の要求・応答」で示されている。

（２） MIB の概要

SNMP を用いた管理では、SNMP マネージャと SNMP エージェントの間で、管理情報（制御の対象となっている情報。例えば工作機械の場合、アラームやステータス情報など）ならびにその値を正しく相互理解できることが不可欠である。MIB（Management Information Base）は、そのために使用される概念であり、MIB により個々の管理情報はあらゆる場合に唯一に特定され、その値は正しく伝達、解読される。

SNMP を使用したネットワークの管理の構成を図 ６－５に、MIB の構造を図 ６－６に示す。

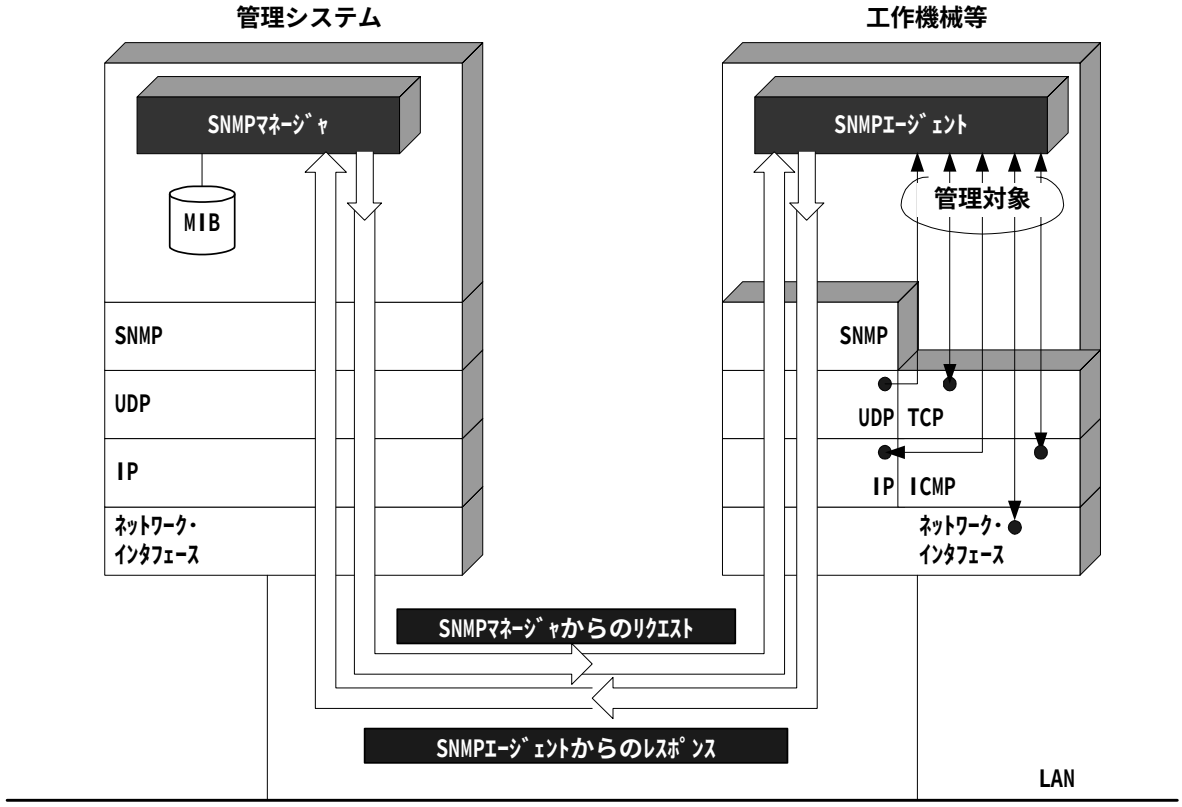

図 ６－５ SNMP を使用したネットワークの管理の構成

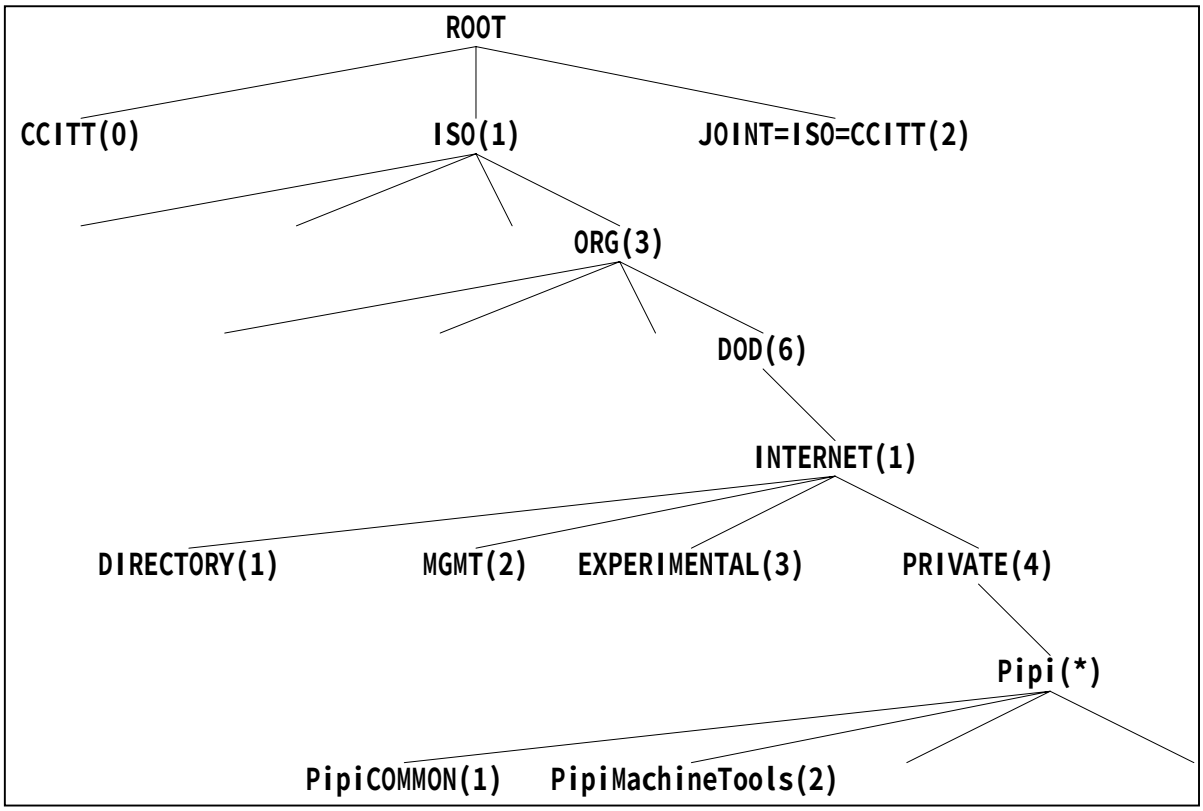

図 ６－６ MIB の構造

MIB は、後述の「６.４.２ MIB 定義」の規定からも分かるように、ある装置群に分類される装置が具備すべき管理情報を規定している。装置によっては、複数の装置群に分類され得る装置もあるが、この場合、当該装置はそれぞれの装置群のために規定される MIB をすべて実装してよい。また、MIB II（RFC 1213 に規定される）の様に、すべての IP 通信機能を持った機器が具備すべき MIB 規定もある。さらに、例えばある装置メーカが、その装置が備える機能についてプライベートな MIB 規定をユーザに公開すれば、これらのプライベートな MIB 規定に含まれる管理情報の要素も SNMP の手続きを介して管理可能となる。

MIB 規定に含まれる個々の管理情報の要素は「オブジェクト」と呼ばれる。MIB II の"system group"に規定される１つのオブジェクト"sysName"をとりあげ、オブジェクトの特定法ならびにその値の規定を解説する。

オブジェクト sysName は、RFC 1213 にて、表 ６－１のように規定されている。

表 6－1　MIB II におけるオブジェクト"sysName"の規定

（実際の規定は、曖昧さを排除するために ASN.1 型の抽象構文を使用している）

項目	規定
構文	0 バイト以上 255 バイト以下のオクテットストリング
アクセス	リード／ライトとも可能
説明	当該管理対象に管理の都合で付与される名前
オブジェクト ID	1.3.6.1.2.1.1.5

(a) オブジェクトの特定

オブジェクトはオブジェクト ID で特定される。上例の"sysName"のオブジェクト ID の値、"1.3.6.1.2.1.1.5" は、世界で唯一" sysName "だけが有する値である。

オブジェクトに付与されるオブジェクト ID の値は、いわゆるディレクトリ構造で一元管理されており、この例の "1.3.6.1" までは、”iso(1) org(3) dod(6) internet(1)” を示し、以下はインターネット協会が管理している（他の団体が管理するオブジェクト ID に ”1.3.6.1.・・” は存在しない）。

“1.3.6.1” 以降は、”mgmt(2) mib-2(1) system(1)” を示し、最後の ”5” で "sysName" が特定される。

SNMP の各メッセージ中では、オブジェクト（管理情報）は、このオブジェクト ID の値によって伝達されている。

SNMP マネージャと SNMP エージェントは、このオブジェクト ID を用いることにより管理の対象のオブジェクトを唯一に特定するのである。

なお、上記のディレクトリ構造は、すべてのオブジェクト ID が唯一であることを保証するための管理法であり、MIB の規定に沿って装置を実装しようとする者、あるいはユーザはオブジェクト ID の各位の値が、何のディレクトリに対応しているかを意識する必要はない。

(b) オブジェクトの値

オブジェクトの値は、オブジェクトの型、取り得る値の範囲、さらに場合によって単位などを表 6－1に示す “構文（SYNTAX）”、”説明（DESCRIPTION）”で規定し、”アクセス（ACCESS）” にて（SNMP マネージャから見て）リード、ライトの可否を規定する。さらに、表 6－1には表れていないが、オブジェクトの値を実際のパケットにコーディングする際のルールも明確に規定されている。これにより、SNMP マネージャと SNMP エージェントは交換するオブジェクトの値を誤解なく認識することができる。

(c) プロトコルスタック

図 6－7に、本仕様に沿って SNMP 管理を実施する FA 機器の一般的なプロトコルスタックを示す。

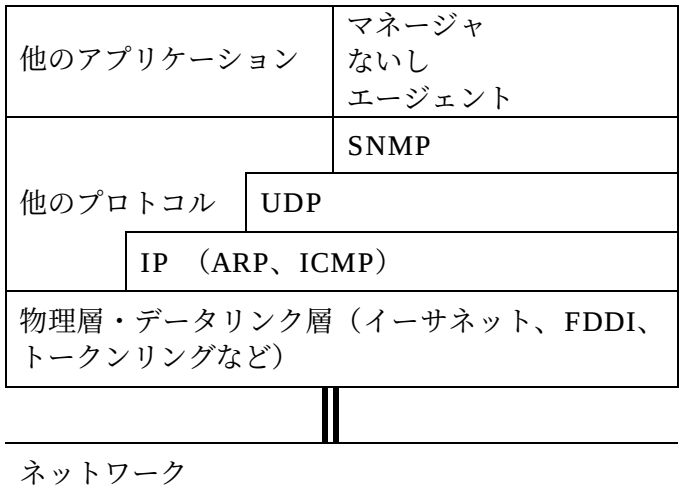

図 6－7 SNMP 管理を実施する FA 機器の一般的なプロトコルスタック

6.4 PIPI 仕様案

6.4.1 SNMP プロトコル規定

本仕様に従う FA 機器は、RFC 1157 “Simple Network Management Protocol (SNMP)” に準拠する。

6.4.2 MIB 定義

(1) PIPI の構造

PIPI における MIB 定義は、図 6－8に示すように、”iso(1) org(3) dod(6) internet(4) private(4) enterprise(4)” の下に PIPI の MIB が定義される。

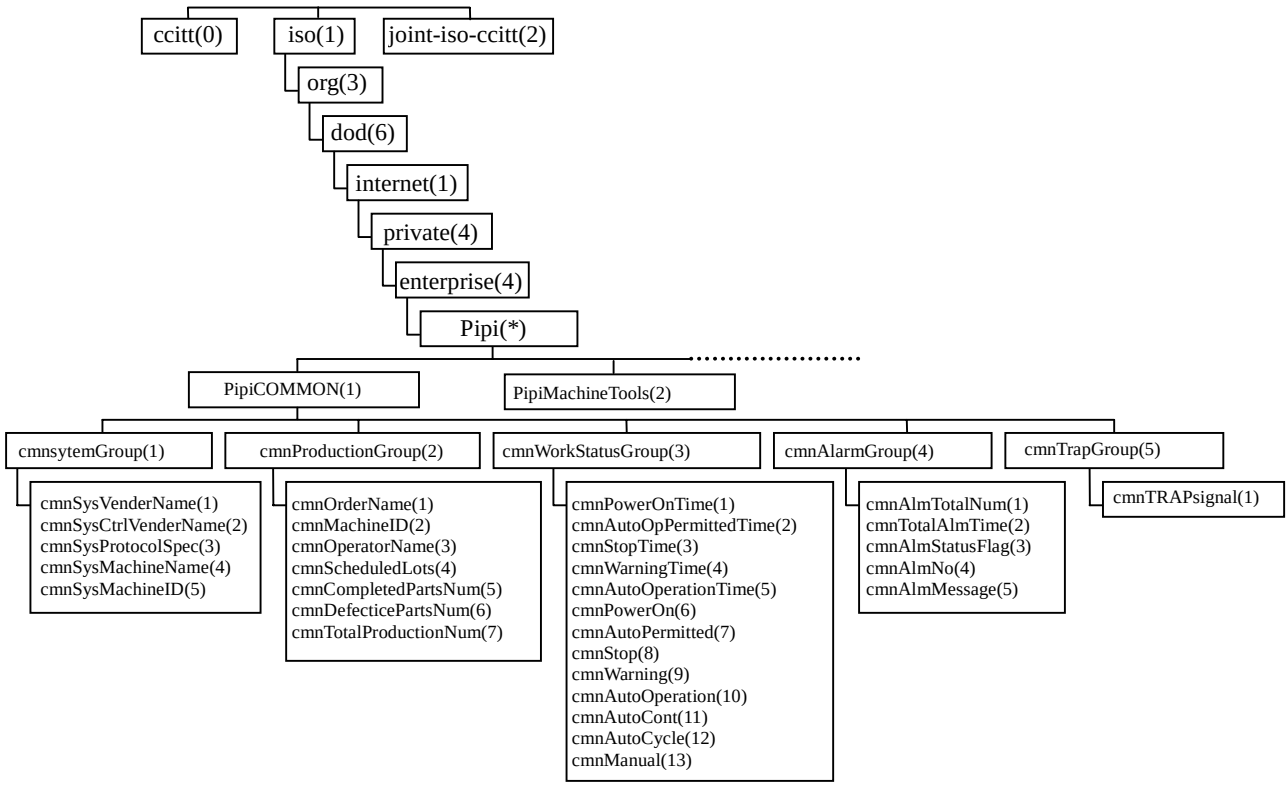

図 ６－８　PIPI 構成テーブル

（２） FA 機器共通管理データ（PipiCOMMON）定義

ここでは、FA 機器の稼動管理に共通して必要となる FA 機器共通管理データ（PipiCOMMON）について定義する。FA 機器共通管理データはベンダ情報（cmnsytemGroup）、生産情報（cmnProductionGroup）、稼動情報（cmnWorkStatusGroup）、アラーム情報（cmnAlarmGroup）、およびトラップ情報（cmnTrapGroup）に分類される。

- ベンダ情報　　：FA 機器の名称や、ベンダ名称など FA 機器固有の情報
- 生産情報　　：オーダー名や生産予定数、生産数など生産管理に必要な情報
- 稼動情報　　：通電時間や停止時間など FA 機器の稼動管理に必要な情報
- アラーム情報　：アラーム番号やアラームメッセージなどのアラーム情報
- トラップ情報　：自動運転状態や警報など FA 機器の状態変化を通知する情報

ベンダ情報、生産情報、稼動情報、アラーム情報、トラップ情報の一覧を「６.４.３ 」項に示すとともに、トラップ情報を例にとり、以下で詳細に説明する。

また、FA 機器共通管理データのコーディング例を「６.４.４ FA 機器共通管理データ（PipiCOMMON）の MIB 定義コーディング例」に示すので、あわせて参照されたい。

（3） トラップ情報（cmnTrapGroup）

「6.4.3 FA 機器共通管理データ一覧」の分類の項で、トラップ情報に分類される情報は、SNMP エージェントが、SNMP マネージャに対して、非同期に警報情報など動作状態の変化などを通知したいときに発行するトラップコマンドで通知する情報である。通知する情報としては、電源 ON／OFF、自動運転状態（自動運転可、自動運転停止、自動運転開始)、操作モード（自動／連続モード、自動／サイクルモード、手動モード)、警報（Warning 発生、アラーム発生）がある。これらの情報は、トラップ種別情報データとしてトラップコマンドに添付して送られる。

また、「6.4.3 FA 機器共通管理データ一覧」の Trap 可否の項目が「可」となっている情報は、その情報が示す状態が変化したとき、TRAP コマンドが発生する情報であることを示している。

(a) トラップ種別情報データ

SNMP エージェントは TRAP 種別情報データを添付してトラップを発行する。

- データ名称　　：TRAP 種別
- データサイズ　：４バイト（32 ビット）
- データタイプ　：正数
- 単位　　：無し
- 取り得る値　　：以下のビット情報で通知される。

(b) TRAP 情報 32 ビットフィールド

- D0　電源 ON　　（電源が ON された）
- D1　電源 OFF　　（電源が OFF された）
- D2　自動運転可　　（自動運転可能状態に遷移した）
- D3　自動運転停止　　（自動運転停止状態に遷移した）
- D4　Warning 発生　　（Warning が発生した）
- D5　自動運転開始　　（自動運転が開始された）
- D6　自動／連続モード　　（自動／連続モードに設定された）
- D7　自動／サイクルモード　　（自動／サイクルモード）
- D8　手動モード　　（手動モードに切り替えられた）
- D9　アラーム発生　　（アラームが発生した）
- D10～D31　予約

(c) 想定されるシーケンス例

SNMP Pエージェントである機械側でアラームが発生した場合の、SNMP マネージャと SNMP

エージェント間の情報授受シーケンスの例を示す。（図 ６－９参照）

① SNMP エージェント（FA 機器）にアラームが発生
② SNMP エージェントからアラーム発生を TRAP（D9=1）コマンドで SNMP マネージャに通知
③ SNMP マネージャは TRAP 種別(32 ビットデータフィールド)の内容を判別し、アラームの発生（D9=1）を確認し、詳細情報の取得を実行
④ SNMP マネージャは GET REQUEST コマンドにより、詳細なアラーム情報を SNMP エージェントに要求
⑤ SNMP エージェントは GET REQUEST コマンドで要求されたアラームデータを収集
⑥ SNMP エージェントは収集したアラームデータを GET RESPONSE で SNMP マネージャに送信
⑦ SNMP マネージャは GET RESPONSE で送られてきたアラームデータに応じて、アラーム処理を実行

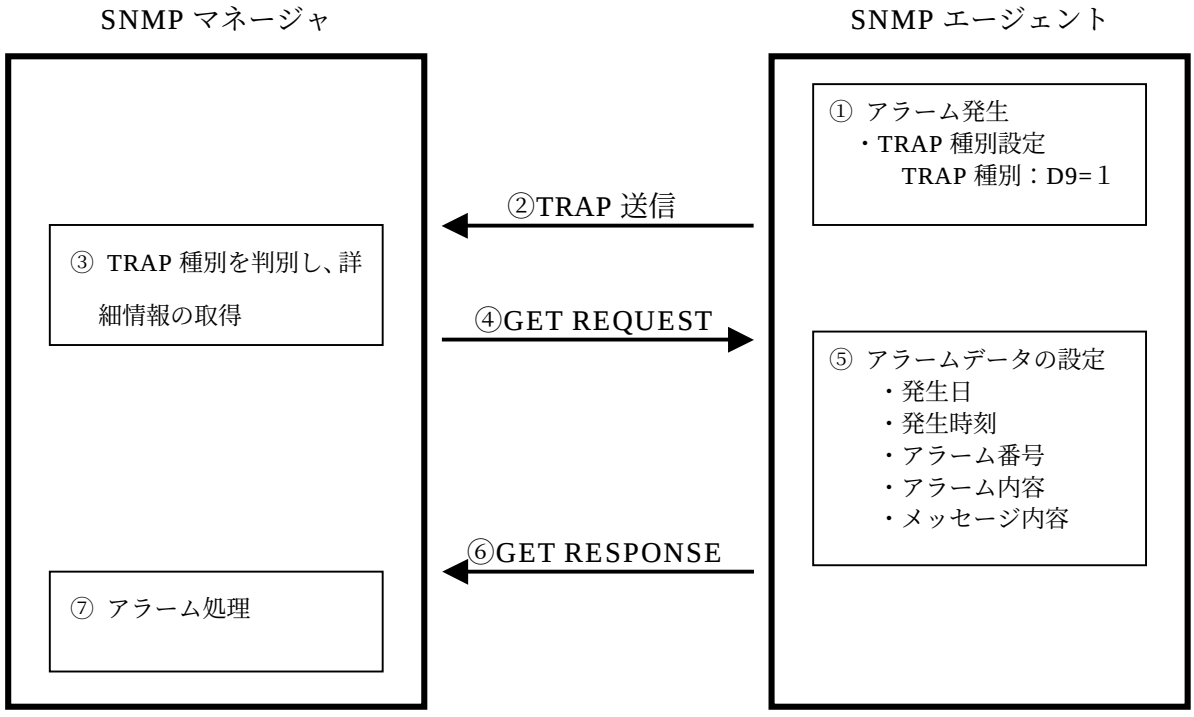

図 ６－９　アラーム発生時の SNMP の情報授受シーケンス例

6.4.3 FA 機器共通管理データ一覧

分類	名称	定義	補足説明	Size	Type	単位	取り得る値	Trap 可否
ベンダ情報	機械ベンダ名	機械ベンダ名称	システム識別情報としての FA 機器ベンダ名称	Max 64Byte	Text	－	規定無し	不可
	制御装置ベンダ名	制御装置ベンダ名称	システム識別情報としての制御装置ベンダ名称	Max 64Byte	Text	－	規定無し	不可
	通信仕様	管理データバージョン	システム識別情報としての管理データバージョン(“ PIPIx.xx”, x.xx はバージョン)	Max 64Byte	Text	－	“PIPI-1.00”	不可
	機械名称	機械名称	システム識別情報としての FA 機器名称	Max 64Byte	Text	－	規定無し	不可
	機械番号	機械番号	システム識別情報としての FA 機器番号	Max 64Byte	Text	－	規定無し	不可
生産情報	オーダー名	オーダー名または生産管理上 ID	生産品目の識別ができる ID。NC の場合にはプログラム名等。	Max 64Byte	Text	－	規定無し	不可
	装置 ID	その装置が特定できる ID	装置番号など、任意に規定できる。	Max 64Byte	Text	－	規定無し	不可
	オペレータ名	作業者名が識別できる ID	任意に規定できる。オペレータ毎の生産効率を管理する場合に用いる。	Max 64Byte	Text	－	規定無し	不可
	生産予定数	生産予定数	オーダーとセットで指示した生産予定数。指示がなければ 0.	4Byte	Integer	個	0～99999999	不可
	生産数	現状までの生産数(良品数)	検査機能を持たない装置では、総生産数に等しい。（不良品数は 0 となる）	4Byte	Integer	個	0～99999999	不可
	不良品数	現状までの不良品数	検査機能を持たない装置では、0 となる。	4Byte	Integer	個	0～99999999	不可
	総生産数	現状までの総生産数	生産数を削除しても、[生産数（良品生産数）=総生産数-不良品数]で求まる。	4Byte	Integer	個	0～99999999	不可
稼動情報	通電時間	現状までの積算通電時間		4Byte	Integer	秒	0～99999999	不可
	自動運転可時間	装置が自動運転できる状態にあった積算時間		4Byte	Integer	秒	0～99999999	不可
	停止時間	装置が停止していた積算時間		4Byte	Integer	秒	0～99999999	不可
	Warning 中積算時間	運転中の Warning 状態発生の積算時間		4Byte	Integer	秒	0～99999999	不可
	自動運転時間	自動運転動作を行っていた積算時間		4Byte	Integer	秒	0～99999999	不可
	通電中状態フラグ	通電中を示す		1Byte	Text	－	0, 1	可
	自動運転可状態フラグ	装置が自動運転できる状態を示す		1Byte	Text	－	0, 1	可
	停止状態フラグ	装置が停止している状態		1Byte	Text	－	0, 1	可
	Warning 中状態フラグ	装置が Warning 状態である事を示す		1Byte	Text	－	0, 1	可
	自動運転状態フラグ	装置が加工動作を行っている状態を示す		1Byte	Text	－	0, 1	可
	自動／連続モードフラグ	自動／連続モードに設定されている状態を示す		1Byte	Text	－	0, 1	可
	自動／サイクルモードフラグ	自動／サイクルモードに設定されている状態を示す		1Byte	Text	－	0, 1	可

分類	名称	定義	補足説明	Size	Type	単位	取り得る値	Trap可否
	手動モードフラグ	手動モードに設定されている状態を示す		1Byte	Text	－	0, 1	可
アラーム情報	アラーム発生件数	現状までのアラーム発生件数		4Byte	Integer	回	0～99999999	不可
	アラーム中積算時間	現状までのアラーム中の積算時間		4Byte	Integer	秒	0～99999999	不可
	アラーム状態フラグ	現状アラーム状態である事を示す		1Byte	Text	－	0, 1	可
	現在のアラーム No.	現在のアラーム番号	重複して発生している場合は最上位レベルのもの。	Max 64Byte	Text	－	規定無し	不可
	現在のアラームメッセージ	現在のアラームメッセージ	重複して発生している場合は最上位レベルのもの。	Max 64Byte	Text	－	規定無し	不可
TRAP情報	電源 ON	電源が ON された	ビットフィールド D0	1bit	bit		0, 1	－
	電源 OFF	電源が OFF された	ビットフィールド D1	1bit	bit		0, 1	－
	自動運転可	自動運転可能状態に遷移した	ビットフィールド D2	1bit	bit		0, 1	－
	自動運転停止	自動運転停止状態に遷移した	ビットフィールド D3	1bit	bit		0, 1	－
	Warning 発生	Warning が発生した	ビットフィールド D4	1bit	bit		0, 1	－
	自動運転開始	自動運転が開始された	ビットフィールド D5	1bit	bit		0, 1	－
	自動／連続モード	自動／連続モードに設定された	ビットフィールド D6	1bit	bit		0, 1	－
	自動／サイクルモード	自動／サイクルモードに設定された	ビットフィールド D7	1bit	bit		0, 1	－
	手動モード	手動モードに切り替えられた	ビットフィールド D8	1bit	bit		0, 1	－
	アラーム発生	アラームが発生した	ビットフィールド D9	1bit	bit		0, 1	－

6.4.4 FA 機器共通管理データ（PipiCOMMON）の MIB 定義コーディング例

```
PIPI-MIB DEFINITIONS ::= BEGIN

    Pipi    OBJECT IDENTIFIER ::= {iso(0) org(3) dod(6) internet(1) private(4) enterprise(1) *}

--  ================================================================
--  注：上記オブジェクト識別子の定義において、最後の '*' は、本来は
--      IETF に申請を行って取得した値（番号）が入る。本ガイドライン
--      発行時には、未取得のため、仮に '*' としておく。
--  ================================================================

    PipiCOMMON OBJECT IDENTIFIER ::= { Pipi 1 }

    cmnSystemGroup OBJECT IDENTIFIER ::= { PipiCOMMON 1 }
    cmnProductionGroup OBJECT IDENTIFIER ::= { PipiCOMMON 2 }
    cmnWorkStatusGroup OBJECT IDENTIFIER ::= { PipiCOMMON 3 }
    cmnAlarmGroup OBJECT IDENTIFIER ::= { PipiCOMMON 4 }
    cmnTrapGroup OBJECT IDENTIFIER ::= {PipiCOMMON 5}

--  ============================================================
--  基本 TRAP 詳細情報
--  ============================================================
```

```
    cmnTRAPsignal OBJECT-TYPE
        SYNTAX      OCTET STRING (SIZE (4))
        ACCESS      read-only
        STATUS      optional
        DESCRIPTION
              "トラップ情報：bit 0 -- 電源 ON
                             bit 1 -- 電源 OFF
                             bit 2 -- 自動運転可
                             bit 3 -- 自動運転停止
                             bit 4 -- Warning 発生
                             bit 5 -- 自動運転開始
                             bit 6 -- 自動／連続モード
                             bit 7 -- 自動／サイクルモード
                             bit 8 -- 手動モード
                             bit 9 -- アラーム発生
                             bit 10 ～ bit 31 -- reserved "
         ::= { cmnTrapGroup 1 }

--  ================================================================
--  cmnSystemGroup Start
--  ================================================================
    cmnSysVenderName OBJECT-TYPE
        SYNTAX      DisplayString (SIZE (0..64))
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
           "機械ベンダ名：システム識別情報としてのＦＡ機器ベンダ名称"
        ::= { cmnSystemGroup 1 }

    cmnSysCtrlVenderName OBJECT-TYPE
        SYNTAX      DisplayString (SIZE (0..64))
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
           "制御装置ベンダ名：システム識別情報としての制御装置ベンダ名称"
        ::= { cmnSystemGroup 2 }

    cmnSysProtocolSpec OBJECT-TYPE
        SYNTAX      DisplayString (SIZE (0..64))
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
           "通信仕様：システム識別情報としての管理データバージョン
            現在は、「PIPI-1.00」で固定"
        ::= { cmnSystemGroup 3 }

    cmnSysMachineName OBJECT-TYPE
        SYNTAX      DisplayString (SIZE (0..64))
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
           "機械名称：システム識別情報としてのＦＡ機器名称"
        ::= { cmnSystemGroup 4 }

    cmnSysMachineID OBJECT-TYPE
        SYNTAX      DisplayString (SIZE (0..64))
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
           "機械番号：システム識別情報としてのＦＡ機器番号"
        ::= { cmnSystemGroup 5 }

--  ================================================================
```

```
-- cmnProductionGroup Start
-- ============================================================
    cmnOrderName OBJECT-TYPE
        SYNTAX      DisplayString (SIZE (0..64))
        ACCESS      read-only
        STATUS      optional
        DESCRIPTION
            "オーダー名：生産品目の識別ができるＩＤ。ＮＣの場合にはプログラム名等"
        ::= { cmnProductionGroup 1 }

    cmnMachineID OBJECT-TYPE
        SYNTAX      DisplayString (SIZE (0..64))
        ACCESS      read-only
        STATUS      optional
        DESCRIPTION
            "装置ＩＤ：装置番号など，任意に規定できる"
        ::= { cmnProductionGroup 2 }

    cmnOperatorName OBJECT-TYPE
        SYNTAX      DisplayString (SIZE (0..64))
        ACCESS      read-only
        STATUS      optional
        DESCRIPTION
            "オペレータ名：任意に規定できる。オペレータ毎の生産効率を管理する場合に用いる。"
        ::= { cmnProductionGroup 3 }

    cmnScheduledLots OBJECT-TYPE
        SYNTAX      INTEGER
        ACCESS      read-only
        STATUS      optional
        DESCRIPTION
            "生産予定数：オーダーとセットで指示した生産予定数。指示がなければ０"
        ::= { cmnProductionGroup 4 }

    cmnCompletedPartsNum OBJECT-TYPE
        SYNTAX      INTEGER
        ACCESS      read-only
        STATUS      optional
        DESCRIPTION
            "生産数（良品）：検査機能を持たない装置では，総生産数に等しい（不良品数は０となる）"
        ::= { cmnProductionGroup 5 }

    cmnDefectivePartsNum OBJECT-TYPE
        SYNTAX      INTEGER
        ACCESS      read-only
        STATUS      optional
        DESCRIPTION
            "不良品数：検査機能を持たない装置では、０になる"
        ::= { cmnProductionGroup 6 }

    cmnTotalProductionNum OBJECT-TYPE
        SYNTAX      INTEGER
        ACCESS      read-only
        STATUS      optional
        DESCRIPTION
            "総生産数：生産数を削除しても、生産数（良品生産数）＝総生産数－不良品数で求まる。"
        ::= { cmnProductionGroup 7 }

-- ============================================================
-- cmnWorkStatusGroup Start
-- ============================================================
    cmnPowerOnTime OBJECT-TYPE
        SYNTAX      INTEGER
        ACCESS      read-only
```

```
        STATUS      optional
        DESCRIPTION
            "通電時間：単位（秒）"
        ::= { cmnWorkStatusGroup 1 }

    cmnAutoOpPermittedTime OBJECT-TYPE
        SYNTAX      INTEGER
        ACCESS      read-only
        STATUS      optional
        DESCRIPTION
            "自動運転可時間：単位（秒）"
        ::= { cmnWorkStatusGroup 2 }

    cmnStopTime OBJECT-TYPE
        SYNTAX      INTEGER
        ACCESS      read-only
        STATUS      optional
        DESCRIPTION
            "停止時間：単位（秒）"
        ::= { cmnWorkStatusGroup 3 }

    cmnWarningTime OBJECT-TYPE
        SYNTAX      INTEGER
        ACCESS      read-only
        STATUS      optional
        DESCRIPTION
            "Warning 中積算時間：単位（秒）"
        ::= { cmnWorkStatusGroup 4 }

    cmnAutoOperationTime OBJECT-TYPE
        SYNTAX      INTEGER
        ACCESS      read-only
        STATUS      optional
        DESCRIPTION
            "自動運転時間：単位（秒）"
        ::= { cmnWorkStatusGroup 5 }

    cmnPowerOn OBJECT-TYPE
        SYNTAX      DisplayString (SIZE (0..1))
        ACCESS      read-only
        STATUS      optional
        DESCRIPTION
            "通電中状態フラグ"
        ::= { cmnWorkStatusGroup 6 }

    cmnAutoPermitted OBJECT-TYPE
        SYNTAX      DisplayString (SIZE (0..1))
        ACCESS      read-only
        STATUS      optional
        DESCRIPTION
            "自動運転可状態フラグ"
        ::= { cmnWorkStatusGroup 7 }

    cmnStop OBJECT-TYPE
        SYNTAX      DisplayString (SIZE (0..1))
        ACCESS      read-only
        STATUS      optional
        DESCRIPTION
            "停止状態フラグ"
        ::= { cmnWorkStatusGroup 8 }

    cmnWarning OBJECT-TYPE
        SYNTAX      DisplayString (SIZE (0..1))
```

```
        ACCESS      read-only
        STATUS      optional
        DESCRIPTION
            "Warning 中状態フラグ"
        ::= { cmnWorkStatusGroup 9 }

    cmnAutoOperation OBJECT-TYPE
        SYNTAX      DisplayString (SIZE (0..1))
        ACCESS      read-only
        STATUS      optional
        DESCRIPTION
            "自動運転状態状態フラグ"
        ::= { cmnWorkStatusGroup 10 }

    cmnAutoCont OBJECT-TYPE
        SYNTAX      DisplayString (SIZE (0..1))
        ACCESS      read-only
        STATUS      optional
        DESCRIPTION
            "自動／連続モードフラグ"
        ::= { cmnWorkStatusGroup 11 }

    cmnAutoCycle OBJECT-TYPE
        SYNTAX      DisplayString (SIZE (0..1))
        ACCESS      read-only
        STATUS      optional
        DESCRIPTION
            "自動／サイクルモードフラグ"
        ::= { cmnWorkStatusGroup 12 }

    cmnManual OBJECT-TYPE
        SYNTAX      DisplayString (SIZE (0..1))
        ACCESS      read-only
        STATUS      optional
        DESCRIPTION
            "手動モードフラグ"
        ::= { cmnWorkStatusGroup 13 }

-- ============================================================
-- cmnAlamGroup Start
-- ============================================================
    cmnAlmTotalNum OBJECT-TYPE
        SYNTAX      INTEGER
        ACCESS      read-only
        STATUS      optional
        DESCRIPTION
            "アラーム発生件数"
        ::= { cmnAlamGroup 1 }

    cmnTotalAlmTime OBJECT-TYPE
        SYNTAX      INTEGER
        ACCESS      read-only
        STATUS      optional
        DESCRIPTION
            "アラーム中積算時間"
        ::= { cmnAlamGroup 2 }

    cmnAlmStatusFlag OBJECT-TYPE
        SYNTAX      DisplayString (SIZE (0..1))
        ACCESS      read-only
        STATUS      optional
        DESCRIPTION
            "アラーム状態フラグ"
```

```
        ::= { cmnAlamGroup 3 }

    cmnAlmNo OBJECT-TYPE
        SYNTAX      DisplayString (SIZE (0..64))
        ACCESS      read-only
        STATUS      optional
        DESCRIPTION
            "現在のアラームＮｏ：重複して発生している場合は最上位レベルのもの"
        ::= { cmnAlamGroup 4 }

    cmnAlmMessage OBJECT-TYPE
        SYNTAX      DisplayString (SIZE (0..64))
        ACCESS      read-only
        STATUS      optional
        DESCRIPTION
            "現在のアラームメッセージ：重複発生している場合は最上位レベルのもの"
        ::= { cmnAlamGroup 5 }

--  ================================================================
--  TRAP 用情報定義
--  ================================================================
    operationStatusChange TRAP-TYPE
        ENTERPRISE   Pipi
        VARIABLES { cmnTRAPsignal }
        DESCRIPTION
            "トラップ情報：運転状態変更時のトラップ"
        ::= 1
END
```

６.５　工作機械への適応例

工作機械への適用事例として、1999 年メカトロテックに JOP と共同出展した FAIPA MIB のデータ内容を表 ６－２および表 ６－３に示す。

本事例の中で、工作機械管理データフォーマットに存在し、上記 MIB 定義にないものを当 WG2 で検討した結果、次のように結論づけた。

- 時刻　　（不要）
  タイムゾーンがあり、それぞれのイベントに時間表記があるため
- タイムゾーン　　（必要）
- 機械の名称　　（不要）
  機械名称は製造者番号、機械の型式で特定できるので省く
- 制御装置識別番号　　（不要）
  WG5 仕様の任意拡張項目のため
- サブプロ番号　　（追加必要）
  プログラム運用方法によってはメインプログラム番号より、サブプログラム番号が重要な場合があるため
- 任意拡張部のバージョン　（不要）
- サーボ状態（主軸含む）　（追加必要）

またデータタイプ・データ単位の取り決めについて次のように結論づけた。

- inch/mm　単位系
  inch/mm の扱いについては、別途検討する必要がある。
  SI 単位系で情報を授受し、inch の場合は、上位アプリケーションで対応する方法もある。
- バイナリデータ・テキストデータ
  XML への世代移行ど将来性を考え、テキストデータを採用する。

表 ６－２　工作機械への適用事例（１）

分類	名前	説明	サイズ	データタイプ	単位
NCモニタ	シーケンス番号	現在実行中のシーケンス番号	４バイト	テキスト	無
	加工プログラム名	現在実行中の加工プログラム名称	２０バイト	テキスト	無
	プログラムデータ	現在実行中のプログラムデータブロック	可変	テキスト	無
	現在値データ	現在値データ	１４バイト×軸数	テキスト	mm | inch
	主軸回転数	現在の主軸回転数	１４バイト×軸数	テキスト	rpm
	工具番号	現在使用中の工具番号	９バイト	テキスト	無
	回転軸角度	現在の回転軸角度	８バイト	テキスト	度
	軸の残り量	現在の軸の残り量	１４バイト×軸数	テキスト	mm | inch
	送り量	現在の送り量	１４バイト×軸数	テキスト	mm / min
稼動時間	通電時間	ＮＣ装置の電源が入っている状態の積算時間	８バイト	テキスト	秒
	稼動時間	メインプログラムが実行されている状態の積算時間	８バイト	テキスト	秒
	切削時間	切削送りが行われている状態の積算時間	８バイト	テキスト	秒
	主軸回転時間	主軸が回転している状態の積算時間	８バイト	テキスト	秒
	外部入力時間	外部入力信号がオンしている状態の積算時間	８バイト	テキスト	秒
	停止時間	加工プログラムが一時停止している状態の積算時間	８バイト	テキスト	秒
	アラーム発生時間	アラームが発生している状態の積算時間	８バイト	テキスト	秒
	非稼動時間	メインプログラムが実行されていない状態の積算時間	８バイト	テキスト	秒
	内段取	“内段取”を理由とした非稼働状態の積算時間	８バイト	テキスト	秒
	オペレータ不在	“オペレータ不在”を理由とした非稼働状態の積算時間	８バイト	テキスト	秒
	待機	“待機”を理由とした非稼働状態の積算時間	８バイト	テキスト	秒
	機械整備	“機械整備”を理由とした非稼働状態の積算時間	８バイト	テキスト	秒
	その他	“その他”を理由とした非稼働状態の積算時間	８バイト	テキスト	秒
軸の負荷情報	サーボ軸の負荷情報	各サーボ軸の負荷情報	３バイト×軸数	テキスト	%
	主軸の負荷情報		３バイト×軸数	テキスト	%

表 ６－３　工作機械への適用事例（２）

分類	名前	説明	サイズ	データタイプ	単位
アラーム発生	発生日	アラームが発生した日付	１０バイト	テキスト	無
	発生時刻	アラームが発生した時刻	８バイト	テキスト	無
	アラーム番号	発生中のアラーム番号	10バイト	テキスト	無
	アラーム内容	発生中のアラームメッセージの内容	最大６４バイト	テキスト	無
	メッセージ内容	表示中のオペレータメッセージの内容	最大２５６バイト	テキスト	無
アラーム解除	解除日	アラームが解除された日付	１０バイト	テキスト	無
	解除時刻	アラームが解除された時刻	８バイト	テキスト	無
	アラーム番号	解除されたアラーム番号	１０バイト	テキスト	無
	アラーム内容	解除されたアラームメッセージの内容	最大６４バイト	テキスト	無
	メッセージ内容	解除されたオペレータメッセージの内容	最大２５６バイト	テキスト	無
ワーニング発生	発生日	ワーニングが発生した日付	１０バイト	テキスト	無
	発生時刻	ワーニングが発生した時刻	８バイト	テキスト	無
	ワーニング番号	発生中のワーニング番号	１０バイト	テキスト	無
	ワーニング内容	発生中のワーニングメッセージの内容	最大６４バイト	テキスト	無
ワーニング解除	発生日	ワーニングが発生した日付	１０バイト	テキスト	無
	発生時刻	ワーニングが発生した時刻	８バイト	テキスト	無
	ワーニング番号	発生中のワーニング番号	１０バイト	テキスト	無
	ワーニング内容	発生中のワーニングメッセージの内容	最大６４バイト	テキスト	無
ＮＣ状態	非常停止状態	非常停止になっている状態	１バイト	テキスト	無
	稼働中状態	加工プログラム実行中（サイクルスタート中）の状態	１バイト	テキスト	無
	主軸回転中	主軸が回転している状態	１バイト	テキスト	無
	停止中	加工プログラムが一時停止している状態	１バイト	テキスト	無
	プログラムストップ中	プログラムストップで停止している状態	１バイト	テキスト	無
	ＳＴＭ動作中	ＳＴＭ動作状態	１バイト	テキスト	無
	リミット	リミット状態	１バイト	テキスト	無
	段取り中	段取中スイッチが入っている状態	１バイト	テキスト	無
	アラーム発生中	アラームが発生している状態	１バイト	テキスト	無
	リセット状態	ＮＣリセット状態	１バイト	テキスト	無
操作モード	自動モード		１バイト	テキスト	無
	手動モード		１バイト	テキスト	無
	ＭＤＩモード		１バイト	テキスト	無
	編集モード		１バイト	テキスト	無
	メーカ固有モード１		１バイト	テキスト	無
	メーカ固有モード２		１バイト	テキスト	無
	メーカ固有モード３		１バイト	テキスト	無
	メーカ固有モード４		１バイト	テキスト	無
プログラム選択	着手日	メインプログラムが選択された日付	１０バイト	テキスト	
	着手時刻	メインプログラムが選択された時刻	８バイト	テキスト	
	メインプログラム	プログラム選択時のプログラム名称（番号）	最大６４バイト	テキスト	無
加工開始	加工開始日	加工プログラム実行開始時の日付	１０バイト	テキスト	無
	加工開始時刻	加工プログラム実行開始時の時刻	８バイト	テキスト	無
	メインプログラム	加工プログラム実行開始時のプログラム名称（番号）	最大６４バイト	テキスト	無
	コメント	０＿（）内のコメント	１７バイト	テキスト	無

加工終了	加工開始日	加工プログラム実行開始時の日付	１０バイト	テキスト	無
	加工開始時刻	加工プログラム実行開始時の時刻	８バイト	テキスト	無
	メインプログラム	加工プログラム実行開始時のプログラム名称（番号）	最大６４バイト	テキスト	無
	コメント	０＿（）内のコメント	１７バイト	テキスト	無
	加工終了日	Ｍ０２、Ｍ３０実行時の日付	１０バイト	テキスト	無
	加工終了時刻	Ｍ０２、Ｍ３０実行時の時刻	８バイト	テキスト	無
	加工終了数	１サイクル当たりの加工ワーク数	２バイト	テキスト	個
	異常終了数	１サイクル当たりのＮＧワーク数	２バイト	テキスト	個
	通電時間	ＮＣ装置の電源が入っている時間	８バイト	テキスト	秒
	稼動時間	メインプログラムが実行されている時間	８バイト	テキスト	秒
	切削時間	切削送りが行われている時間	８バイト	テキスト	秒
	主軸回転時間	主軸が回転している時間	８バイト	テキスト	秒
	外部入力時間	外部入力信号がオンしている時間	８バイト	テキスト	秒
	アラーム発生時間	アラームが発生している時間	８バイト	テキスト	秒
	非稼動時間	メインプログラムが実行されていない時間	８バイト	テキスト	秒
	内段取	“内段取”を理由とした非稼働時間	８バイト	テキスト	秒
	オペレータ不在	“オペレータ不在”を理由とした非稼働状態時間	８バイト	テキスト	秒
	待機	“待機”を理由とした非稼働時間	８バイト	テキスト	秒
	機械整備	“機械整備”を理由とした非稼働時間	８バイト	テキスト	秒
	その他	“その他”を理由とした非稼働時間	８バイト	テキスト	秒
部品測定	測定実施日付	機内計測が行われた日付	１０バイト	テキスト	無
	測定時刻	機内計測が行われた時刻	８バイト	テキスト	無
	計測モード		２バイト	テキスト	
	計測結果判定		１バイト	テキスト	
	測定番号	機内計測における計測個所を特定する番号	２バイト	テキスト	無
	測定データ	機内計測のデータ	１０バイト ×計測個所数	テキスト	mm | inch
	マクロデータ	マクロＤＰＲＮの出力	５６バイト	テキスト	無
工具実績	工具番号	工具交換時の主軸工具番号	９バイト	テキスト	無
	コメント	工具名称	１７バイト	テキスト	無
	回転数	工具ごとの回転数	６バイト	テキスト	rpm
	送り速度	工具ごとの送り速度	６バイト	テキスト	mm / min
	加工時間	工具ごとの加工時間	４バイト	テキスト	秒
	切削時間	工具ごとの切削時間	４バイト	テキスト	秒
	早送り時間	工具ごとの早送り時間	４バイト	テキスト	秒
	その他時間	工具ごとのその他時間	４バイト	テキスト	秒
	使用データ	工具寿命の使用値	４バイト	テキスト	寿命単位
	状態	工具寿命の状態（０:未、１:ＯＫ、４:寿命、８:破損）	１バイト	テキスト	無
	設定単位	工具寿命の単位（Ｍ:分、Ｔ:回、Ｌ:長、Ｈ:穴）	１バイト	テキスト	無
	設定データ	工具寿命の設定値	４バイト	テキスト	寿命単位

６.６　考察・展望

WG2 では平成 11 年度まで活動を行ってきた JOP オープンコントローラ専門委員会･管理系データモデルワーキンググループ（WG5）から提案された“工作機械管理データの標準化案”[1][2]と FA イントラネット推進協会から提案された“SNMP プロトコルを使用したデータ管理”について討議が行われ、生産設備の稼働管理データと通信プロトコルをとりまとめ標準案として提案した。具体的には、稼働管理データの標準化では、SNMP を運用する場合に参照される MIB 定

義として標準案を作成し、通信プロトコルに関しては OSI７階層モデルのトランスポート層より上位の３階層であるセッション層、プレゼンテーション層そしてアプリケーション層を実現した SNMP を標準案として提案した。本WGの本年度の活動を踏まえ、以下の３つの視点から考察し、将来の展望について述べる。

- 管理データの標準について
- 通信プロトコルについて
- 実証について

（１） 管理データの標準化についての考察・展望

本 WG では、生産設備機械一般を対象に管理データの抽出、および項目の検討を行ってきた。ここで検討された管理データの項目は設備機械一般で使用されうる項目に関してのみ定義した。

今後は以下の５種類に分類されるコントローラで、必要とする管理データ項目についても検討する必要がある。

- CNC
- PLC
- ロボットコントローラ
- AGV コントローラ
- その他、FA コントローラ

さらに、コントローラ種別とは別に産業機械別に作業や業種に則した管理データ項目についても検討し、企業内 FA 機器全ての管理状態を把握できるように、管理データ項目の拡充を考える必要がある。

- フライス盤
- 研削盤
- ボール盤
- 放電加工機
- 鍛圧機械
- レーザ加工機
- 食品製造機械
- チップマウンタ
- 洗浄装置

- その他の産業機械

PIPI の MIB 定義に関しての課題として、MIB のデータ構造は番号体系で管理される実績データとの意味付け管理が必要となり、これらを統一的に管理する組織の永続性が必要となる。

（2） 通信プロトコルに関する課題と展望

本標準化案で採用されている通信プロトコル SNMP は、IP ネットワーク機器の保守情報を管理するために規定されたプロトコルであり、その目的から非常に汎用的で、トランスポートレイヤに UDP（User Datagram Protocol）を使用していることから、実装もシンプルである反面下記に示す制約が存在する。

- UDP は非接続型のプロトコルであり、転送の失敗、メッセージの重複に対し、トランスポートレイヤではリカバー出来ないため、上位のアプリケーションでの対応が必要である。
- 一般的なネットワークファイアーウォールの設定では、UDP を通過させないため、インターネット環境での直接的な応用が出来ない。

しかしながら、独自に TCP/IP、UDP/IP 上に保守情報管理用のアプリケーションプロトコルをかぶせる一般的アプローチに比較し、すでに標準化が完成し広く普及したアプリケーションプロトコルである SNMP を採用するメリットは非常に大きく、今後、多くの生産設備が本仕様をサポートされることが期待される。

本標準仕様で規定するのは、生産設備機械の運転状況を収集するための通信プロトコルであり、その情報の上位システムへの接続のためのインタフェース、上位 API を規定するものではない。したがって、SNMP マネージャと上位アプリケーションとのインタフェースは別途実装する必要がある。上位システムとしては、上位アプリケーションの種類やプラットホームに依存して、マイクロソフト Windows をプラットホームとする DCS (Distributed Control System)や SCADA (Supervisory Control and Data Acquisition)などのアプリケーション、OpenMES 等の MES アプリケーション、あるいは Java ベースの監視アプリケーション等が想定される。それぞれの上位システムの要求用件に合わせ、OPC（OLE for Process Control）サーバ API、OpenMES 等で要求される API、JIM (Web based industrial monitoring based on Java technology)のデータプロバイダとしての API 等を実装する必要がある。また今後、これらの上位システムとのインタフェースには、XML (eXtensible Markup Language)による標準化の方向も考えられる。

（3） 実証

来年度は、本標準化案に基づき、数社の工作機械に実装し、さらに本専門委員会 WG1 で標準化活動を行っている OpenMES フレームワークとこれらの工作機械を接続し、2001 年 10 月にポ

ートメッセ名古屋で開催されるメカトロテック JAPAN2001 で実証試験を行う予定である。

WG1、WG2 の両 WG が提案しているオープン化された標準案を使用した上位情報系と工作機械を接続し、統合的な生産管理システムの検証を行う予定である。

また、今後の活動としては本標準案を国際的な標準仕様として提案していくこと、さらに産業界に広く認知されるため、また広く使用するために普及活動を行うことが必要である。

参考資料

[1] 製造科学技術センター、”平成 10 年度　通商産業省工業技術院委託　統合化 FA に関する調査研究成果報告書 ”、1999
[2] 製造科学技術センター、”通商産業省工業技術院委託　統合化 FA に関する調査研究　オープンコントローラ専門委員会　WG5　工作機械管理データフォーマット 1.00 ”、2000
[3] FA イントラ推進協会、 http://www.iijnet.or.jp/fa-intranet/

７． 生産システム情報の統合

Adhoc-WG において、WG1 および WG2 における活動成果についてその統合の方向性を探り、統合する際に指針となるモデルの策定を行った。

７.１　検討経過

今年度３回の会議における検討経過を、以下に述べる。

（１） 第１回会議（９月２２日開催）

統合の方向性を探るに先立ち、統合の対象となるこれまでの活動成果について、各委員から報告された。その概要は以下のとおりである。

- OpenMES の報告
  Network Computing 環境を基盤とする製造管理システム（MES）の構築を支援するミドルウエア（フレームワーク）を提供する。
  テクニカルアプローチとして、共通モデルの導入（再利用性、オープン化)、ビルディングブロックアプローチ（再利用性、オープン化)、分散処理（スケーラビリティ）が特徴。これらを実現するため CORBA と JAVA を採用した。
  システムは、MES Core Framework、MES Application Frameworks、ERP Connection Frameworks、VMC からなり、制御機械のコントローラとは VMC を介して MES と接続される。
- FA イントラネット推進協会ネット推進協会の報告
  1997 年から活動を開始し、現在４５会員で運営されている。製造現場の加工に関わる作業（加工ワークフロー）を分析し、作業をそのとき必要となる情報を整理した。さらに、それらの情報を生産現場と情報システムをつなぐために WEB 技術や情報伝達仕様、伝達手段を提案してきた。情報収集手段としては、SNMP プロトコルと MIB データ構造を定義することにより各社のコントローラから同一のフォーマットで情報収集できる仕組みを考え出した。これを SNMP 通信プロトコルガイドラインとしてまとめている。
- ＪＯＰオープンコントローラ専門委員会ワーキンググループ５
  オープンコントローラ専門委員会 WG５（管理データ WG）では、生産現場の情報系ネットワークモデルのうち、稼動管理情報に関して検討してきた。アンケートから収集の周期や項目を検討し、稼動の実績情報の標準化案をつくった。それは、稼動情報の必須項目（１０項目）と任意項目からなり、CSV 形式でデータ保存することが決められている。

上記の報告を受けて、互いの理解を深めるための討議を行った。結果として、OpenMES と、

FAイントラネット推進協会ネット推進協会提案およびJOP/OC-WG5提案との擦りあわせ部分について、確認できた。

（２） 第２回会議（１０月１９日開催）

図 ７－１に示すOpenMESとFAイントラネット推進協会の接続を検討するための実装案が提示され、これを足掛かりに、討議を進めた。

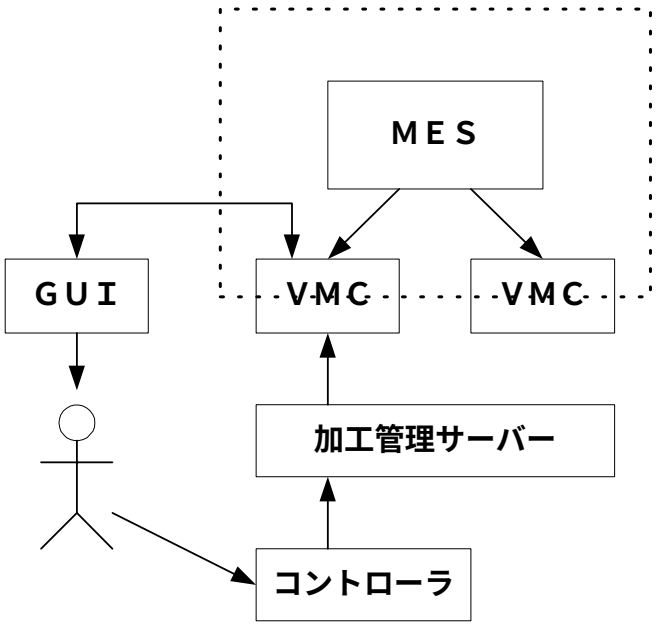

図 ７－１ OpenMES と FA イントラネット推進協会の接続検討案

議論の過程で、OpenMES として、標準化される範囲が図中の太線で示される範囲である事が確認された。OpenMES としては、機能を持ったモデルの標準化を行い、接続される個別の機器に付いての切り口に対しては、標準化する事は行わない。従って、FA イントラネット推進協会ネット推進協会が提唱する SNMP(MIB)に付いては、OpenMES の標準化の対象ではなく、標準化された VMC に準拠したアドオン機能としての位置付けとする。図で示される実装方法では、以下の様な情報の流れとなる。

- OpenMES からの作業指示
  OpenMES からの作業指示が、VMC を介して GUI アプリケーション上に通知される。
- 作業指示の実行
  オペレータが、作業指示を受けコントローラを操作する。
- 実績情報の通知
  加工管理サーバが、SNMP マネージャとなり、コントローラが、エージェントとして実績情報、加工管理情報を吸い上げ、VMC に通知する。

- コールバック
  加工管理マネージャが、VMC に対し実績情報を通知し、VMC が OpenMES に対しコールバックを行う。FA イントラネット推進協会が提唱する細かな加工管理情報は、GUI アプリケーション上に表示され、モニタ可能となる。

この検討により、OpenMES への実装が可能である事が確認された。また FA イントラネット推進協会が提唱する SNMP(MIB)の位置付けが明確になった。

OpenMES を中心とした、生産システムのモデル図のたたき台を作成した。

（3） 第３回会議（１２月１１日開催）

これまでの、討議結果のとりまとめを行った。OpenMES を中心とした生産システムのモデル図を作成した。また、OpenMES と SNMP の統合モデルを作成し、実証実験の可能性を探った。この会議での、取り纏め結果については、７.２ 項で詳述する。

７.２ 検討結果

今年度３回の会議で得られた検討結果を、以下にまとめる。

（1） OpenMES を中心とした生産システムモデル

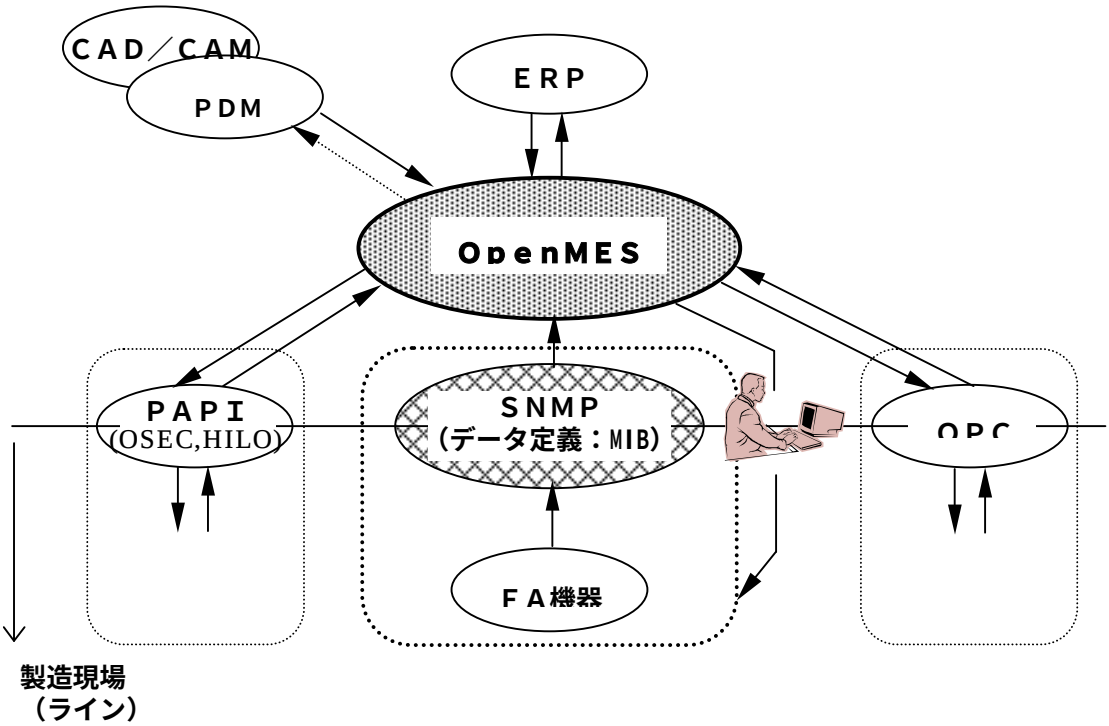

図 ７－２ OpenMES を中心とした生産システムモデル

図 ７－２に、OpenMES を中心とした生産システムモデルを示す。この図は、OpenMES と SNMP との接続関係を示し、双方の位置付けを明確化するために作成された。

本委員会の WG1 で討議されている OpenMES は、Network Computing 環境を基盤とする製造管理システム（MES）の構築を支援するミドルウエア（フレームワーク）を提供する。OpenMES は、MES Core Framework、MES Application Frameworks、ERP Connection Framewroks、VMC（Virtual Manufacturing Cell）から構成され、FA 機器のコントローラとは VMC を介して接続される。

一方、本委員会の WG2 で策定中である SNMP 通信プロトコルガイドラインでは、SNMP プロトコルと MIB データ構造を定義することにより（以下、SNMP プロトコルと MIB データ定義を併せて、SNMP と称す）、各社の FA 機器のコントローラから同一のフォーマットで情報収集できる仕組みを提案している。また、これには、オープンコントローラ専門委員会 WG５（管理データ WG）の活動成果も反映される予定である。

図 ７－２において、SNMP は、OpenMES の標準化された VMC に準拠したアドオン機能のひとつと位置付けられる。SNMP では、FA 機器のコントローラからの情報収集の仕組みは提供されているが、FA 機器のコントローラへ直接指示を行う手段が提供されていないため、GUI（人手）を介すことになる。図 ７－２では、OpenMES にとって FA 機器との接続が定義される VMC は１つであり、VMC の中を覗いてみると、SNMP であったり、PAPI であったり、OPC であったりする。CAD/CAM で生成されたファイルは、PDM を介して OpenMES に提供される。また、OpenMES からのフィードバックが CAD/CAM に対し行われるのであれば、OpenMES から PDM への矢印は実線となる。中央の横線より下は製造現場を示している。

（２） OpenMES と SNMP の統合モデル

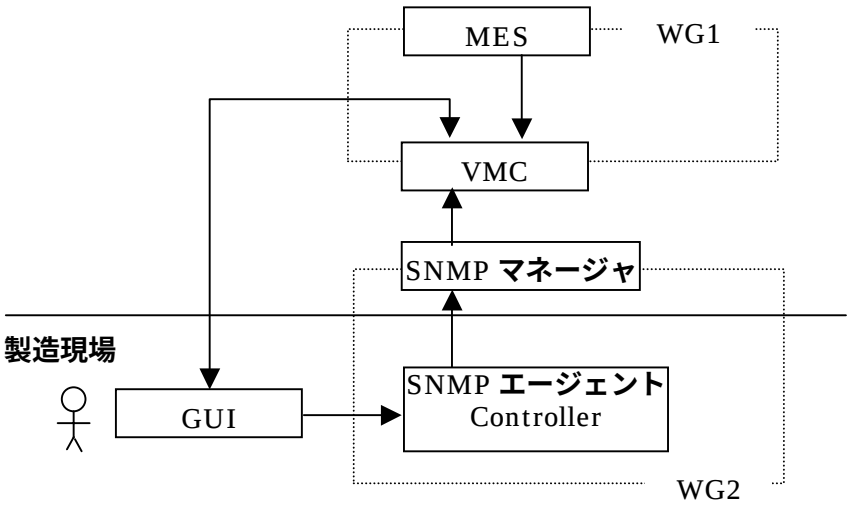

図 ７－３　OpenMES と SNMP の統合モデル

図 ７－３に、OpenMES と SNMP の統合イメージを示す。OpenMES からの作業指示が VMC を介して GUI アプリケーション上に通知される。オペレータが GUI アプリケーションをとおして作業指示を受け、コントローラを操作する。SNMP マネージャ（加工管理サーバ）が、SNMP

エージェント（コントローラ）から実績情報・加工管理情報を吸い上げ、VMC に通知する。VMC が OpenMES に対しコールバックを行う。また、実績情報・加工管理情報は、GUI アプリケーション上に表示されモニタ可能となる。

（3） OpenMES と SNMP 統合の実証実験に向けた検討

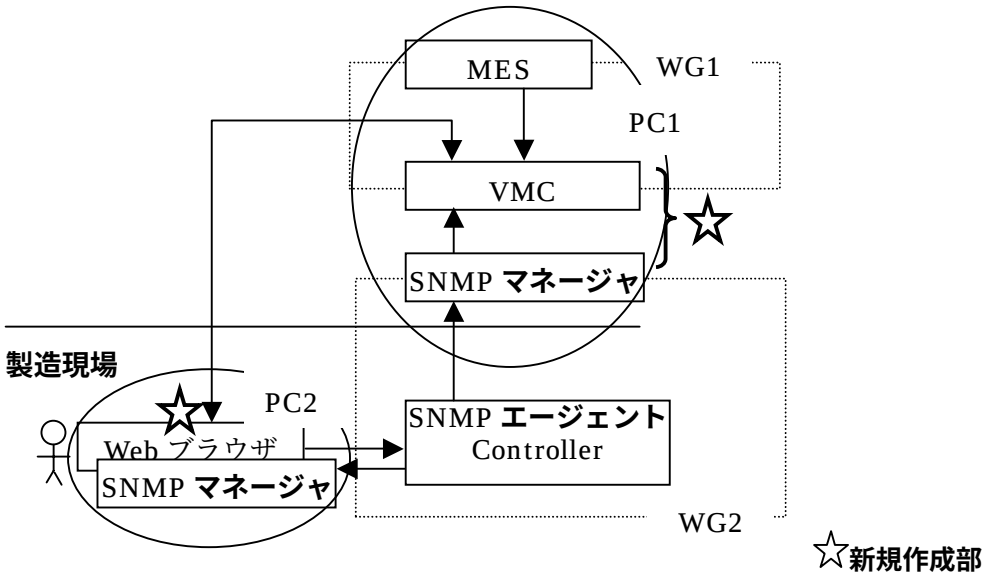

図 7－4　OpenMES と SNMP 統合の実装

図 7－4に、OpenMES と SNMP の統合を実装する場合の様子を示す。VMC の下位部分及び SNMP からコールする上位部分は、ソフトウエアベンダ(ユーザ責任)が作り込む必要がある。加工プログラムの UP／DOWN ロードについては OpenMES の機能を使う（VMC 経由でコントローラに接続する)。VMC を含んだ OpenMES と SNMP マネージャは、同一のパソコン上に構成できる。また、GUI アプリケーションの部分は、Web ブラウザと SNMP マネージャにより構成し、SNMP エージェントから得られる実績情報画面に加えて OpenMES からの処理をできるようにする。

８．今後の課題

本専門委員会では、OpenMES と SNMP について、各々の仕様定義を本年度に行い、さらに、これらを統合して、実際の生産システムにおいて利用可能であるかどうかについて検討してきた。

OpenMES および SNMP ともに、それぞれ開発してきた経緯があり、システム的には重複する部分もあるが、さまざまな機器の情報を統合し、生産計画や進捗管理の処理ソフトウェアとの連携を取ることは、火急の課題であり、今後このようなシステムを普及するうえでも重要な課題であると認識して、AdHoc グループで提案されたシステム構成で情報統合の実証試験を行うことにした。

これらの実証試験にあたっては、実証実験ワーキンググループを設置して、実証実験のシナリオ作成、変更せざるを得ないソフトウェアなど具体的な仕様の確定を行う予定である。

本実証実験で、類似の異なる経緯で作られた規格を統合することが実証できることによってより広範囲なオープン化の礎になることが期待されている。

添付資料

OpenMES 1.00J

JOP 生産システム情報統合専門委員会
© 2001 MSTC/JOP

Open MES 仕様書 Ver 1.00J

JOP 生産システム情報統合専門委員会

平成１２年度 成果報告書 添付版

目次

JOP 生産システム情報統合専門委員会 © 2001 MSTC/JOP

JOP 生産システム情報統合専門委員会 © 2001 MSTC/JOP

改定履歴

α版	99/05/27	JOP/FAM 内リリース
α１版	99/08/26	
α２版	99/09/14	JOP 内リリース
β１版	99/12/29	JOP/FAM 成果報告書
1.00J	01/04/20	JOP 生産システム情報統合専門委員会 成果報告書 添付

１．要約

本仕様書は加工組み立て生産ラインのための MES アプリケーションフレームワーク（以下 Open MES フレームワーク）の仕様を規定しており、本書の読者として、フレームワーク、コンポーネントおよびアプリケーション開発者を想定している。

ディスクリート系 MES の開発においては、これまで共通したインタフェースが存在せず、またビジネスの競争上、各企業・工場ごとに管理ノウハウが異なっていたため、個別の作り込みを行なっていた。しかし実は MES には共通なモデルが存在し、そのモデルに従ったソフトウェア部品を用意することにより、個々の生産現場にあった MES の構築を、必要なソフトウェア部品の組み合わせとそのカスタマイゼーションによって行うことが可能である。

本書に記述された MES フレームワークを用いるメリットは以下のとおりである。

- SCM/ERP との連携によって全社的な最適化が可能になる。現場の情報が SCM/ERP に反映されるため、販売、生産、調達、物流にわたる最適化が行なえるようになる。
- 生産ライン改善のための Plan Do Check Action に必要なデータが得られる。それぞれの局面において必要なデータが生成・加工される。
- Web Browser によって生産指示情報、生産実績情報にどこからでもアクセスすることが可能になり、意志決定が迅速・正確になる。
- 生産装置のマルチベンダ環境を容易に構築できる。
- これまでに開発・使用されたコンポーネントを用いることで短期間で高品質な MES を構築できる。
- ノウハウを組み込んだコンポーネントの提供によるビジネスが可能となる。

本書で記述されているフレームワークの対象範囲は、

- 工場管理
- 製造指示管理
- 製造仕様管理
- 工程仕様管理
- 工程管理
- 設備管理
- 搬送管理
- 資源管理
- スケジュール管理

であり、この仕様をもとに GUI 開発、アプリケーション開発、データのカスタマイズ、CORBA インタフェースのカスタマイズ、フレームワーク機能の追加を行なうことができる。

２．はじめに

本仕様書は、電子商取引共通基盤整備事業の一環として情報処理振興事業協会より製造科学技術センタが開発を請け負った OpenMES アプリケーションフレームワークの仕様を記述している。

一般に仕様書を記述する目的は、プロジェクトの成果を普及させるために

- 標準化すべき仕様を規定する。
- プロジェクトで開発したプログラムをベースにして、個々の MES を開発するために必要な仕様をまとめる。
- プロジェクトの成果をまとめる。

ことである。ただしこれらは一意の仕様にはならない。たとえば「プロジェクトの成果をまとめ」た仕様には、そこで開発した全ての仕様が記述されており、必づしも「標準化すべき仕様」とは一致しない。「標準化すべき仕様」とはインターオペラビリティを確保し、プラグインを可能にすることにより、オープンな世界を実現するために必要十分な仕様である。実装に依存する仕様は含まれない。

本仕様書では 2 番目の「プロジェクトで開発したプログラムをベースにして、個々の MES を開発するために必要な仕様」を規定する。プロジェクト全体の仕様に関しては、これをベースにして、フレームワークのいわゆる “ Cooking Book ” を後日執筆したい。「標準化すべき仕様」に関しては、製造科学技術センタの FA オープン推進協議会生産システムモデル専門委員会での今後の議論にゆずる。

２.１　想定している読者

本書が想定している読者は次の 3 者である。

- アプリケーションプログラマ
- コンポーネントプロバイダー
- フレームワークプロバイダー

アプリケーションプログラマーは Javadoc の仕様を参照してアプリケーションを開発することができる。

コンポーネントプロバイダーは同様に Javadoc の仕様を参照して再利用を前提としたアプリケーションフレームワークを開発することができる。

フレームワークプロバイダーは Javadoc の仕様および UML による仕様を参照にしてフレームワークを実装し、また拡張することができる。ただし、実装のための詳細な情報は本書に記述されていないので、各プロバイダでパフォーマンス、パーシステンス、セキュリティ、その他実用時に必要とされる事項について設計を行なう必要がある。

２.２　なぜフレームワークなのか

MES（Manufacturing Execution System）とは生産システムにおいて ERP 等からの生産計画および生産指示を受け、製造設備に作業指示を出し、実績収集、進捗管理を行なう工程管理を中心とした統制システムである。MES の構築、調達において次の問題が存在する。

- 生産システムは、数多くのサブ・システムから構成されるが、標準化されたインターフェースが存在せず有機的に統合することが難しい。
- 生産システムは、他種類の製造設備から構成されるマルチ・ベンダー環境が普通であるが、製造設備を管理する標準的なインターフェースが存在しないため、個別の作り込みによって製造設備をシステムに取り込む必要がある。
- 生産現場の環境の多様さに起因する様々な特殊要件が存在するため、システムのパッケージ化が難しく、生産現場ごとに個別に作り直す必要がある。

こうした理由から、生産システムの開発・保守には多大なコストがかかり、また真に統合された生産システムが存在しないために、システム全体にわたる生産の最適化を計ることが困難となっている。

しかし実は MES には共通なモデルが存在し、そのモデルに従ったソフトウェア部品を用意することにより、個々の生産現場にあった MES の構築を、必要なソフトウェア部品の組み合わせとそのカスタマイゼーションによって行うことが可能であると我々は考えている。このようなアプローチの実現のためにオブジェクト指向を採用した。オブジェクト指向によって、生産システムをモデル化しソフトウェア部品に分解することができる。また同時に、これらのソフトウェア部品間の連携を定義することにより、構築される生産システムのテンプレートを予め与えることができる。このテンプレートは MES アプリケーションフレームワークと呼ばれ、新たな機能部品の追加や、既存機能部品の振る舞いの変更などにより、必要なカスタマイゼーションを容易に行うことが可能である。

MES アプリケーションフレームワークには機能面以外にも次の要件が存在する。

- 小規模な生産ラインから大規模な生産ラインまで対応できるスケーラビリティを持つこと
- ハードウェア、OS 等のプラットフォームに依存しないオープン性を持つこと

このため CORBA の分散オブジェクトを用い、複数のサーバにオブジェクトを分散させ、また実装言語として JAVA を採用することによりスケーラビリティとオープン性を確保する。

こういった MES アプリケーションフレームワークを導入することにより、現場の情報化が実現し、必要なデータにどこからでもアクセスできるようなシステムを短期間・低コストでユーザ自身が構築することが可能になる。さらにこのような情報化を推し進めることにより、ERP（Enterprise Resource Planning）、SCM（Supply Chain Management）、設備制御システムと協調して最適な生産が実現可能となる。

２.３　なぜ CORBA なのか

MES は複数の設備、サーバ、クライアント端末から構成され、本質的に分散処理が要求される。また分散処理によって小規模な生産システムから複数の工場を含む生産システムまでサポートできるスケーラビリティを実現することができる。さらに拡張性、オープン性の観点からソフトウェアのコンポーネント化、プラグイン性が要求されている。このような環境を実現するプラットホームとして最近は

- DCOM
- CORBA

が注目されている。MES の場合、ハードウェアに関しては、計算機の種類が多岐に渡り、PC、ワークステーション、ビジネスコンピュータ、大型コンピュータから構成される。このような環境でシームレスなシステムを構築していくためにはプラットホームに依存しない CORBA が適している。

２.４　フレームワークの範囲

MES が管理対象とする製造工程には、一般的に下記の三つの形態があると考えられる。

- 連続プロセス
- バッチ・プロセス
- ディスクリート・プロセス

オープン MES では、この内のディスクリート・プロセスをモデル化の対象として選択した。ディスクリート・プロセスとは、ロットなどの単位でまとめられたワークが、工程ごとに区切られた製造設備間を移動しながら、加工を施されるものである。機械加工や組立の工場がこれに相当する。

ディスクリート・プロセスには、さらにコンベヤ・ラインの様に固定された工程経路をとるフロー・ショップと、工程経路に自由度のあるジョブ・ショップに分かれる。オープン MES では、ジョブ・ショップに対応したモデルを用意し、その特殊形としてフロー・ショップに対応する。

フレームワークに含まれる機能は以下の通りである。詳しくは以下の章で説明する。

- 工場管理
- 製造指示管理
- 製造仕様管理
- 工程仕様管理
- 工程管理
- 設備管理
- 搬送管理
- 資源管理
- スケジュール管理

２.５　フレームワークのメリット

２.５.１　エンドユーザ

MES 導入によるメリットは次の通りである。

- Web Browser によって生産指示情報、生産実績情報にどこからでもアクセスすることが可能になり、意志決定が迅速・正確になる。例えば営業員は在庫状況、仕掛かり状況から納期を返答することが可能になり、また現場管理者は来期の生産スケジュールを知ることにより、負荷の集中や仕掛かり在庫を減少させることが可能になる。
- SCM/ERP との連携によって全社的な最適化が可能になる。現場の情報が SCM/ERP に反映されるため、販売、生産、調達、物流にわたる最適化が行なえるようになる。
- 生産ライン改善のための Plan Do Check Action に必要なデータが得られる。Plan はスケジューリング、Do は作業指示、Check は進捗管理、Action は実績データであり、それぞれの局面において必要なデータが生成・加工される。また作業日報等をわざわざ作成することはない。
- MES による指示により複数の工場があたかも一つの工場として機能することができる。製品種別ごとに工場の組み合わせを柔軟に変更することができる。
- MES フレームワークのメリットとしては既存のソフトウェアコンポーネントを調達してシステムを構築できることがあげられる。

２.５.２　システムインテグレータ

MES アプリケーションフレームワーク導入によるメリットは次の通りである。

- 生産装置のマルチベンダ環境を容易に構築できる。
- これまでに開発・使用されたコンポーネントを用いることで短期間で高品質な MES を構築できる。
- Web を用いた先進的なソリューションを提案、提供できる。

２.５.３　コンポーネントプロバイダ

MES アプリケーションフレームワーク導入によるメリットは次の通りである。

- 自社のノウハウを組み込んだコンポーネントの提供によるビジネスが可能となる。

２.５.４　装置メーカ

MES アプリケーションフレームワーク導入によるメリットは次の通りである。

- 生産システムごとの、装置の繋ぎ込みが不要となる。

２.６　仕様に関する規定・記述法

通常 CORBA を採用したフレームワークの仕様は IDL で記述されることが多い。しかし、本プロジェクトでは、次の 2 つの理由から Java の API として仕様を定める。

- アプリケーションプログラマが CORBA の仕様に精通していなくとも開発できるようにする。
- フレームワークのパフォーマンスを高めるため、必要なオブジェクトだけを CORBA オブジェクトとする。サーバとクライアントとで共有する必要がある非 CORBA オブジェクトは両者の間での値渡しとする。この機能は現在 CORBA2.0 では規約化されていない。Visibroker のベンダ固有の機能として提供されている。したがって、この機能は標準的な IDL では記述できない。

Java クラスは階層的なグループに分類されている。最上位のグループを機能グループ、それを分割したものをコンポーネントと呼ぶ。コンポーネントはクラスおよびインタフェースから構成される。本仕様書はクラスまたはインタフェースの機能について記述している。メソッドおよびその引数についての詳細は付録（別冊？）の Javadoc による仕様書を参照していただきたい。

グラフィカル表記は UML に準じている。

クラス階層図、シーケンス図において導出クラスの例が現れることがある。本仕様書では導出クラスの Prefix として“ FMS ”を用いる。

２.７　この仕様書の使用法

フレームワークのカスタマイズには次の 5 段階が考えられる。

- GUI 開発

  Visual Age Java, Jbuilder, Visual Cafe 等の開発環境あるいは JDK を用いて、フレームワーク API を呼び出す Graphical User Interface の開発・変更を行なう。

- アプリケーション開発

  フレームワーク API を呼び出すアプリケーションのロジックをクライアント側あるいはサーバ側に実装する。サーバ側の場合、クライアントとの HTML の授受にはサーブレットや JSP を用いるのであれば、その設計・実装も行なう。

- データのカスタマイズ

  個々の MES ごとにデータをカスタマイズすることを想定しているクラスが用意されているので、必要なメンバ変数・メソッドを追加する。

- CORBA インタフェースのカスタマイズ

  設備インタフェース、スケジューラインタフェースなど CORBA インタフェースにメンバ変数・メソッドを追加する場合は、CORBA インタフェースの導出インタフェースを定義し、そのインタフェースを実装する。それに伴い、新しいメソッドを呼び出すアプリケーションを開発することになる。また CORBA クラスに変数を追加する場合、実装クラスの導出クラスを定義・実装する。CORBA インタフェースを変更せず、ロジックを変更する場合は、実装クラスの導出クラスを定義し、メソッドを実装する。

- フレームワーク機能の追加

  仕様書で規定されているクラス定義を呼び出すことにより、再利用を考慮した MES アプリケーションフレームワークの機能グループ・コンポーネントを追加する。分散処理が必要な場合、あるいは適切な場合は CORBA インタフェースを新規に定義し、実装する。

２.８　関連活動

２.８.１　OMG

Manufacturing Domain Task Force では MES の RFI が提出され、Response が 6 個提出された。現在それを基に 3 つの RFP が準備されている。

２.８.２　ISO

TC 184/SC5/WG1 (Enterprise Integration)で New Work Item を審議中であり、日本国内対策委員会として OpenMES を検討している。

２.８.３　MSTC/JOP

生産システムモデル専門委員会で仕様について審議中である。

３．Open MES Objects

Open MES フレームワークは次の 10 機能グループから構成される。

- 工場管理機能グループ
- 製造指示管理機能グループ
- 製造仕様機能グループ
- 工程仕様機能グループ
- 工程管理機能グループ
- 設備管理機能グループ
- 搬送管理機能グループ
- 資源管理機能グループ
- スケジュール管理機能グループ
- 共通機能グループ

本章では、これらの機能グループに含まれるクラスおよびインターフェースについて記述する。

図 3-1に製造指示クラス（MesProductionOrder）、ロットクラス(MesProductionLot)、ロットジョブクラス(MesLotJob)、プロセスジョブクラス(MesProcessJob)、作業指示クラス(MesWorkOrder)、抽象工程設備クラス（MesProcessEquipment）の関係を示す。製造指示を表現する MesProductionOrder オブジェクトはロットを表現する複数の MesProductionLot オブジェクトから構成される。MesProductionOrderManager オブジェクトはシステム内に存在する複数の MesProductionOrder オブジェクトを管理している。MesProductionLot オブジェクトに関する作業を MesLotJob オブジェクトは表現しており、個々の工程（加工、検査、洗浄等）を表現する MesProcessJob オブジェクトから構成される。作業指示を表現する MesWorkOrder オブジェクトは MesLotJob オブジェクトと関連付けられており、ディスパッチャ（MesDispatcher）によって抽象工程設備 MesProcessEquipment オブジェクトに差し立てられる。

図 3-2 にこれらのクラス間のコラボレーション図を示す。アプリケーション（図中 AP 製造指示管理）が MesProductionOrderManager の addProductionOrder メソッドを呼び出し、製造指示（MesProductionOrder オブジェクト）を作成する。次に createProductionLots メソッドを呼び出すことによって、MesProductionLot オブジェクトが生成され、releaseProductionOrders メソッドの呼び出しによって、MesLotJobManager に対し MesLotJob の生成を指示する。次にアプリケーション（図中 AP 製造ロット管理）が createSchedule メソッドを呼び出すことにより、スケジュールを作成し、outputSchedule メソッドの呼び出しにより、作成されたスケジュールをフレームワークに取り込む。releaseLotJobs メソッドにより、生産ラインにロットが投入され、addWorkOrder メソッドにより、作成されたスケジュールに基づき MesWorkOrder(作業指示)を MesProcessEquipment(工程設備)に割り付ける。作業が終了したら finishWorkOrder メソッドが呼び出され、setWorkResult メソッドにより、関係している MesWorkOrder, MesLotJob, MesProcessJob, MesProductionOrder オブジェクトに作業実績が関連づけられる。

クラス図/製造関係

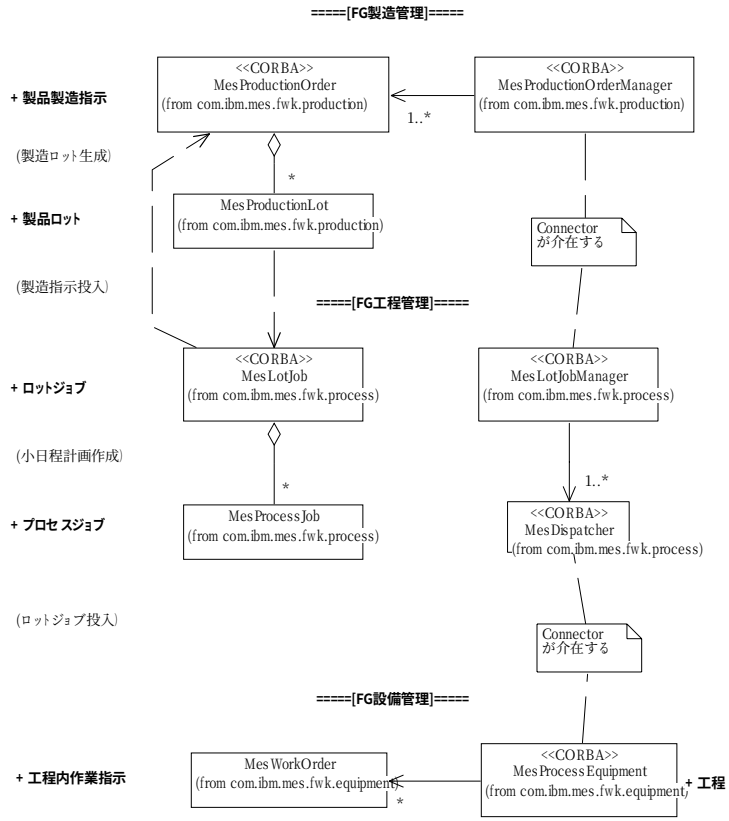

図 3-1 製造関連クラス図

コラボレーション図/UG計画管理

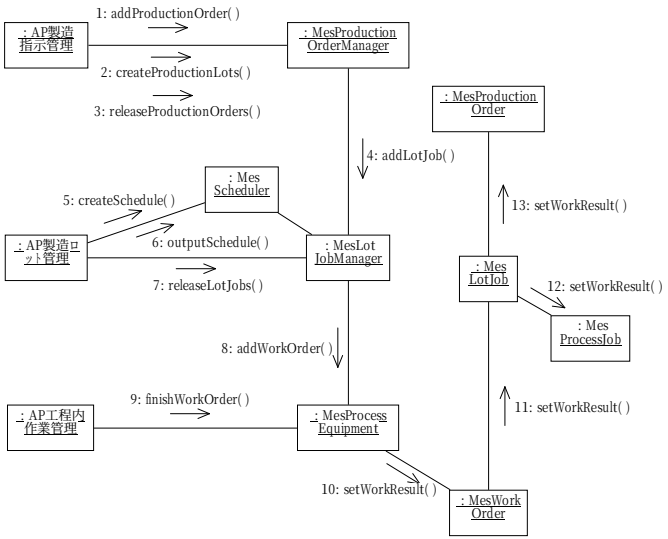

図 3-2 製造関連クラスコラボレーション図

実際にフレームワークを稼動させるために CORBA インタフェースを実装したクラスが必要である。CORBA インタフェース名を“ Foo ”とした場合、実装クラス名を“ FooImpl ”そのオブジェクトを立ち上げ管理するクラスを“ FooServer ”としている。“ FooImpl ”に関してはメソッド定義が“ Foo ”と同一のため、本仕様書では記述しない。ただし、クラス図やシーケンス図においては理解を容易にするため出現することがある。

３．１　工場管理機能グループ

本機能グループは工場の構成と稼動スケジュールを管理する。

- 工場構成管理コンポーネント
- 設備スケジュール管理コンポーネント

から構成される。

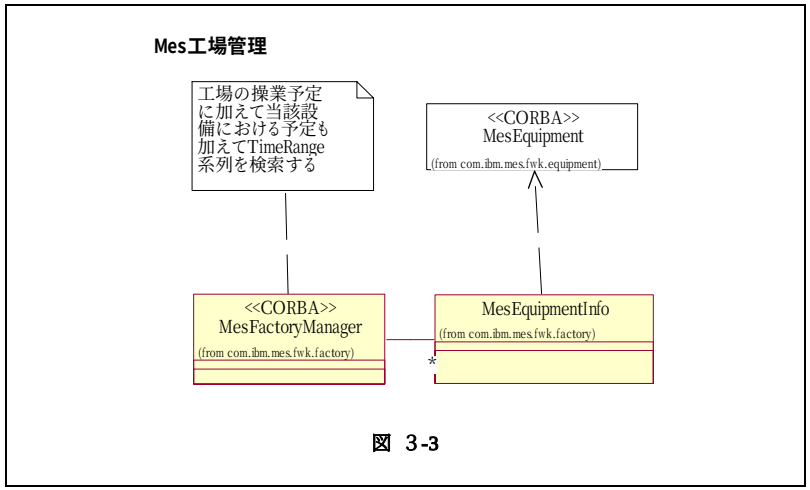

図 3-3

３．１．１　工場構成管理コンポーネント

本コンポーネントは工場に設置されている設備情報を管理する。

（１） MesFactoryManager

本インタフェースは製造設備情報、工場オペレーション・プラン、設備オペレーション・プランおよび保守計画に対する操作（設定、検索、変更、削除）を定義する。さらに、設備休止時間帯、ある時間帯における使用可能設備を取得するメソッドを定義している。

（２） MesEquipmentInfo

本クラスは製造設備の情報すなわち種別、型番、設置場所の値を管理する。MES ごとにカスタマイズすることを想定している。

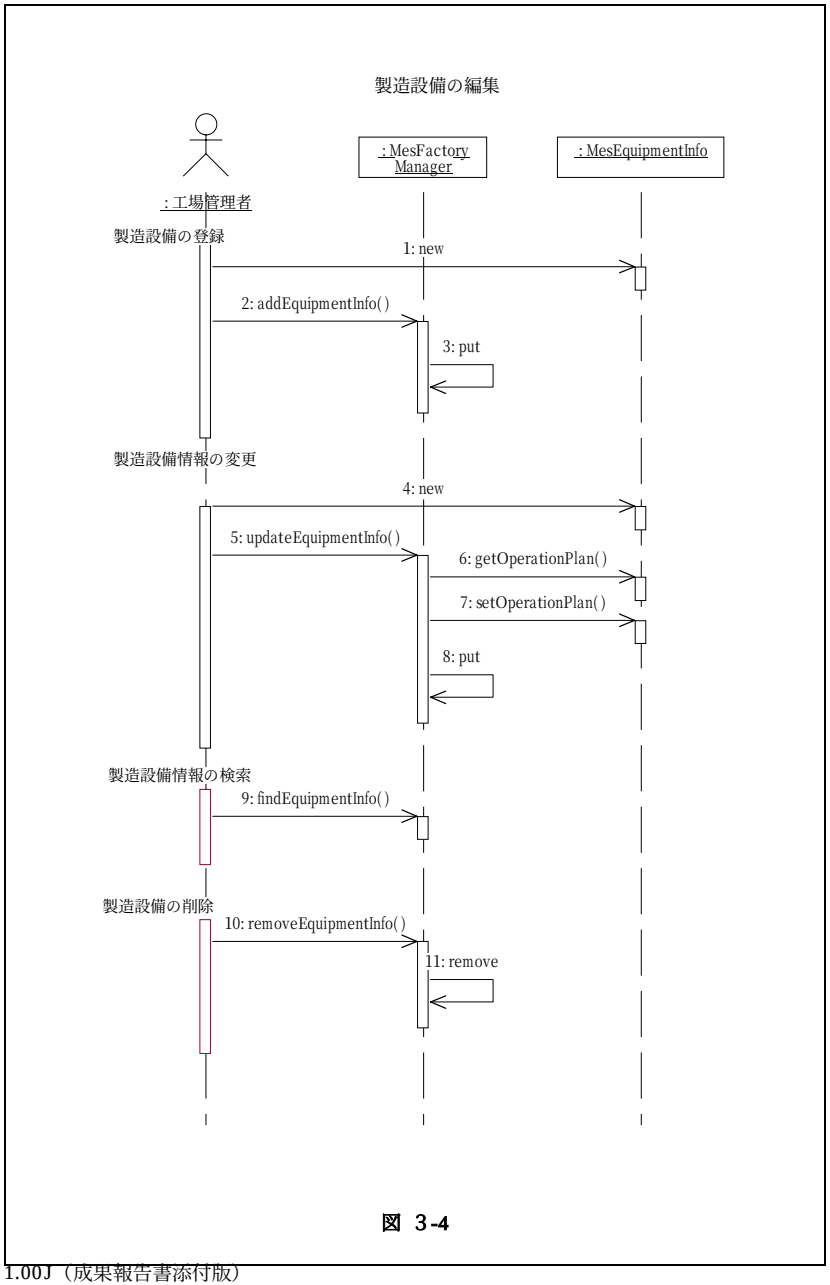

図 3-4

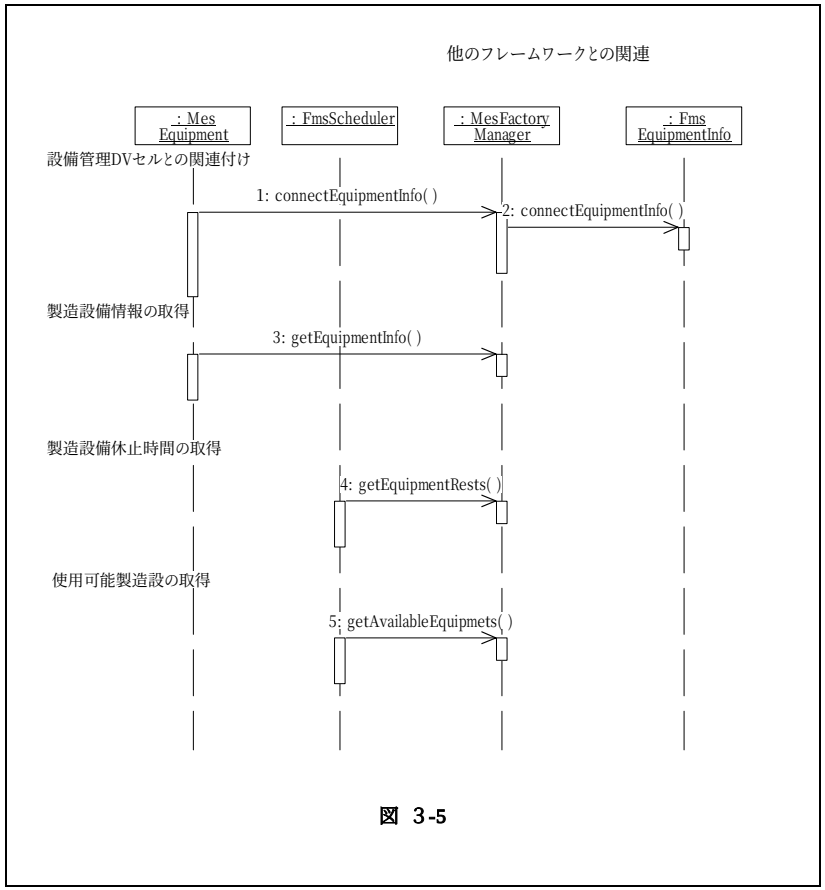

図 3-5

３．１．２　設備スケジュール管理コンポーネント

本コンポーネントは工場および個々の設備スケジュールを管理する機能を提供するクラスから構成される。設備の保守時期も本コンポーネントを用いて表現される。またスケジュールに対する実績値も管理する。

スケジュールは休止期間の集合として定義される。工場スケジュールは工場の休止期間、設備スケジュールは設備の休止期間の集合を表す。保守に関しては保守期間を表す。スケジューラが稼動期間として認識するのは、工場の休止期間と設備の休止期間の論理和の補

集合に相当する期間である。

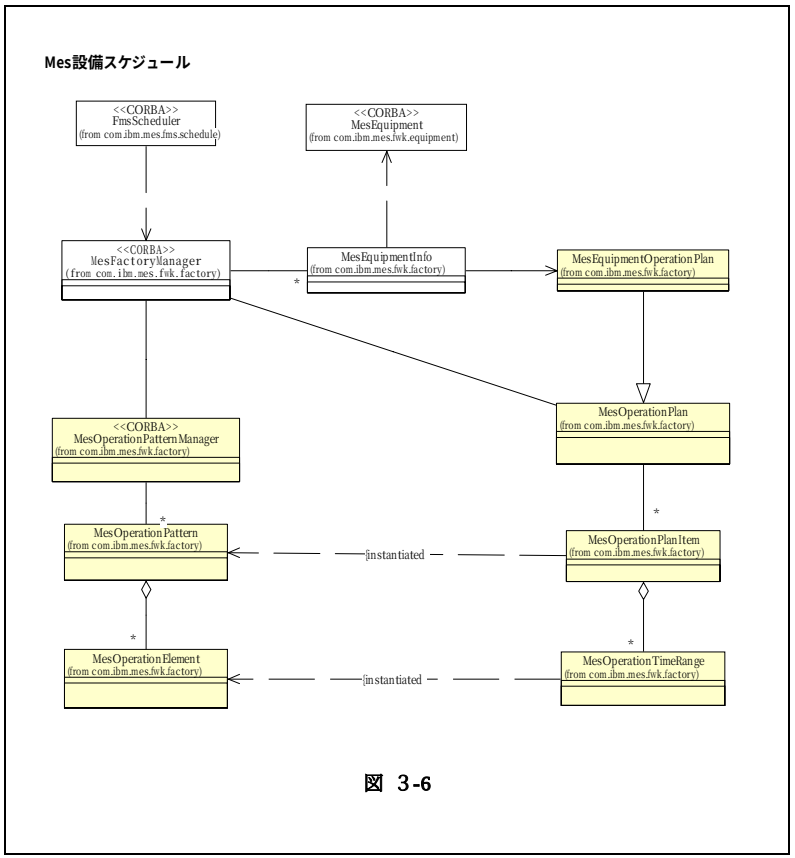

図 3-6

（1） **MesOperationPatternManager**

本インタフェースは製造設備の管理と工場オペレーションの計画パターン に対する操作（設定、取得、検索、変更、削除）をおこなうメソッドを定義する。

（2） **MesOperationPattern**

本クラスは MesOperationElement の集合体である。MesOperationElement 間の Interval を指定することにより、スケジュールのパターンを規定する。Interval の単位は分である。週毎であれば、7*24*60=10080 となる。

（3） **MesOperationElement**

本クラスはスケジュールの時間パターン要素を表す。開始時間からの Offset と期間で定義される。単位は両者とも分である。

（4） **MesOperationPlan**

本クラスはスケジュールに対する操作（設定、取得、削除）を提供する。異なるパターンを持つ MesOperationPlanItem の集合を保持する。

（5） **MesEquipmentOperationPlan**

本クラスは設備スケジュールおよび保守管理 に対する計画を持つ。

（6） **MesOperationPlanItem**

本クラスは MesOperationPattern を参照して、生成される一つのスケジュール要素である。開始日時と繰り返し回数が指定される。MesOperationTimeRange の集合体を保持する。

（7） **MesOperationTimeRange**

本クラスはタイム・レンジを表し、予定と実績を管理する。開始日時、期間で定義される。期間の単位は分である。

（8） **MesPlanTimeRange**

タイム・レンジを定義する。休止開始時刻、期間で定義される。このクラスは、休止時間の計算結果として Hashtable の要素として返される。

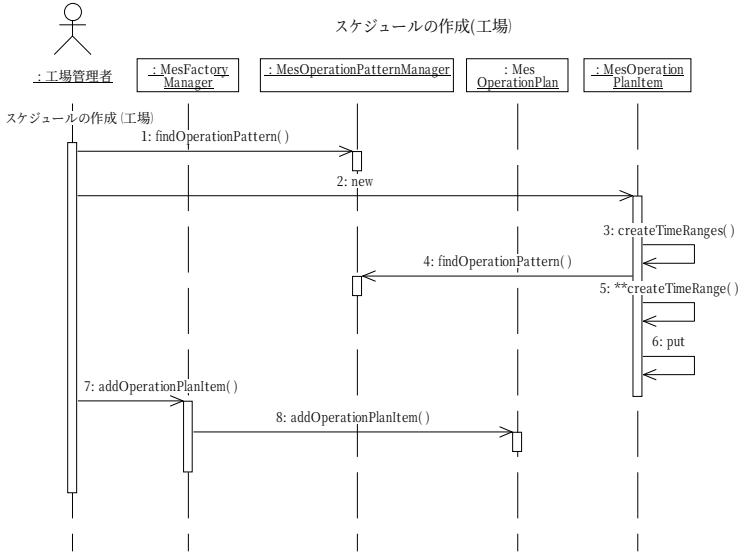
スケジュールの作成(工場)
：工場管理者
：MesFactory Manager
：MesOperationPatternManager
：Mes OperationPlan
：MesOperation PlanItem
スケジュールの作成（工場）
1: findOperationPattern( )
2: new
3: createTimeRanges( )
4: findOperationPattern( )
5: **createTimeRange( )
6: put
7: addOperationPlanItem( )
8: addOperationPlanItem( )

図 3-7

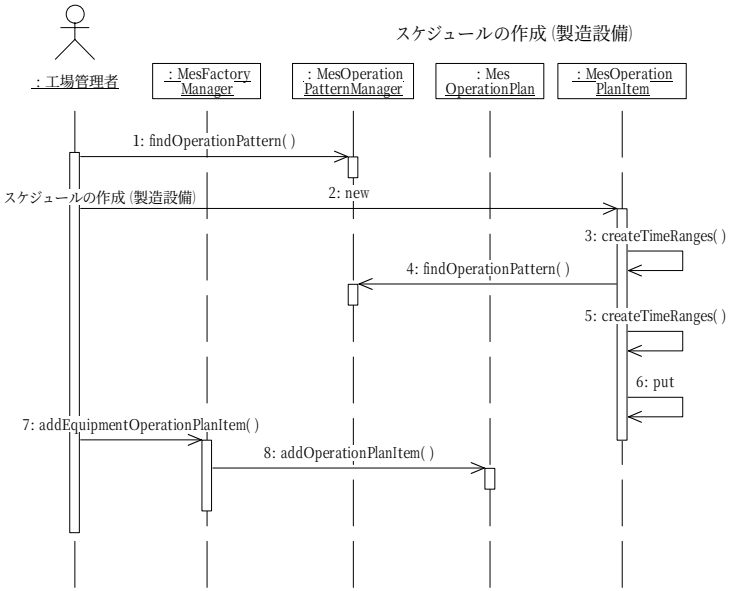
スケジュールの作成（製造設備）
：工場管理者
：MesFactory Manager
：MesOperation PatternManager
：Mes OperationPlan
：MesOperation PlanItem
1: findOperationPattern( )
2: new
スケジュールの作成（製造設備）
3: createTimeRanges( )
4: findOperationPattern( )
5: createTimeRanges( )
6: put
7: addEquipmentOperationPlanItem( )
8: addOperationPlanItem( )

図 3-8

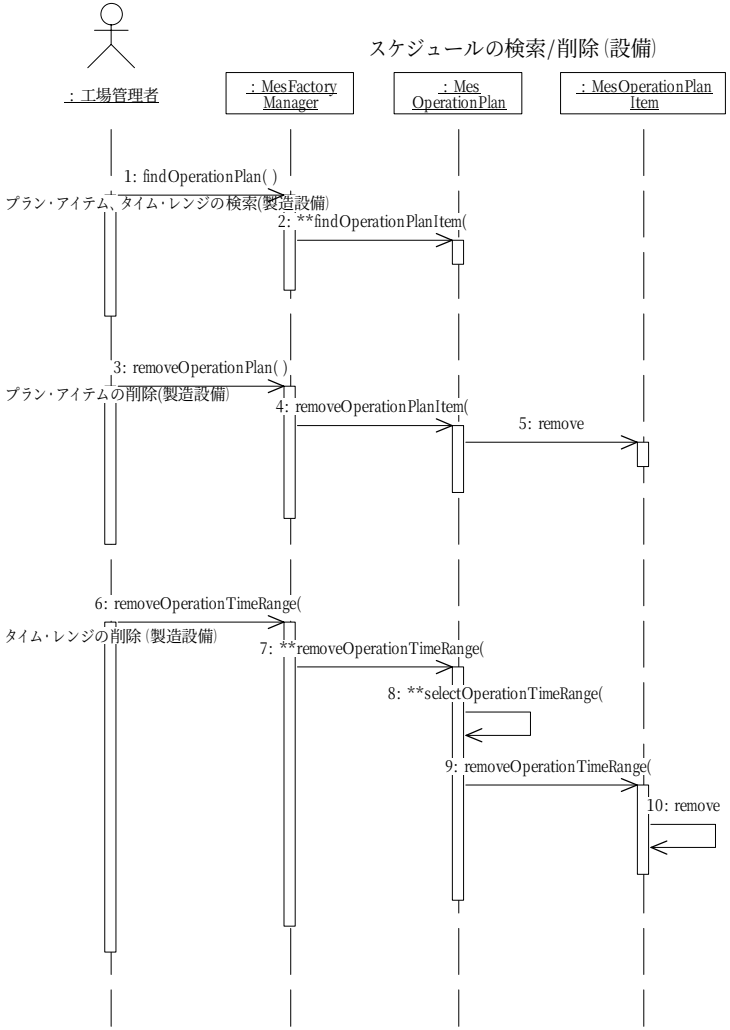
スケジュールの検索/削除（設備）
：工場管理者
：MesFactory Manager
：Mes OperationPlan
：MesOperationPlan Item
1: findOperationPlan( )
プラン・アイテム、タイム・レンジの検索(製造設備)
2: **findOperationPlanItem(
3: removeOperationPlan( )
プラン・アイテムの削除(製造設備)
4: removeOperationPlanItem(
5: remove
6: removeOperationTimeRange(
タイム・レンジの削除（製造設備）
7: **removeOperationTimeRange(
8: **selectOperationTimeRange(
9: removeOperationTimeRange(
10: remove

図 3-9

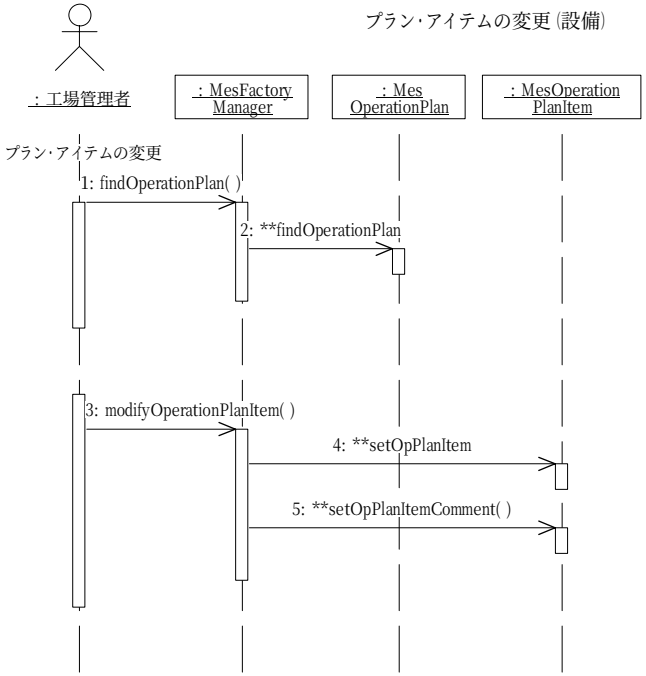
プラン・アイテムの変更（設備）
：工場管理者
：MesFactory Manager
：Mes OperationPlan
：MesOperation PlanItem
プラン・アイテムの変更
1: findOperationPlan( )
2: **findOperationPlan
3: modifyOperationPlanItem( )
4: **setOpPlanItem
5: **setOpPlanItemComment( )

図 3-10

３．２　製造指示管理機能グループ

製造指示管理機能グループは、生産計画システムとのインターフェースを提供する。製造指示を受けて必要な数のロットを作成し、工程に投入し、その製造指示の進捗を監視することができる。すなわち、製造指示と工程内を流れるロットの間の紐付けを管理する。

この紐付けは、受注生産か見込み生産かといった生産形態、MRP や製番管理といった管理方式、ロット分割など、様々な要因で変わってくる可能性がある。

３．２．１　製造指示管理コンポーネント

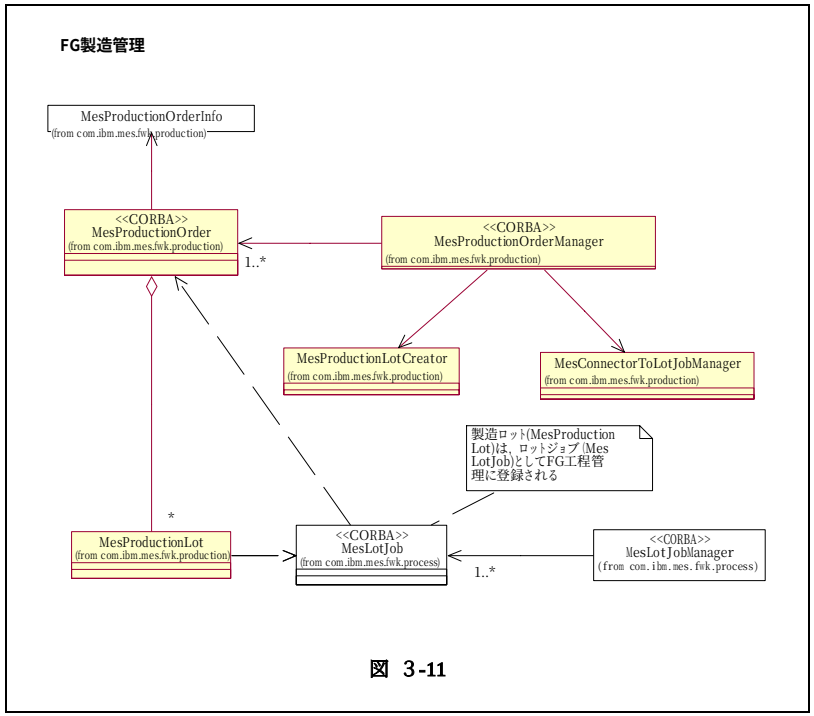

図 3-11

（１）　MesProductionOrderManager

本インタフェースは MesProductionOrder を管理（登録、削除、検索、編集）するメソッドを定義する。

（２）　MesProductionOrder

本インタフェースは製造指示を表す。以下の機能を持つメソッドが定義されている。

- 製造指示 ID の取得

- 製品 ID の設定、取得
- 製品仕様 ID の取得
- 製品仕様バージョンの設定、取得
- 製造ロットの登録、取得
- 製造ロット数の取得
- 製造ロットの選択
- 製造指示優先順位の設定、取得
- 製造指示の状態の設定、取得
- 製造指示情報の設定、取得
- 現在の製造実績数量のカウント
- 製造予定数量の設定、取得
- 製造実績数量の設定、取得
- 着手予定日時の設定、取得
- 着手実績日時の設定、取得
- 最早製造指示投入時刻の設定、取得
- 最遅製造指示完了時刻の設定、取得
- 終了予定日時の設定、取得
- 終了実績日時の設定、取得
- 製造指示が登録された時刻の設定、取得。
- 製造指示状態集合の取得。
- すべての製造ロットの終了実績日時が設定されたかチェックする。
- すべての製造ロットの着手実績日時が設定されていないかチェックする。
- 全ての製造ロットの削除
- 特定の製造ロットの削除
- 製造指示を一時停止状態にする。
- 製造指示の一時停止状態を解除する。
- ProductionLot の納期設定。
- ProductionLot の着手予定日時の設定
- ProductionLot の着手実績日時の設定
- ProductionLot の終了予定日時の設定
- ProductionLot の終了実績日時の設定
- ProductionLot の実績数量の設定

（３）　MesProductionOrderState

本クラスは製造指示状態集合を表す。製造指示には以下の状態が存在する。

- 未処理

- 製造ロット生成済
- 製造指示投入済
- 小日程計画生成済
- 製造指示投入済
- ロットジョブ開始後
- 製造指示完了
- 製造指示一時停止
- 破棄

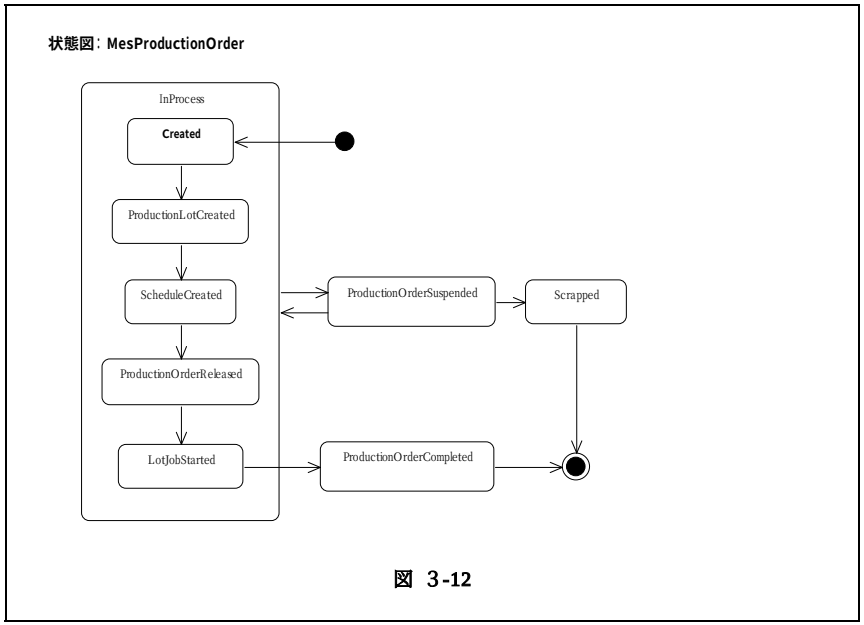

図 3-12

（4） **MesProductionOrderInfo**

本クラスは製造指示情報を表す。このクラスは適用事例において派生させることによって, MesProductionOrder では 提供できない個別の属性やそれに関わる操作を定義するために用意されている。

3.2.2 製造ロット作成管理コンポーネント

（1） **MesProductionLot**

本クラスは製造ロットを表す。以下の機能を持つメソッドが定義されている。

- 製造ロット ID の設定、取得
- 製造指示 ID の設定、取得
- 納期の設定、取得
- 与えられた製造ロットがこの製造ロットと同じかどうか判定する。
- 予定数量の設定、取得
- 実績数量の設定、取得
- 着手予定日時の設定、取得
- 着手実績日時の設定、取得
- 終了予定日時の設定、取得
- 終了実績日時の設定、取得
- ロットジョブの優先順位を設定
- ロットジョブを一時停止状態に設定
- ロットジョブの再開

（2） **MesProductionLotCreator**

本クラスは MesProductionLot を生成する。

（3） **MesConnectorToLotJobManager**

本クラスは製造指示管理機能グループと工程管理機能グループとのインタフェース機能を持つ。MesProductionLot を MesLotJobManager に送付する。

３.３　製品仕様管理機能グループ

３.３.１　製品仕様管理コンポーネント

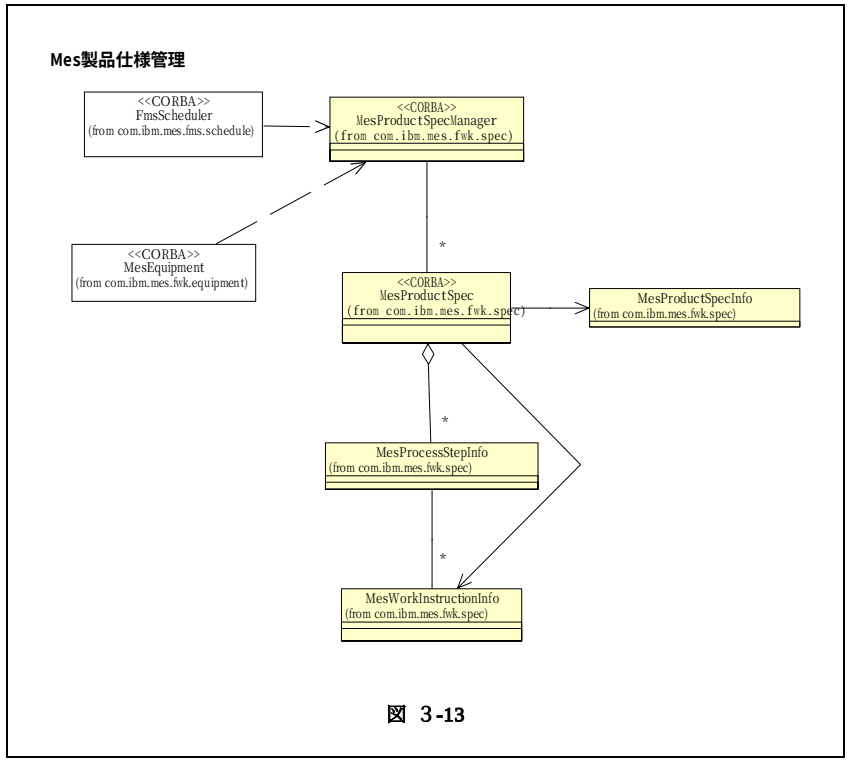

図 3-13

（１） **MesProductSpecManager**

本インタフェースは製品仕様（MesProductSpec）に対する操作（登録、削除、検索、変更）を行なうメソッドを定義する。

（２） **MesProductSpec**

本インタフェースは製品仕様（MesProductSpecInfo）と工程ステップ情報(MesProcessStepInfo)を管理する次のメソッドを定義する。

- 製品 ID の設定、取得。
- 製品仕様名の設定、取得
- MesProcessStepInfo の登録、削除、検索
- MesWorkInstructionInfo の登録、削除

- MesProductSpecInfo の設定、取得
- 仕様の参照回数を 1 増減する。参照回数が 0 のときのみ当該 MesProductSpec オブジェクトを削除できる。
- 製品仕様の参照回数の設定、取得
- この製品仕様が使用中であるかどうかチェックする。
- 工程経路の設定、取得
- 製品バージョンの設定、取得

（３） **MesProcessStepInfo**

本クラスは工程内作業指図の登録と工程ステップ属性の変更をする。

- MesWorkInstructionInfo 工程内作業指図の登録、取得、削除
- 関連先 MesProductSpecInfo オブジェクト・リファレンスを設定、取得する。
- 工程ステップ番号の設定、取得
- 工程ステップ名の設定、取得

（４） **MesWorkInstructionInfo**

本クラスは工程内作業指図を表す。MES ごとにカスタマイズすることを想定している。

- 製造設備 ID の設定、取得
- 製品設備型番の設定、取得
- 関連 ProcessStepInfo オブジェクトの設定、取得
- 設備モードの設定、取得

（５） **MesProductSpecInfo**

本クラスは製品仕様の追加情報を表しており、MES ごとにカスタマイズすることを想定している。

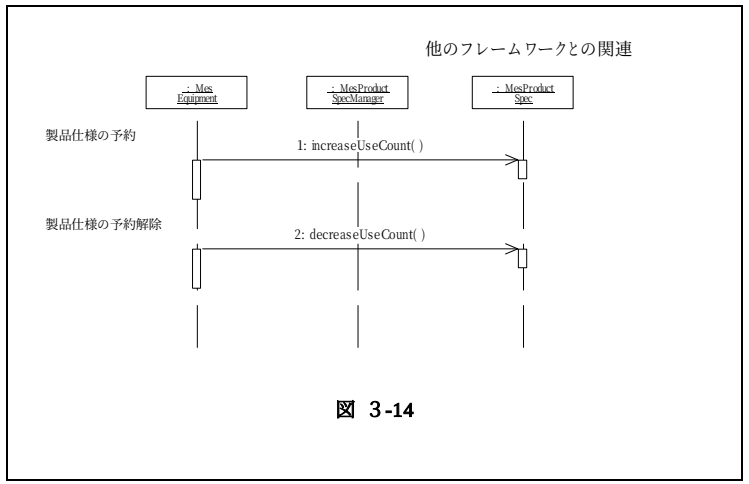

図 3-14

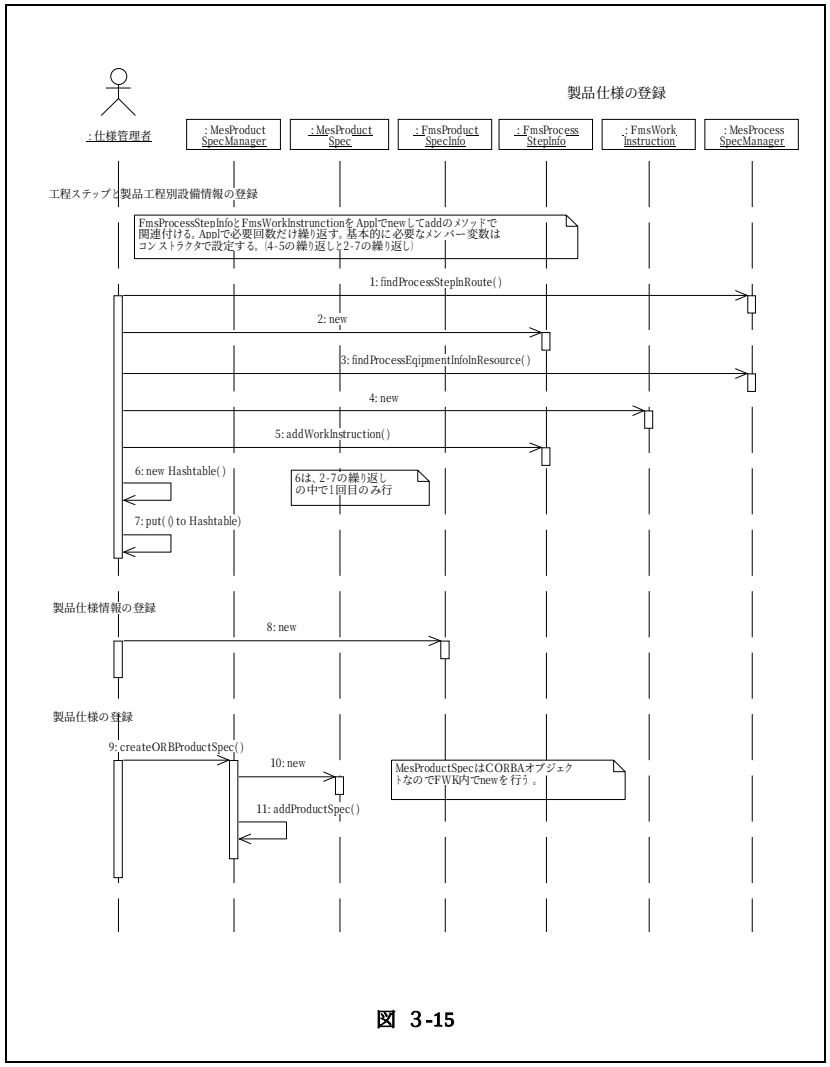
製品仕様の登録
:仕様管理者
:MesProduct SpecManager
:MesProduct Spec
:FmsProduct SpecInfo
:FmsProcess StepInfo
:FmsWork Instruction
:MesProcess SpecManager
工程ステップと製品工程別設備情報の登録
FmsProcessStepInfoとFmsWorkInstructionをApplでnewしてaddのメソッドで関連付ける。Applで必要回数だけ繰り返す。基本的に必要なメンバー変数はコンストラクタで設定する。(4-5の繰り返しと2-7の繰り返し)
1: findProcessStepInRoute( )
2: new
3: findProcessEqipmentInfoInResource( )
4: new
5: addWorkInstruction( )
6: new Hashtable( )
6は、2-7の繰り返しの中で1回目のみ行
7: put( ) to Hashtable)
製品仕様情報の登録
8: new
製品仕様の登録
9: createORBProductSpec( )
10: new
MesProductSpecはCORBAオブジェクトなのでFWK内でnewを行う。
11: addProductSpec( )

図 3-15

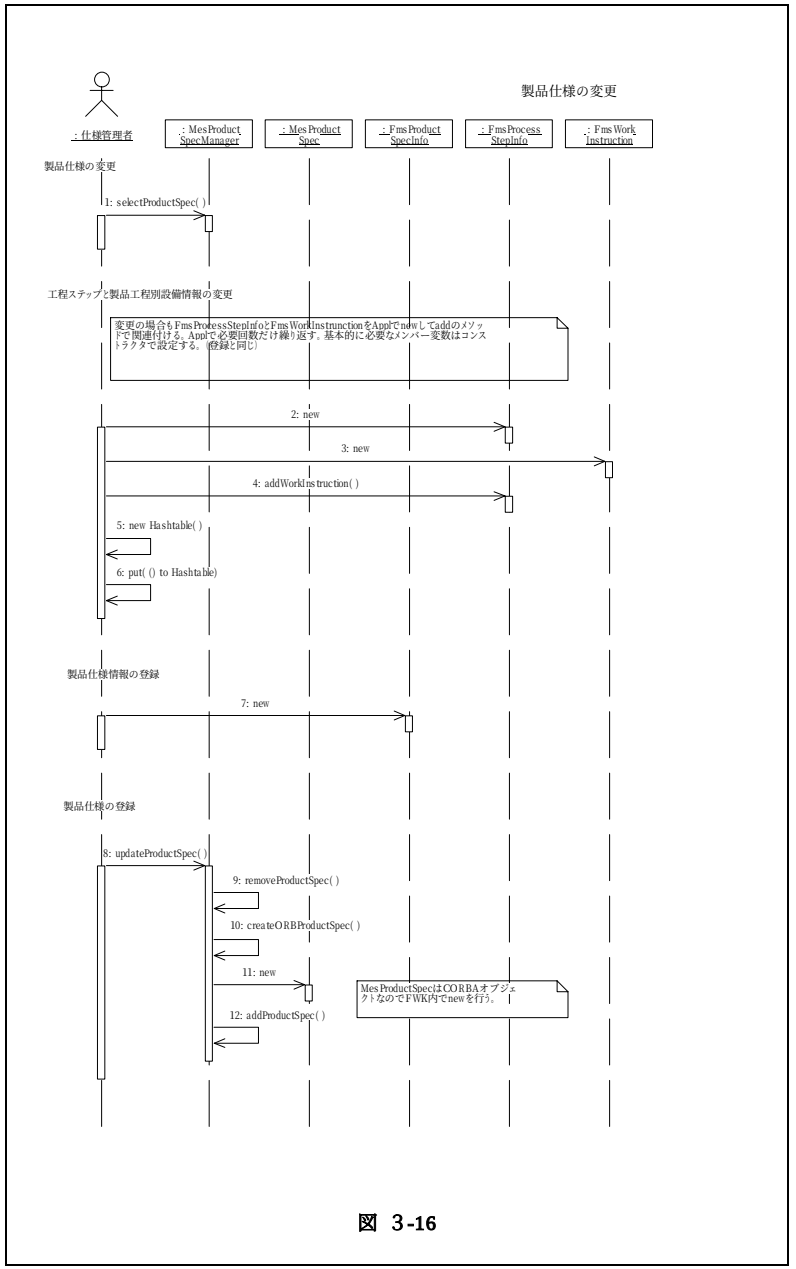
製品仕様の変更
:仕様管理者
:MesProduct SpecManager
:MesProduct Spec
:FmsProduct SpecInfo
:FmsProcess StepInfo
:FmsWork Instruction
製品仕様の変更
1: selectProductSpec( )
工程ステップと製品工程別設備情報の変更
変更の場合もFmsProcessStepInfoとFmsWorkInstructionをApplでnewしてaddのメソッドで関連付ける。Applで必要回数だけ繰り返す。基本的に必要なメンバー変数はコンストラクタで設定する。(登録と同じ)
2: new
3: new
4: addWorkInstruction( )
5: new Hashtable( )
6: put( ) to Hashtable)
製品仕様情報の登録
7: new
製品仕様の登録
8: updateProductSpec( )
9: removeProductSpec( )
10: createORBProductSpec( )
11: new
MesProductSpecはCORBAオブジェクトなのでFWK内でnewを行う。
12: addProductSpec( )

図 3-16

３．４　工程管理機能グループ

工程管理機能グループは、ロットに対する製造工程の実行すなわち設備に対し作業指示を行ない、作業進捗を追跡する。工程管理の中心になるのは、ロットに対する作業計画と作業実績を管理する LotJob と呼ばれるクラスと、作業計画に基づいて作業指示を各設備に差し立てる Dispatcher と呼ばれるクラスである。

ここでロットと呼ばれるのは、工程内を流れる物理的なロットと対応する製造ロットである。今回は、投入から完了までロットの統合や分割は行わないものとしてモデル化した。また、製造指示との紐付けは、製造指示管理機能グループで取り扱うものとした。

次のコンポーネントから構成される。

- 製造ロット管理コンポーネント
- 工程内作業指示管理コンポーネント
- 投入指示管理コンポーネント
- 搬送指示管理コンポーネント

３．４．１　製造ロット管理コンポーネント

（１）　MesLotJobServer

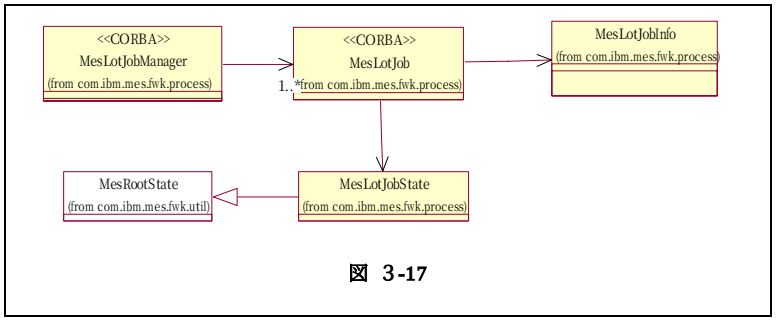

図 3-17

本クラスはCORBA オブジェクトである MesLotJob を実装したオブジェクトを立ち上げ、管理する。

（２）　MesLotJobManagerServer

本クラスはCORBA オブジェクトである MesLotJobManager を実装したオブジェクトを立ち上げ、管理する。

（３）　MesLotJobCreator

本クラスはロットジョブを生成する。Creator に関しては共通機能グループを参照していただきたい。

（４）　MesLotJobHandlerServer

本クラスは CORBA オブジェクトである MesLotJobHandler を実装したオブジェクトを立ち上げ、管理する。

（５）　MesLotJobHandler

このインタフェースは MesLotJob を異なるアドレス空間に配置するために用意されている。本仕様書では詳述しない。

（６）　MesLotJobManager

MesLotJob を管理するインタフェースであり、以下の機能を提供するメソッドが宣言されている。

- ロットの作成、削除、検索
- プロセスジョブの検索
- 全ロットジョブの投入（実行されるのはディスパッチ以後）
- 特定のプロセスジョブの投入
- 全プロセスジョブのディスパッチ
- 優先順位の高いプロセスジョブのディスパッチ

（７）　MesLotJob

ロットジョブを表現するインタフェースであり、以下の機能を提供するメソッドが宣言されている。

- プロセスジョブの登録、検索
- 優先順位の設定・取得
- 開始予定時刻、終了予定時刻、開始実績時刻、終了実績時刻の設定・取得
- 最早投入時刻、最遅完了時刻の設定・取得
- 製造予定数量の設定・取得
- 製造実績数量の取得
- 実績欠陥数量の取得
- ロットジョブ情報の設定・取得
- 状態の設定・取得
- 実行開始要求
- 一時停止、再開通知
- プロセスジョブ完了通知
- プロセスジョブ中断通知
- プロセスジョブ取り消し通知
- プロセスジョブ一時停止通知
- プロセスジョブ再開通知

変更内容が MesWorkOrder(作業指示)まで伝達されるものとして以下のものがある。

- 本オブジェクトを構成している全プロセスジョブの優先順位
- 本オブジェクトを構成している特定のプロセスジョブの優先順位
- 本オブジェクトを構成している特定のプロセスジョブの開始予定時刻

以下のものについては実行時に履歴情報をデータベースに記録することができる。

- 実行開始要求
- プロセスジョブ完了通知
- プロセスジョブ中断通知
- プロセスジョブ取り消し通知
- プロセスジョブ一時停止通知
- プロセスジョブ再開通知

（８） **MesLotJobState**

本クラスはロットジョブの状態を表現する。

状態図：**MesLotJob**

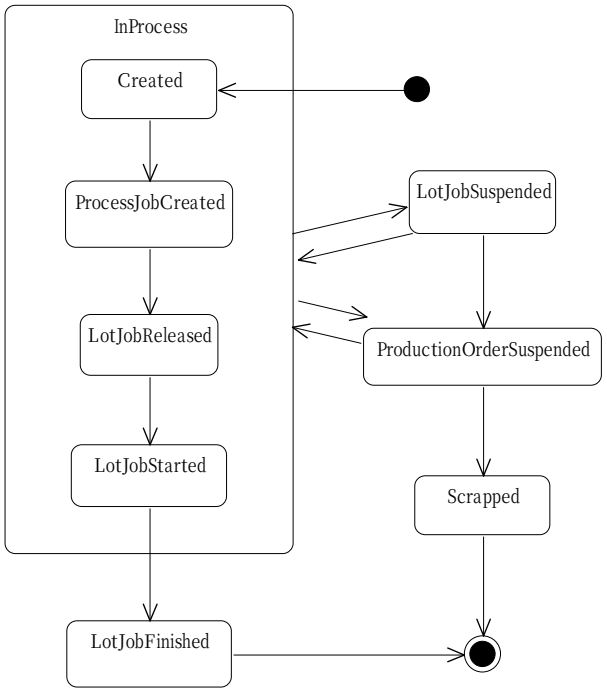

図 3-18

（９） **MesLotJobInfo**

このクラスは適用事例において派生させることによって, MesLotJob では 提供できない個別の属性やそれに関わる操作を定義するために用意されている。アプリケーション開発者は CORBA オブジェクトである MesLotJob をカスタマイズする必要がない。

３．４．２　工程内作業管理コンポーネント

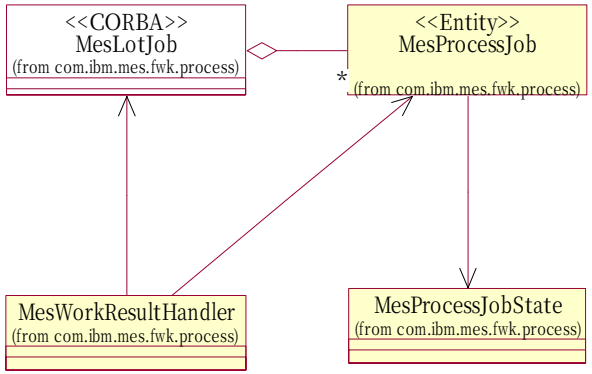

図　3-19

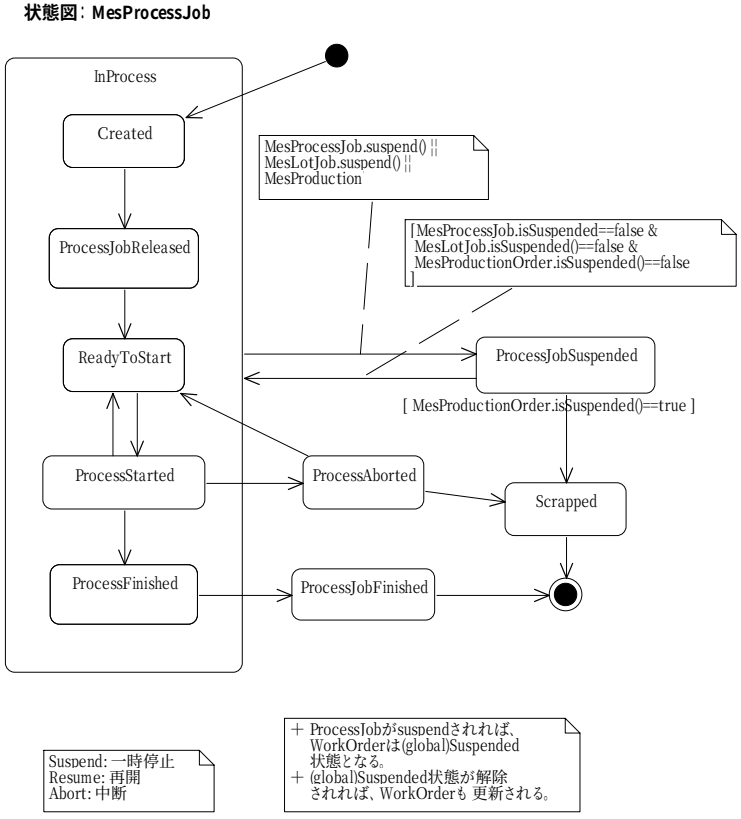

図　3-20

（１）　**MesProcessJob**

本クラスはプロセスジョブを表現する。以下の機能を持つ。

- 作業予定数量の設定、取得
- 作業実績数量の設定、取得
- 作業予定設備の設定、取得

- 作業実績設備の設定・取得
- 優先順位の設定、取得
- 工程資源の設定、取得
- 作業開始予定時刻の設定、取得
- 作業終了予定時刻の設定、取得
- 作業開始実績時刻の設定、取得
- 作業終了実績時刻の設定、取得
- 作業実績データの設定
- 実績欠陥数量の設定、取得
- 作業取り消し時刻の設定、取得
- 状態の設定、取得
- 投入要求
- 開始要求
- 中断要求
- 取り消し要求
- 終了要求
- 一時停止通知
- 再開通知

（２） **MesProcessJobState**

本クラスはプロセスジョブの状態を表現する。

（３） **MesWorkResultHandler**

本クラスは MesProductionOrder に、関連を辿って次のものを通知する。

- 最初の MesProcessJob の開始
- 最後の MesProcessJob の完了
- MesWorkResult（作業開始と作業完了と両方ある）

３．４．３　投入指示管理コンポーネント

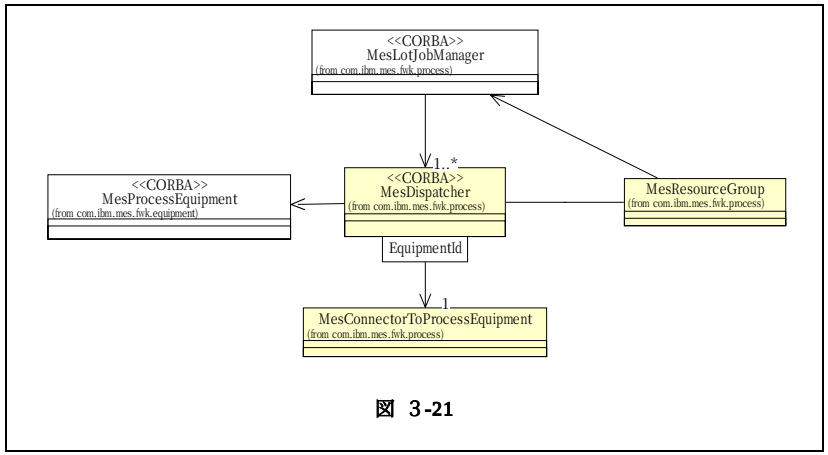

図 3-21

（１） **MesDispatcherServer**

本クラスは CORBA オブジェクトである MesDispatcher を実装したオブジェクトを管理する。

（２） **MesDispatcher**

本インタフェースは MesWorkOrder を MesProcessEquipment に差し立てるメソッドを定義する。

- ディスパッチングに関する履歴情報を登録する。
- プロセスジョブを工程設備コネクタに登録する。
- リリースされた全プロセスジョブをディスパッチする。
- リリースされた全プロセスジョブを指定された工程設備にディスパッチする。
- ディスパッチング順位の最も高いものから指定された数のプロセスジョブを工程設備にディスパッチする。
- ディスパッチング順位の最も高いものから指定された数のプロセスジョブを特定の工程設備にディスパッチする。
- 指定された工程設備に登録されたプロセスジョブを検索する。
- ロットジョブ･マネジャーID を返す。
- プロセスジョブの比較演算子を返す。
- 指定された工程設備コネクタが登録されているかどうかを返答する。
- 代替設備に該当するコネクタに登録されたプロセスジョブ状態を開始不可とする。
- プロセスジョブが削除されたことを工程設備コネクタに通知する。
- 代替設備に該当するコネクタに登録されたプロセスジョブを削除する。

- 工程設備コネクタ･クラス名を設定する。
- ロットジョブ･マネジャーID を設定する。
- プロセスジョブの比較演算子を設定する。
- プロセスジョブのパラメタが更新されたことを工程設備コネクタに通知する。

（３） **MesConnectorToProcessEquipment**

本クラスは工程管理機能グループと設備管理機能グループとのインタフェースを取る。この両者の接続は特に MES でカスタマイズする必要性が高く、機能グループ間での依存関係を単純化するために用意されている。

カスタマイズする必要性があるのは、次ぎの２つである。

- 作業指示（MesWorkOrder）の工程設備への登録・削除
- プロセスジョブに対する優先順位、作業開始予定時刻、一時停止状態の設定・解除について、作業指示への反映

（４） **MesResourceGroup**

代替設備を表す。

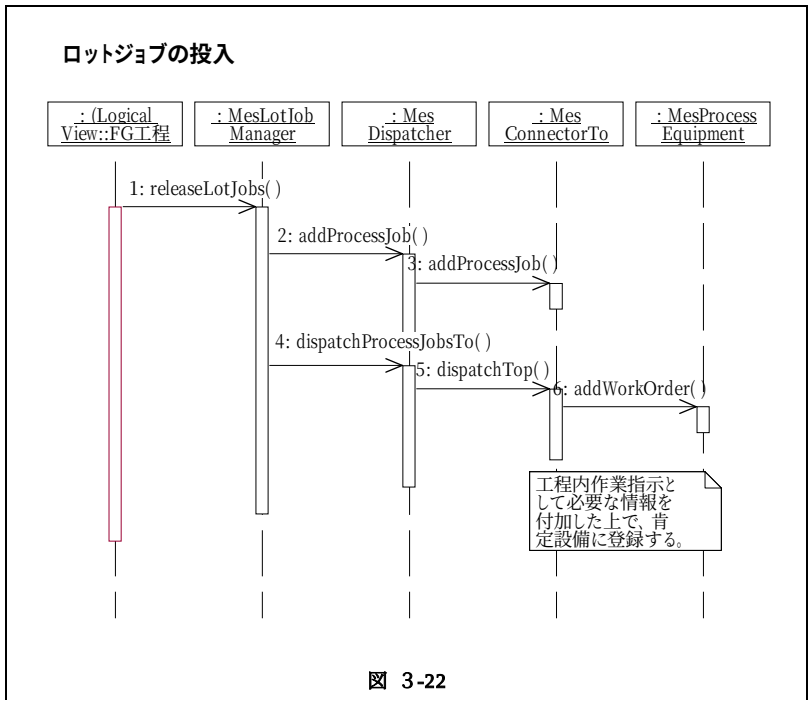

図 3-22

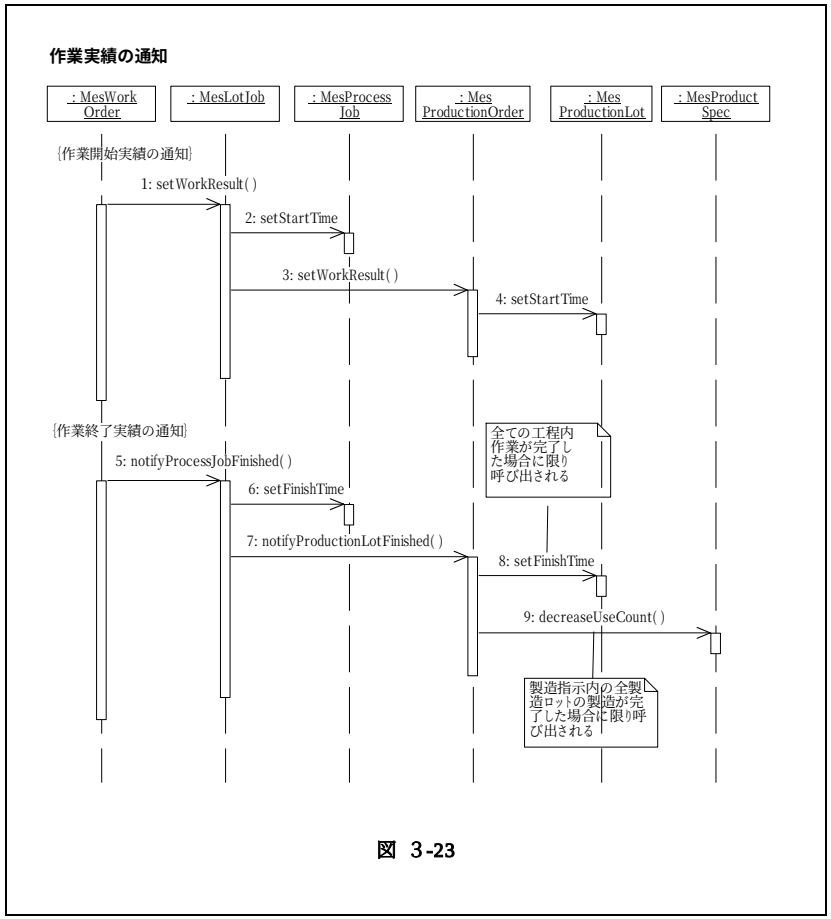

図 3-23

３．４．４ 搬送指示管理コンポーネント

本コンポーネントは搬送指示を管理する機能を提供する。

（１） **MesTransferRequestManager**

本インタフェースは、ロットを設備に対して配送する要求または設備からの引き取りの要求を受け付け、実行可能になったら、例えば対象設備の作業が完了したら、搬送設備に

対し搬送を要求する以下の機能を提供するメソッドを定義している。

- ロットを設備に配送するように要求する。
- セットアップのために、空パレットの搬入を要求する。
- 空になったパレットの搬出を要求する。

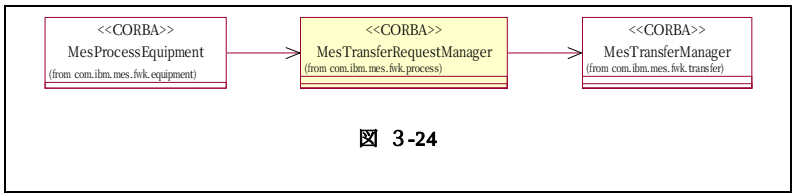

図 3-24

- ロットを設備から引き取るように要求する。
- 搬送完了の通知を受け取る。

３．５　資材管理機能グループ

資材管理機能グループは、資材の利用実績を管理する。Consumable クラスと呼ぶ消費財と Durable クラスと呼ぶ耐久財を管理対象として定義している。ただし、資材の在庫管理全体まではカバーしておらず、例えば資材の種別ごとのマスタ情報などは持っていない。これは、資材の在庫管理は ERP などの外部システムが行うものと仮定しており、ここで用意したクラスは、外部システムに対して利用状況を通知するためのインタフェースとして使われることを想定しているためである。

３．５．１　資材管理コンポーネント

（１）MesInventoryManagerServer

本クラスは MesInventoryManager を実装したオブジェクトを立ち上げ、管理する。

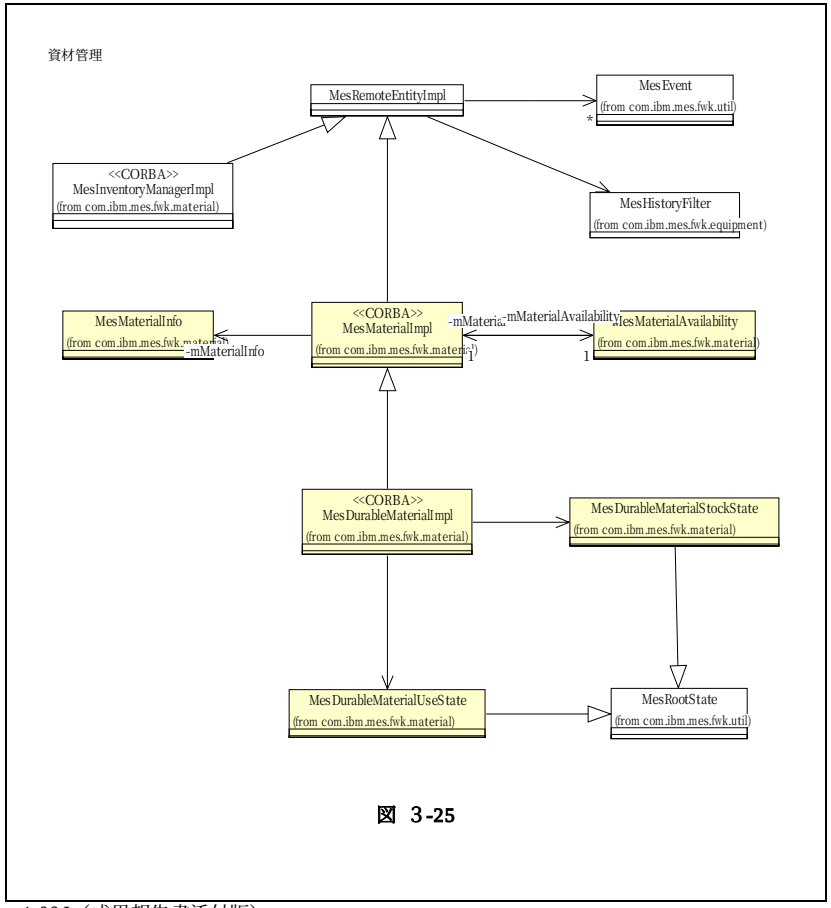

図 3-25

（２） **MesInventoryManager**

このインタフェースは、資材（MesMaterial）を管理（登録、検索、削除）するメソッドを定義する。

（３） **MesMaterial**

このインタフェースは、資材を定義する。

- 所在の設定、取得
- MesMaterialInfo の設定、取得
- 資材名の設定、取得
- 資材タイプの取得
- 寿命に達したかどうかを調べる。

（４） **MesMaterialAvailability**

この抽象クラスは、資材の寿命を調べる機能を定義する。

（５） **MesMaterialInfo**

この抽象クラスは、資材情報を定義する。

（６） **MesDurableMaterial**

このインターフェースは、資材から継承した耐久資材の機能を定義する。

- 使用回数の実績を設定、取得する。
- 使用時間の実績を設定、取得する。
- 残りの可能使用回数を取得する。
- 残りの可能使用時間を取得する。
- MesDurableMaterialStockState を設定、取得する。
- 使用回数の上限値を設定、取得する。
- 使用時間の上限値を設定、取得する。
- MesDurableMaterialUseState を設定、取得する。
- 実績使用回数を 1 回加算する。
- 使用時間の実績を加算する。

（７） **MesDurableMaterialUseState**

このクラスは、耐久資材の使用状態集合を定義する。

- 使用可　(Ready)
- 使用中　(BeingUsed)
- 予約中　(BeingReserved)
- 使用不可　(NotUsable)
- 廃棄　(Disposed)

（８） **MesDurableMaterialStockState**

このクラスは、耐久資材のストック状態集合を定義する。

- 保管中　(InStock)

- 出庫中　(BeingStockOut)
- 設備内　(InEquipment)
- 入庫中　(BeingStockIn)
- 廃棄　(Disposed)

（９） **MesDurableMaterialAvaiability**

このクラスは、耐久資材の寿命を調べる機能を定義する。寿命判定のロジックを実装する必要がある。

３．６　搬送管理機能グループ

搬送管理機能グループは、ロットを搬送する機能を提供する。

ただし搬送システムは、通常専門の搬送装置メーカによって一括してインストールされることが多い。従って、搬送システム内の AGV や自動倉庫の標準モデルが存在し得るかどうかは不明である。このためオープン MES では、搬送システムのインタフェースの抽象度に段階を設け、最も高い抽象度では、搬送システム全体をブラック・ボックスとして扱えるように考慮した。

３．６．１　搬送管理コンポーネント

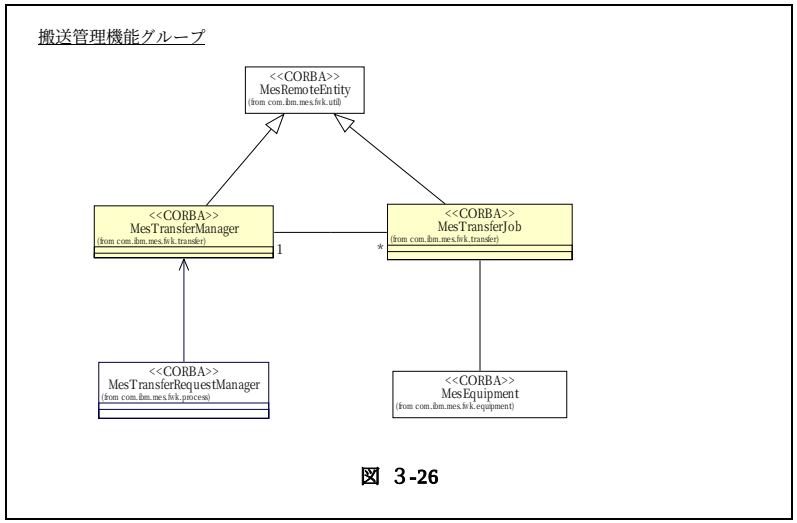

図 3-26

（１）**MesTransferManager**

このインターフェースは、搬送管理機能を定義する。

- MesTransferRequestManager を設定、取得する。
- 貯蔵庫から設備へロットの配送を行う。
- 設備から貯蔵庫へロットの配送を行う。
- ロットの設備間搬送を行う。
- 実行中の全ての搬送ジョブを取得する。
- 搬送が完了した通知を受け取る。
- 実行中の搬送ジョブの個数を取得する。
- 搬送ジョブを投入する。
- MesTransferJob オブジェクトを取得する。

（２）**MesTransferJob**

このインターフェースは、搬送ジョブの機能を定義する。

- 搬送元設備を取得する。
- 搬送先設備を取得する。
- ロットを取得する。
- 優先順位を設定、取得する。
- 搬送ジョブを要求した日時を取得する。
- 搬送ジョブを開始した日時を取得する。
- 搬送ジョブを中断する。
- 搬送ジョブを取り消す。
- 搬送ジョブを完了する。
- 搬送ジョブを完了した日時を取得する。
- 搬送ジョブの状態を取得する。
- 搬送ジョブの終了通知を受け取る。
- 搬送ジョブの開始通知を受け取る。
- 搬送ジョブを選択する。
- 搬送ジョブの選択を解除する。

（３）**MesTransferJobState**

本クラスは搬送ジョブの状態を表す。 搬送ジョブには以下の状態が存在する。

- 搬送可能　(ReadyForTransfer)
- 搬送待ち　(WaitingForTransfer)
- 搬送中　(UnderTransfer)
- 搬送アボート　(TransferAborted)
- 搬送中止　(TransferCanceled)
- 搬送完了　(TransferCompleted)

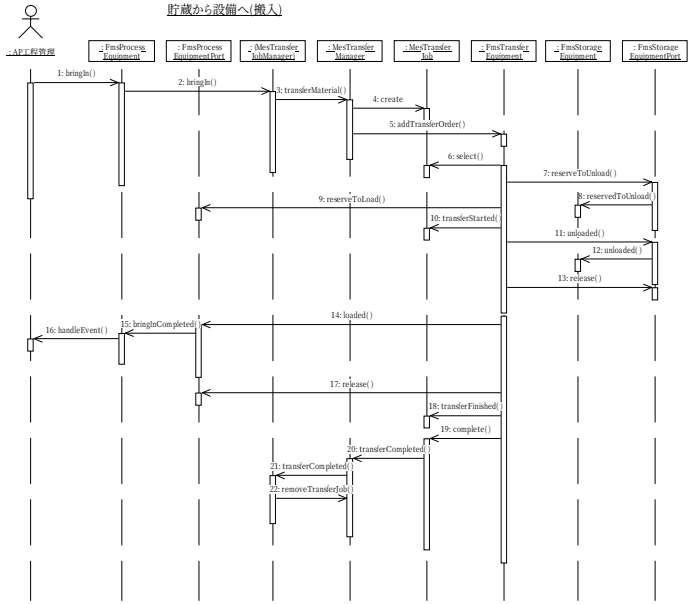
貯蔵から設備へ(搬入)
: AP工程管理
: FmsProcess Equipment
: FmsProcess EquipmentPort
: (MesTransfer JobManager)
: MesTransfer Manager
: MesTransfer Job
: FmsTransfer Equipment
: FmsStorage Equipment
: FmsStorage EquipmentPort
1: bringIn( )
2: bringIn( )
3: transferMaterial( )
4: create
5: addTransferOrder( )
6: select( )
7: reserveToUnload( )
8: reservedToUnload( )
9: reserveToLoad( )
10: transferStarted( )
11: unloaded( )
12: unloaded( )
13: release( )
14: loaded( )
15: bringInCompleted( )
16: handleEvent( )
17: release( )
18: transferFinished( )
19: complete( )
20: transferCompleted( )
21: transferCompleted( )
22: removeTransferJob( )

図 3-27

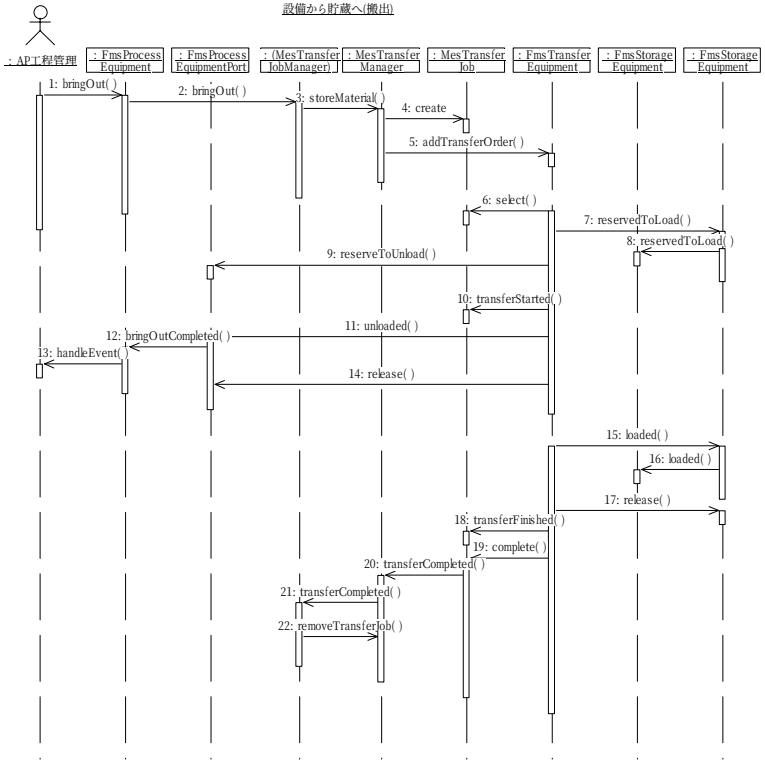
設備から貯蔵へ(搬出)
: AP工程管理
: FmsProcess Equipment
: FmsProcess EquipmentPort
: (MesTransfer JobManager)
: MesTransfer Manager
: MesTransfer Job
: FmsTransfer Equipment
: FmsStorage Equipment
: FmsStorage Equipment
1: bringOut( )
2: bringOut( )
3: storeMaterial( )
4: create
5: addTransferOrder( )
6: select( )
7: reservedToLoad( )
8: reservedToLoad( )
9: reserveToUnload( )
10: transferStarted( )
11: unloaded( )
12: bringOutCompleted( )
13: handleEvent( )
14: release( )
15: loaded( )
16: loaded( )
17: release( )
18: transferFinished( )
19: complete( )
20: transferCompleted( )
21: transferCompleted( )
22: removeTransferJob( )

図 3-28

３．７　工程仕様管理機能グループ

本機能グループは工程経路、工程資源を管理する機能を提供する。MesProcessRoute（工程経路）は MesProcessStep（工程ステップ）のシーケンスであり、この情報をもとに MesLotJob から MesProcessJob が生成される。

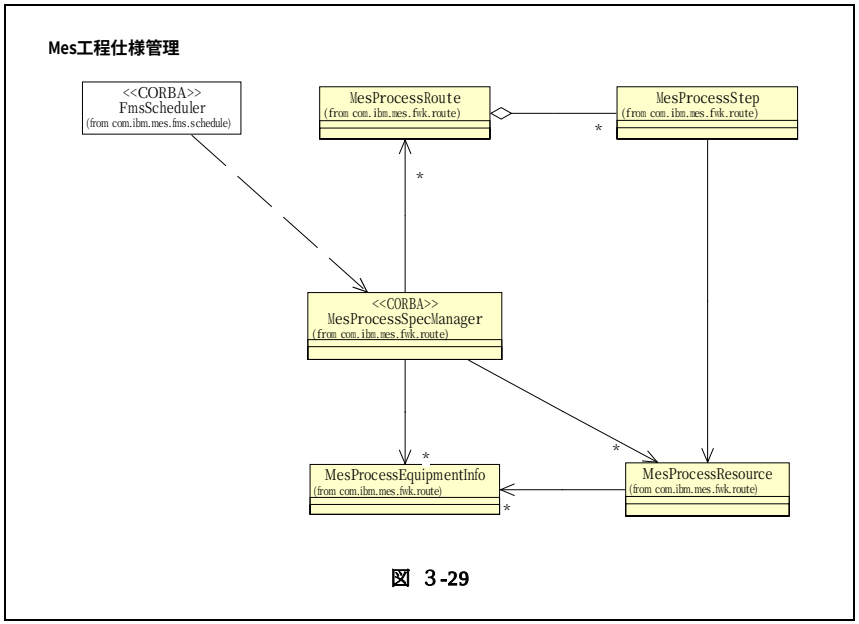

図 3-29

３．７．１　工程経路管理コンポーネント

（１） MesProcessSpecManager

本インタフェースは工程経路、工程ステップ、工程資源および製造設備のオブジェクトに対し操作するメソッドを定義する。

- MesProcessResorce（工程資源）を登録、検索、削除する。
- MesProcessRoute（工程経路）を登録、検索、削除する。
- MesProcessStep（工程ステップ）を工程経路に登録する。
- 工程経路内の工程ステップを検索する。
- 工程経路の参照カウントを１つ増減する。
- MesProcessEquipmentInfo（工程設備情報）を設定、検索、変更、削除する。
- 工程設備情報を工程資源に関連づける。
- 工程資源に登録されている工程設備情報を検索、削除する。

- 工程経路に対する参照が存在するかどうかチェックする。
- 工程ステップに工程資源を割り当てる。
- 工程ステップから工程資源の割り当てを削除する。
- 工程ステップを工程経路から削除する。

（２） MesProcessRoute

本クラスは工程経路を表す。

- 工程ステップを工程経路に登録する。
- 工程経路の工程ステップを削除する
- 工程ステップを取得する
- 参照カウンタを 1 増減する。
- 参照カウンタを設定、取得する。
- 使用中かどうかを返す。

（３） MesProcessStep

本クラスは工程ステップを表す。

- 関連先 MesProcessRoute を設定、取得する。
- 関連工程資源を設定、取得する。
- 工程資源を工程ステップに登録する。
- 工程ステップの工程資源を削除する

（４） MesProcessResource

本クラスはある工程ステップを実行できる製造設備のグルーピング機能を提供する。

- 製造設備資源情報を設定、取得する。
- 製造設備種別を設定、取得する。
- 関連先 MesProcessStep を設定、取得する。
- 製造設備種別を取得する。
- 工程資源の製造設備を削除する
- 製造設備情報を工程資源に登録する。

（５） MesProcessEquipmentInfo

本クラスは製造設備情報を表す。

- 関連先 MesProcessResource を取得、設定する。
- 製造設備型番を設定、取得する。
- 製造設備種別を設定、取得する。

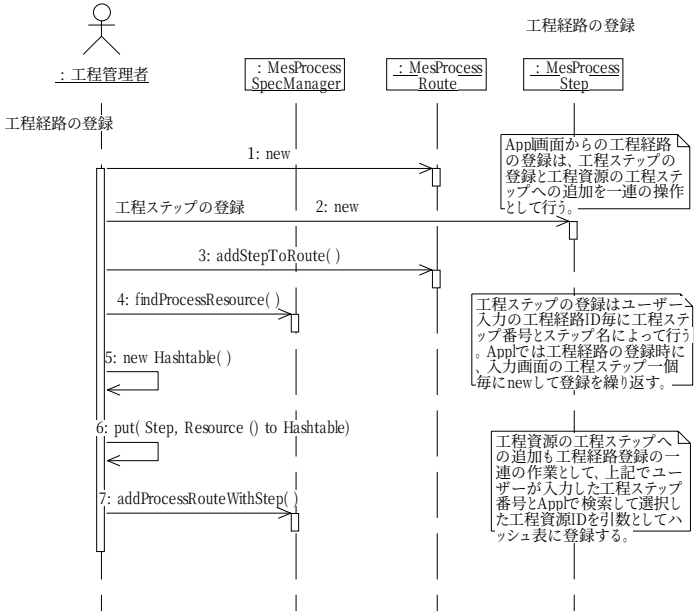
工程経路の登録
：工程管理者
：MesProcess SpecManager
：MesProcess Route
：MesProcess Step
工程経路の登録
1: new
Appl画面からの工程経路の登録は、工程ステップの登録と工程資源の工程ステップへの追加を一連の操作として行う。
工程ステップの登録
2: new
3: addStepToRoute( )
4: findProcessResource( )
工程ステップの登録はユーザー入力の工程経路ID毎に工程ステップ番号とステップ名によって行う。Applでは工程経路の登録時に、入力画面の工程ステップ一個毎にnewして登録を繰り返す。
5: new Hashtable( )
6: put(Step, Resource () to Hashtable)
工程資源の工程ステップへの追加も工程経路登録の一連の作業として、上記でユーザーが入力した工程ステップ番号とApplで検索して選択した工程資源IDを引数としてハッシュ表に登録する。
7: addProcessRouteWithStep( )

図 3-30

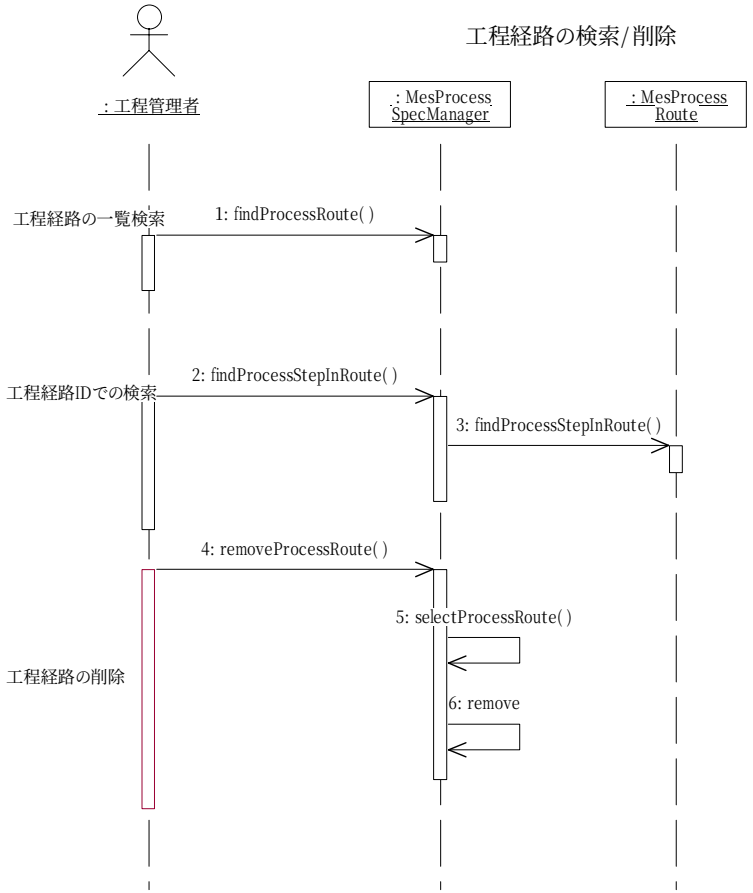
工程経路の検索/削除
：工程管理者
：MesProcess SpecManager
：MesProcess Route
工程経路の一覧検索
1: findProcessRoute( )
2: findProcessStepInRoute( )
工程経路IDでの検索
3: findProcessStepInRoute( )
4: removeProcessRoute( )
5: selectProcessRoute( )
工程経路の削除
6: remove

図 3-31

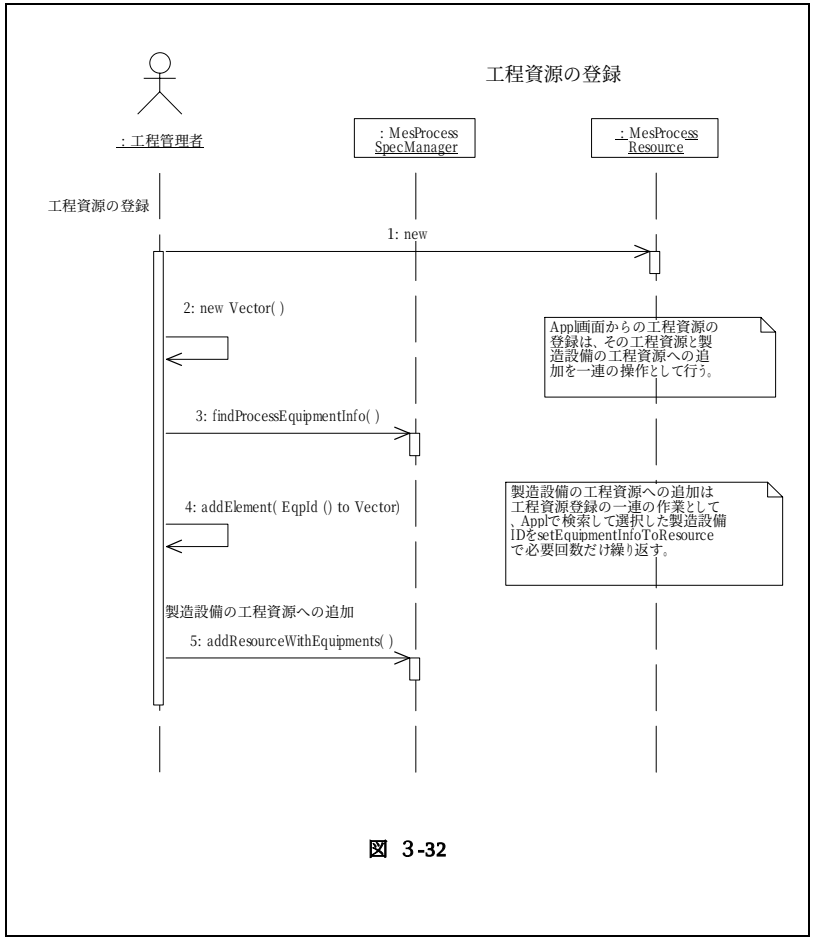
工程資源の登録
：工程管理者
：MesProcess SpecManager
：MesProcess Resource
工程資源の登録
1: new
2: new Vector( )
App画面からの工程資源の登録は、その工程資源と製造設備の工程資源への追加を一連の操作として行う。
3: findProcessEquipmentInfo( )
4: addElement(EqpId () to Vector)
製造設備の工程資源への追加は工程資源登録の一連の作業として、Applで検索して選択した製造設備IDをsetEquipmentInfoToResourceで必要回数だけ繰り返す。
製造設備の工程資源への追加
5: addResourceWithEquipments( )

図 3-32

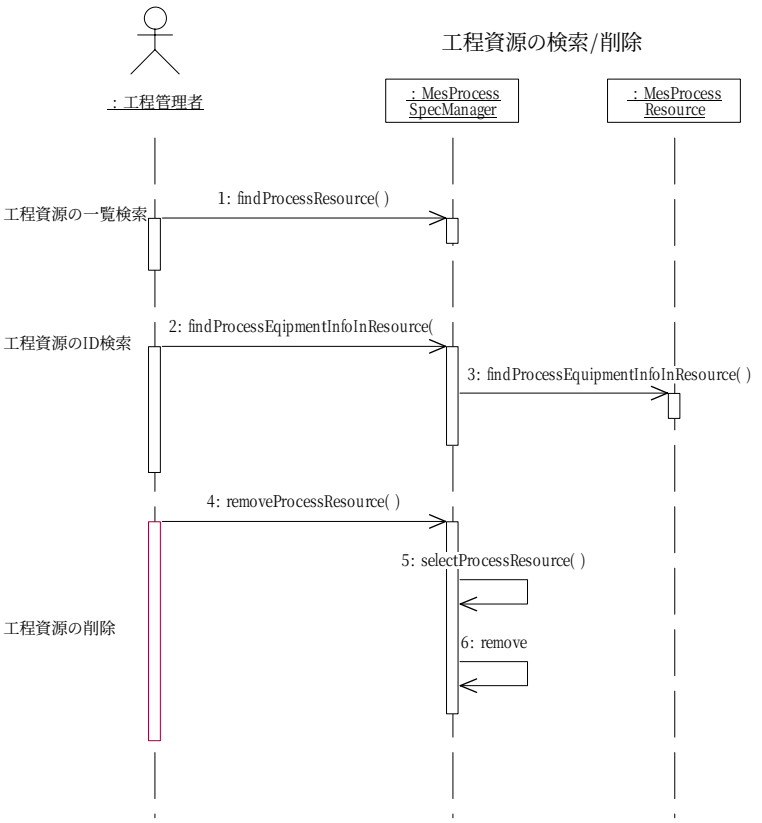
工程資源の検索/削除
：工程管理者
：MesProcess SpecManager
：MesProcess Resource
1: findProcessResource( )
工程資源の一覧検索
2: findProcessEqipmentInfoInResource(
工程資源のID検索
3: findProcessEquipmentInfoInResource( )
4: removeProcessResource( )
5: selectProcessResource( )
工程資源の削除
6: remove

図 3-33

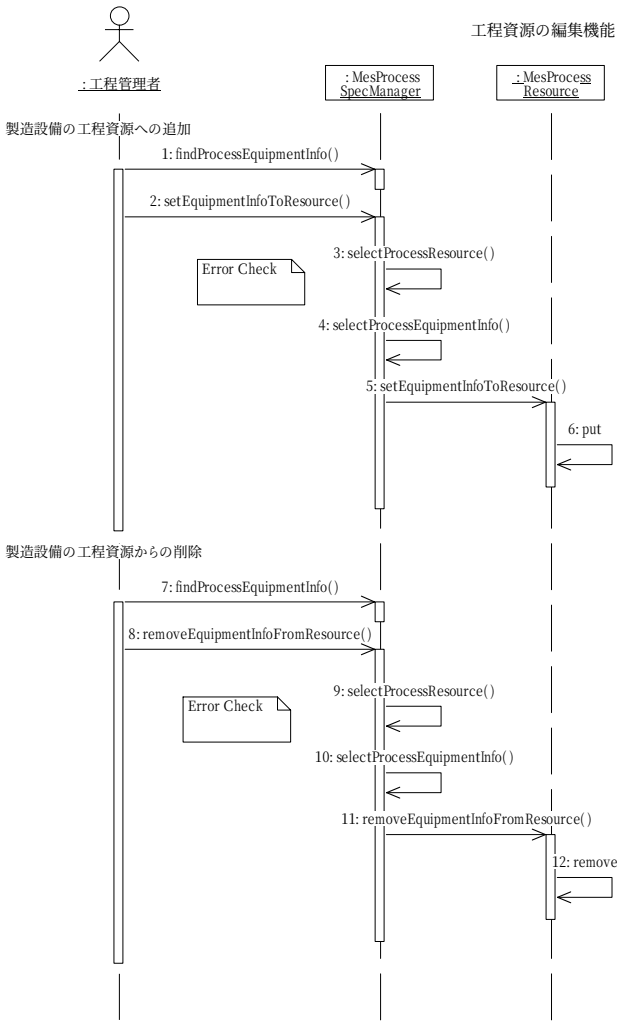
工程資源の編集機能
:工程管理者
: MesProcess SpecManager
: MesProcess Resource
製造設備の工程資源への追加
1: findProcessEquipmentInfo()
2: setEquipmentInfoToResource()
3: selectProcessResource()
Error Check
4: selectProcessEquipmentInfo()
5: setEquipmentInfoToResource()
6: put
製造設備の工程資源からの削除
7: findProcessEquipmentInfo()
8: removeEquipmentInfoFromResource()
9: selectProcessResource()
Error Check
10: selectProcessEquipmentInfo()
11: removeEquipmentInfoFromResource()
12: remove

図 3-34

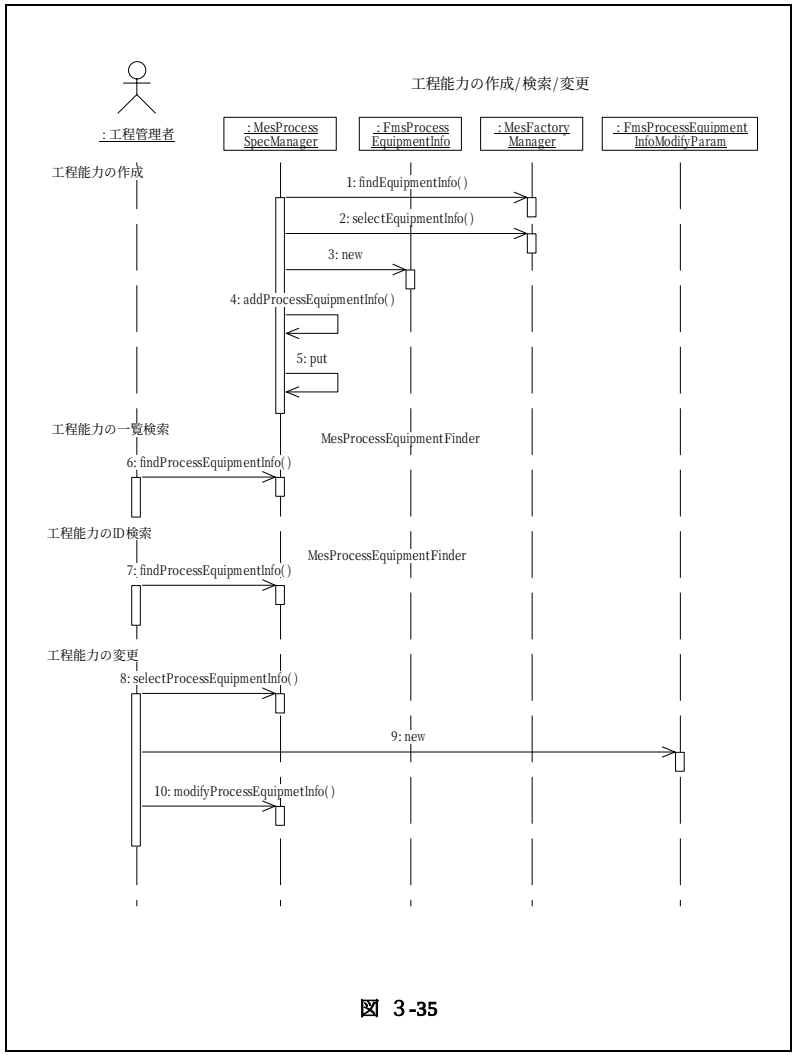
工程能力の作成/検索/変更
:工程管理者
: MesProcess SpecManager
: FmsProcess EquipmentInfo
: MesFactory Manager
: FmsProcessEquipment InfoModifyParam
工程能力の作成
1: findEquipmentInfo()
2: selectEquipmentInfo()
3: new
4: addProcessEquipmentInfo()
5: put
工程能力の一覧検索
MesProcessEquipmentFinder
6: findProcessEquipmentInfo()
工程能力のID検索
MesProcessEquipmentFinder
7: findProcessEquipmentInfo()
工程能力の変更
8: selectProcessEquipmentInfo()
9: new
10: modifyProcessEquipmetInfo()

図 3-35

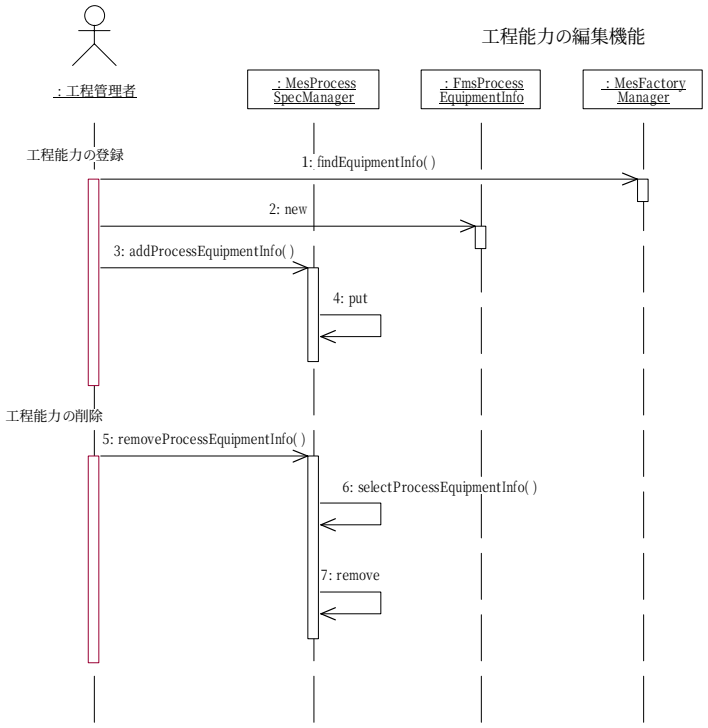

図 3-36

３.８ 設備管理機能グループ

設備管理機能グループは、工場内の製造設備と MES 間の作業指示、作業実績、アラーム等のインタフェースとなる機能を提供する。この機能グループは、個々の製造設備に対応するクラスと、それらの設備をまとめて管理するクラスから構成されている。

ここで扱う設備は、物理的な実体を持たない抽象的な設備である。抽象設備は、下記の三つの性質を持つものとしてモデル化されている。

- 稼働状態の管理単位としての設備（MesEquipment）
- 工程の実行単位としての設備（MesProcessEquipment）
- 制御装置の単位としての設備（MesProcessEquipmentAdaptor）

３.８.１ 設備管理コンポーネント

工程管理機能グループからは MesProcessEquipment に対し、MesWorkOrder が差し立てられ、MesProcessEquipment が MesEquipmentAdapter 経由で制御装置の API を呼び出す。作業が終了すると、制御装置からは MesProcessEquipmentCallback が呼び出され、作業実績（MesWorkResult）が MesWorkOrder と関連づけられる。

MesProcessEquipment が工程管理機能グループとのインタフェース機能を持ち、MesProcessEquipmentAdapter が制御装置とのインタフェース機能を持つため、設備との接続に関しては CORBA の知識を必要としない。

抽象工程設備

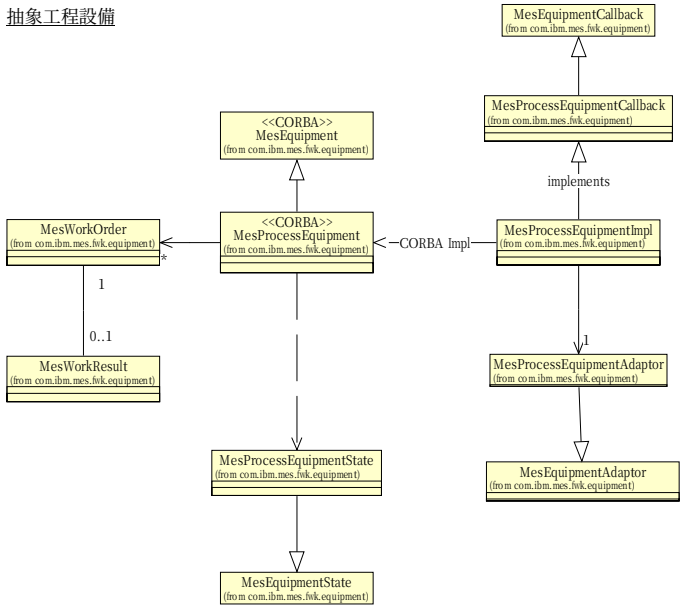

図 3-37

加工設備（工作機械）

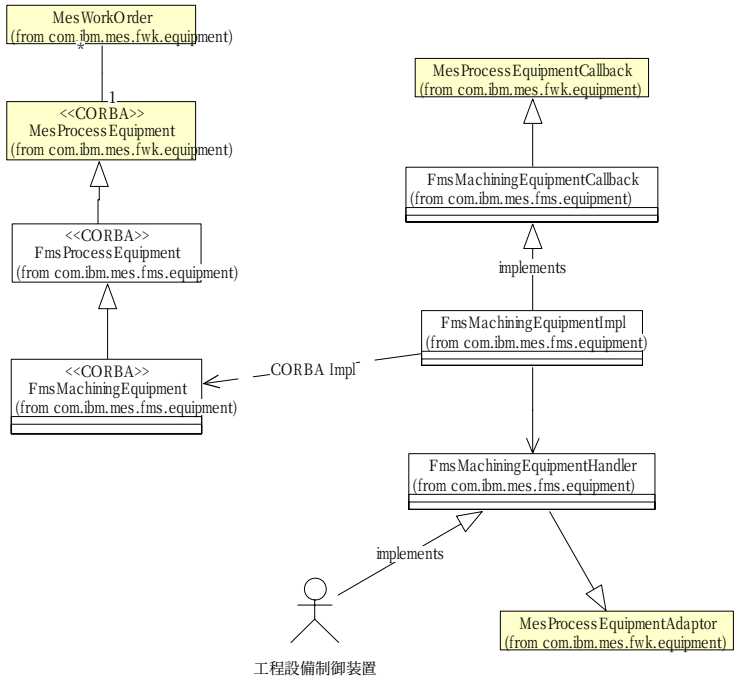

図 3-38

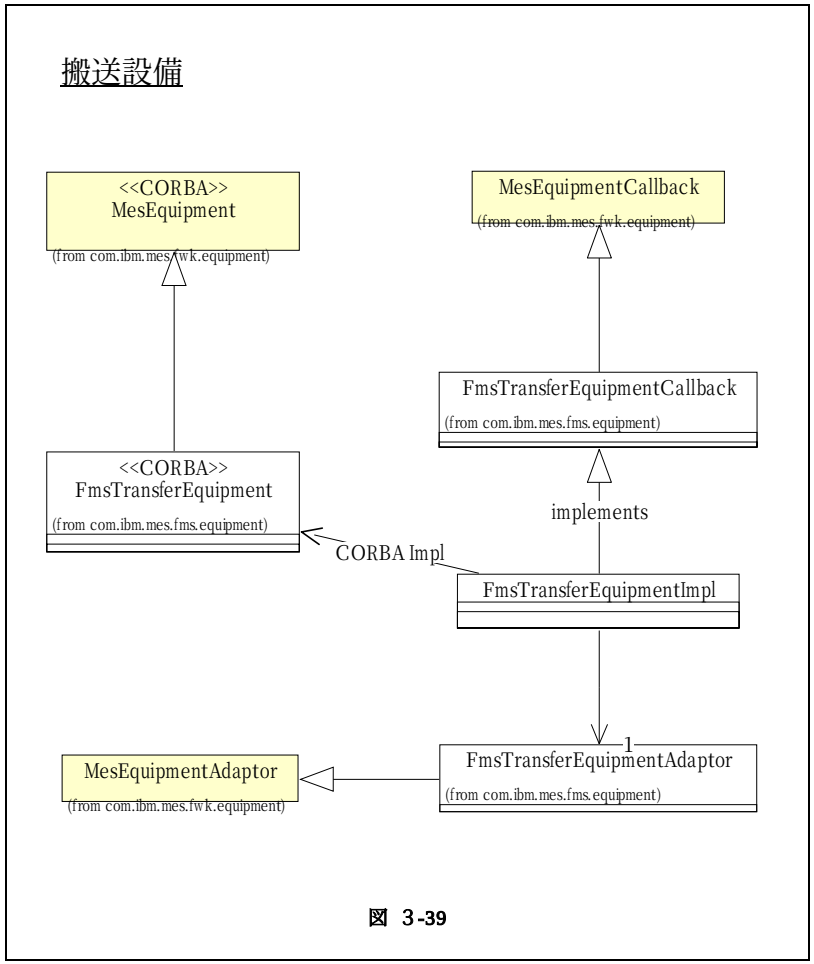

図 3-39

（１） **MesEquipment**

このインタフェースは、抽象設備のための以下の機能を定義する。

- 設備の設置場所、型番、種別を設定する。
- 設備の設置場所、型番、種別を取得する。
- 設備の始動、遮断を行う。
- アラームの発生を通知する。
- 現在発生中のアラームがあるかどうかをチェックする。
- 発生中のアラームを取得する。
- 全てのアラームを解除する。
- 作業者による介入を要求する。
- 作業者による介入要求が発生しているかどうかをチェックする。
- 作業者による介入要求を解除する。
- 設備がオンライン状態かどうかをチェックする。
- 設備をオフラインにし、設備を製造工程内で利用不可とする。
- 設備をオンラインにし、設備を製造工程内で利用可能とする。
- 設備に搬送装置に対するポート(MesPort)を登録する。
- 設備に登録されている全てのポートを検索する。
- 設備からポートを削除する。
- 設備に登録されているポートを取得する。

（２） **MesEquipmentAdapter**

このインタフェースは、抽象設備からの要求に従い、その制御装置を呼び出すための機能を定義する。

- 設備の始動を行う。
- 設備の遮断を行う。
- 発生しているアラームの解除を行う。
- 設備をオフラインにする。
- 設備をオンラインにする。

（３） **MesEquipmentCallback**

このインタフェースは、抽象設備がその制御装置から呼び出されるメソッドを定義する。

- 設備の始動が開始された通知を受け取る。
- 設備の始動が完了した通知を受け取る。
- 設備の始動が取り消された通知を受け取る。
- 設備の遮断が開始された通知を受け取る。
- 設備の遮断が完了した通知を受け取る。
- 設備の遮断が取り消された通知を受け取る。

- アラームが解除された通知を受け取る。
- 全てのアラームが解除された通知を受け取る。
- 設備がオンラインになった通知を受け取る。
- 設備がオフラインになった通知を受け取る。
- 設備の緊急停止状態が解除された通知を受け取る。
- 設備の緊急停止が発生した通知を受け取る。

（４） **MesEquipmentManager**

このインターフェースは、MesEquipment を管理（登録、取得）する。

（５） **MesPort**

このインタフェースは、ポート(搬送装置との接続部分。パレットチェンジャなど)の持つ機能を定義する。

（６） **MesProcessEquipment**

このインタフェースは、工程の割付対象となる抽象工程設備の機能を定義する。

- 工程内作業指示を登録する。
- 全ての工程内作業指示を取得する。
- 次に着手すべき工程内作業指示を取得する。

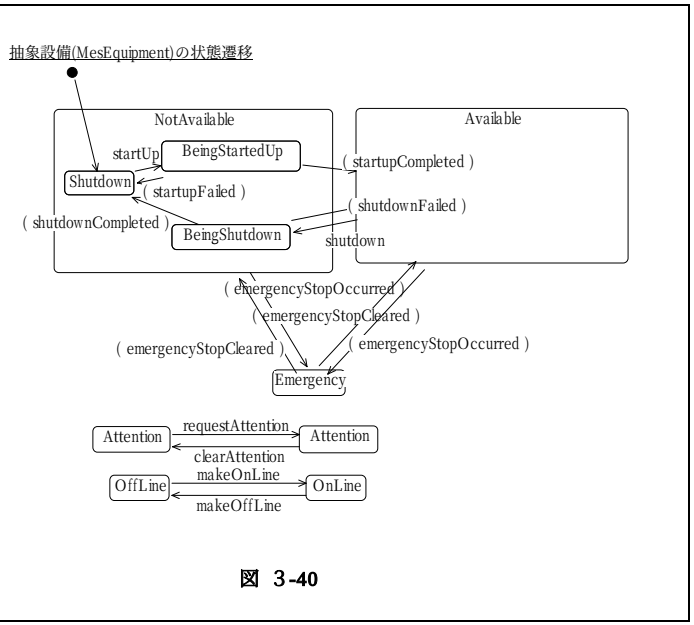

図 3-40

- 状態を指定して工程内作業指示を取得する。

- 工程内作業指示に対応した作業指図を取得する。
- 工程内作業を一時停止状態にする。
- 工程内作業の一時停止状態を解除する。
- 工程内作業指示を取得する。
- 工程内作業を取り消す。
- 工程内作業を中断する。
- 工程内作業の開始を要求する。
- 工程内作業を完了する。
- 工程内作業履歴を追加する。
- 工程設備に現在設定されている運転モード（自動、半自動、手動）を取得する。
- 運転モードを自動運転モードに設定する。
- 運転モードを手動運転モードに設定する。
- 運転モードを半自動運転モードに設定する。
- 工程設備に固有の運転モードを設定、取得する。固有の運転モードとは自動、半自動、手動のいずれかである。
- 中断された工程内作業指示を削除する。
- 完了している工程内作業指示を削除する。
- 指定された工程内作業指示を削除する。
- 工程内作業指示の情報を MesWorkOrderUpdateParam の値に従って更新する。

（７） **MesProcessEquipmentAdapter**

このインタフェースは、抽象工程設備からその制御装置を呼び出すための機能を定義する。

- 設備の運転モードを自動運転モードに変更する。
- 設備の運転モードを手動運転モードに変更する。
- 設備の運転モードを半自動運転モードに変更する。
- 工程内作業を開始する。
- 工程内作業指示が追加された通知を受け取る。

（８） **MesProcessEquipmentCallback**

このインタフェースは、抽象工程設備がその制御装置から受け取るコールバックを定義する。

- 作業開始の許可を要求する。
- 作業の開始通知を受け取る。
- 作業の完了通知を受け取る。
- 作業の停止通知を受け取る。

（９） **MesAlarm**

このクラスは、設備内で発生するアラームを定義する。

（１０） **MesAlarmEvent**

このクラスは、アラームの発生状況の変化を通知（発生通知、解除通知）するイベントを定義する。

（１１） **MesEquipmentAttentionEvent**

このクラスは、アテンション（作業者が介入する状態）の発生状況の変化を通知するイベントを定義する。

（１２） **MesEquipmentLineModeEvent**

このクラスは、ライン・モード（オンライン、オフライン）の変化を通知するイベントを定義する。

（１３） **MesEquipmentManagerServer**

このクラスは、MesEquipment の起動サーバを定義する。

（１４） **MesEquipmentOpMode**

このクラスは、設備の運転モード（自動、半自動、手動）を定義する。運転モードとは、MesProcessEquipment が工程作業を実行する際の実行方法を定義するものである。MesProcessEquipment の運転モードを問い合わせた結果として戻される。

（１５） **MesEquipmentOpModeEvent**

このクラスは、運転モード（MesEquipmentOpMode）の変更を通知するイベントを定義する。

（１６） **MesEquipmentProcessingEvent**

このクラスは、設備内工程処理状態（開始、完了、停止）の変更を通知するイベントを定義する。

（１７） **MesEquipmentServer**

このクラスは、抽象設備の起動サーバを定義する。

（１８） **MesEquipmentState**

このクラスは、設備の稼働状態を定義する。

- 利用可能状態
- 利用不可状態
- 始動中状態
- 遮断中状態
- 遮断状態
- 緊急停止状態
- 不明な状態

（１９） **MesEquipmentStateEvent**

このクラスは、設備の稼働状態（MesEquipmentState）の変更を通知するイベントを定義する。

（２０） **MesProcessEquipmentServer**

このクラスは、抽象工程設備の起動サーバを定義する。

（２１） **MesProcessEquipmentState**

このクラスは、設備の稼働状態（待機中、実行中）を定義する。

（２２） **MesProcessEquipmentStateEvent**

このクラスは、設備の稼働状態の変化を通知するイベントを定義する。

（２３） **MesWorkHistoryElement**

このクラスは、工程内作業と共に記録される履歴項目（タイムスタンプ、ロット ID、工程 ID、工程内作業指示 ID、コメント）を定義する。

（２４） **MesWorkInstruction**

このクラスは、工程設備に渡される工程内作業指図を定義する。

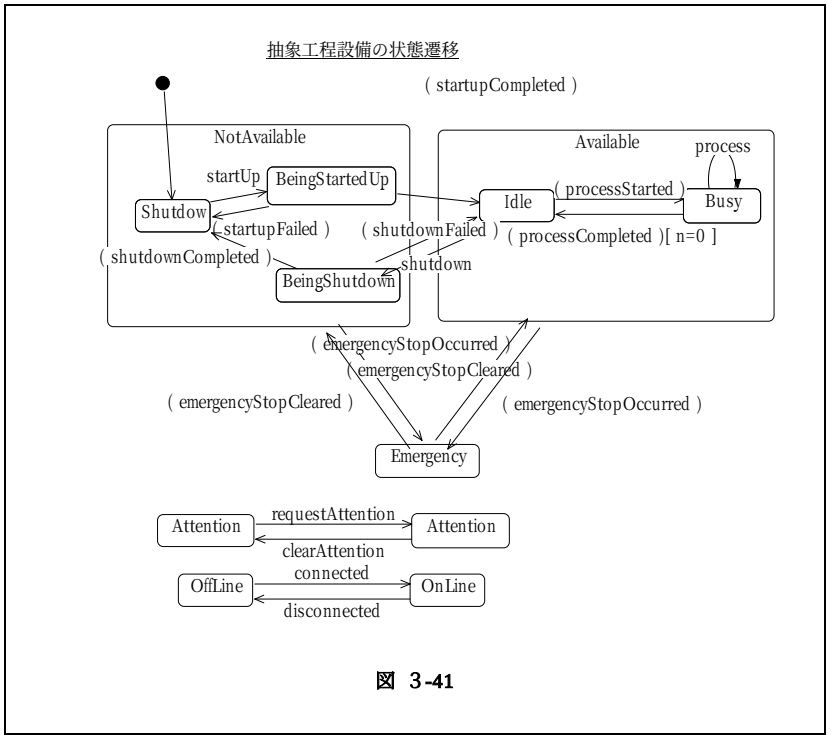

図 3-41

（２５） **MesWorkOrder**

このクラスは、工程設備の受け取る工程内作業指示を定義する。

- 工程内作業指示の ID を設定、取得する。
- 工程内作業指示の工程 ID を設定、取得する。
- 工程内作業指示を含むロットの ID を設定、取得する。
- 作業予定数量を設定、取得する。
- 作業開始予定日時を設定、取得する。
- 作業完了予定日時を設定、取得する。
- 工程内作業指示の優先順位を設定、取得する。
- MesLotJob を設定、取得する。
- 工程内作業指示が割り付けられた設備の ID を設定する。
- 差し立て内での工程内作業指示の順位を取得する。
- 工程内作業指示の状態（MesWorkOrderState）を取得する。
- 段取り替えの必要性を設定、取得する。
- 工程内作業を着手不可にする。
- 工程内作業を着手可能にする。
- 工程処理の開始、完了、停止を通知する。
- 工程内作業の開始許可を要求する。
- 工程内作業を中断する。
- 工程内作業を取り消す。
- 工程内作業を完了する。
- 工程内作業指示を一時停止状態とする。
- 工程内作業指示の一時停止状態を取得する。
- 工程内作業指示の一時停止状態を解除する。
- 工程内作業指示のグローバルな一時停止状態を取得する。
- 工程内作業指示のグローバル（代替設備全部）な一時停止を設定・解除する。
- 工程内作業指示のローカル（特定の設備）な一時停止を設定・解除する。
- 差し立て内での工程内作業指示の順位を設定する。
- 作業結果を登録する。
- 工程内作業履歴を登録する。

（２６） **MesWorkOrderAddParam**

このクラスは、工程設備に対して工程内作業指示を割り付けるためのパラメータ（工程内作業を要求したオブジェクト、工程内作業指図、工程内作業指示）を定義する。

（２７） **MesWorkOrderEvent**

このクラスは、設備の持つ工程内作業指示の状態(MesWorkOrderState)の変更を通知するイベントを定義する。

（２８） **MesWorkOrderState**

このクラスは、工程内作業指示の状態を定義する。

- 作業中断
- 作業完了
- 作業中
- 着手不可
- 作業中でかつ工程処理が完了している状態
- 作業中でかつ工程処理が開始不可能な状態
- 作業中でかつ工程処理が開始可能な状態
- 作業中でかつ工程処理が開始されている状態
- 作業中でかつ工程処理が停止中の状態
- 着手可能
- 前工程が完了して着手可能であるが、グローバルまたはローカルにサスペンドしている状態

（２９） **MesWorkOrderTransferState**

このクラスは、工程内作業指示で指示されたロットの搬送状態を定義する。

- 搬入指示により搬入を実行中である状態
- 搬出中の状態
- 搬入済みの状態
- 搬出済みの状態
- 搬入未着手の状態

（３０） **MesWorkOrderUpdateParam**

このクラスは、工程設備に登録された工程内作業指示の情報を更新するためのパラメータを定義する。

- 作業の完了予定日時
- 作業の着手予定日時
- 優先順位
- グローバル・サスペンド・フラグ
- 差し立て内での順位
- 工程内作業指図
- 工程内作業指示 ID

（３１） **MesWorkResult**

このクラスは、工程設備の返す工程内作業結果を定義する。工程内作業結果は、工程内作業指示に対して、その作業結果の更新を要求することができる。

- 工程内作業を行った設備の ID を設定、取得する。
- 工程内作業で発生した不良品数を設定、取得する。
- 工程内作業の完了日時を設定、取得する。
- 工程内作業で処理が完了した数量を設定、取得する。

- 工程内作業を担当した担当者の ID を設定、取得する。
- 工程内作業の開始日時を設定、取得する。

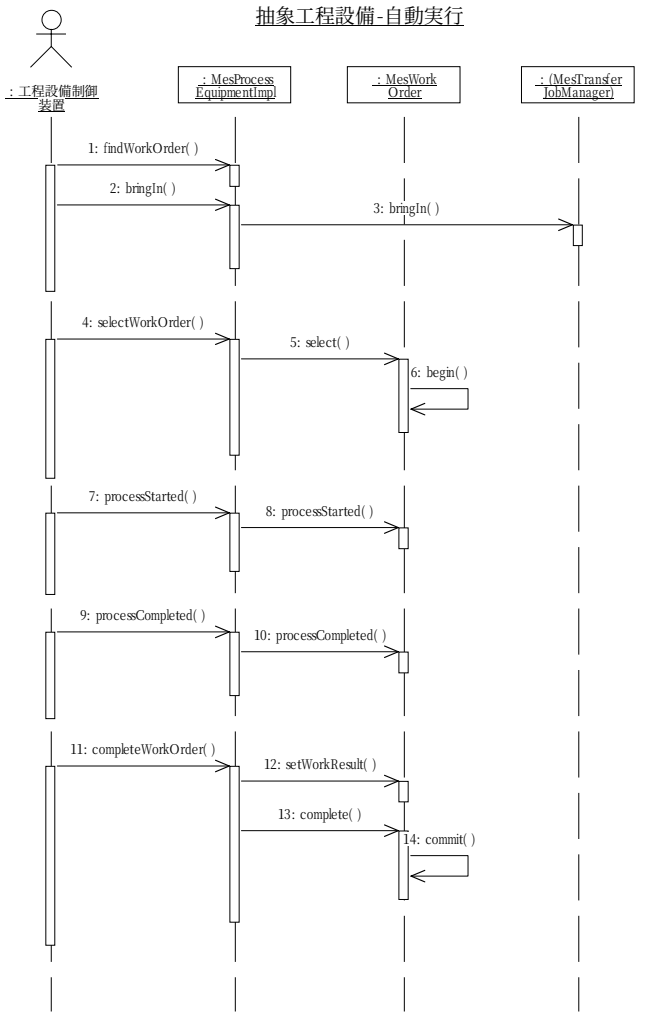

図 3-42

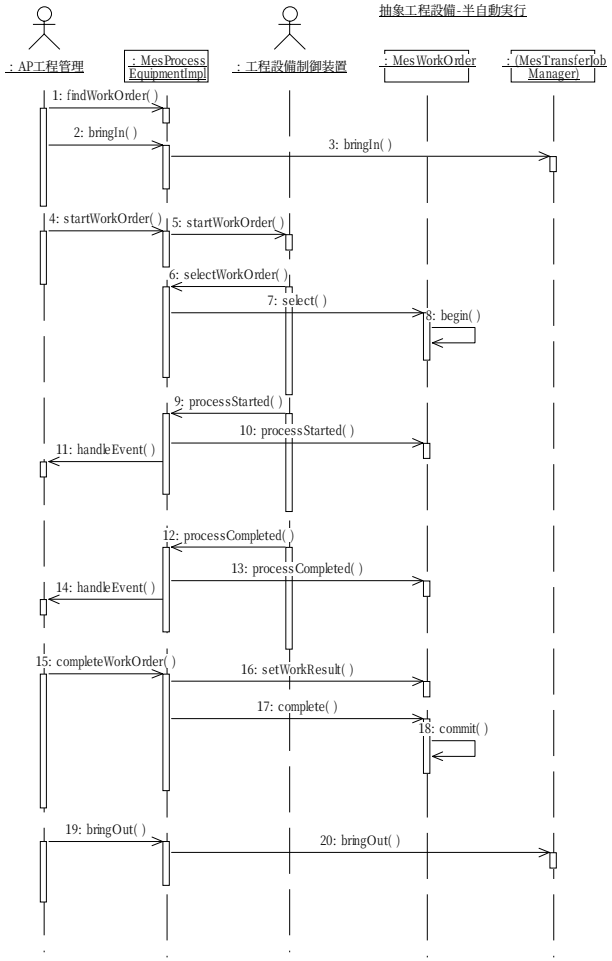
抽象工程設備-半自動実行
：AP工程管理
：MesProcess EquipmentImpl
：工程設備制御装置
：MesWorkOrder
：(MesTransferJob Manager)
1: findWorkOrder( )
2: bringIn( )
3: bringIn( )
4: startWorkOrder( )
5: startWorkOrder( )
6: selectWorkOrder( )
7: select( )
8: begin( )
9: processStarted( )
10: processStarted( )
11: handleEvent( )
12: processCompleted( )
13: processCompleted( )
14: handleEvent( )
15: completeWorkOrder( )
16: setWorkResult( )
17: complete( )
18: commit( )
19: bringOut( )
20: bringOut( )

図 3-43

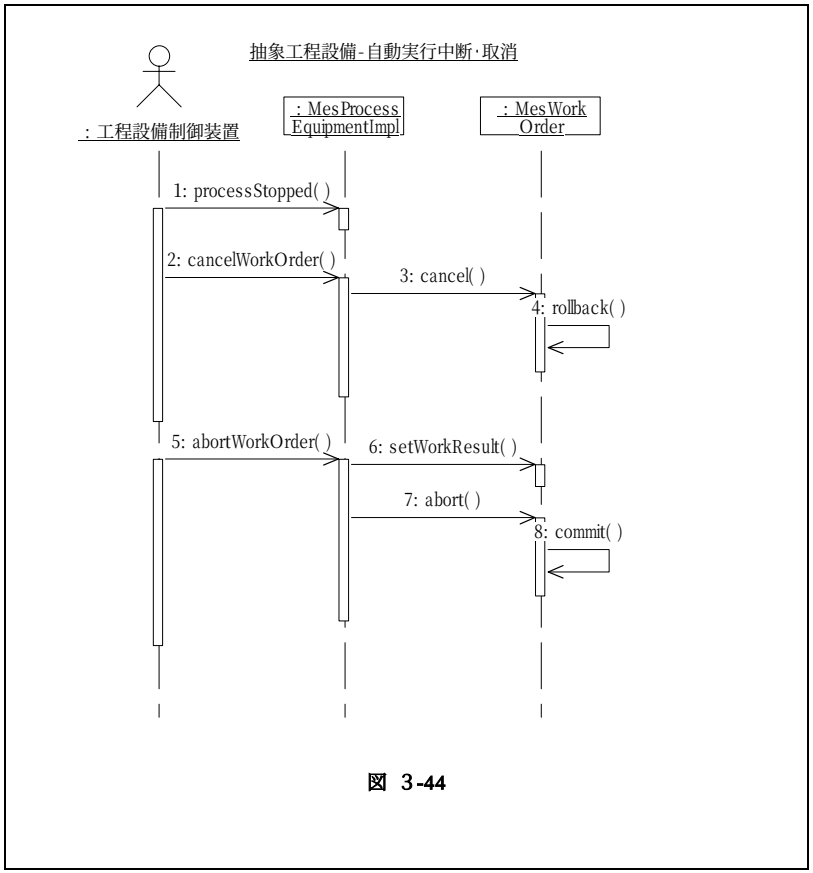
抽象工程設備-自動実行中断･取消
：工程設備制御装置
：MesProcess EquipmentImpl
：MesWork Order
1: processStopped( )
2: cancelWorkOrder( )
3: cancel( )
4: rollback( )
5: abortWorkOrder( )
6: setWorkResult( )
7: abort( )
8: commit( )

図 3-44

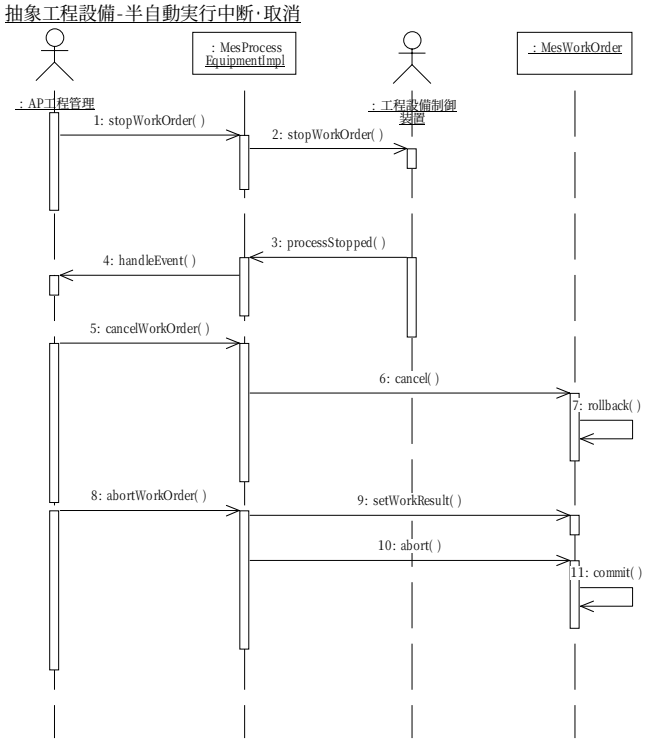

図 3-45

３.９ スケジュール管理機能グループ

３.９.１ スケジューラインタフェースコンポーネント

（１） MesScheduler

このインタフェースはスケジューラへのインタフェース機能を定義する。既存のスケジューラと接続するため、抽象的なメソッドしか定義されていない。

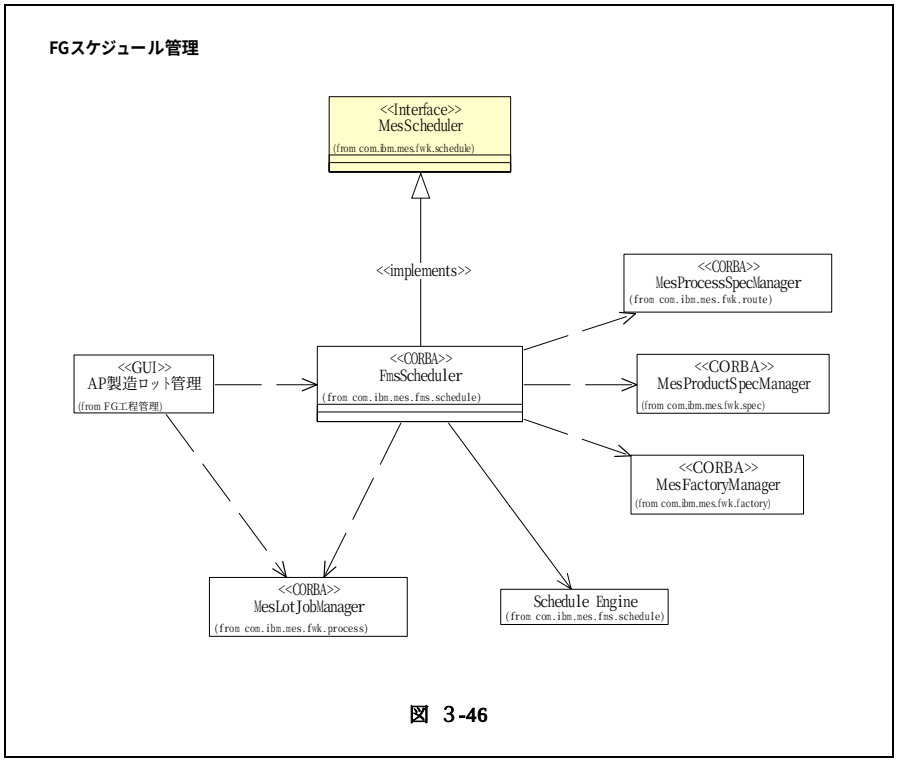

図 3-46

３．１０　共通機能グループ

３．１０．１　イベント通知管理コンポーネント

CORBA の Event Service を使用している。イベントモデルについては、CORBA の仕様書を参照のこと。

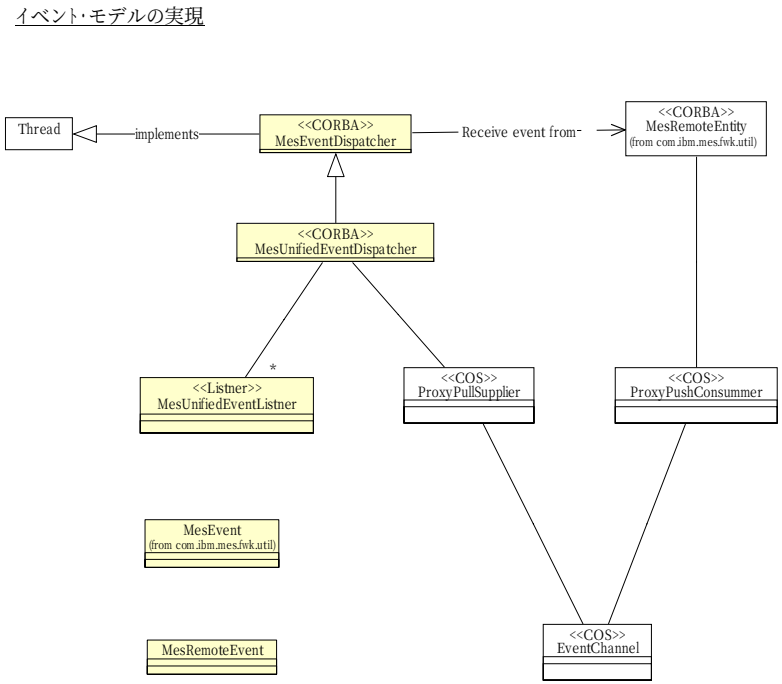

図 3-47

（１） **MesEvent**

このクラスは、コンポーネント内に発生するイベントを定義する。

（２） **MesEventDispatcher**

このクラスは、CORBA サーバからのイベントを受け取り、ローカル空間内の Java オブジェクトに配信するディスパッチャを定義する。

（３） **MesRemoteEvent**

このクラスは、CORBA の Event Channel を通してイベントを通知するためのパケットを

定義する。このクラスは、IDL のマッピング上は struct に変換される。 Event Channel が struct マッピングしか受け付けないため、MesEvent からこのクラスに変換して送る必要がある。

（４） **MesUnifiedEventDipatchcer**

このクラスは、共通のインターフェースを通して全てのイベントの配信を行う

- イベントのリスナを登録する。
- イベント（MesRemoteEvent）の配信を行う。
- イベントのリスナの登録を削除する。

（５） **MesUnifiedEventListener**

このインタフェースは、CORBA サーバからのイベントを受け取るリスナを定義する。

３．１０．２　リモートエンティティコンポーネント

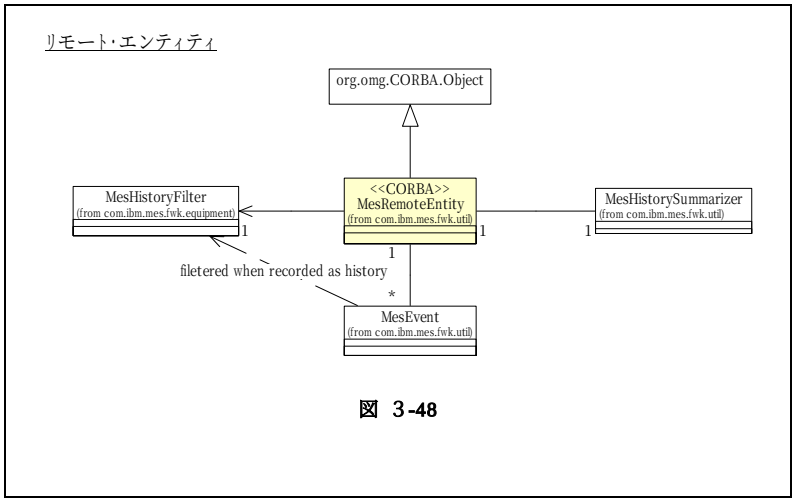

図 3-48

（１） **MesRemoteEntity**

このインターフェースは、フレームワークを構成するリモート・オブジェクトの持つ共通機能を定義する。 このインターフェースから取得された各種の属性はサーバ内部の属性のコピーであり、取得された属性に対する変更操作はサーバ内部の属性には反映されない。

（２） **MesHistoryElement**

このクラスは、コンポーネント内で記録される履歴項目を定義する。

（3） **MesHistoryFilter**

このクラスは、履歴を記録するかどうかのチェックを行うためのフィルタを定義する。

（4） **MesHistorySummarizer**

このクラスは、MesRemoteEntity のオブジェクトの履歴からサマリを作成するた

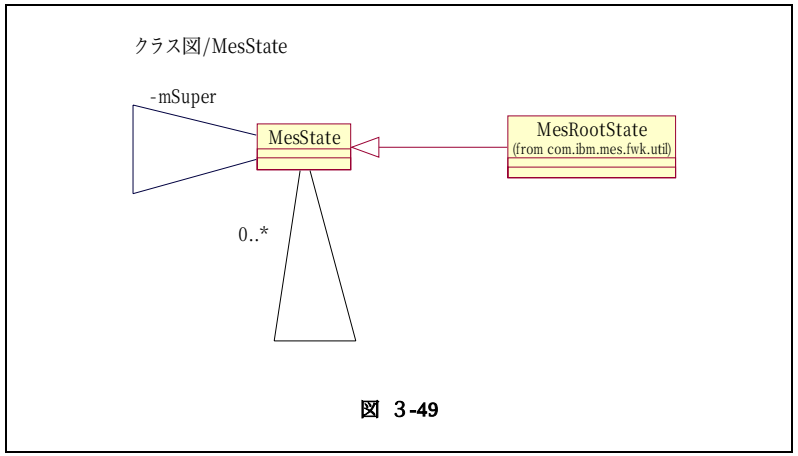

図 3-49

めの抽象クラスである。

３．１０．３ 状態管理コンポーネント

（1） **MesState**

階層化された状態集合を木構造によって保持する。以下の機能を提供する。

- 状態 ID を取得する。
- 状態レベルを取得する。
- 状態名を設定、取得する。
- 上位状態を設定、取得する。
- 下位状態を設定、取得する。
- 与えられた状態がこの状態と同じかどうか判定する。
- 下位状態が登録されているかどうかを取得する。
- 指定された下位状態をその全ての下位状態とともに削除する。
- 与えられた状態 ID がこの状態あるいは下位状態として登録されている場合 その状態を返す。
- 与えられた状態 ID がこの状態あるいは上位状態として登録されている場合 その状態を返す。

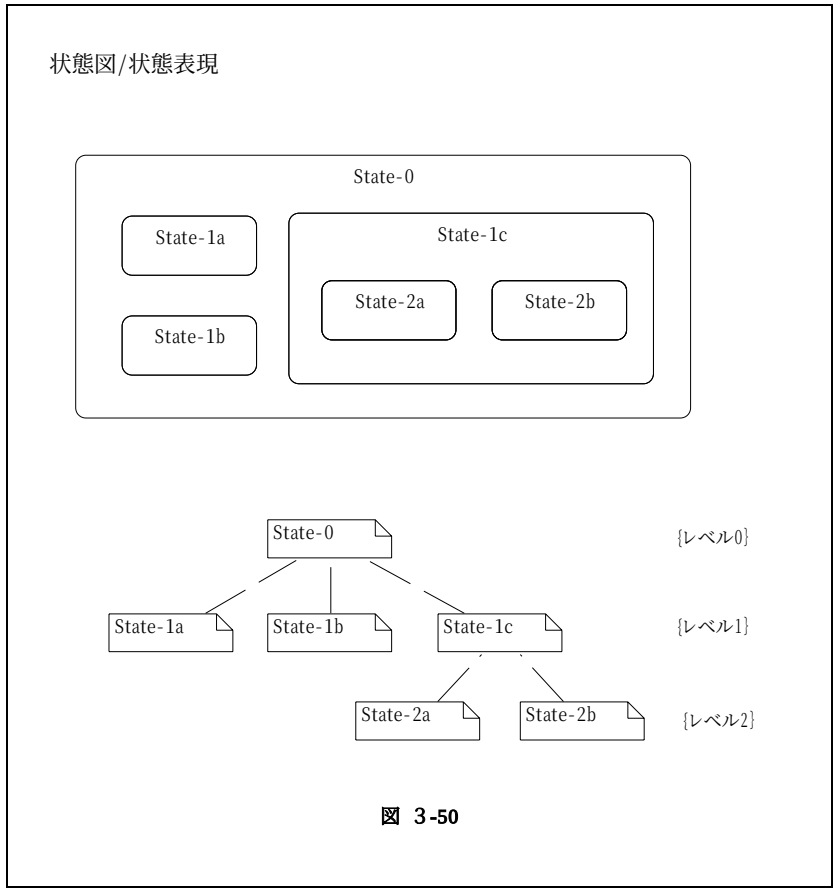

図 3-50

（2） **MesRootState**

最上位状態。 木構造によって階層化された全状態集合のルート(根)となる。このクラスは 個別の状態に相当する MesState から派生したものであるが、特にその時点での 状態を保持することができる。さらに、この状態集合に属する状態は addState (最上位状態の直下に属する場合)、もしくは addStateTo(既に登録されている 状態の下位状態を登録する場合)によって登録される。これらのメソッドによって 登録された状態についてはこの状態集合内での一意性が保証される。 様々な状態を取り得る実体クラス(例えば、製造指示、製造ロット)には、該当する状態集合が関連付けられる。そのような実体クラスには、public な

メソッドとして 例えば isInState()、getCurrentStateId()、getRootState()を提供する。isInState はそのオブジェクトが与えられた状態もしくはその上位状態にあるかどうかを返答 する。getCurrentStateId はその時点でのオブジェクトの状態 ID を返す。さらに、getRootState は状態集合の全要素が関連付けられた木構造の根となる MesRootState もしくはその派生クラス(例えば MesLotJobState)を返す。getRootState によって 得られた木構造を辿ることによって状態集合に伴われる全ての状態 ID と対応する 状態名(例えば画面表示に使用)を参照することができる。

平成１２年度　ＦＡオープン推進協議会
生産システム情報統合専門委員会　成果報告書

発　行　　平成１３年４月

発行者　　財団法人　製造科学技術センター
〒105－0002　東京都港区愛宕１－２－２
電　話　（０３）５４７２－２５６１