APSOM 000001-9:2018(CD)

APSOM/MESX/WG 1

Secretariat: Kiminobu Kodama

生産管理のための階層間システム連携

– 階層間連携言語（iHCl 2.0）によるメッセージとプロトコル

CD

Contents

Foreword

本規格書，APSOM/MESX-JP 000001 は，MESX-JP により準備され，APSOM 標準化委員会によって採択された。

本規格書の附属書 A から H は，情報提供のみを目的としている。

Introduction

一般に，巨大なシステムを一つの一枚岩（monolithic）なソフトウェアとして構築することはシステム理論から見て得策ではない。脆弱で，環境の変化への対応にコストがかかるためである。システム工学では一般に，巨大なシステムをいくつかのサブシステムに分割してソフトウェアを作成し，サブシステムが適切に連携するように設計する。これにより，変更容易性，拡張性，耐故障性が高まると期待される。実際，IEC 62264-1:2013 では，経営管理レベルから生産現場の制御レベルを統合的に扱うシステムを，レベル 4 からレベル 0 までの機能の階層に分割し，各階層で実施される活動と対話を定義している。

サブシステムの分割が適切であるためには，それが持つべき機能に対して，処理とデータが最少かつ完備であるように設計し，他のサブシステムとの間の相互参照性または依存性をできるだけ弱めなければならない。処理とデータに過不足がないことを凝集性が高い（high cohesion）といい，相互依存性が弱いことを結合度が低い（low coupling）という。サブシステム分割が適切であれば，サブシステムごとのポータビリティが高まり，相互運用性も高まる。

本規格では，サブシステム間をメッセージによって疎に連携するための言語を規定する。メッセージによる連携は結合度がきわめて低いとされる[12]。一方で，機能の配置とサブシステムの分割はニワトリと卵の関係にあり，サーバや通信設備の性能なども含めて総合的に考慮しなければならず，適切なサブシステムの分割点は自明ではない。

このようにして適切に分割されたサブシステムに対し，ISO 16100-5 で述べられるケイパビリティプロファイルを記述することで，サブシステムごとにケイパビリティを識別し，対話プロトコルを定義することで，サブシステムの代替可能性を検討できる。

Annex A では，本文で規定する基準に基づいて分割したサブシステムと，そのサブシステムから構成される全体システムの構造の例を示す。

Annex B では，第 5 章で述べる二つのサブシステム間で交わされる対話構造のベースとなった Customer-Performer モデルの原理を説明する。

Annex C では，注文メッセージに記載する注文内容の属性について詳細を示す。

Annex D では，第 7 章で定めた階層間連携メッセージの例の一部を具体的に示す。

Annex E では，オリジナルの対話構造を，日本の商習慣を扱えるように拡張したモデルを提示し，その拡張に基づく拡張階層間業務メッセージを定義する。また，Annex F では，日本の商習慣を反

映した曖昧な仕様のための，未定の仕様値，代替の仕様値の表現を導入する。これにより，仕様の遅延束縛（late binding）を実現できる。

Annex G では，サブシステム間の通信チャネルの実装方法の例を解説する。これは，実際にメッセージをやりとりする上で必要な仕組みである。ここでは，ISO 16100 のケイパビリティプロファイルを利用する。

Annex H では，メッセージを組み立てる上で必須のメッセージ番号の発番についての注意と，一つのメッセージに複数の注文をパッキングすることについて解説する。

生産管理のための階層間システム連携 – 階層間連携言語（iHCl 2.0）によるメッセージとプロトコル

1 Scope

この文書は，ソフトウェアエージェントが互いに協調してシステム全体の機能を創発するためのメッセージ言語とプロトコルを記述する。ISO 16100-3：2005 で定義される MSU ケイパビリティプロファイルを提示することによって，各エージェントの製造活動に関するケイパビリティと認識可能なメッセージ形式について相互認識する。製造活動を要求するエージェントを顧客（Customer）と呼び，製造活動を提供するエージェントを実行者（Performer）と呼ぶ。Customer は，製造活動に対する要求メッセージをメッセージ言語によって記述し，Performer はその製造活動の実行結果の報告メッセージをメッセージ言語によって記述する。

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IEC 62264-1:2013, Enterprise-control system integration — Part 1: Models and terminology

IEC 62264-3:2007, Enterprise-control system integration — Part 3: Activity models of manufacturing operations management

ISO 16100-3:2005, Manufacturing software capability profiling for interoperability —Part 3: Interface services, protocols and capability templates

ISO 16100-5:2009, Manufacturing software capability profiling for interoperability — Part 5: Methodology for profile matching using multiple capability class structures

ISO/IEC 14977:1996(E), Syntactic meta language - Extended BNF

ISO 8601:2004, Data elements and interchange formats – Information interchange – Representation of dates and times

ISO 4217, Currency codes

ISO/IEC 19505-2:2012, Information technology -- Object Management Group Unified Modeling Language (OMG UML) -- Part 2: Superstructure

JIS Z 8203:2000, 国際単位系 (SI) 及びその使い方

JIS B 3900-5:2014, 産業オートメーションシステムと統合 – 製造用ソフトウェア相互運用のためのケイパビリティプロファイリング – 第 5 部：複数のケイパビリティ構造を用いたプロファイル照合の手法

3 Terms and definitions

3.1
サブシステム（Subsystem）
全体システムにおける実装されたシステム要素で，独立したソフトウェアシステムまたはソフトウェアユニット。MSU は製造目的に特化したソフトウェアユニットをいう。「ドメイン」はデータの構造を含む論理的なシステム要素を指す。

3.2
対話構造（Interactional Structure）
サブシステム間連携の文脈において行われる対話の状態遷移の全体

3.3
注文（Order）
発注者が，一連の対話構造を通して獲得したい業務上の状態。注文には，「発注者が，受注者に，何を，いつ（までに），どれくらい，（いくらで），どうしてほしいか」が含まれる。

3.4
発注者（Customer）サブシステム（C-Subsystem）
対話構造において，注文を発行して，その処理結果を受け取るサブシステム

3.5
注文語（Order Verb）
発注者が受注者に対して「どうしてほしいか」を表現する語（動詞）。「注文」に含める。日本語では終止形で，英語では命令形にする。たとえば，「作る（Make）」「買う（Buy）」など。

3.6
受注者（Performer）サブシステム（P-Subsystem）
対話構造において，注文を受けて，処理を実行し，その結果を返すサブシステム

3.7
制御メッセージ（Control Message）
業務メッセージの交換を制御するためのメッセージ

3.8
業務メッセージ（Performative Message）
業務上の意思（例：注文，処理結果）を伝えるメッセージ

3.9
制御動詞（Control Verb）
制御メッセージのタイプを表す動詞。たとえば，「取りに来い（Notify）」「応答を乞う（RSVP）」など。

3.10
遂行動詞（Performative）
対話構造において，状態を遷移させる業務メッセージのタイプを表す動詞

3.11
ケイパビリティプロファイル（Capability Profile）
ISO 16100 シリーズで定めるサブシステムが実行できる機能のプロファイルデータ。本規約では，サブシステムが受け付け可能な業務単体または業務のリストで表現される。

3.12
生産システム（Production System）
生産を目的とする実際に稼働するシステムの実体。受注から製造指示，調達，実績記録，請求まで，またはその一部。

3.13
商品
発注者が要求し，受注者が提供するものまたはサービス。サービスには製造・加工サービス，保守サービス，輸送サービスが含まれる。

4. Symbols

4.1 Abbreviation

MSU Manufacturing Software Unit

UML Unified Modelling Language

ESB Enterprise Service Bus

REST Representational State Transfer

5. 概要

5.1 本規約の位置づけ

RAMI 参照モデル[14]における本規格の位置づけは，Figure 1 にオレンジ色で示したとおり，Life Cycle & Value Stream 軸においては Production，Layers 軸においては Business，Hierarchy Levels 軸においては Enterprise から Control Devices にまたがっている。すなわち，製造ビジネスにおける受注から設備コントロールに至るまでの，段階的な作業計画立案から，その指示と実績に基づく再計画に関わっている。

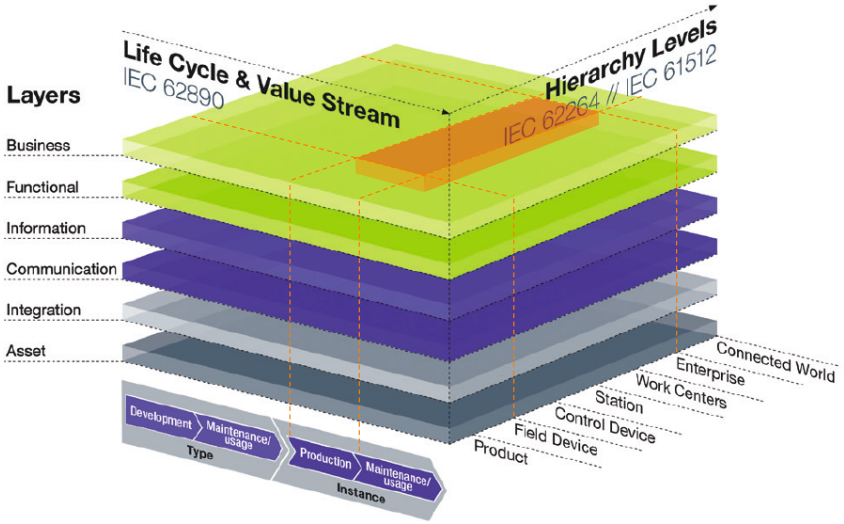

Figure 1　RAMI 4.0 参照モデル[14]における本規格の位置づけ

5.2 サブシステム間の対話構造

生産システム内のサブシステム間の連携方式を，発注者-受注者（Customer-Performer）間の対話に還元してとらえる。

ある業務における発注者（C）と受注者（P）の二者間で起こりうる対話構造[7]は，Figure 2 の状態遷移図で表される。

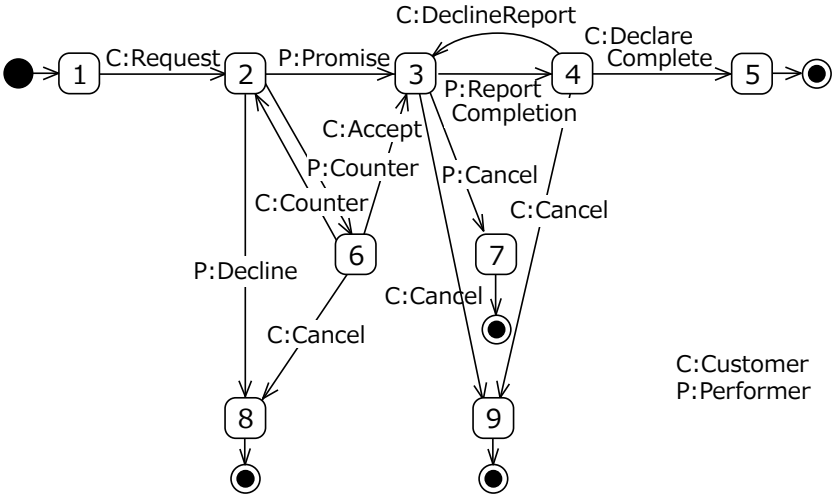

Figure 2　発注者-受注者間で起こりうる対話構造[7]

Figure 2 は，C を発注者，P を受注者とする二者間で起こりうる対話構造を表す UML のプロトコル状態機械図である。この図は，状態を遷移させる遂行動詞を説明している。遷移の矢印上にある Request, Promise などを遂行動詞と呼ぶ。たとえば，状態 1 から状態 2 への遷移の矢印上に C: Request と表記されている。これは，状態 1 において発注者が遂行動詞 Request を受注者に送ると，状態は 2 に遷移するという意味である。その Request に対して受注者が約束（P: Promise）したり，拒絶（P: Decline）したり，対案を提示（P: Counter）したりすることがある。状態 2 から状態 3 に至るすべてのルートは「約束」のフェーズに対応する。以下，同様に読む。

状態 3 から状態 4 に遷移するとき，受注者が Promise した条件，たとえば製造数が不足した，完成時期が遅れたなどを満足できない場合も起こりうる。この場合も，受注者は ReportCompletion で報告し，発注者はその対応を DeclineReport するか DeclareComplete することで返答する。

生産システムをいくつかの階層的なサブシステムに分割して構築するとき，注文を発するサブシステム（C-Subsystem）と注文を受けるサブシステム（P-Subsystem）間においてこの対話構造が成立する。サブシステム分割の例を Annex A に示す。また，対話構造について Annex B で説明し，この日本版拡張を Annex E に示す。

なお，原則として Figure 2 で記述した以外の状態は存在せず，そのような遷移を引き起こす遂行動詞もない。

6 階層間連携言語

階層間連携言語（inter-Hierarchy Collaboration Language, iHCl）を規定する。

6.1 構文の表記

構文の表記には**拡張 BNF** を用いる。

6.2 特殊文字

iHCl で用いる特殊文字を次のように定める。

```
" = '"'.
: = ":".
{ = "{".
} = "}".
[ = "[".
] = "]".
comma = ",".
```

これらは連結記号とともに用いられる。

Note: 連結記号は拡張 BNF で定義されている記号，すなわち特殊記号ではないカンマ（","）である。

拡張 BNF の基本要素（syntactic primary）は連結記号を伴わないことに注意。たとえば，次のような表記において，両端の{}は連結記号を伴うので特殊文字，内側の{}は連結記号を伴わないので基本要素の反復記号と解釈される。

```
{,o-attr,{comma,o-attr},}.
```


6.3 基本型の表記

頻出する属性（基本型，Foundation Type）について，次のように表現の形式を定める。

6.3.1 基本語の表現

iHCl で用いる基本的な語を次のように定義する。

```
value = 即値.
即値 = 数値 | 文字列.
数値 = [符号],{数字},["."{数字}] | 整数. (* 符号を省略したときは+，小数点付きも可 *)
整数 = [符号],{数字}.
符号 = "+"|"-".
文字列 = {? UTF-8 で定義されて，人が読める文字 ?}.
数字 = "0"|"1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9".
```

6.3.2 数値の表現

数と量を区別する。

量は，たとえば，長さや重さのような「もの」の計測値であり，多くは単位をもつ。連続値の場合もあるし，色名やランクなどの離散値の場合もある。量は，たとえば，高さ，幅，奥行きのように多次元で表現されることもあるが，ここでは 1 次元だけで扱う。量の単位は，JIS Z 8203:2000 による。

金額も量と同様に扱う。通貨の表現は ISO 4217 による。

数は，集合の要素数であり，常に整数で単位をもたない（日本語では便宜的に「個」や「本」などの数詞を使うことがあるが，これは単位とはみなさない）。

```
数量 = ["(",量,","],数,[")"].
量 = 数値,"_",単位.
数 = 整数.
```

たとえば，次のようになる。

```
"(338_g, 3)"     量が 338 グラムのものが 3 個
"1362_JPY"       1362 円
"200"          200 個
```

6.3.3 日時のエンコード

日時は ISO 8601:2004 に準拠して，次の形式で表現する。

```
日時 = 日付,comma,時刻,[標準時識別].        (* 標準時識別の省略時は JST，以下同じ *)
日付 = 年,"-",月,"-",日,[標準時識別].
年 = ? 2000 以上の 4 桁の整数 ?.
月 = ? 01 から 12 の 2 桁の整数 ?.
日 = ? 01 から 31 の 2 桁の整数 ?.           (* 上限値は年月の組合せによって決まる*)
時刻 = 時,[:,分,[:,秒]],[標準時識別].       (* 省略時は JST *)
```

```
時 = ? 00 から 23 の 2 桁の整数 ?.
分 = ? 00 から 59 までの 2 桁の整数 ?.
秒 = ? 00 から 60 までの 2 桁の整数 ?.        (* 60 は閏秒*)
標準時識別 = "GMT"|"JST "                    (* 日付と時刻を分離するときは同一であること*)
```

たとえば，日本時間で平成 28 年 2 月 3 日の午後 6 時 30 分は，次のようになる。

```
"2016-02-03JST,18:30JST"
"2016-02-03,18:30"
```

6.3.4 時間間隔のエンコード

時間間隔（duration）は ISO 8601:2004 に準拠して，次の形式で表現する。

```
日時間隔 = "P",[日間隔],[時間隔].
日間隔 = [年数,"Y"],[月数,"M"],[日数,"D"].
時間隔 = "T",[時数,"H"],[分数,"M"],[秒数,"S"].
年数 = ? 0 以上の整数 ?.
月数 = ? 1 から 12 の整数 ?.
日数 = ? 0 以上の整数 ?.
時数 = ? 0 から 23 の整数 ?.
分数 = ? 0 から 59 までの整数 ?.
秒数 = ? 0 から 60（閏秒）までの整数 ?.
```

たとえば，2011 年 3 月 11 日から 2016 年 3 月 11 日までの時間間隔は，次のように異なる複数の表現が可能である。

```
"P5Y0M0DT0H"
"P4Y366D"
"P1827D"
```

6.3.5 場所のエンコード

場所（place）は任意の形式で表現する。住所，緯度経度，法人名，組織名などであり得るため，実装ごとに定める。

```
場所 = ? 任意の形式 ?.
```

6.3.6 商品のエンコードと仕様表現

発注者–受注者サブシステム間のメッセージ連携において，サブシステムごとに設計されたドメインモデルを共有することを前提にしない。たとえば，商品を商品コードで識別することは，少量多品種や個別受注品を扱う場合に，システム間連携の柔軟性を削ぐ。サブシステム内はどうあれ，サブシステム間では商品の識別を次のように行う。

商品の識別は，あらかじめサブシステム間で合意された商品を識別する文字列（商品識別）と仕様を表現する文字列（仕様表現）からなる。商品が直ちに品目に対応する場合，JIS や業界で定められている合意された品名を用い，それ以外のものは，商品が修理などのサービスである場合も含め，当該生産システムへの参加者で協議して決める。JIS で合意された品名には，たとえば，「ねじ」（JIS B 205:1997, JIS B 207:1982），「ばね」（JIS B 2704:2009）などがある。


```
商品 = {,",'what',",:,", 商品識別,",[comma,",'spec',",:,{,仕様表現,}],}.
商品識別 =文字列.        (*「ねじ」，「ばね」，「修理」など *)
仕様表現 = (仕様名,:,仕様値)|(仕様名,:,仕様値,{comma,仕様名,:,仕様値}).
仕様名 = ",文字列,".                    (* 仕様名は品目ごとに別途規定する *)
仕様値 = ["],即値,['_',単位],["].       (* 即値が整数で単位を省略する場合，"を省略可 *)
```

仕様表現は，仕様名と仕様値（単位付き）の対のリストで表現する。仕様名も，品名ごとに JIS や業界で定められているものを優先的に用いる。

たとえば，品目「玉軸受（ボールベアリング）」の仕様表現は，仕様名「D(外径)」，「d(内径)」，「B(幅)」などのサイズと「シール」の有無（○か−）で表現されるとするとき，次のように表現される。

```
{"what":"玉軸受","spec":{"D":"120_mm","d":"100_mm","B":12,"シール":"○"}}
```

6.3.7 曖昧な仕様値

注文の状態によって，商品の仕様表現が曖昧な場合がある。たとえば，幅をもたせたり，いくつかの候補値を挙げたり，未定の場合もある。これらは製造に取りかかる前には確定されることが期待される。具体的な表現については Annex F に示す。

6.3.8 個品の識別

個品とは，集合である品目の要素で，具体的な一つひとつの現物を言う。物理的に一つのものを個品として扱うこともあるし，ロットを量を持った一つの個品として扱うこともある。個品単位で残量を管理したり，トレーサビリティを記録するために用いる。

商品が品目を含みかつ個品を識別して注文する場合は，品目の記述の中に次のように個品を識別する文字列（個品識別）を追加できる。個品識別にはシリアル番号などが用いられるが，具体的には，当該生産システムへの設計者間で協議して決める。

```
品目 = {,",'what',",:,",品目識別,",[comma,",'spec',",:,[,{,仕様表現,},]],
        ["','serialno',",:,[,",個品識別,"{comma",個品識別,"}],].
```

6.3.9 パーティのエンコード

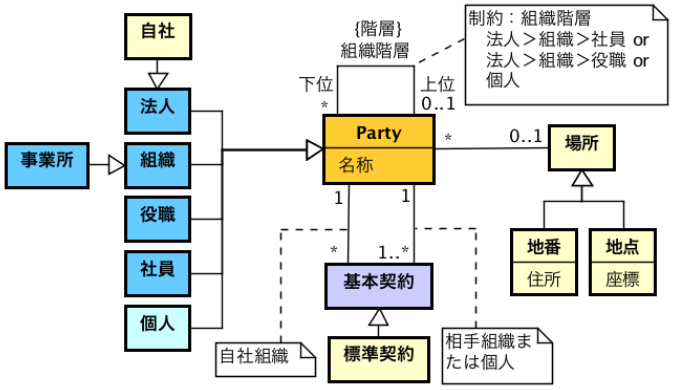

Figure 3　Party の概念モデル

パーティ（Party）とは，契約や注文の主体となり得る法人-組織-役職または社員，あるいは個人を表す複合概念である（Figure 3 の概念モデル参照）。取引によって，パーティをどのような階層

で識別するかが異なる。その具体的なエンコードの形式は，当該生産システムへの設計者間で協議して決める。

7　階層間連携メッセージ

階層間連携言語（iHCl）を用いて，サブシステム間で交わすメッセージの内容を記述する。

7.1　メッセージ設計の原則

7.1.1　ドメインモデルからの独立

メッセージ内容は，階層間連携言語を用いて記述する。発注者および受注者のサブシステムがそれぞれに想定する「ドメインモデル」を前提としないメッセージとする。

メッセージ内容は，発注者および受注者サブシステム間の相対的関係でのみ規定され，それ以外の文脈に依存しない。

7.1.2　エンティティの識別

品目やパーティなどのエンティティの識別方法については，当該生産システムの設計者間で協議して決める。それ以前に，IEC Common Data Dictionary（IEC 61360 シリーズ）や業界標準によって識別や略称が決まっているもの，たとえば ISO 13616（銀行口座，IBAN コード）などはそれに従う。

7.1.3　任意の符号化の忌避

フラグや区分，省略語などのドメインモデルおよびプログラムによる解釈に依存するデータをメッセージに含めない。任意の符号化を避け，それ自身で説明できる表現を用いること。たとえば，消費税の扱いを，消費税区分として"1"や"2"のように符号化して表すのではなく，"内税"，"外税"などと明示的に表現する。これも符号化の一種といえるが，多くの発注者と受注者の間であらためて約束を取り交わす必要がない点で後者がよい。

7.1.4　制御メッセージと業務メッセージ

メッセージには，制御メッセージと業務メッセージの二種類がある。制御メッセージは業務メッセージの通信を制御する支援的役割をもつ。業務メッセージは一つの注文に関する対話構造を実現するメッセージである。

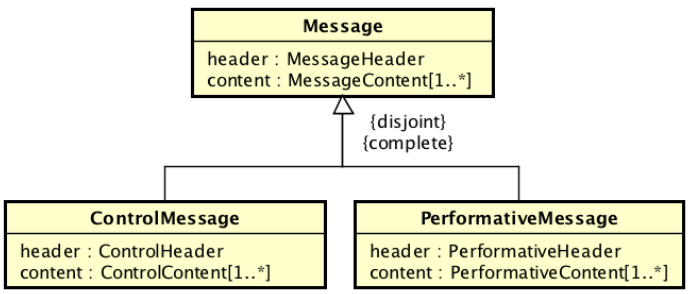

Figure 4　2 種類のメッセージ

7.1.5 非同期通信

個々のメッセージは送信側が送信した後，受信側が一旦受け取り，送信側はそのメッセージに対する処理の完了を待つことはない（非同期通信）。非同期通信では，送信側がなんらかの処理結果メッセージを受け取ったとき，それがかつて送ったどのメッセージまたは注文に対応するのかが明確に分かるようにメッセージ識別または注文識別を付す。

メッセージ識別および注文識別は通常，C-Subsystem で発番するが，注文なし実績の場合は，P-Subsystem でも発番され，Customer に通知されて，Customer は受信した旨を Performer に返す。このとき，メッセージ識別および注文識別がシステム全体で重複しないように制御される。

7.1.6 メッセージの内容および通信相手の特定

C-Subsystem は，自分が発行する注文を処理できるサブシステムはどれか，使用する遂行動詞はどれか，業務メッセージに含める項目はどれかをシステム全体が稼働する前に知っている必要がある。また，P-Subsystem は，自分が処理すべき注文を送って来るサブシステムはどれか，使用する遂行動詞はどれか，業務メッセージに含まれる項目はどれかを，システム全体が稼働する前に知っている必要がある。通信アダプタは，相互に通信すべき相手を特定して通信チャネルを確立し，サブシステム間で異なる対話プロトコルに対応し，サブシステム間で異なる業務メッセージの内容に対応するために，ケイパビリティプロファイルで定義し，これを必要に応じて参照する。具体例を Annex G に示す。

7.1.7 通信異常の通知

メッセージの送受信においてデータがロストしないことを保証するために，再通信を行い，重複メッセージを送信することを許すが，業務メッセージが重複していることは検知され，送信元に警告を送り，受信側は無視するように処置する。業務メッセージの重複はメッセージ識別の重複を見ることで検知される。

それ以外の通信異常は，より低レベルのネットワークプロトコルで検知され，処置される。

7.2 メッセージの基本構文

メッセージの基本構文を次のとおりとする。メッセージの具体例を Annex D および Annex E に示す。

7.2.1 メッセージの構成

メッセージはメッセージヘッダとメッセージ本文から構成される。

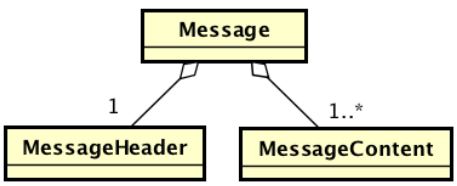

Figure 5　メッセージの構造

```
メッセージ = {,header,comma,content,}.
```

7.2.2 メッセージヘッダ

メッセージヘッダは，制御動詞または遂行動詞，送信元，送信先，送信時刻印，メッセージ識別からなる。制御動詞は制御メッセージで，遂行動詞は業務メッセージで使われる。

```
header = verb,comma,clause|(clause,{comma,clause}).
verb = (",'controlVerb',",:,",制御動詞,")|
     (",'performative',",:,",遂行動詞,").
clause =              (* 重複しないこと *)
   ",'sender',",:,",subsystem-id,"|    (* 送信元，Sell などのサブシステム名 *)
   ",'receiver',",:,",subsystem-id,"|  (* 送信先識別 *)
   ",'time-stamp',",:,",timestamp,"|   (* 送信時刻印 *)
   ",'message-id',",:,",message-id,"|  (* メッセージ識別 *)
   ",'in-reply-to',",:,",message-id,"| (* 制御メッセージで，元のメッセージ識別 *)
   ",'language',",:,",'iHCl',"         (* content の記述言語，常に iHCl *)
subsystem-id = ? サブシステムの識別 ?.  (* 実装環境ごとに決定 *)
timestamp = 日時.                     (* 時刻印を文字列に変換 *)
message-id = ? ビット列 ?.            (* システムごとに実装判断 *)
```

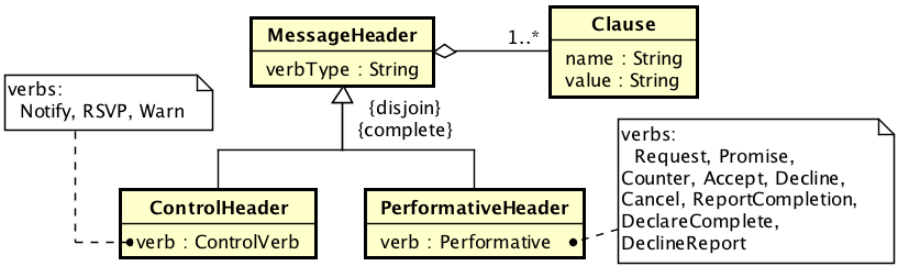

Figure 6　メッセージヘッダの構造

Table 1　Clause に記述する項目の一覧

name	value
sender:	送信元の識別
receiver:	受信先の識別
time-stamp:	送信時の時刻印
message-id:	メッセージ識別
in-reply-to:	制御メッセージのとき，元のメッセージ識別
language:	常に"iHCl"

7.2.3 メッセージ本文

メッセージ本文は，content の値として，カギ括弧内に属性名と属性値の対で記述する。

```
content = ",'content',",:,[,message-content,].
```

メッセージ本文の構文および意味は，メッセージごとに説明する。

7.3 制御メッセージ

7.3.1 メッセージ識別

制御メッセージでは，制御のライフサイクルを管理する識別子として，メッセージヘッダのメッセージ識別を用いる。制御メッセージに対する返信のメッセージヘッダには，"in-reply-to"で，元のメッセージ識別を明示する。メッセージ識別の発番方法について，Annex H で補足的に説明する。

7.3.2 制御動詞

制御メッセージで用いる制御動詞には，「取りに来い（Notify）」「応答を乞う（RSVP）」「警告する（Warn）」の 3 種類がある。

```
制御動詞 = 'Notify'|'RSVP'|'Warn'.
```

7.3.3 「取りに来い（Notify）」メッセージの構文

C-Subsystem は，P-Subsystem に渡したい注文があることを P-Subsystem に通知する。このメッセージの使用は任意である。

C-Subsystem が発行する「取りに来い」メッセージの構文は次のとおりとする。すなわち，空文とする。

```
制御動詞 = 'Notify'.
message-content = ''.
```

7.3.4 「応答を乞う（RSVP）」メッセージの構文

P-Subsystem は，ケイパビリティプロファイルによりあらかじめ定められている C-Subsystem に，適当なタイミングで「応答を乞う」メッセージを送る。その際，返答を受ける P-Subsystem の送信先（callback）を伝える。

C-Subsystem が送るべきメッセージをもっているかどうか分からない状態で，いきなり「応答を乞う」メッセージを送るのでは無駄なやりとりになる可能性が高い（ポーリング）。そのため，C-Subsystem から「取りに来い（Notify）」メッセージを受けてから，P-Subsystem の都合の良いタイミングで「応答を乞う」メッセージを送るようにするとよい。ただし，「取りに来い」メッセージを使うかどうかは任意である。

C-Subsystem は「応答を乞う」メッセージに応えて，それを発行した P-Subsystem に「注文提示」などの下りのメッセージを送る。

この構文は次のとおり。

```
制御動詞 = 'RSVP'.
message-content = ',
   "select(",選択対象,"),", (* 受け取るデータ項目を指定する *)
   "where(",検索条件,"),", (* データの範囲を指定する*)
```



```
"callback(",返答の送信先,")" (* 送信先のアドレス *)
,'.
選択対象 = '*'|データ名|データ名,",",選択対象. (* '*'は一度だけ *)
検索条件 = ,"time-stamp",">",前回受信 time-stamp. (* 要素は重複しないこと *)
```

データ名は，業務メッセージで使用するデータ項目（o-attr）を指定する。検索条件では，二重に受け取らない，データをロスしないように受け取るデータの範囲を指定する。

下りのメッセージを受け取るに当たって，P-Subsystem は検索条件を提示できる。前回受信 time-stamp を検索条件に指定することで，C-Subsystem は前回注文を送信した以降に発生したメッセージを抽出する。

送信するメッセージがないとき，C-Subsystem は「応答を乞う」メッセージに対して何も応答しない。

7.3.5 「警告する（Warn）」メッセージの構文

対話中に何らかの異常を検知した場合，C-Subsystem または P-Subsystem は，相手にその旨を警告して，間接的に対応を求める。起こりうるエラーは，Figure 2 の対話構造に従わないメッセージを受け取った（プロトコルシーケンスエラー），メッセージを重複して受け取った（重複メッセージエラー），送信注文件数と受信注文件数が一致しない（件数不一致），チェックサムが一致しない（チェックサム不一致）場合などである。本メッセージに対してどのように対応するかは，本規格書では定めない。

```
制御動詞 = 'Warn'.
message-content = error,' at ',message-id,' in ',遂行動詞.
error =
   'Protocol Out of Sequence'|          (* プロトコルシーケンスエラー *)
   'Message Duplicated'|                (* メッセージ重複エラー *)
   'Timed out'|       (* 許容応答時間を過ぎた *)
   'NoR Unmatched'|                     (* 注文件数不一致 *)
   'Checksum Unmatched'.                (* チェックサム不一致 *)
```

7.4 業務メッセージ

7.4.1 注文識別

業務メッセージでは，注文のライフサイクルを管理する識別子として，メッセージ本文中の注文識別を用いる。注文識別の発番方法について，Annex H で補足的に説明する。

7.4.2 遂行動詞

業務メッセージで用いる遂行動詞は，Figure 2 の対話構造の遂行動詞に対応して 9 種類がある。

```
遂行動詞 = 'Request'|'Promise'|'Counter'|'Accept'|'Decline'|'Cancel'|
     'ReportCompletion'|'DeclareComplete'|'DeclineReport'.
```


C-Subsystem は P-Subsystem ごとに，どの遂行動詞を用いるかをケイパビリティプロファイルで明示しておく。具体例を Annex G に示す。

7.4.3 業務メッセージの内容

業務メッセージの内容は大きく 3 つの部分に分かれる。最初は注文語を提示する部分，二番目は注文内容を提示する部分，最後が通信上の誤りをチェックする部分である。

```
message-content =                          (* 要素は重複しないこと *)
   {,",'order',",:,",注文語,",},           (* 注文語の提示部分 *)
   [comma,[,注文内容,]],                   (* 注文内容の部分 *)
   [comma,{,checker,{comma,cheker},}].      (* 誤りチェックの部分，任意 *)
注文語 = 文字列.                           (* あらかじめ定められた文字列 *)
checker =                                  (* 要素は重複しないこと *)
   ",'number-of-records',",:,",nor,"]|      (* 注文の件数，任意 *)
   ",'check-sum',",:,",値,".                (* チェックサム値，任意 *)
nor = 整数                                 (* このメッセージで送られる注文の件数 *)
```

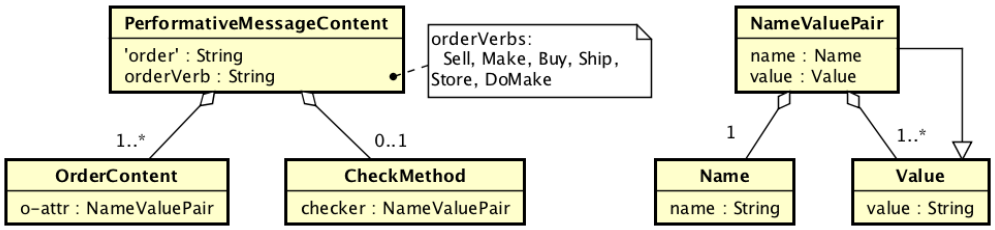

Figure 7　業務メッセージの注文内容の構造

誤りチェックの number-of-records および check-sum は，同時に複数の注文を提示する場合に用いることができる。通信中のデータのロストや文字化けを早期に発見するための措置である。通信の安全が保証される環境では，このオプションを用いる必要はなく，その扱いはシステム全体で別途定める。

注文語は，注文提示メッセージに含まれ，発注者が受注者候補に対して「どうしてほしいか」を示す動詞（英語では命令形，日本語では終止形）であり，その意味は生産システムに参加するサブシステム間で共有されている必要がある。「どうしてほしいか」は，達成される結果の記述によって定義される。また，注文語は，ケイパビリティプロファイルの Activity タグに対応する。

注文語の定義は，当該生産システムの主催者（Owner）が定める。

たとえば，注文語'Make'の意味は，Sell サブシステムが注文した品目が指定した仕様どおりに，希望日までに完成していることであり，注文語'Ship'の意味は，注文した品目が，希望日までに，希望の倉庫から，希望の届け先に到着していることである。注文語の具体的例を Annex A に示す。

以降，注文内容について遂行動詞別に説明する。

7.4.4 「注文提示（Request）」メッセージの構文

C-Subsystem は P-Subsystem の「応答を乞う（RSVP）」メッセージに応えて，「注文提示（C: Request）」メッセージを返す（Figure 2 の状態 1 から 2 への遷移）。なお，「注文提示」メッセ

ージ中に 1 つ以上の注文が含まれることがある。注文の実行は注文識別ごとに進度管理され，「注文完了宣言」「中止」「取り下げ」または「辞退」されるまで対話構造が維持される。

構文は次のとおり。

```
遂行動詞 = 'Request'.
注文内容 = {,o-attr,{comma,o-attr},}.
o-attr = ",'o-id',",:,",注文識別,".     (* 他の注文属性については Annex C を参照 *)
```

C-Subsystem は，業務メッセージ内に，どの注文属性（o-attr）を含めるかをケイパビリティプロファイルで定義する。Annex C に注文語別に選択可能な注文属性の一覧を，Annex G に選択した注文属性をケイパビリティプロファイル上で記述した例を，Annex D に選択した注文属性を含むメッセージの具体例を示す。

7.4.5 「約束（Promise）」メッセージの構文

P-Subsystem が，C-Subsystem からの「注文提示」メッセージを受け取って，注文ごとに，品目仕様，数，納期などを一定の水準で許容できると判定されたときは，「約束（P: Promise）」メッセージを返す（状態 2 から 3 への遷移）。これは，一般に「注文請書」などと呼ばれる。注文に対する対価の見積もりおよび／または推定納期を，このメッセージに含めて返すかどうかは，当該生産システムの設計者間で別途定める。

メッセージの内容は，提示された注文内容に対する差分となる。すべての項目が元の注文と違いがなければ，注文識別だけとなる。

構文は次のとおり。

```
遂行動詞 = 'Promise'.
注文内容 = {,o-attr,{comma,o-attr},}.
o-attr = ",'o-id',",:,",注文識別,".     (* 他の注文属性については Annex C を参照 *)
```

7.4.6 「対案提示（Counter）」メッセージの構文

P-Subsystem が，C-Subsystem からの「注文提示」メッセージを受け取って，注文ごとに，品目の仕様，数|量，納期などについて許容範囲を超えた場合，P-Subsystem は「辞退」するか「対案提示」するかを判断しなければならない。対案提示する場合は，「対案提示（P: Counter）」メッセージを返す（状態 2 から 6 への遷移）。この間，注文はペンディングの状態にある。

構文は次のとおり。

```
遂行動詞 = 'Counter'.
注文内容 = {,o-attr,{comma,o-attr},}.
o-attr = ",'o-id',",:,",注文識別,".     (* 他の注文属性については Annex C を参照 *)
```


7.4.7 「対案受入（Accept）」メッセージの構文

先の「注文提示」に対して P-Subsystem が「対案提示」メッセージを送り返してきた場合，C-Subsystem は，それを受け入れるか取り下げるかを判断する。受け入れる場合には「対案受入（C: Accept）」メッセージを返す（状態 6 から 3 への遷移）。

構文は次のとおり。

```
遂行動詞 = 'Accept'.
注文内容 = ",'o-id',",:,",注文識別,".    (* 対案を受け入れる注文の識別，必須 *)
```

7.4.8 「取り下げ（Cancel）」メッセージの構文

先の「注文提示」に対して P-Subsystem が「対案提示」メッセージを送り返してきた場合，C-Subsystem が対案を受け入れないときは「取り下げ（C: Cancel）」メッセージを返す（状態 6 から 8 への遷移）。

構文は次のとおり。

```
遂行動詞 = 'Cancel'.
注文内容 = ",'o-id',",:,",注文識別,".    (* 取り下げる注文の識別，必須 *)
```

7.4.9 「逆対案提示（Counter）」メッセージの構文

C-Subsystem が，P-Subsystem からの「対案提示」メッセージを受け取って，その内容が満足できないにしても，品目，仕様，納期，数量のどれか（またはすべて）を変更することで，成約の可能性がある場合，発注者は逆に対案を提示できる。この場合，C-Subsystem「逆対案提示（C: Counter）」メッセージを返す（状態 6 から 2 への遷移）。この間，注文はペンディングの状態にある。

構文は次のとおり。

```
遂行動詞 = 'Counter'.
注文内容 = {,o-attr,{comma,o-attr},}.
o-attr = ",'o-id',",:,",注文識別,".    (* 他の注文属性については Annex C を参照 *)
```

7.4.10 「辞退（Decline）」メッセージの構文

C-Subsystem からの「注文提示」や「逆対案提示」に対して P-Subsystem は「約束」を拒否できる。この場合，P-Subsystem は「辞退（P: Decline）」メッセージを返す（状態 2 から 8 への遷移）。

構文は次のとおり。

```
遂行動詞 = 'Decline'.
注文内容 = ",'o-id',",:,",注文識別,".    (* 辞退する注文の識別，必須 *)
```

7.4.11 「終了報告（ReportCompletion）」メッセージの構文

P-Subsystem が注文に関する作業を終了または途中で停止したとき，その注文を提示した C-Subsystem に「終了報告（P: Report Completion）」メッセージを送る（状態 3 から 4 への遷移）。

終了報告は，数量や納期，さらには実現された仕様が Promise したものと異なる場合も，P-Subsystem が終了と判断すれば，それぞれの注文属性に値を設定してこのメッセージを送る。それで十分（Complete）かどうかは C-Subsystem が判定を下す。注文属性が Promise したとおりであれば，注文識別だけを送信すればよい。

構文は次のとおり。

```
遂行動詞 = 'ReportCompletion'.
注文内容 = {,o-attr,{comma,o-attr},}.
o-attr = ",'o-id',",:,",注文識別,".     (* 他の注文属性については Annex C を参照 *)
```

製造現場の判断による注文に対応しない製造が行われる（注文なし実績）ことが稀にある。この場合も P-Subsystem は終了報告を送り，C-Subsystem は形式的に，受信した旨を P-Subsystem に返す。このときのメッセージ番号および注文番号は P-Subsystem で発番されるが，生産システム全体で重複しないように制御される。

7.4.12 「受注者中止（Cancel）」メッセージの構文

P-Subsystem の何らかの都合によって，一旦約束した注文を途中で停止するときは，基本的に「終了報告」（P: Report Completion）」メッセージを送る。

P-Subsystem は「受注者中止（P: Cancel）」メッセージを送る（状態 3 から 7 への遷移）こともできるが，C-Subsystem との対話を行わない一方的で特別な状況に限られる。

構文は次のとおり。

```
遂行動詞 = 'Cancel'.
注文内容 = ",'o-id',",:,",注文識別,".    (* 中止する注文の識別，必須 *)
```

7.4.13 「発注者中止（Cancel）」メッセージの構文

C-Subsystem の何らかの都合によって，着手している注文を途中で停止するとき，または完了した注文を中止するときは，「発注者中止（C: Cancel）」メッセージを送る（状態 3 から 9 への遷移，または状態 4 から 9 への遷移）。

発注者中止の場合，途中まで出来上がっている製品をどう扱うかが問題になる。これについては，システムの外で決定し，システムでの対処を決める。

構文は次のとおり。

```
遂行動詞 = 'Cancel'.
注文内容 = ",'o-id',",:,",注文識別,".    (* 中止する注文の識別，必須 *)
```

7.4.14 「注文完了宣言（DeclareComplete）」メッセージの構文

P-Subsystem の「終了報告」メッセージに対して，C-Subsystem は，たとえ数が不足していたり，納期をオーバしていても，それを受け入れる（検収する）か，拒否するかを判断しなければならない。受け入れる場合は「注文完了宣言（C: DeclareComplete）」メッセージを送る（状態 4 から 5 への遷移）。

構文は次のとおり。

```
遂行動詞 = 'DeclareComplete'.
注文内容 = ",'o-id',",:,",注文識別,".    (* 完了宣言する注文の識別，必須 *)
```

7.4.15 「終了報告の拒否（DeclineReport）」メッセージの構文

C-Subsystem が「終了報告」を拒否する場合は，P-Subsystem に「終了報告の拒否（C: DeclineReport）」メッセージを送る（状態 4 から状態 3 への遷移）。

このメッセージを受けて P-Subsystem は，一律に自動的に再作業するわけには行かないので，一旦システムの外で判定（営業判断など）する。

構文は次のとおり。

```
遂行動詞 = 'DeclineReport'.
注文内容 = ",'o-id',",:,",注文識別,".    (* 終了報告を拒否する注文の識別，必須 *)
```

Annex A (informative) サブシステム連携の例

生産管理システム全体を Figure A.1 のような機能（Activity）を受け持つ MSU（サブシステム）間を本書のメッセージとプロトコルによって，疎に連携させる。機能の構造は，垂直方向には関心対象（concern）であるオブジェクトに基づいて決定する。各 MSU には，関心対象の生成から消滅までのライフサイクル事象が記録され，オブジェクトの一生を追跡することができる。

IEC 62264 では，生産実行の文脈において，レベル 0 から 4 までの機能階層を定めている。レベル 4 は資源の引当調整，レベル 3 は製造オペレーション，レベル 2 以下は物理的な機器制御の層としている。これらは，レベルごとにタイムフレームが異なるという程度で，機能分割を説明しようとしている。本設計では，関心対象が異なるということで，明確なサブシステム分割の基準を定め，それに伴う機能の配置を定めている。

A.1 階層構造（Management Layer）

垂直方向のサブシステム連携は管理水準に基づく。管理水準は，組織の役割に対応することが多いため，組織階層のように見えるが，本質的には，組織ごとの関心対象に基づく。

IEC 62264 のレベルと管理階層の水準を対応づけるとすると，レベル 4 が Sell サブシステム，レベル 3 が Buy/Make サブシステム，レベル 2 以下が Do サブシステムにあたる。Selling I/F は対外的な通信サブシステム（EDI など）で，IEC 62264 では定義されていない。

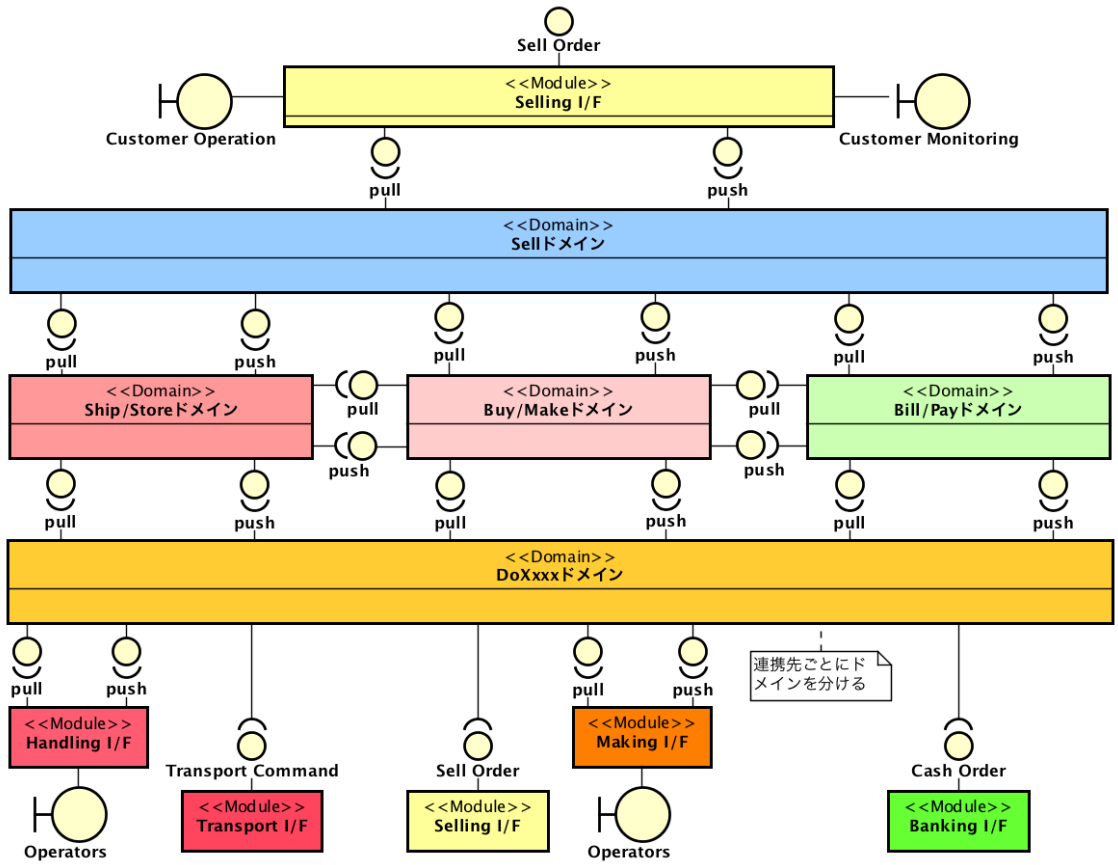

Figure A.1　生産管理システムの全体構造の例

A.2　注文語の定義（Definition of Request Verb）

注文語は，C-Subsystem 別に注文に記述すべき主要な内容を決定する。これを， Table A.1 のとおり定義する。

Table A.1　注文語の定義

C-Subsystem	注文語	説明	P-Subsystem
Selling I/F	Sell	商品を販売せよ	Sell
Sell	Ship	商品を出荷せよ	Ship/Store
Sell	Buy	商品を購入せよ	Buy/Make
Sell	Make	商品を製造せよ	Buy/Make
Sell	Store	商品を入荷せよ	Ship/Store
Sell	Bill	代金を売掛にせよ	Bill/Pay
Buy/Make	Buy	部品を購入せよ	Buy/Make
Buy/Make	Make	部品を製造せよ	Buy/Make
Buy/Make	Store	部品を入荷せよ	Ship/Store
Buy/Make	Ship	製品を出庫せよ	Ship/Store
Buy/Make	Store	製品を入庫せよ	Ship/Store
Buy/Make	Pay	代金を買掛にせよ	Bill/Pay
Buy/Make	Do(Make)	製造を実施せよ	Do(Make)
Ship/Store	Do(Ship)	出荷を実施せよ	Do(Ship)
Ship/Store	Do(Store)	入荷を実施せよ	Do(Store)
Bill/Pay	Do(Bill)	請求額を請求せよ	Do(Bill)
Bill/Pay	Do(Pay)	被請求額を支払え	Do(Pay)

A.3　システム構造（System Structure）

A.3.1　Sell サブシステム

Figure A.1 の上から 2 段目の管理階層は「Sell サブシステム」である。ここでは，購入（Sell）注文の確定から注文の完了（請求あるいは入金）までのライフサイクル事象が扱われる。また仕入の注文は内部的に発生し，ここで注文も確定から注文の完了（支払）までのライフサイクル事象が扱われる。仕入の注文は，対象の品目によって購買（Buy）か生産（Make）かどちらかの注文に置き換わる。

A.3.2　Buy/Make サブシステム

同じく上から 3 段目の管理階層で中央に配置されているのは「Buy/Make サブシステム」である。ここでは，Sell サブシステムで生成された購買または生産注文（Buy/Make）に基づきこれらの計画を立てる。計画の関心対象は，材料，部品，設備，作業者，作業空間，時間などのうち，有限な「資源」の引当であり，そのライフサイクルが記録される。引当に当たっては，歩留まり率や部品欠品率，代替設備などに関する製造知識（たとえば IEC 62264-1:2013 Figure 27 を参照）が必要である。資源引当のムダを解消するための，作業実行の時刻を調整するし，資源の引当を最適化する処理をスケジューリングと呼ぶ。

A.3.3 Ship/Store サブシステム

同じく上から 3 段目の管理階層で左に配置されているのは「Ship/Store サブシステム」である。ここでは，Buy/Make サブシステムで生成された部品の受入，出庫など，物流作業の計画を立てる。

A.3.4 Bill/Pay サブシステム

同じく上から 3 段目の管理階層で右に配置されているのは「Bill/Pay サブシステム」である。ここでは，Sell サブシステムで生成された商品の代金請求と仕入・購買の代金支払いの計画を立てる。

A.3.5 Do サブシステム

同じく上から 4 段目の階層は「Do サブシステム」である。Do サブシステムのインスタンスは図中 3 段目の各サブシステムに対応して配置され，対上位においては作業指示系との接点をもち，対下位においては実世界との接点を構成するためのインタフェースをもつ。各 Do サブシステムのインスタンスは，上位サブシステムが計画した物流の Do(ship), Do(Store)注文，製造および購買の Do(Make), Do(Buy)注文，入出金の Do(Bill), Do(Pay)注文を具体的に実施し，物流または製造機器を制御し，その実績を記録する。実績が予定と異なる場合は，上位サブシステムで定めた資源引当を変更しない範囲で指示内容を変更してもよい。Make ドメインの下位に位置する Do(Make)サブシステムは，かつて MES（Manufacturing Execution System）と呼ばれていた。

作業効率を高めるために実行順序を調整する処理もスケジューリングと呼ぶ。ただし，Buy/Make サブシステムのそれと区別するため，シーケンシングまたはオプティマイゼーションと呼び換えることがある。

実世界との接点では，さまざまな事態が発生しうる。このため，上位システムとは異なり，作業展開を 1 ステップずつ行って，発生した事態に適切に対応できるリアクティブなアーキテクチャを採用することが多い。

A.3.6 Handling, Transport, Selling, Making, Banking インタフェース

Figure A.1 のシステムの最下段にある「インタフェース」は，制御可能な設備とのインタフェースや，作業者への指示および実績の収集を行うモジュールである。たとえば，マテハン（Material Handling），輸送（Transport），製造実行（Making），外部発注（自分が Customer の立場での Selling），銀行のサービス API との接続などがある。これらの機器インタフェースはその媒体，データ形式，ビジネスプロトコルなどにおいてさまざまなバリエーションをもつ。Do サブシステムとこれらのインタフェース間の通信規約は機器ごとに定義される。

設備が作業者の場合は，指示の表示と実績の登録を行う画面などのユーザインタフェースを提供する。

A.3.7 Selling Interface

Figure A.1 のシステムの最上段は管理階層ではなく，システムの外部とインタフェースするモジュールである。「Selling Interface」は，顧客とメッセージの送受信を行う。顧客から受け取る/顧客へ返すメッセージの形式は，媒体，データ形式，ビジネスプロトコルなどにおいてさまざまなバ

リエーションをもつ。これを iHCl に/から変換する。これらの条件は，基本契約の各条項に記述される。

Annex B (informative) Customer-Performer モデル

本生産システムでは，サブシステム間の対話の原則を，Customer-Performer モデル[6]（Figure B.1）に基づいて定める。このモデルは，業務の流れの中で，発注者（Customer）と受注者（Performer）の間で交わされる対話の構造を規定している。このモデルはそのまま発注者サブシステムと受注者サブシステム間の対話の構造に適用できる。その概要を説明する。

B.1 対話の構造

発注者は，ある仕事を注文するに当たって，受注者の候補と注文内容と条件を提示する（注文）。受注者は注文内容を検討して，請けると決めたら契約を締結する（約束）。一般的には注文書であったり，基本契約を結んだ上での個別契約だったりする。社内での注文であれば口頭で行われることもある。その後，受注者は注文内容を実施しし，その結果を提出する（実行）。発注者はその結果を見て，条件を満足していれば受け取る。不満であれば，その点を示してやり直してもらう（検収）。

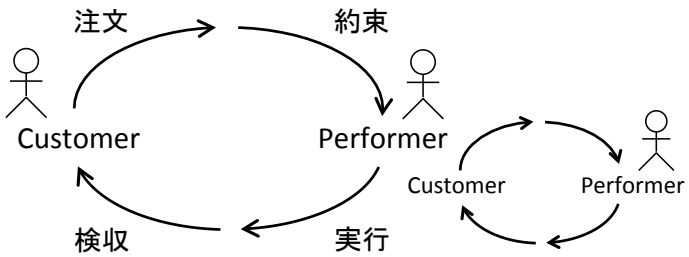

Figure B.1 Customer-Performer モデル

B.2 対話の意味

このモデルのポイントは，①発注者と受注者間に四つの対話のフェーズがあること，②受注者が請けた注文内容をいくつかの部分に分けて，今度はその仕事の発注者に役割を変えて，同じ対話のフェーズを再帰的に繰り返すこと，すなわち，どの階層においても二者間で交わされる対話は同型であることを示したことにある。

B.3 手順分解と再委託

受注者は受託した内容を自分自身で実施してもよいが，他者に再委託することもできる。そのときは，受託した業務の内容を手順分解して，自らは発注者にロールを変えて，分解した手順の内容を下位の一つ以上のサブシステムに注文を出す。あとは同様に進む。各手順の注文が完了したら，今度はロールを受注者に変えて，結果を集約して発注者に報告する。このように，手順分解・詳細化をしながら次々と下位のサブシステムと連携すことで，Enterprise レベルから Device Control レベルまでの統合的な業務連携が実現できる。

途中の業務を外部の事業者に注文（外注）することもある。

Annex C (informative)　注文の構造

業務メッセージの注文属性（o-attr）の内容は，注文語や取り扱う品目の特性によって異なる。ここでは注文語ごと，遂行動詞ごとに選択可能な注文属性の例を挙げる。実際にメッセージングで使用する注文属性は，P-Subsystem のケイパビリティプロファイルの<Data>タグで定義する。

注文属性のうち，注文識別，パーティ，組織，品目識別，勘定名，場所などの識別子の具体的な表記法（コード化のルールなど）については，当該システムの参加者の合意で個別に定める。

注文属性を重複して選択しないこと。注文属性のうち，任意とされている属性を省略できるが，その省略値の解釈は使用する当該システムの参加者の合意で個別に定める。

数と量の使い分けは，品目の特性に依存する。たとえば，一つひとつが分離されて存在する品目は数，ビンの内容物や線材のように一部を取り出して使うような品目は量を用いる。数は常に正数である。品目のこのような特性は，製品定義情報（IEC 62264-3）において述べられるものであり，本規格では数と量の使い分けについては定めない。

C.1　Sell 注文

C.1.1　Request メッセージ

注文語 Sell（売れ：販売せよ）の Request メッセージに記載する注文属性の例を示す。

```
o-attr = ",'o-id',",:,",注文識別,"|
   ",'who',",:,",パーティ|組織,"|              (* 注文主（Customer） *)
   ",'what',",:,",商品識別,"|                  (* 注文する品目またはサービス *)
   ",'quantity',",:,数|(",量,")|               (* 注文する数または量 *)
   ["','who_pay',",:,",パーティ|組織,"]|       (* 請求先，注文主と異なる場合のみ，任意 *)
   ["','whom_incharge',",:,",組織,"]           (* 担当部署の指定，任意 *)
   ["','spec',",:,[,{,仕様表現,},]]|           (* 注文する品目またはサービスの仕様，任意 *)
   ["','contract',",:,",契約識別,"]|           (* この注文が依拠する契約の識別，任意 *)
   ["','when_from',",:,日時]|                  (* 希望する開始日時，任意 *)
   ["','when_to',",:,日時|'?']|                (* 希望する納期または終了日時，任意 *)
   ["','where_from',",:,",場所|パーティ,"]|    (* 発地，where_to とは別の値，任意 *)
   ["','where_to',",:,",場所|パーティ,"]|      (* 着地，where_from とは別の値，任意 *)
   ["','how_pay',",:,",金額|'?']['/'金種,"]|   (* 注文する代金（指値），任意 *)
   ["','option',",:,JSON].                     (* 特記事項，任意 *)
```

who は注文の主体となるパーティ，who_pay は請求先で，通常は who と同じなので省略される事が多い。whom_incharge は担当部署（自社の組織）の指名で，通常は使われない。what は商品の識別で，必要に応じて spec を指定する。数および/または量を指定する。

when_to は終了日時で，納期を指定するときに使用する。通常，when_from を指定することはないが，サービスを要求する際には，いつから（when_from）いつまで（when_to）を指定することがある。納期を問い合わせる場合は，when_to に"?"を設定する。納期の問い合せに対しては，Counter メッセージで回答する。

where_from は出荷場所，where_to は届け先であるが，物理的な場所（地番）を指すことも，論理的なパーティを指すこともある。

how_pay は金額と，必要に応じて，現金，振込，クレジット，請求書払いなどの支払い方法（金種と呼ぶ）を"/"に続いて指定する。金額の表現は量の表現に準拠する。金種の表現は，システムの参加者間で別途定める。たとえば，次のようにする。

```
"how_pay":"16800_JPY/cash"
"how_pay":"130.74_EUR/banktransfer"
"how_pay":"151.35_USD/creditcard"
```

Request メッセージに金額を入れる意図は指値である。見積もり金額を問い合わせる場合は，how_pay に"?"を設定する。見積もり金額の問い合せに対しては，P:Counter メッセージで回答する。

option は，注文に関する特記事項欄であり，上記以外に Customer が Performer に伝えたいことを，システムの参加者間で取り決めたうえで，JSON 形式で自由に記述する。たとえば，出荷条件（持ち届けか倉取りか，荷姿に対する注文，分納の可否，輸送車の指定など），請求条件（基本契約に定める請求条件を上書きするなど），商品仕様の追加（代替品の条件など）などに使用する。

C.1.2 Promise メッセージ

Request メッセージのとおりに履行できる場合は，注文識別だけを記載した Promise メッセージを返す。Request メッセージの納期および/または金額の記載がない場合，または空欄であった場合は，それぞれの回答を記載する。特記事項があればそれを記載する。記載可能な注文属性の例は，次のとおりである。

```
o-attr = ",'o-id',",:,",注文識別,"|
   [",'when_to',",:,日時]|                    (* 希望する納期または終了日時，任意 *)
   [",'how_pay',",:,",金額['/'金種],"]|       (* 注文する代金（指値），任意 *)
   [",'option',",:,JSON].                     (* 特記事項，任意 *)
```

C.1.3 P: Counter メッセージ

Request メッセージのすべての注文属性に対して対案を提示できる。P:Counter メッセージには，対案の分だけを記載する。記載可能な注文属性の例を示す。

```
o-attr = ",'o-id',",:,",注文識別,"|
   [",'whom_incharge',",:,",組織,"]           (* 担当部署の対案，任意 *)
   [",'what',",:,",商品識別,"]|               (* 品目またはサービスの対案，任意 *)
   [",'spec',",:,[,{,仕様表現,},]]|           (* 品目またはサービスの仕様の対案，任意 *)
   [",'quantity',",:,数|(",量,")]|            (* 数または量の対案，任意 *)
   [",'when_from',",:,日時]|                  (* 開始日時の対案，任意 *)
   [",'when_to',",:,日時|'?']|                (* 納期または終了日時の対案，任意 *)
   [",'where_from',",:,",場所|パーティ,"]|    (* 発地の対案，任意 *)
   [",'where_to',",:,",場所|パーティ,"]|      (* 着地の対案，任意 *)
   [",'how_pay',",:,",金額|'?']['/'金種,"]|   (* 代金の対案，任意 *)
   [",'option',",:,JSON].                     (* 特記事項，任意 *)
```

C.1.4 C: Counter メッセージ

C:Counter メッセージは，かつて Performer が提示した対案（P:Counter）に対する逆対案を提示する。よって，P:Counter の記載属性のうち対案のあるもののみを記載する。記載可能な注文属性は，P:Counter と同じである。

C.1.5 ReportCompletion メッセージ

ReportCompletion メッセージは，以前に Promise した内容に対する実績の報告である。実績として変わりうるのは商品とその仕様，数量，納期である。よって，本メッセージに記載可能な注文属性の例は，次のとおりである。

```
o-attr = ",'o-id',",:,",注文識別,"|
   ",'what',",:,",商品識別,"|              (* 実績の品目またはサービス *)
   ",'quantity',",:,数|(",量,")|            (* 実績の数/量 *)
   [",'whom_incharge',",:,",組織,"]         (* 担当部署の実績，任意 *)
   [",'spec',",:,[,{,仕様表現,},]]|         (* 実績の品目またはサービスの仕様，任意 *)
   [",'serialno',",:,[,{,個体識別,},]]|     (* 実績の個品の識別，任意 *)
   [",'when_to',",:,日時|'?']|              (* 実績の日時，任意 *)
   [",'where_from',",:,",場所|パーティ,"]|  (* 実績の発地，任意 *)
   [",'where_to',",:,",場所|パーティ,"]|    (* 実績の着地，任意 *)
   [",'option',",:,JSON].                   (* 特記事項，任意 *)
```

serialno は，商品を個品管理する場合，実績の個体識別またはロット番号をリスト形式で記載する。それ以外は Request で説明したとおりである。

商品識別が予定と実績で異なることは現実には考えにくいが，ある仕様が異なると別商品と見なすなどの商品の識別ルールに依存する。仕様が異なる，数/量が異なる，納期や金額が異なる場合は，報告した上で Customer の評価を待つ。

C.1.6 Cancel, Decline, DeclineReport, DeclareComplete メッセージ

Cancel, Decline, DeclineReport, DeclareComplete は注文識別のみを記載する。

C.2 Make 注文および Buy 注文

C.2.1 Request メッセージ

注文語 Make（作れ：生産を計画せよ）および Buy（買え：購入を計画せよ）の Request メッセージに記載する注文属性の例を示す。

```
o-attr = ",'o-id',",:,",注文識別,"|
   ",'what',",:,",商品識別,"|              (* 注文する品目またはサービス *)
   ",'quantity',",:,数|(",量,")|            (* 注文する数または量 *)
   [",'who',",:,",組織,"]|                  (* Buy/Make 注文を発行する自社組織，任意 *)
   [",'spec',",:,[,{,仕様表現,},]]|         (* 注文する品目またはサービスの仕様，任意 *)
   [",'when_from',",:,日時]|                (* 希望する開始日時，任意 *)
   [",'when_to',",:,日時]|                  (* 希望する納期または終了日時，任意 *)
   [",'where_from',",:,",場所,"]|           (* 発地，where_to とは別の値，任意 *)
   [",'where_to',",:,",場所,"]|             (* 着地，where_from とは別の値，任意 *)
```

```
   [",'option',",:,JSON].                    (* 特記事項，任意 *)
```

各注文属性の意味は C.1.1 Sell 注文の Request メッセージと同じ。

C.2.2 Promise メッセージ

Request メッセージのとおりに履行できるときは，注文識別だけを記載した Promise メッセージを返す。

C.2.3 P: Counter メッセージ

Request のすべての注文属性に対して対案を提示できる。メッセージには，対案のある分だけを記載する。記載可能な注文属性の例は，次のとおりである。

```
o-attr = ",'o-id',",:,",注文識別,"|
   [",'what',",:,",商品識別,"]|              (* 品目またはサービスの対案，任意 *)
   [",'spec',",:,[,{,仕様表現,},]]|          (* 品目またはサービスの仕様の対案，任意 *)
   [",'quantity',",:,数|(",量,")]|           (* 数または量の対案，任意 *)
   [",'when_to',",:,日時|'?']|               (* 納期または終了日時の対案，任意 *)
   [",'option',",:,JSON].                    (* 特記事項，任意 *)
```

C.2.4 C: Counter メッセージ

C:Counter メッセージは，かつて Performer が提示した対案（P:Counter）に対する逆対案を提示する。よって，P:Counter の記載属性のうち対案のあるもののみを記載する。記載可能な注文属性は，P:Counter と同じである。

C.2.5 ReportCompletion メッセージ

ReportCompletion メッセージは，上で Promise した内容に対する実績の報告である。実績として変わりうるのは品目の仕様，数/量，納期である。よって，本メッセージに記載可能な注文属性の例は，次のとおりである。

```
o-attr = ",'o-id',",:,",注文識別,"|
   ",'what',",:,",商品識別,"|               (* 実績の品目またはサービス *)
   ",'quantity',",:,数|(",量,")|            (* 実績の数/量 *)
   [",'whom_incharge',",:,",組織,"]         (* 担当部署の実績，任意 *)
   [",'spec',",:,[,{,仕様表現,},]]|         (* 実績の品目またはサービスの仕様，任意 *)
   [",'serialno',",:,[,{,個体識別,},]]|     (* 実績の個品の識別，任意 *)
   [",'when_to',",:,日時|'?']|               (* 実績の終了日時，任意 *)
   [",'where_from',",:,",場所|パーティ,"]|  (* 実績の発地，任意 *)
   [",'where_to',",:,",場所|パーティ,"]|    (* 実績の着地，任意 *)
   [",'option',",:,JSON].                    (* 特記事項，任意 *)
```

各注文属性の意味は，serialno を除いて C.1.5 Sell 注文の ReportCompletion メッセージと同じ。serialno は実績の個体識別またはロット番号をリスト形式で記載する。

C.2.6 Cancel, Decline, DeclineReport, DeclareComplete メッセージ

Cancel, Decline, DeclineReport, DeclareComplete は注文識別のみを記載する。

C.3 Ship/Store 注文

C.3.1 Request メッセージ

注文語 Ship（出庫せよ）および Store（入庫せよ）の Request メッセージに記載する注文属性の例を示す。

```
o-attr = ",'o-id',",:,",注文識別,"|
   ",'what',",:,",品目識別,"|              (* 入出庫する品目 *)
   ",'quantity',",:,数|(",量,")|           (* 入出庫する数または量 *)
   [",'who',",:,",組織,"]|                 (* Ship/Store 注文を発行する自社組織，任意 *)
   [",'whom_incharge',",:,",組織,"]        (* 担当部署の指定，任意 *)
   [",'spec',",:,[,{,仕様表現,},]]|        (* 入出庫する品目の仕様，任意 *)
   [",'serialno',",:,[,{,個体識別,},]]|    (* 入出庫する個品の識別，任意 *)
   [",'when_from',",:,日時]|                (* 出庫日時，任意 *)
   [",'when_to',",:,日時]|                  (* 入庫日時，任意 *)
   [",'where_from',",:,",場所,"]|           (* 送り元，任意 *)
   [",'where_to',",:,",場所,"]|             (* 届け先，任意 *)
   [",'option',",:,JSON].                   (* 特記事項，任意 *)
```

各注文属性の意味は，serialno を除いて C.1.1 Sell 注文の Request メッセージと同じ。serialno は，個体識別またはロット番号をリスト形式で指定する。

C.3.2 Promise メッセージ

Request メッセージのとおりに履行できるときは，注文識別だけを記載した Promise メッセージを返す。

C.3.3 ReportCompletion メッセージ

ReportCompletion メッセージは，上で Promise した内容に対する実績の報告である。実績として変わりうるのは品目の仕様，個品識別，数/量，入出庫日時である。よって，本メッセージに記載可能な注文属性の例は，次のとおりである。

```
o-attr = ",'o-id',",:,",注文識別,"|
   ",'what',",:,",品目識別,"|              (* 入出庫した品目 *)
   ",'quantity',",:,数|(",量,")|           (* 入出庫した数/量 *)
   [",'whom_incharge',",:,",組織,"]        (* 担当部署の実績，任意 *)
   [",'spec',",:,[,{,仕様表現,},]]|        (* 入出庫した品目の仕様，任意 *)
   [",'serialno',",:,[,{,個体識別,},]]|    (* 入出庫した個品の識別，任意 *)
   [",'when_to',",:,日時|'?']|              (* 入出庫した日時，任意 *)
   [",'where_from',",:,",場所|パーティ,"]| (* 実績の発地，任意 *)
   [",'where_to',",:,",場所|パーティ,"]|   (* 実績の着地，任意 *)
   [",'option',",:,JSON].                   (* 特記事項，任意 *)
```

各注文属性の意味は，serialno を除いて C.1.5 Sell 注文の ReportCompletion メッセージと同じ。serialno は実績の個体識別またはロット番号をリスト形式で記載する。

C.3.6 Cancel, Decline, DeclineReport, DeclareComplete メッセージ

Cancel, Decline, DeclineReport, DeclareComplete は注文識別のみを記載する。

C.4 DoMake 注文

C.4.1 Request メッセージ

注文語 DoMake（製造を実施せよ）の Request メッセージに記載する注文属性の例を示す。

```
o-attr = ",'o-id',",:,",注文識別,"|
   ",'what',",:,",品目識別,"|              (* 製造実施する品目 *)
   ",'quantity',",:,数|(",量,")|           (* 製造実施する数または量 *)
   [",'who',",:,",組織,"]|                 (* DoMake 注文を発行する自社組織，任意 *)
   [",'whom_incharge',",:,",組織,"]        (* 作業場（shop）の指定，任意 *)
   [",'spec',",:,[,{,仕様表現,},]]|        (* 製造実施する品目の仕様，任意 *)
   [",'serialno',",:,[,{,個体識別,},]]|    (* 製造実施する個品の識別，任意 *)
   [",'when_from',",:,日時]|               (* 製造開始日時，任意 *)
   [",'when_to',",:,日時]|                 (* 製造終了日時，任意 *)
   [",'detals',",:,JSON]|                  (* 製造指示の詳細，任意 *)
   [",'option',",:,JSON].                  (* 特記事項，任意 *)
```

各注文属性の意味は，serialno および details を除いて C.1.1 Sell 注文の Request メッセージと同じ。serialno は，個体識別またはロット番号をリスト形式で指定する。details は製造指示の詳細，すなわち投入品，使用設備，副産物を JSON 形式で指示する。

C.4.2 Promise メッセージ

Request メッセージのとおりに履行することとし，注文識別だけを記載した Promise メッセージを返す。

C.4.3 ReportCompletion メッセージ

ReportCompletion メッセージは，上で Promise した内容に対する実績の報告である。実績として変わりうるのは品目とその仕様，数/量，製造開始日時，終了日時である。よって，本メッセージに記載可能な注文属性の例は，次のとおりである。

```
o-attr = ",'o-id',",:,",注文識別,"|
   ",'what',",:,",品目識別,"|              (* 製造実施した品目 *)
   ",'quantity',",:,数|(",量,")|           (* 製造実施した数または量 *)
   [",'whom_incharge',",:,",組織,"]        (* 作業場（shop）の実績，任意 *)
   [",'spec',",:,[,{,仕様表現,},]]|        (* 製造実施した品目の仕様，任意 *)
   [",'serialno',",:,[,{,個体識別,},]]|    (* 製造実施した個品の識別，任意 *)
   [",'when_from',",:,日時]|               (* 製造開始した日時，任意 *)
   [",'when_to',",:,日時]|                 (* 製造終了した日時，任意 *)
   [",'detals',",:,JSON]|                  (* 製造実績の詳細，任意 *)
   [",'option',",:,JSON].                  (* 特記事項，任意 *)
```


各注文属性の意味は，serialno および details を除いて C.1.5 Sell 注文の ReportCompletion メッセージと同じ。serialno は実績の個体識別またはロット番号をリスト形式で記載する。details は製造実績の詳細，すなわち投入品実績，使用設備の実績，副産物の実績を JSON 形式で報告する。

C.4 Control 注文

C.4.1 Request メッセージ

注文語 Control（制御せよ）の Request メッセージに記載する注文属性の例を示す。

```
o-attr = ",'o-id',",:,",注文識別,"|
   ",'target',",:,",制御対象,"|          (* 制御対象の識別 *)
   ",'what',",:,",制御動作,"|            (* 制御動作の名称 *)
   [",'detals',",:,JSON]|                (* 制御動作に対する指示の詳細，任意 *)
   [",'when_from',",:,日時]|             (* 制御開始日時，任意 *)
   [",'when_to',",:,日時].               (* 制御終了日時，任意 *)
```

target は制御対象の識別，what はあらかじめ制御機器に用意されている制御動作名，details は制御指示の詳細を JSON 形式で指定する。when_from，when_to は制御開始および終了時刻を指定する。これが省略されているときは，即時実行を意味する。

C.4.2 Promise メッセージ

Request メッセージのとおりに履行することとし，注文識別だけを記載した Promise メッセージを返す。

C.4.3 ReportCompletion メッセージ

ReportCompletion メッセージは，上で Promise した内容に対する実績の報告である。実績として記載可能な注文属性は，次のとおりである。

```
o-attr = ",'o-id',",:,",注文識別,"|
   ",'result',",:,真理値|                (* OK/NG *)
   [",'when_from',",:,日時]|             (* 制御開始実績日時，任意 *)
   [",'when_to',",:,日時]|               (* 制御終了実績日時，任意 *)
   [",'observations',",:,JSON].          (* 観測データ，任意 *)
```

result は制御結果の評価で，OK か NG かの二値で表現する。when_from，when_to は実績の制御開始および終了時刻，observations は制御動作時に観測されたセンサデータを任意の形式で報告する。

C.5 Bill/Pay 注文

C.5.1 Request メッセージ

注文語 Bill（請求せよ）および Pay（支払え）の Request メッセージに記載する注文属性の例を示す。

```
o-attr = ",'o-id',",:,",注文識別,"|
   [",'who',",:,",申請者,"]|             (* 申請者の識別，自明なことが多い *)
```

```
",'who_pay',",:,",請求先|支払先,"|        (* 請求先/支払先の識別 *)
[",'where_to',",:,",口座情報,"]|          (* 入金/支払先口座の情報 *)
[",'when_at|_by',",:,日時]|               (* 決済日|決済期限，どちらか必須 *)
",'how_pay',",:,",方法と金額,"|           (* 金種別金額のリスト *)
[",'detals',",:,",内訳,"].                (* 金額の内訳リスト，任意 *)
```

who はこの注文の発行者で，Bill 注文の場合は営業部署，Pay 注文の場合は購入部署となる。who_pay は Bill 注文の場合は請求先，Pay 注文の場合は支払先のパーティである。how_pay は合計の請求金額または支払金額だが，その実績は，現金，振込，クレジット，ポイントなど方法が多様であり，通貨も混ざっていることもある。金種のエンコードは当該システムの参加者の合意で個別に定める。details は金額の内訳で，記載は任意である。

C.5.2 Promise メッセージ

Request メッセージのとおりに履行することとし，注文識別だけを記載した Promise メッセージを返す。

C.5.3 ReportCompletion メッセージ

ReportCompletion メッセージは，上で Promise した内容に対する実績の報告である。実績として記載可能な注文属性の例は，次のとおりである。

```
o-attr = ",'o-id',",:,",注文識別,"|
   ",'when_at',",:,日時|                  (* 入金/支払日 *)
   ",'how_paied',",:,",方法と金額,"|      (* 入金実績の金種別金額のリスト *)
   [",'details',",:,",内訳,"].            (* 予定と異なるとき，金額の内訳リスト*)
```

予定と実績が異なる場合は，すべて当メッセージで報告する。

Annex D (informative) 階層間連携メッセージの例

第 7 章で定めた個別のメッセージの，簡単なシナリオに基づく一部の例を挙げる。C-Subsystem の識別を「C」，P-Subsystem の識別を「P」と表記する。

注文の識別は〝#〞から始まる 5 桁の連番，品目は「箱」で，その仕様は W, H, D の仕様名に仕様値を指定することで特定されると仮定する。

D.1 制御メッセージ

D.1.1 「取りに来い」メッセージの例

```
{"controlVerb":"Notify",
   "sender":"C",
   "receiver":"P",
   "time-stamp":"2015-10-10,10:15:30",
   "message-id":"3781209846309",
   "language":"iHCl",
   "content":""
}
```

D.1.2 「応答を乞う」メッセージの例

```
{"controlVerb":"RSVP",
   "sender":"P",
   "receiver":"C",
   "time-stamp":"2015-10-10,10:15:45",
   "message-id":"3781209848823",
   "language":"iHCl",
   "content":"select(*),where(time-stamp>"2015-10-10,09:12:28")"
}
```

D.2 業務メッセージ

D.2.1 「注文提示」メッセージの例

```
{"performtive":"Request",
   "sender":"C",
   "receiver":"P",
   "time-stamp":"2015-10-10,10:15:45",
   "message-id":"37812309854398",
   "language":"iHCl",
   "content":[
      {"order":"Buy"},[
         {"o-id":"#00004","what":"箱","spec":{"W":100,"H":20,"D":30},
           "quantity":10,"when_to":"2015-10-20"},
         {"o-id":"#00005","what":"箱","spec":{"W":110,"H":30,"D":30},
           "quantity":15,"when_to":"2015-10-20"}
         ],
      {"number-of-records":2,"check-sum":13}
    ]
}
```

D.2.2 「約束」メッセージの例

2 件の注文とも要求を満たせることを確認し，「約束」メッセージを返して，製造に取りかかる。

```
{"performtive":"Promise",
   "sender":"P",
   "receiver":"C",
   "time-stamp":"2015-10-10,10:20:03",
   "message-id":"37812311059863",
   "language":"iHCl",
   "content":[
      {"order":"Buy"},[
         {"o-id":"#00004"},
         {"o-id":"#00005"}
      ]
   ]
}
```

D.2.3 「終了報告」メッセージの例

2 件の注文のうち 1 件のみ製造が終了したものとする。

```
{"performtive":"ReportCompletion",
   "sender":"P",
   "receiver":"C",
   "time-stamp":"2015-10-13,14:16:54",
   "message-id":"37812312234458",
   "language":"iHCl",
   "content":[
      {"order":"Buy"},[
         {"o-id":"#00004","what":"箱","spec":{"W":100,"H":20,"D":30},
           "quantity":10,"when_to":"2015-10-13"}
      ]
   ]
}
```

D.2.4 「注文完了宣言」メッセージの例

2 件の注文のうち 1 件のみ注文の終了を宣言する。

```
{"performtive":"DeclareComplete",
   "sender":"C",
   "receiver":"P",
   "time-stamp":"2015-10-13,14:20:32",
   "message-id":"378123123487633",
   "language":"iHCl",
   "content":[
      {"order":"Buy"},[
         {"o-id":"#00004"}
      ]
   ]
}
```


Annex E (informative)　対話構造の拡張

E.1　対話構造の日本向け拡張

Figure E.1 は，Figure 2 を日本のビジネスプロトコルに対応できるように拡張したものである。C: ChangeRequest, C:Confirm, および P:ChangeRequest が拡張部分である。拡張部分について説明する。

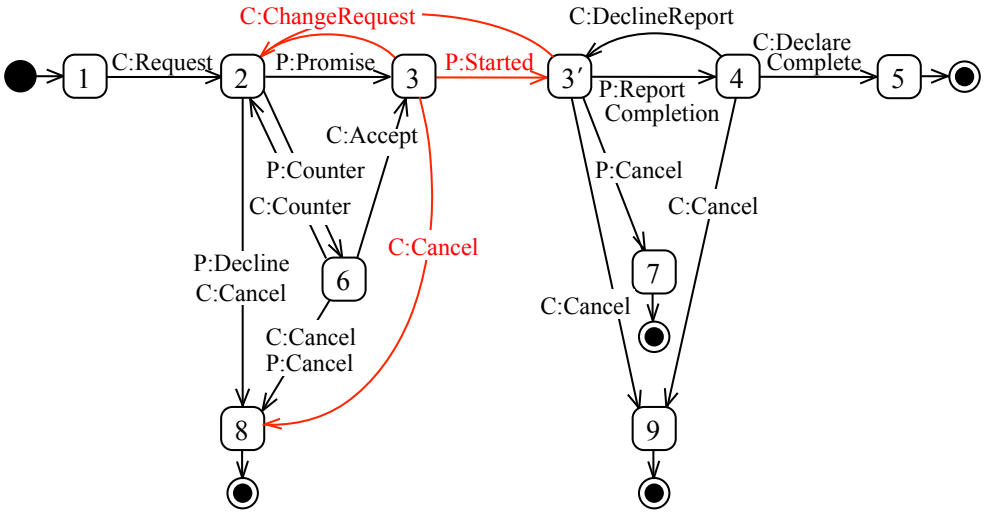

Figure E.1　「発注者-受注者間で起こりうる対話構造」の拡張

状態 3 から状態 2 への逆流の遷移は，日本で特有のビジネスプロトコルを反映している。すなわち，一旦注文に対する約束が確立されたにもかかわらず，発注者が「注文変更」する（C: Change-Request）ことを許す場合がある。

状態 3 から状態 3′への遷移は，注文に対する「約束を確定する」処置（C: Confirm）である。これは，最初の「約束」が日本では「内示」と呼ばれる注文と約束の仮の状態があることを意味する。

状態 3 から状態 3′へ遷移するとは，時間的な期限または変更回数などによる期限切れを契機として，両者とは独立の契約条件などに基づき，強制的に「約束」が確定することである。

状態 3′から状態 6 への遷移は，機器の故障や部品の不良などの不測の事態により，受注者が一旦約束した注文の内容が，着手前に達成できないことが判明した場合の「受注者による注文変更」処置である。この遷移では，納期や数量（場合によっては代替品へ）の変更を発注者に対して要求することを表す。P: Counter ではなく，P: ChangeRequest である理由は，受注者が一度約束した内容に対する変更要求だからである。なお，機器故障による保守作業の計画と作業展開，当該作業のリスケジュール，および資源の引当は，その資源を保有する管理階層内で完結させることになる。

E.2　拡張業務メッセージ

上記の対話構造の拡張に対して，拡張業務メッセージの構文を定義する。

遂行動詞を 2 つ追加する。

```
遂行動詞 = 'ChangeRequest'|'Started'
```

E.2.1 「注文変更（ChangeRequest）」メッセージの構文

発注者側の都合により，一度「約束」された注文の内容を変更せざるを得なくなることもある。C-Subsystem は注文の納期変更，数変更，代替可能な製品への変更など，注文内容の変更を申し出ることができる。この場合 C-Subsystem は「注文変更（C: ChangeRequest）」メッセージを P-Subsystem に送る（Figure E.1 の状態 3 から 2 への遷移）。

構文は次のとおりとする。message-content は変更した項目だけを記載する。

```
遂行動詞 = 'ChangeRequest'.
注文内容 = {,o-attr,{comma,o-attr},}.
o-attr =                (* 要素は重複しないこと *)
   ",'o-id',",:,",注文識別,"|
   [",'who',",:,",パーティ|組織,"]|          (* 注文主（Customer） *)
   [",'who_pay',",:,",パーティ|組織,"]|      (* 請求先，注文主と異なる場合のみ，任意 *)
   [",'whom_incharge',",:,",組織,"]          (* 担当部署または実施工程指定，任意 *)
   [",'what',",:,",品目識別,"]|              (* 注文するものの識別 *)
   [",'spec',",:,[,{,仕様表現,},]]|          (* 注文するものの仕様，任意 *)
   [",'contract',",:,",契約識別,"]|          (* この注文が依拠する契約の識別，任意 *)
   [",'quantity',",:,数|(",量,")]|           (* 注文する数または量 *)
   [",'how_pay',",:,",金額|'?'][/'金種,"]|  (* 注文する代金（指値），任意 *)
   [",'when_from',",:,日時]|                 (* 希望する着手開始日時，任意 *)
   [",'when_to',",:,日時|'?']|               (* 希望する納期または終了日時，任意 *)
   [",'where_from',",:,",場所|パーティ,"]|   (* 発地，where_to とは別，任意 *)
   [",'where_to',",:,",場所|パーティ,"]|     (* 着地，where_from とは別，任意 *)
   [",'option',",:,JSON].                  (* 特記事項，任意 *)
```

P-Subsystem はこのメッセージを受けて，注文変更を受け入れるか「対案提示」するか「取り下げ」るか判断する。これに対して，一般的には受注者側の営業判断が必要となる。

E.2.2 「受注者による注文変更（ChangeRequest）」メッセージの構文

受注者側の都合（装置の故障など）により，一度「約束」した注文の内容を変更せざるを得なくなることがある。「受注者中止」にして必ず再注文してもらうことができればいいが，それが叶わなくて注文を維持したい場合，P-Subsystem は「受注者による注文変更（P: ChangeRequest）」メッセージを C-Subsystem に送る（状態 3 から 3'への遷移）。

構文は次のとおり。変更を提案する主体が P-Subsystem である点（メッセージヘッダの sender が P-subsystem になる）を除いて，E.2.1 と同じ内容になる。

C-Subsystem はこのメッセージを受けて，注文変更を受け入れるか「対案提示」するか「取り下げ」るかを判断する。

E.2.3 「着手した（P:Started）」メッセージの構文

P-Subsystem は C-Subsystem に対して，着手したことを通知することで今後の変更には違約金が発生することを暗示する。

構文は次のとおり。


```
遂行動詞 = 'Started'.
注文内容 = ",'o-id',",:,",注文識別,".    (* 確定する注文の識別，必須 *)
```

E.3　注文なし実績

先行着手などの目的により，発注者からの明示的な注文のプロセスがないにも関わらず，実質的にある行為が実行されて完了（P: ReportCompletion）することがある。

この場合は，内部の発注者を明示した注文に基づく対話構造があるとみなす。たとえば，生産管理システムにおける不良品の廃棄処理，保守点検の注文などでは，工場長など，その業務の責任者を内部の発注者とする「注文」が存在するものとして，状態 3 から遷移する。

E.4　拡張業務メッセージの例

E.4.1　「受注者による注文変更」メッセージの例

2 件の注文のうち 1 件のみ納期を変更したい旨を申し出る。

```
{"performtive":"ChangeRequest",
   "sender":"P",
   "receiver":"C",
   "time-stamp":"2015-10-10,11:22:39",
   "message-id":"37812311063441",
   "language":"iHCl",
   "content":[
      {"order":"Buy"},[
         {"o-id":"#00004","when_to":"2015-10-22"}
      ]
   ]
}
```

Annex F (informative)　構文の拡張

F.1　即値の拡張

比較演算において，即値の表現に，正規表現を用いることができる。

即値の文字列中にある”?”は，その位置にある一つの文字をマスクする，すなわちどんな文字があっても真とみなすことを意味する。

”*”は，同様に 0 個以上の文字をマスクして可変長の判定をすることを意味する。

また，連続値の即値を範囲指定したい場合は，“最小値..最大値”の形式で記述できる。

F.2　仕様値表現の拡張

F.2.1　曖昧な仕様値の表現

仕様値の表現には，幅や候補値をもたせた指定，「未定」の指定も可能とする。業務的には，曖昧な仕様であっても，受注確定前あるいは製造指示発行前に，必要な資源の仮引当をしておくなどのために利用できる。この場合は，仕様表現を次のように拡張する。

```
value = 即値|
      (即値,"..",即値)|                    (* 幅で指定するときは，左側が小さい値 *)
      (即値,"|",value).                    (* 代替仕様の表現 *)
即値 = 数値|"未定"|"".                     (* 未定の仕様値を許す *)
```

仕様値の幅は，たとえば，品目が「箱」で，その仕様が仕様名「W」,「H」,「D」で表現されるとすると，「W」の仕様値が 100mm から 110mm の間で，「H」が 20mm か 25mm から 30mm の間，「D」が未定の仕様表現は次のように書ける。

```
{"what":"箱","spec":{"W":"100..110_mm","H":"20|25..30_mm","D":"未定"}}
```

F.2.2　代替可能性の表現

仕様値に，代替仕様を指定する方法を提供する。これは，製品というより清掃過程で用いる部品の要求において，指定した仕様値に該当する部品がない場合，品質が保証される範囲で，互換の部品を引き当てるなどに用いられる。この場合は，仕様表現を次のように拡張する。

```
value = 即値,[,代替仕様,].
代替仕様 = 即値.                           (* 未定の仕様値を許す *)
```

上記の例で，「W」の仕様値が第 1 候補としては 100mm を引き当てようとするが，それが在庫として存在しない場合は 110mm を代替することを指定する仕様表現は次のように書ける。それ以外の仕様値は上記の例のままとする。

```
{"what":"箱","spec":{"W":"100_mm[110_mm]","H":"20|25..30_mm","D":"未定"}}
```

Annex G (informative) サブシステム間の通信チャネルの実装

G.1 メッセージングのシステム要素

MSU を改変して，直接 iHCl に基づく対話ができるようにしてもよいが，より簡単に iHCl を使えるように通信アダプタを利用することが現実的であろう。各 MSU は，この通信アダプタの関数を呼ぶように最小限のコードを改変するだけですむ。以下では，その通信アダプタの機能仕様について述べる。

通信アダプタを用いる場合の最小構成は，Figure G.1 に示すように，MSU のアダプタ（Adapter），通信チャネル（pull, push, notify, callback）の 2 つのシステム要素と 2 つの定義体（ケイパビリティプロファイルと環境定義ファイル）からなる。

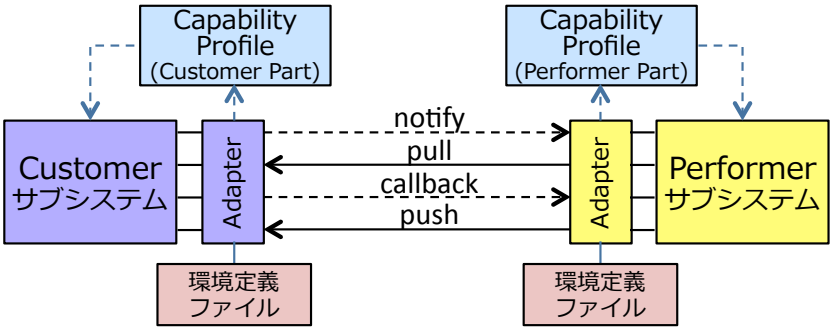

Figure G.1 メッセージングのシステム要素

G.1.1 アダプタ（Adapter）

アダプタは，物理的なネットワークを抽象化してメッセージの送受信処理を単純化する。アダプタは，サブシステムから受け取ったメッセージの送り先を認識して，そのサブシステムに送信する。その際，受け取ったメッセージが妥当な通信相手であることを確認する。アダプタは C-Subsystem に対して 2 つの通信チャネル（pull, push）を，P-Subsystem に対して 2 つの通信チャネル（notify, callback）をもつ。

G.1.2 通信チャネル（Communication Channel）

通信チャネルは，サブシステム間の通信インタフェースにおけるメッセージの経路である。

通信チャネルには，通信相手のサブシステムごとに，P-Subsystem から C-Subsystem に向かう上りメッセージのための 2 つのチャネル（pull, push）と，下りメッセージ（notify）のための 1 つのチャネル，pull に対する返答の送信先（callback）である 1 つのチャネル，合計 4 つのチャネルがある。

Performer のアダプタは通信を開始する前に，Customer の Capability Profile にある pull および push チャネルの情報を，自分の環境（定義ファイルなど）に記録することで通信先を設定する。

対話は常に P-Subsystem 側から起動される。例外は「取りに来い（Notify）」制御メッセージである。notify は，C-Subsystem が P-Subsystem に対して送信したいメッセージがあることを通知する特殊なメッセージである。notify チャネルはそれだけを流す。

notify メッセージを用いなくても C-Subsystem と P-Subsystem 間の対話は可能である。notify メッセージを用いない場合，P-Subsystem は一定の時間間隔で RSVP 制御メッセージを C-Subsystem に送って，メッセージの有無を確かめる（polling）ことになる。この時間間隔が短いと，C-Subsystem 側に送るべきメッセージがないこと（空振り）が多くなり，むだな通信と空振りの処理が増える。時間間隔が長いと，C-Subsystem 側に送るべきメッセージがたまっていることが多くなり，もっと早くに処理できたものが不当に遅延してしまう。notify メッセージを用いることで，空振りがなくなり，処理の遅延もなくなる。notify メッセージは通知だけなので，密結合にはならない。

通信チャネルは，REST や ESB などで実装されると想定する。

G.1.2.1 pull チャネル

pull チャネルは，C-Subsystem が P-Subsystem から下りの業務メッセージの要求を受け取るためのチャネルである。たとえば，P-Subsystem は C-Subsystem に対して「応答を乞う（P:RSVP）」制御メッセージをこの pull チャネルに対して送り，C-Subsystem はその応答として C-Subsystem に対して「注文提示（C:Request）」などの下りの業務メッセージを callback チャネルに送る。

下りの業務メッセージには，「注文提示（C:Request）」のほかに，「対案受入（C:Accept）」「逆提案（C:Counter）」「取り下げ（C:Cancel）」「発注者中止（C:Cancel）」「作業結果の拒否（C:DeclineReport）」「注文完了宣言（C:DeclareComplete）」がある。これらも上と同様に行う。

なお，拡張された対話構造における「発注者変更（C:ChangeRequest）」業務メッセージも同様である。

G.1.2.2 push チャネル

push チャネルは，P-Subsystem が C-Subsystem に対して上りの業務メッセージを送るチャネルである。上りの業務メッセージの中には，「対案提示（P:Counter）」「約束（P:Promise）」「実施結果の報告（P:ReportCompletion）」「辞退（Decline）」「中止（P: Cancel）」メッセージがある。

拡張された対話構造における上りの業務メッセージ「注文変更の申し出（P:ChangeRequest）」も同様である。

G.1.2.3 notify チャネル

notify チャネルは P-Subsystem が C-Subsystem から notify 制御メッセージを受け取るためのチャネルである。このチャネルは notify 制御メッセージしか受け取らない。

G.2 通信チャネルの確立

P-Subsystem は，自分が処理すべき注文を送ってくれる MSU はどれかを，システム全体が稼働する前に知っている必要がある。相互に通信すべき相手を特定し，そのケイパビリティプロファイルを基に，通信チャネルの定義をアダプタの環境定義ファイルなどに書き込む。通信アダプタは，その起動時に環境定義ファイルに基づいて対話を開始する。

当該生産システムに参加するサブシステムの構成が変化した場合は，アダプタを再立ち上げして対応する。

このイメージを Figure G.2 に示す。

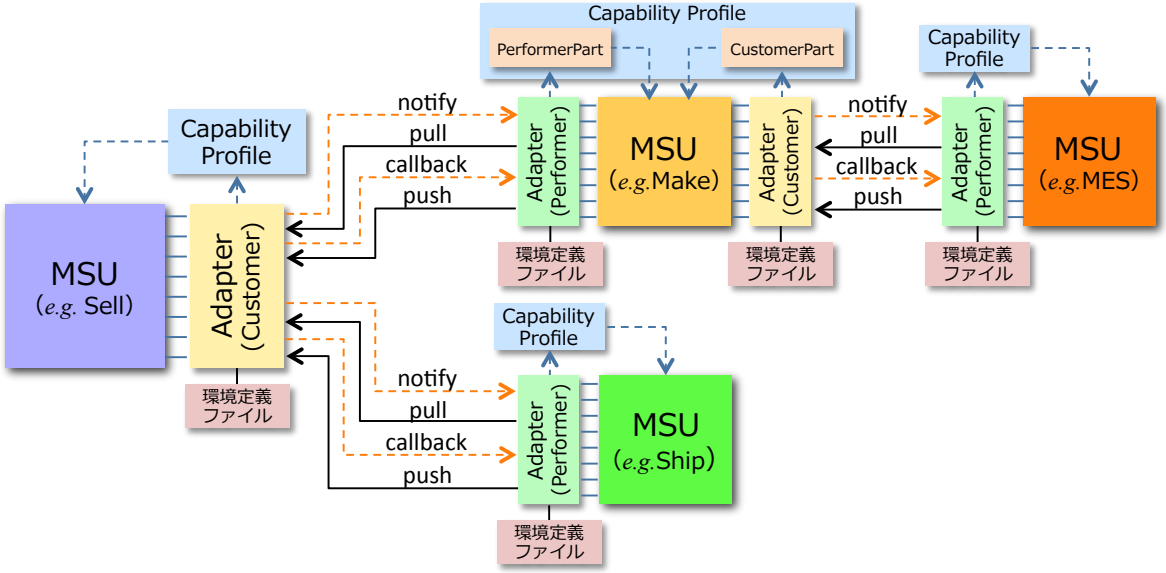

Figure G.2 対話チャネルとケイパビリティプロファイル

G.3　ケイパビリティプロファイル

MSU 自身のケイパビリティを記述するために，ケイパビリティプロファイルを用いる。P-Subsystem は通信相手の C-Subsystem を特定するために，ケイパビリティプロファイルを用いる。

C-Subsystem のケイパビリティプロファイルには，pull/push のチャネルに関する記述が必要である。P-Subsystem のケイパビリティプロファイルには，C-Subsystem からの Notify 制御メッセージを受ける場合は，Notify チャネルの定義が必要である。対話の確立後にケイパビリティプロファイルは参照されない。

こうした情報は，ケイパビリティプロファイルのテンプレート（ISO 16100-3: 2005）に準拠して，Specific パートで記述される。その説明と記述例を下に示す。

G.3.1　Specific パートの説明

Specific パートに追加した XML タグは<Activity>である。この中でアダプタが必要とする情報を記述する。

<Activity>は，当該 MSU がケイパビリティを提供する生産活動の名称を表す。以下では，その活動名を Figure A.1「生産管理システムの全体構造」のドメインに対応させる。このサブタグの<InformationExchange>は，一連のサブサブタグ（<BasicProtocol>や<Channel>）によって pull と push および notify と callback の通信チャネルを定義する。

<Message>では，当該 MSU が送信予定（<Customer>）および受入可能（<Performer>）の遂行動詞を定義し，それぞれのメッセージの項目（<Data>）の型情報を与える。

G.3.2 ケイパビリティプロファイルの例

次の例は，Sell 機能を持つ MSU のケイパビリティのうち，下位機能の MSU との対話のための通信チャネルの定義例を示している。下線部は，実装環境に合わせて設定する。

```
<?xml version="1.0" encoding="UTF-8"?>
<CapabilityProfiling xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Type id="iHCl Capability Profile"/>
  <CapabilityProfile date="2017-08-10">
    <Common>
      <MSU Capability ID="Order Management" />
      <Owner>
        <name>APSOM MESX-JP</name>
        <country>Japan</country>
      </Owner>
    </Common>
    <Specific>
      <Activity id="Sell" name="販売せよ">
        <InformationExchange>
          <ApplicationProtocol id="Customer-Performer" />
          <BasicProtocol id="REST" />
          <Part type="Customer" >
            <Channel type="pull" address="http://pull のアドレス"/>
              <Method type="POST" name="..." />
            </Channel>
            <Channel type="push" address="http://push のアドレス"/>
              <Method type="POST" name="..." />
            </Channel>
            <Message name="Request">
              <Data name="o-id" type="Identifier"/>  <!-- 注文識別 -->
              <Data name="who" type="Party"/>        <!-- 注文主 -->
              <Data name="what" type="Service"/>           <!-- 注文品 -->
              <Data name="spec" type="JSON"/>        <!-- 注文仕様 -->
              <Data name="qtty" type="Numeric" unit="Kg"/>    <!-- 量 -->
              <Data name="when_to" type="DateTime"/> <!-- 納期 -->
              <Data name="where_to" type="Address"/> <!-- 送付先 -->
              <Data name="option" type="JSON"/>            <!-- 追加情報 -->
            </Message>
            <Message name="DeclineReport" />
            <Message name="DeclareComplete" />




          </Part>
        </InformationExchange>
      </Activity>
    </Specific>
  </CapabilityProfile>
</CapabilityProfiling>
```

次の例は，Buy/Make 機能を持つ MSU のケイパビリティのうち，上位機能（Sell を想定）の MSU および下位機能（DoMake を想定）との対話のための通信チャネルの定義例を示している。

```
<?xml version="1.0" encoding="UTF-8"?>
<CapabilityProfiling xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
```



```
<Type id="iHCl Capability Profile"/>
<CapabilityProfile date="2017-08-10">
   <Common>
      <MSU Capability ID="Production Management System" />
      <Owner>
         <name>APSOM MESX-JP</name>
         <country>Japan</country>
      </Owner>
   </Common>
   <Specific>
      <Activity id="Buy/Make" name="購買/製造せよ">
        <InformationExchange>
          <ApplicationProtocol id="Customer-Performer" />
          <BasicProtocol id="REST" />
          <Part id="Performer" />
            <Channel type="notify" address="http://notify のアドレス"/>
              <Method type="POST" name="..." />
            </Channel>
            <Message name="Counter">
              <Data name="o-id" type="Identifier"/>  <!-- 注文識別 -->
              <Data name="who" type="Party"/>        <!-- 注文主 -->
              <Data name="what" type="Service"/>           <!-- 注文品 -->
              <Data name="spec" type="JSON"/>        <!-- 注文仕様 -->
              <Data name="qtty" type="Numeric" unit="Kg"/> <!-- 数(量) -->
              <Data name="when_to" type="DateTime"/> <!-- 納期 -->
              <Data name="where_to" type="Address"/> <!-- 送付先 -->
              <Data name="option" type="JSON"/>            <!-- 追加情報 -->
            </Message>
            <Message name="Promise" />
            <Message name="ReportCompletion" />

          </Part>
          <Part id="Customer" />
            <Channel type="pull" address="http://pull のアドレス">
              <Method type="POST" name="..." />
            </Channel>
            <Message name="Request">
              <Data name="o-id" type="Identifier"/>  <!-- 注文識別 -->
              <Data name="what" type="Service"/>           <!-- 操作名 -->
              <Data name="spec" type="JSON"/>        <!-- 注文仕様 -->
              <Data name="qtty" type="Size"/>        <!-- 量 -->
              <Data name="start_from" type="DateTime"/>    <!-- 開始時刻 -->
              <Data name="end_to" type="DateTime"/>  <!-- 終了時刻 -->
              <Data name="option" type="JSON"/>            <!-- 追加情報 -->
            </Message>

            <Channel type="push" address="http://push のアドレス">
              <Method type="POST" name="..." />
            </Channel>
          </Part>
        </InformationExchange>
      </Activity>
   </Specific>
  </CapabilityProfile>
</CapabilityProfiling>
```

Annex H (informative) メッセージの組立

H.1 識別の発番

H.1.1 識別の発番管理

非同期通信であるものを，業務的に同期させるためにメッセージ識別と注文識別を用いる。これらは，システム全体でユニークであることを保証する。このために，システム全体で識別の発番を管理する独立したシステムを設ける。

メッセージの重複は，メッセージ識別が重複していることをもって検知する。

注文識別は，対話構造のインスタンスを識別するために用いる。

H.1.2 iHCl 変換ライブラリの「注文提示」インタフェース

C-Subsystem は「注文提示」する前に，メッセージ識別および注文識別を知っている必要がある。このために，iHCl 変換ライブラリは「注文提示」メッセージを iHCl 形式に変換するときに，メッセージ識別および注文識別をメッセージに埋め込むと同時に，呼び出し元にもメッセージ識別および注文識別を返す。

H.2 メッセージの一個流しとパッキング

業務メッセージは，一つの注文について発生の都度，送信する。これを一個流しと呼ぶ。ただし，通信効率を考慮して，通信チャネルごとに，送受信するタイミングで複数の注文が滞留しているときは，それらの注文を一つのメッセージにパッキング（封入）してもよい。

メッセージ識別はパッキングの単位で発番し，注文識別は注文ごとに発番する。

Bibliography

[1] 高橋達也ほか，「標準技術の相互活用による工場内情報連携（MESX プロトコルによる製販一体化)」，日本機械学会論文集（C 編），Vol.76, No.772，pp.58-63, 2011
[2] 児玉公信，「企業情報システムのための早期アーキテクティングの一方法」，情報処理学会 情報システムと社会環境研究会，2012/6/4
[3] 児玉公信，「MES（製造実行システム）のモデルに関する一考察」，情報処理学会 情報システムと社会環境研究会，2012/12/3
[4] 児玉公信：企業情報システムのための早期アーキテクティングの一方法，情報処理学会研究会報告，Vol.2012-IS-120 No.3, 2012/6/4
[5] 児玉公信，水野忠則，「少量多品種型生産管理システムの一般モデル CHARM の提案」，情報処理学会論文誌，Vol. 49(2), 902-909, 2008
[6] Medina-Mora, Winograd, Flores, and Flores, "The Action Workflow Approach to Workflow Management Technology," Proc. of CSCW 92, pp1-10, 1992
[7] Winograd, T.: A Language/Action Perspective on the Design of Cooperative Work, HUMAN-COMPUTER INTERACTION, Vol.3, pp.3-30 (1987-1988)
[8] R. Brooks, “A Robust Layered Control System For a Mobile Robot,” IEEE J. of Robotics and Automation, 1986
[9] Yourdon, and Constantine: “Structured design: fundamentals of a discipline of computer program and systems design,” Prentice Hall, 1979
[10] Finin, T. et al: “Specification of the KQML Agent-Communication Language (DRAFT),” http:www.csee.umbc.edu/csee/research/kqml/papers/kqmlspec.ps, 1993
[11] Labrou, Y. and Finin, T.: “A Proposal for a new KQML Specification,” TR CS-97-03, Computer Science and Electrical Engineering Department, University of Maryland, 1997
[12] Page-Jones M., “The Practical Guide to Structured Systems Design,” Prentice Hall, 1988.
[13] MESX-JP，MESX プロトコルブック，2011
[14] IEC PAS 63088 Ed.1, Smart manufacturing – Reference architecture model industry 4.0 (RAMI4.0)