ICS 25.040.40
CCS J07

中华人民共和国国家标准

GB/T XXXXX—XXXX/ISO/TR 16161:2019

自动化系统与集成 制造软件单元间互操作能力专规应用案例

Automation systems and integration-Use case of capability profiles for cooperation between manufacturing software units

(ISO/TR 16161:2019, IDT)

（征求意见稿）

在提交反馈意见时，请将您知道的相关专利连同支持性文件一并附上。

XXXX－XX－XX 发布　　XXXX－XX－XX 实施

国家市场监督管理总局
国家标准化管理委员会　发布

目　　次

前　言

本文件按照GB/T 1.1—2020《标准化工作导则　第1部分：标准化文件的结构和起草规则》的规定起草。

本文件等同采用国际标准ISO/TR 16161:2019 Automation systems and integration-Use case of capability profiles for cooperation between manufacturing software units《自动化系统与集成制造软件单元间互操作能力专规应用案例》。

请注意本文件的某些内容可能涉及专利。本文件的发布机构不承担识别专利的责任。

本文件由中国机械工业联合会提出。

本文件由全国自动化系统与集成标准化技术委员会（SAC/TC159）归口。

本文件起草单位：北京机械工业自动化研究所有限公司，浙江大学等。

本文件主要起草人：

引　言

ISO 16100的动机源于工业和经济环境，尤其是：

a) 供应商专用解决方案的增长基础；

b) 用户在应用标准方面的困难；

c) 需要转型为模块化的系统集成工具集；

d) 认识到应用软件和应用该软件的专业知识是企业的资产。

ISO 16100是一项国际标准，用于表示计算机可理解和人类可读的能力专规。其目标是提供一种独立于特定系统体系结构或实现平台的方法，用于表示制造应用软件中的制造软件单元（MSU）与其在整个制造应用生命周期中的作用相关的能力。这可以减少制造应用的用户和供应商/供货方的生产和信息管理成本。

本文件描述了ISO 16100的应用。制造软件代理是MSU的一种，使用ISO 16100中规定的能力专规实现互操作。

本文件描述了软件代理相互协作以实现系统功能的消息语言和协议。在介绍ISO 16100-3中定义的MSU能力专规时，代理相互识别制造活动的能力和可识别的信息。需要制造活动的软件代理称为消费者，提供制造活动的代理称为实施者。消费者通过消息语言描述制造活动的请求消息。实施者通过消息语言描述制造活动结果的报告消息。

智能物理代理基金会（FIPA）提出的代理通信语言（ACL）是一种与多个代理交换的消息语言，定义消息序列的协议是交互协议的框架，应用于代理并对其进行识别以使用FIPA所规定的本体。相比之下，本文件描述的协议和消息语言中，作为消费者的软件代理和作为实施者的软件代理以一对一的方式交互，并且每个软件代理使用ISO 16100-3中描述的能力专规进行识别。因此，ACL和本文件中描述的协议和消息语言是不同的。

自动化系统与集成 制造软件单元间互操作能力专规应用案例

1 范围

本文件描述了通过交换制造软件单元（MSU）能力专规的方式以实现使用ISO 16100标准进行软件代理之间互操作的方法。代理之间交换的专规描述了请求者请求并将由实施者完成的制造能力。

2 规范性引用文件

本文件没有规范性引用文件。

3 术语和定义

下列术语和定义适用于本文件。

ISO和IEC维护的在标准化中使用的术语数据库为以下网址：

ISO在线浏览平台：https://www.iso.org/obp。

IEC电工词汇：http://www.electropedia.org/。

3.1

消费者 customer

制造活动的请求者。

3.2

C子系统 C-subsystem

请求制造活动的制造软件单元。

3.3

实施者 performer

制造活动的提供者。

3.4

P子系统 P-subsystem

提供制造活动的制造软件单元。

3.5

能力专规服务提供者 capability profile service provider

实现能力专规接口的软件。

[来源：GB/T 19902.3-2006, 3.1.2]

3.6

服务提供者 service provider

实体，扮演**能力专规服务提供者**(3.5)的角色，并负责准备并提供一对**消费者**(3.1)和**实施者**(3.3)。

4 缩略语

下列缩略语适用于本文件。

MSU：制造软件单元（Manufacturing Software Unit）
UML：统一建模语言（Unified Modelling Language）
URI：统一资源标识符（Uniform Resource Identifier）

5 C子系统与P子系统之间的对话

订购者和承包商之间的交互由消费者和实施者之间的对话替代。消费者与实施者之间的对话可由与订单相关的一系列行为来表示。行为顺序可由对话状态转换图表示。在生产系统中，消费者为C子系统，而实施者则为P子系统。

图1显示了C子系统与P子系统之间产生的对话。图1中的“C：请求”是C子系统向P子系统请求的一种操作，“P：接受”是承诺P子系统已接受C子系统请求的一种行为，“P：拒绝”是P子系统拒绝C子系统请求的一种行为。有时P子系统提出一个建议，C子系统接受这个提议。状态2到状态2’的转换显示了这一系列行为。另一方面，C子系统还可对P子系统提供的建议再提出另外一种建议，从状态2’到状态2的转换过程则显示了这一系列行为。

假设订购者和承包商之间的对话可以由图1的状态转换图来表示，且消息是根据此假设来进行设计的。以下是对每种状态的描述：

——状态1：初始状态；
——状态2：下订单；
——状态3：接受订单；
——状态4：完成订单；
——状态5：最终状态；
——状态2’：提供建议；
——状态3’，2’’，3’’：最终状态。

附录E给出了对话状态转换图的定制示例。

附录F提出了用于处理对话状态转换、消息和能力专规的解决方案。

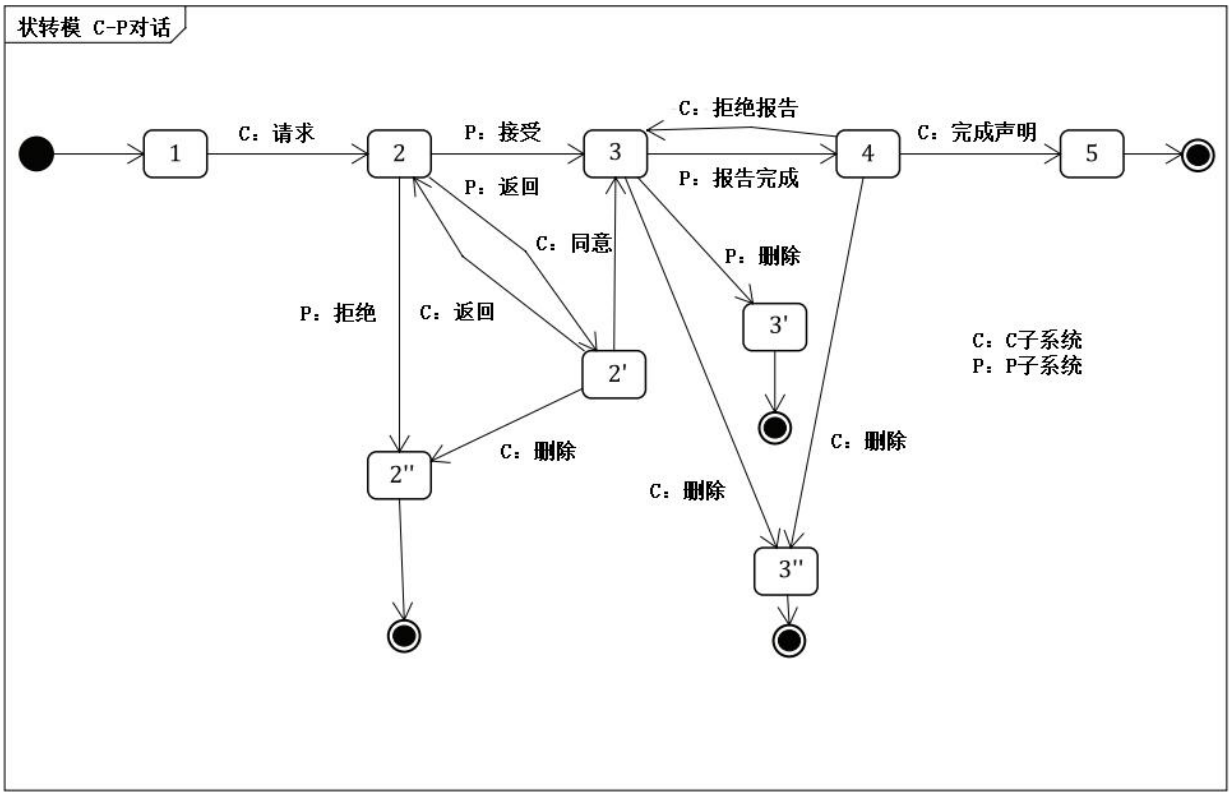


图 1　对话状态转换图

6　C 子系统与 P 子系统之间对话启动前的规程

6.1　概述

C子系统从服务提供者那里获取可以执行特定制造活动的P子系统的统一资源标识符（URI），而被请求执行活动的P子系统从服务提供者那里获取执行活动所需的C子系统的URI。C子系统和P子系统使用各自所获得的URI开始进行对话。当C子系统向P子系统发送第一个请求消息时，图1所示的对话启动，并根据图1的步骤来交换消息。本文件假设第3章定义的服务提供者集中控制投标，而其他投标过程（如分布式投标过程）是今后要做的工作，将不在本文件中描述。

6.2　使用能力专规识别对话伙伴的过程

识别对话伙伴和建立对话的过程如下所示：

a)　C 子系统和 P 子系统预先向服务提供者注册其每一个能力专规。C 子系统的能力专规描述 C 子系统所请求的活动的详细信息，而 P 子系统的能力专规描述 P 子系统可以执行的活动的详细信息。

b)　C 子系统从服务提供者那里获取能满足其自身请求的 P 子系统能力专规，而 P 子系统从服务提供者那里获取发送请求的 C 子系统的能力专规。能力专规匹配的规程符合 ISO 16100 系列标准中所指定的规程。

c)　在启动图 1 中的对话之前，C 子系统与 P 子系统之间存在两种信息交换情况。第一种情况是，C 子系统向所获取的 P 子系统的 URI 发送一个“通知”消息，而接收到“通知”消息的 P 子系统立即将一个“询问响应”消息发送给 C 子系统。另一种情况是，即使没有接收到 C 子系统

的“通知”消息，P 子系统也会发送“询问响应”消息。P 子系统可以定期地或在任意时间重复发送“询问响应”消息，直到 C 子系统将响应消息发送到“询问响应”消息为止。

d) 当 C 子系统向 P 子系统发送“请求”消息时，对话启动。随后，C 子系统和 C 子系统将根据图 1 进行消息交换。

制造软件单元（MSU）可实现多种不同功能。此处关注的是通信功能，包括C子系统和P子系统如何建立对话。所以，这里描述的重点在于MSU作为C子系统的角色与MSU作为P子系统的角色之间的通信功能以及通信中能力专规的使用。如图2中能力专规注册所示，在MSU之间开始交互之前，每个MSU需事先向服务提供者注册每一个能力专规，由服务提供者来决定MSU之间消费者与实施者之间的关系并确认两者的可连性。本文件假设服务提供者已经完成MSU之间消费者与实施者的关联，并确认已关联的MSU之间能进行连接。消费者-实施者模式在附录C中阐述。

图2展示了名为“C-P合作”的顺序图，包括名为“能力专规注册”的顺序图、名为“C-P对话”的顺序图以及名为“C-P对话基础”的顺序图。名为“能力专规注册”的顺序图显示了C子系统和P子系统预先在服务提供者中注册其能力专规。因为本文件是本文件的能力专规的一个应用示例，所以不对“能力专规注册”和服务提供者的细节做解释。图2中名为“准备C-P对话”的顺序图包含一个名为“得到能力专规”和一个名为“发送通知”的顺序图。第一个“得到能力专规”显示MSU与服务提供者之间的交互。充当C子系统的MSU从服务提供者那里获取充当P子系统的MSU的能力专规。另一方面，该P子系统从服务提供者获得作为P子系统伙伴的C子系统的能力专规。第二个“发送通知”有两种情况。第一种情况是当C子系统向P子系统发送通知消息，然后P子系统向C子系统发送询问响应时。另一种情况是当P子系统即使没有收到来自C子系统的通知消息也仍向C子系统发送询问响应时。“发送通知”中的“询问请求消息”表明P子系统可以重复发送询问响应，直到C子系统以请求消息回应为止。

图2中名为“C-P对话基础”的顺序图是图1中C子系统与P子系统之间的通信顺序图。因此，只有C子系统才需要获取P子系统的URI，所以参数p依赖于实施状况，在本文件中不做特别说明。附录B中描述了C-P对话基础顺序图。

C子系统和P子系统之间交换的消息在第7章进行了说明。C子系统和P子系统之间的消息通信在第8章中给出了更详细的说明。

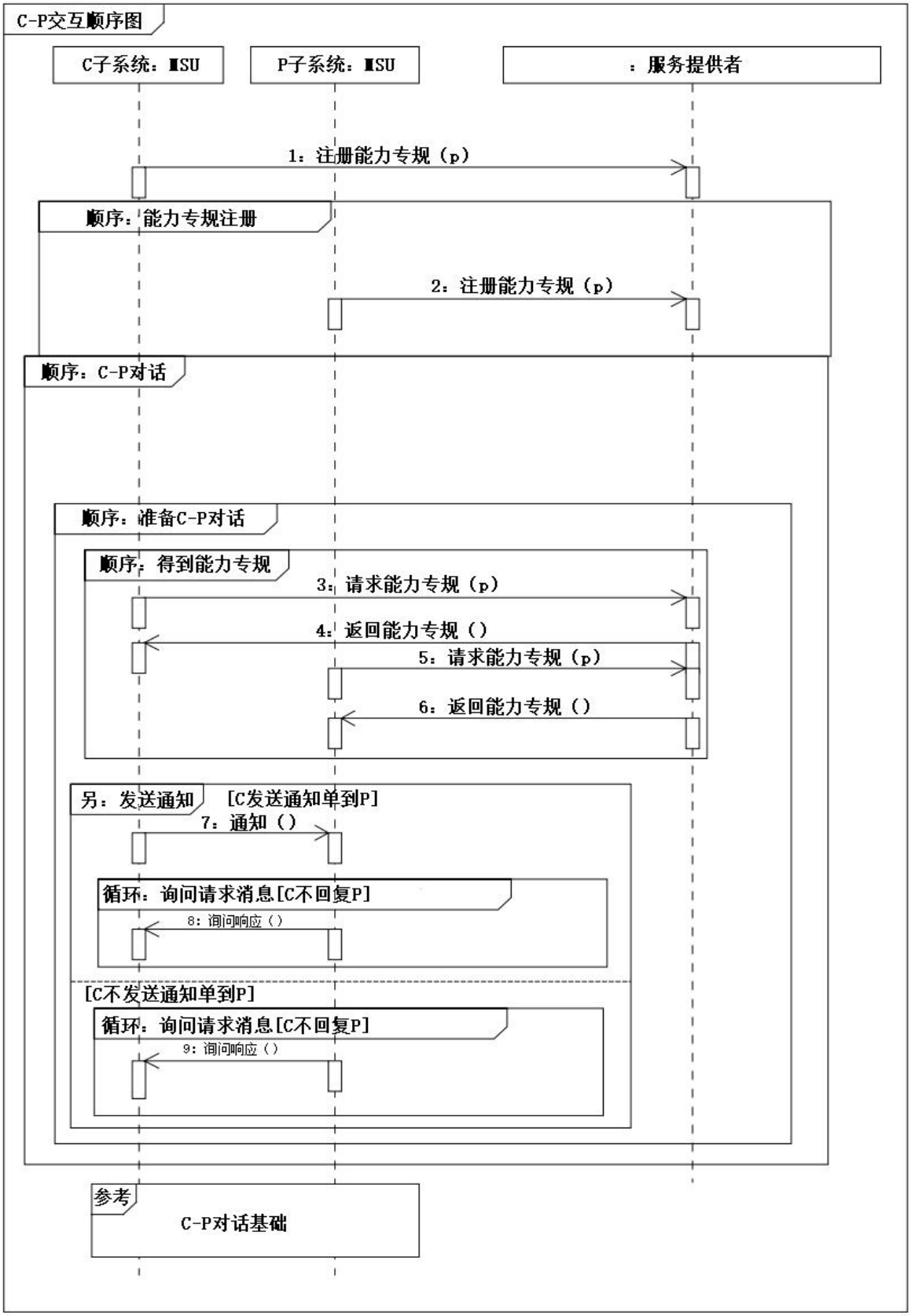

C-P交互顺序图
C子系统：ISU
P子系统：ISU
：服务提供者
1：注册能力专规（p）
顺序：能力专规注册
2：注册能力专规（p）
顺序：C-P对话
顺序：准备C-P对话
顺序：得到能力专规
3：请求能力专规（p）
4：返回能力专规（）
5：请求能力专规（p）
6：返回能力专规（）
另：发送通知
[C发送通知单到P]
7：通知（）
循环：询问请求消息[C不回复P]
8：询问响应（）
[C不发送通知单到P]
循环：询问请求消息[C不回复P]
9：询问响应（）
参考
C-P对话基础

图 2　对话交互概图

6.3 能力专规

6.3.1 概述

MSU使用能力专规来识别与其对话的伙伴。即使C子系统首先发送附录B中的“通知”消息，每个对话仍由P子系统的“邀请”消息来启动，因此需要定义C子系统的通信通道。在C子系统发送“通知”消息的情况下，需要有P子系统的地址。

6.3.2 特定部分描述

以上信息已在能力专规的特定部分中根据能力专规的模板（参见ISO 19100-3）进行了描述。6.4节中展示了向本文件添加必要元素的特定部分模板和该特定部分的示例。

添加到特定部分的XML标签是<Activity>（活动）和<Performatives>（执行），这些用于描述采用者所需的信息。

<Activity>标签使用动词来表示MSU提供功能的一种制造活动。图B.1“生产管理系统总体结构”中，作为MSU而实现的每个域的生产活动均以动词来表示，如卖、制造、买、开票、支付等等。在<Activity>部分的<InformationExchange>（信息交换）中的<Channel>（频道）标签拥有MSU之间通信的信息。通信方法由此标签的属性名称“类型”定义，而URI则由属性名称“地址”定义，通信方法将在第8章中说明。

<Performatives>标签表示可以接收的MSU执行动词（<Receive>（接受）标签）和可以发送的执行动词（<Send>（发送）标签）。<MessageFormat>（消息格式）标签给出了每个消息项中数据类型的信息。

6.4 能力专规模板和示例

本节中能力专规(参见ISO 16100-3）特定部分中所添加的必要元素的模板如6.4.1所示。6.4.2所示为应用此模板的能力专规示例。

6.4.1 能力专规模板

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="CapabilityProfiling">
    <xs:complexType>
      <xs:sequence maxOccurs="unbounded">
        <xs:element name="type">
          <xs:complexType>
            <xs:attribute name="id" type="xs:string" use="required"/>
          </xs:complexType>
        </xs:element>
        <xs:element name="CapabilityProfile">
          <xs:complexType>
                  <xs:sequence>
```

```
                <xs:element name="pkgtype">
                  <xs:complexType>
                    <xs:attribute name="version" type="xs:string"
form="unqualified"/>
                  </xs:complexType>
                </xs:element>
                              <xs:element name="Common" type="CommonPartType"/>
                <xs:element name="Specific" type="SpecificPartType"/>
              </xs:sequence>
              <xs:attribute name="date" type="xs:string"
              form="unqualified"/>
            </xs:complexType>
              </xs:element>
      </xs:sequence>
        </xs:complexType>
  </xs:element>
  <xs:complexType name="CommonPartType">
        <xs:sequence>
            <xs:choice>
        <xs:element name="Requirement">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="ID" type="xs:string"/>
            </xs:sequence>
            <xs:attribute name="id" type="xs:string"form="unqualified"/>
          </xs:complexType>
        </xs:element>
        <xs:element name="MSU_Capability">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="ID" type="xs:string"/>
            </xs:sequence>
            <xs:attribute name="id" type="xs:string"form="unqualified"/>
```

```
      </xs:complexType>
    </xs:element>
  </xs:choice>
  <xs:sequence maxOccurs="unbounded">
    <xs:element name="ReferenceCapabilityClassStructure">
      <xs:complexType>
        <xs:attribute name="id" type="xs:string" form="unqualified"/>
        <xs:attribute name="name" type="xs:string" form="unqualified"/>
       <xs:attributename="version" type="xs:string" form="unqualified"/>
        <xs:attribute name="url" type="xs:string" form="unqualified"/>
      </xs:complexType>
    </xs:element>
    <xs:element name="TemplateID">
      <xs:complexType>
        <xs:attribute name="ID" type="xs:string" form="unqualified"/>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
  <xs:element name="Version">
    <xs:complexType>
      <xs:attribute name="major" type="xs:string" form="unqualified"/>
      <xs:attribute name="minor" type="xs:string" form="unqualified"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="Owner">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="name" type="xs:string" minOccurs="0"/>
        <xs:element name="street" type="xs:string" minOccurs="0"/>
        <xs:element name="city" type="xs:string" minOccurs="0"/>
        <xs:element name="zip" type="xs:string" minOccurs="0"/>
        <xs:element name="state" type="xs:string" minOccurs="0"/>
        <xs:element name="country" type="xs:string" minOccurs="0"/>
```

```
            <xs:element name="comment" type="xs:string" minOccurs="0"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="ComputingFacilities" minOccurs="0"
maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
           <xs:element name="Processor0" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
               <xs:attribute name="type" type="xs:string" form="unqualified"/>
              </xs:complexType>
            </xs:element>
            <xs:element name="OperatingSystem0" minOccurs="0"
maxOccurs="unbounded">
              <xs:complexType>
               <xs:attribute name="type" type="xs:string" form="unqualified"/>
              </xs:complexType>
            </xs:element>
           <xs:element name="Language" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
               <xs:attribute name="name" type="xs:string" form="unqualified"/>
              </xs:complexType>
            </xs:element>
            <xs:element name="Memory" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
               <xs:attributename="size" type="xs:string" form="unqualified"/>
               <xs:attributename="unit" type="xs:string" form="unqualified"/>
              </xs:complexType>
            </xs:element>
           <xs:element name="DiskSpace" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
               <xs:attribute name="size" type="xs:string" form="unqualified"/>
```

```
                <xs:attribute name="unit" type="xs:string" form="unqualified"/>
                  </xs:complexType>
                </xs:element>
              </xs:sequence>
              <xs:attribute name="type" type="xs:string" form="unqualified"/>
            </xs:complexType>
          </xs:element>
          <xs:element name="Performance" minOccurs="0" maxOccurs="unbounded">
            <xs:complexType>
          <xs:attribute name="ElapsedTime" type="xs:string" form="unqualified"/>
              <xs:attribute name="TransactionsPerUnitTime"
              type="xs:string"form="unqualified"/>
            </xs:complexType>
                </xs:element>
           <xs:element name="ReliabilityData" minOccurs="0" maxOccurs="unbounded">
            <xs:complexType>
              <xs:sequence>
                  <xs:element name="UsageHistory" type="xs:string" minOccurs="0"/>
                 <xs:element name="Shipments" minOccurs="0" maxOccurs="unbounded">
                  <xs:complexType>
                  <xs:attribute name="number" type="xs:string"  form="unqualified"/>
                  </xs:complexType>
                </xs:element>
                <xs:element name="IntendedSafetyIntegrity" minOccurs="0"
maxOccurs="unbounded">
                  <xs:complexType>
                    <xs:attributename="level"type="xs:string"form="unqualified"/>
                  </xs:complexType>
                </xs:element>
              <xs:element name="Certification" minOccurs="0"  maxOccurs="unbounded">
                  <xs:complexType>
                     <xs:attributename="no1"type="xs:string"form="unqualified"/>
                        </xs:complexType>
```

```
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
             <xs:element name="SupportPolicy" minOccurs="0" maxOccurs="unbounded">
                    <xs:complexType>
                <xs:attribute name="index" type="xs:string" form="unqualified"/>
                  </xs:complexType>
              </xs:element>
              <xs:element name="PriceData" minOccurs="0" maxOccurs="unbounded">
                    <xs:complexType>
                <xs:attribute name="invest" type="xs:string" form="unqualified"/>
                <xs:attribute name="annualSupport"
                type="xs:string"form="unqualified"/>
                <xs:attribute name="unit" type="xs:string"form="unqualified"/>
                  </xs:complexType>
                </xs:element>
          </xs:sequence>
              </xs:complexType>
        <xs:complexType name="SpecificPartType">
          <xs:sequence>
            <xs:element name="Activity" type="Activity" maxOccurs="unbounded"/>
            <xs:element name="Performatives" type="Performatives" />
            <xs:element name="MessageFormat" type="MessageFormat" />
          </xs:sequence>
        </xs:complexType>

        <xs:complexType name="Activity">
          <xs:sequence>
            <xs:element name="InformationExchange" type="InformationExchange"
maxOccurs="unbounded" />
          </xs:sequence>
          <xs:attribute name="id" type="xs:string" form="unqualified"/>
```

```
  <xs:attribute name="name" type="xs:string" form="unqualified"/>
</xs:complexType>
<xs:complexType name="InformationExchange">
  <xs:sequence>
    <xs:element name="BasicProtocol">
      <xs:complexType>
        <xs:attribute name="id" type="xs:string" form="unqualified"/>
       <xs:attribute name="type" type="ProtocolType"form="unqualified"/>
      </xs:complexType>
    </xs:element>
    <xs:element name="ApplicationProtocol">
      <xs:complexType>
        <xs:attribute name="id" type="xs:string" form="unqualified"/>
      </xs:complexType>
    </xs:element>
    <xs:element name="TalkTo">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="Method" type="Method" />
        </xs:sequence>
        <xs:attribute name="id" type="xs:string" form="unqualified"/>
       <xs:attribute name="address" type="xs:string"form="unqualified"/>
      </xs:complexType>
    </xs:element>
    <xs:element name="CallbackTo" minOccurs="0">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="Method" type="Method" />
        </xs:sequence>
        <xs:attribute name="id" type="xs:string" form="unqualified"/>
       <xs:attribute name="address" type="xs:string"form="unqualified"/>
      </xs:complexType>
    </xs:element>
```

```
      <xs:element name="ContentEditor" minOccurs="0">
        <xs:complexType>
          <xs:attribute name="name" type="xs:string"form="unqualified"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="Method">
    <xs:attribute name="type" form="unqualified">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="GET" />
          <xs:enumeration value="POST" />
          <xs:enumeration value="PUT" />
          <xs:enumeration value="DELETE" />
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="name" type="xs:string" form="unqualified"/>
</xs:complexType>
<xs:simpleType name="ProtocolType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="PUSH" />
      <xs:enumeration value="PULL" />
      <xs:enumeration value="NOTIFY" />
      <xs:enumeration value="CALLBACK" />
    </xs:restriction>
</xs:simpleType>

<xs:complexType name="Performatives">
    <xs:choice maxOccurs="unbounded">
      <xs:element name="Receive">
                <xs:complexType>
```

```
            <xs:attribute name="name" type="xs:string" form="unqualified"/>
          </xs:complexType>
        </xs:element>
        <xs:element name="Send">
          <xs:complexType>
            <xs:attribute name="name" type="xs:string" form="unqualified"/>
          </xs:complexType>
        </xs:element>
      </xs:choice>
    </xs:complexType>
    <xs:complexType name="MessageFormat">
      <xs:sequence>
        <xs:element name="Data" maxOccurs="unbounded">
          <xs:complexType>
            <xs:attribute name="name" type="xs:string" form="unqualified"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
 </xs:schema>
```

6.4.2 能力专规示例

以下示例是执行“卖”活动的MSU（C子系统）的能力专规。此示例仅限于与充当P子系统的MSU进行对话所需的元素。

```
 <?xml version="1.0" encoding="UTF-8"?>
 <CapabilityProfiling xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <type id="MSU_Profile"/>
    <CapabilityProfile date="2017-08-10">
      <pkgtype version="1.0" />
      <Common>
        <MSU_Capability>
          <ID>Buy/Make Subsystem</ID>
        </MSU_Capability>
        <ReferenceCapabilityClassStructure />
```

```
    <TemplateID />
    <Version />
    <Owner>
      <name>APSOM MESX-JP</name>
      <country>Japan</country>
    </Owner>
  </Common>
  <Specific>
    <Activity id="Sell" name="SELL">
      <!-- Pull channel from Sell-function as C-subsystem -->
      <InformationExchange>
        <BasicProtocol id="REST" type="PULL" />
        <ApplicationProtocol id="C-subsystem-P-subsystem" />
        <TalkTo id="Sell" address="http:// URL of P-subsystem /ihcl/">
          <Method type="POST" name="ihcladaptor" />
        </TalkTo>
        <CallbackTo id="Sell" address="http://localhost/ihcl/">
          <Method type="POST" name="ihcladaptor" />
        </CallbackTo>
       <ContentEditor name="IHCL.message.editor.DefaultContentEditor" />
      </InformationExchange>
      <!-- Push channel to Sell-function as C-subsystem -->
      <InformationExchange>
        <BasicProtocol id="REST" type="PUSH"/>
        <ApplicationProtocol id="C-subsystem-P-subsystem" />
        <TalkTo id="Sell" address="http:// URL of P-subsystem /ihcl/">
          <Method type="POST" name="ihcladaptor" />
        </TalkTo>
      </InformationExchange>
    </Activity>
    <Activity id="Notify" name="CallFor">
      <!-- Notify channel to P-subsystem Do -->
      <InformationExchange>
   <BasicProtocol id="REST" type="PUSH" />
```

```
        <ApplicationProtocol id="C-subsystem-P-subsystem" />
       <TalkTo id="Do" address="http:// address of do subsystem /ihcl/">
           <Method type="POST" name="IHCLAdaptorDefaultServlet" />
        </TalkTo>
       <ContentEditor name="IHCL.message.editor.DefaultContentEditor" />
     </InformationExchange>
   </Activity>
   <Performatives>
     <Receive name="Request" />
     <Send name="Promise" />
     <Send name="Decline" />
     <Send name="ReportCompletion" />
     <Receive name="DeclineReport" />
     <Receive name="DeclareComplete" />
   </Performatives>
   <MessageFormat>
     <Data name="o-id" />           <!--   Order identifier -->
     <Data name="who" />            <!--   Orderer -->
     <Data name="what" />           <!--   Order item -->
     <Data name="spec" />           <!--   Specification-->
     <Data name="how-many" /> <!-- quantity -->
     <Data name="when_by" />        <!--   Due date -->
     <Data name="where_to" /> <!--          Destination -->
     <Data name="option" />         <!--   Extra information-->
   </MessageFormat>
 </Specific>
</CapabilityProfile>
</CapabilityProfiling>
```

7 消息

7.1 消息结构

消息分为控制消息和执行消息。执行消息会导致对话状态转换，而控制消息则不会改变对话的状态。图3显示了消息的结构。

有些像IEC 62264-5和OAGiS这样的标准定义了有关应用程序之间信息交换的事务或消息，且动词在IEC 62264-5和OAGiS中都扮演着重要角色，从这点来说，本文件中的消息也是如此。另一方面，本文件中的消息用于那些运用能力专规的应用程序之间的对话，并且基于一个不同于OAGiS消息和IEC 62264-5事务的框架。由于IEC 62264-5的事务动词与OAGiS的消息动词除了动词“通知”外均相同，所以附录G显示了本文件的消息动词与OAGiS动词之间的映射表。

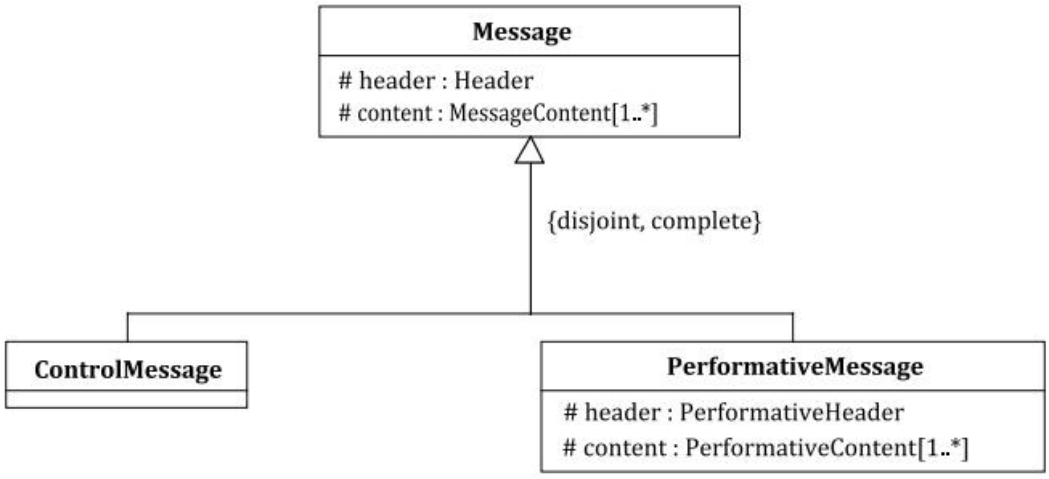


图 3　消息结构

消息由属性标头和属性内容组成。标头是具有标头类型的属性，而内容是具有多重性1..*MessageContent类型的属性。

标头分为ControlHeader和PerformativeHeader。图4显示了标头的结构。

标头由具有字符串类型的动词类属性、具有字符串类型的动词属性和被称为从句的属性组成。从句是具有多重性1..*的从句（Clause）类型的属性。标头（Header）实例是“控制标头”（ControlHeader）实例还是“执行标头”（PerformativeHeader）实例则取决于动词类型属性的值。动词类型属性的值限定为“controlVerb”或“performative”。当动词类型（verbtype）的值为“控制动词”（controlVerb）时，它是一个ControlHeader的实例。当“动词类型”（verbtype）的值为“performative”时，它则是一个PerformativeHeader的实例。从句由具有字符串类型的属性名称和具有字符串类型的属性值组成。

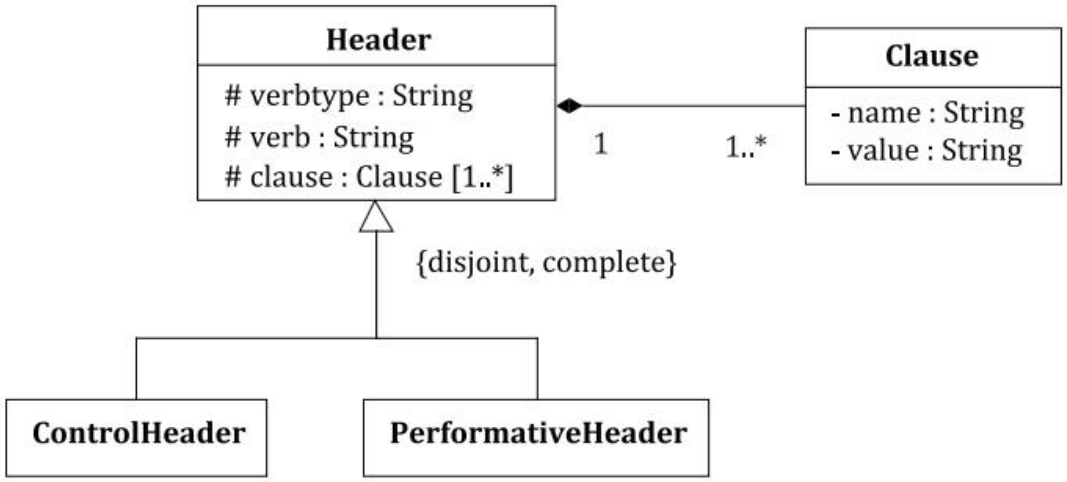


图 4　标头结构

表 1　从句值

名称	值
发送者	发送者的标识符
接收者	接收者的标识符
时间戳	发送消息的时间戳
消息ID	消息标识符
回复	原始消息标识符
语言	“zzzz”

MessageContent分为ControlContent和PerformativeContent。图5显示了MessageContent的结构。

控制内容（Control Content）具有字符串类型的message-content属性。执行内容（Performative Content）具有执行消息内容（Performative Message Content）类型的message-content属性。

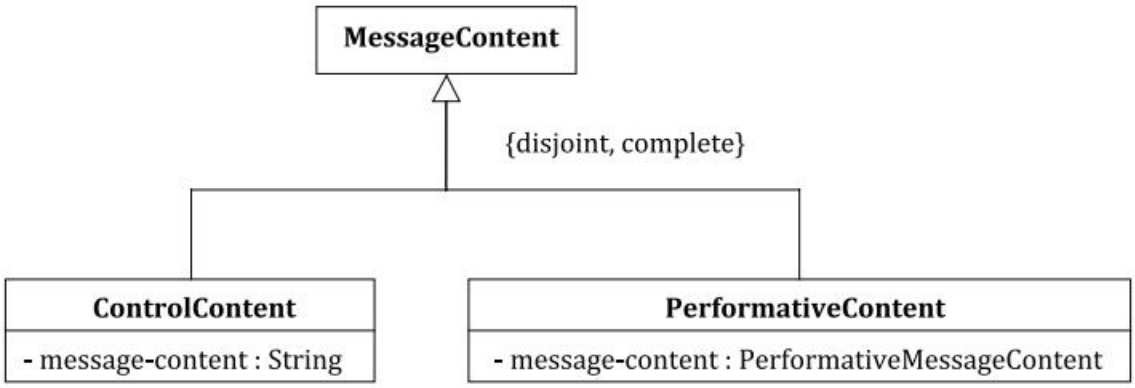


图 5　MessageContent 的结构

PerformativeMessageContent由具有字符串类型的订单属性、具有多重性1..*的NameValuePair类型的订单项目（ordereditems）属性以及具有多重性0..2的NameValuePair类型的检验器（checker）属性组成。图6显示了PerformativeMessageContent的结构。

图 6　PerformativeMessageContent 的结构

订单属性的实例由字符串类型的值组成。订单项目的实例由一个或多个订单单项目（Ordered Item）实例组成。复合检验器（Checkers）的实例由一个或多个检验器实例组成。

NamedValuePair分为NumberOfPieces、Quantity、StringValue或CollectionValue。数量由两个属性组成，一个属性名为值，另一个属性名为度量。属性值是双类型，属性度量是字符串类型。数量用于表示物理量，例如长度和重量。CollectionValue具有多重性1..*的NamedValuePair类型的属性值。CollectionValue用于表示NameValuePair的嵌套实例。

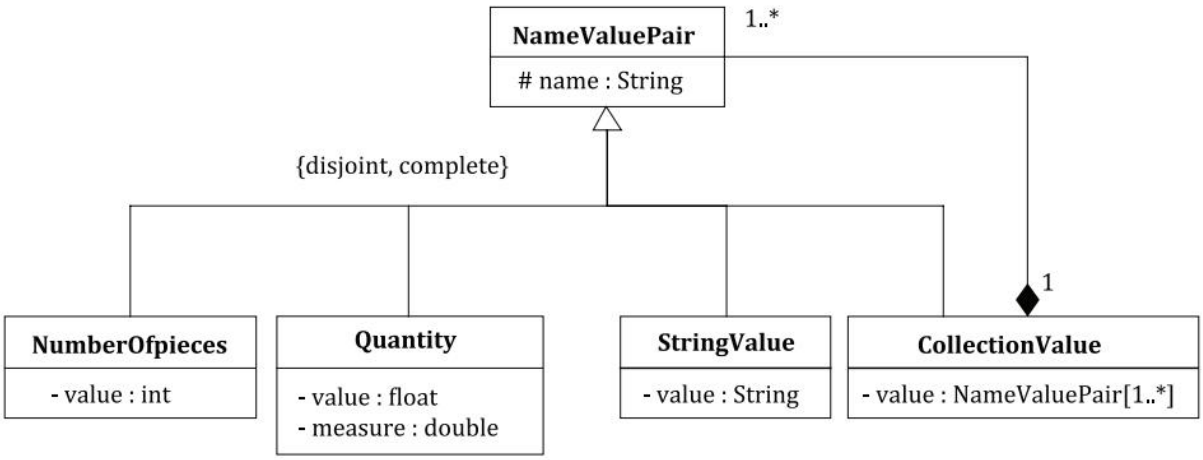


图 7　NameValuePair 的结构

7.2　控制消息

控制消息包含以下消息：

——通知消息；

——询问响应消息；

——警告消息。

下列中的每一个消息都使用了示例进行解释。

7.2.1　通知消息

C子系统通知P子系统，其已准备好为P子系统下订单。当C子系统将消息发送到P子系统时，便会触发P子系统发送一个邀请消息。

消息是 ControlMessage的实例。属性标头拥有一个ControlHeaderMessage的实例。属性内容拥有一个ControlContent的实例。当C子系统对P子系统有要求时，便会向P子系统发送此消息。询问响应消息（Ask Response message）中解释了为何此消息具有任意性的原因。

下列是ControlHeader实例的属性值。

——verbtype="controlVerb"；

——verb="Notify"；

——clause = (("sender","Cxxxx"),("receiver", "piny"), ,("language","zzzz"))。

下列是ControlContent实例的属性值。

message-content=""。

7.2.2 询问响应消息

此消息既用于针对通知(Notify)消息而返回的一个响应消息，又用于在等待C子系统响应时的一个提示响应消息。P子系统接收到C子系统的通知(Notify))消息后，发送一个“询问响应”(Ask Response)消息。C子系统对“询问响应”（Ask Response）消息做出响应，向P子系统发送如请求消息之类的消息。如6.2中所述，当P子系统要求C子系统对“C：请求消息”进行响应时，P子系统可以向C子系统重复发送“询问响应（Ask Response ）消息。除此情况外，当P子系统向C子系统询问 C:还盘（Counter）,C:取消（Cancel）或C:声明完成（DeclareComplete）这些消息时，P子系统可以向C子系统重复发送“询问响应”（Ask Response）消息，直至返回所需的消息为止。

消息是ControlMessage的实例，属性标头是ControlHeader的实例，属性内容是ControlContent的实例。P子系统将此消息发送到C子系统。

下列是ControlHeader实例的属性值：

——verbtype="controlVerb"；

——verb="AskResponse"；

——clause = (("sender","Cxxxx"),("receiver","Pyyyy"),...,("language","zzzz"))。

下列是ControlContent实例的属性值：

message-content=select (*) where (‘ConditionStatement’)

ConditionStatement是"ID=RequestIdentifier" 。

RequestIdentifier是Request或"NEW"的标识符。

示例 1：P 子系统已经接收到 P 子系统的请求，该请求标识为“ RE12345”。

ConditionStatement 是"ID=RE12345"。

此conditonStatement意为P子系统要求C子系统重新发送请求。

示例 2：示例 2：P 子系统在收到一个另外请求后又收到一个请求。

ConditionStatement 是"ID=NEW"。

P 子系统通过指定上次接收的时间戳来识别从 C 子系统接收到的消息。P 子系统将对此消息的答复发送给 C 子系统。

7.2.3 警示消息

如果消息接收者在对话期间检测到异常，它会向该消息发送者发送一个警示消息。

消息是ControlMessage的实例，属性标头具有一个ControlHeader的实例，属性内容具有一个ControlContent的实例。当P子系统检测到异常状态时，P子系统会将此消息发送给C子系统。当C子系统检测到异常状态时，C子系统会将该消息发送到P子系统。

下列是ControlHeader实例的属性值：

——verbtype="controlVerb";
——verb="Warn";
——clause = (("sender","Cxxxx"),("receiver","piny"),...,("language","zzzz"))。
下列是ControlContent实例的属性值：
message-content=error at 01000001 Request
01000001是message-id，是“请求”（Request）消息出现异常消息的标识符。
以下用词是表达错误的词语：
——“协议顺序错误” （Protocol Sequence Error）： 收到意外消息，在当前对话状态下无法接受该消息；
——或 "消息重复”（Message Duplicated）；
——或 "请求的技术不匹配”（NoR Unmatched）；
——或 "校验和不是同一个值”（Checksum Unmatch）。

7.3 执行消息

执行消息分为九种类型。每一种类型对应于图1中的一种行为。表2显示了这些对应关系。

表 2 执行消息的分类类型

消息类型	行动	描述
请求消息 Request message	C:Request	C子系统向P子系统发送包含订单的请求消息
接受消息 Promise message	P:Promise	P子系统将Promise消息发送到C子系统。
还盘消息 Counter message	P:Counter,C:Counter	P子系统向C子系统发送还盘消息。 C子系统向P子系统发送还盘消息。
接收消息 Accept message	C:Accept	C子系统接受来自P子系统的还盘。
拒绝消息 Decline message	P:Decline	P子系统拒绝来自C子系统的C：请求或C：还盘。
取消消息 Cancel message	C:Cancel, P:Cancel	C子系统取消C子系统发出的订单。 P子系统取消C子系统发出的订单。
报告完成消息 Report Completion message	P:Report Completion	P子系统向C子系统发送报告完成消息。
声明完成消息 Declare Complete message	C:Declare Complete	当C子系统接受来自P子系统的“报告完成”消息时，C子系统发送“声明完成”消息。
拒绝报告消息 Decline Report message	C:Decline Report	当C子系统拒绝来自P子系统的报告完成消息时，C子系统发送拒绝报告消息。

表 3 执行消息内容订购货品名称清单

姓名	描述
订单ID（o-id）	订单标识符
谁（who）	订购者，团体或组织

表3 执行消息内容订购货品名称清单（续）

姓名	描述
什么（what）	货品或帐户名称的标识符
规格（spec）	订购商品的收取值规格，收取值
序列号（serialno）	订购商品的标识符
数量（how many）	订购数量（件数或代表实物数量的数量）
数量（how much）	价格、数量
何时完成（when_by）	所需的交货日期和时间或完成日期、日期和时间
何时（when_at）	所需的开始日期和时间
发货地（where_from）	发货位置、地点或团体
发送地（where_to）	接收点，地点或团体
负责人（whom_incharge）	主管部门或实施过程、组织
选择（option）	备注、文字

表 4　订购商品与实施消息之间的关系

订货名称	M/O	请求消息	承诺消息	换盘消息	接受消息	拒绝消息	取消消息	报告完成消息	宣布完成消息	拒绝报告消息
订单ID o-id	M	M	M	M	M	M	M	M	M	M
谁who	M	M	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
什么what	M	M	M	M	N/A	N/A	N/A	M	N/A	N/A
规格spec	O	O	O	O	N/A	N/A	N/A	O	N/A	N/A
序列号 serialno	O	O	O	O	N/A	N/A	N/A	O	N/A	N/A
数量 how many	M	M	M	M	N/A	N/A	N/A	M	N/A	N/A
数量 how much	M	M	M	M	N/A	N/A	N/A	M	N/A	N/A
何时完成 when_by	O	O	O	O	N/A	N/A	N/A	M	N/A	N/A
何时 when_at	O	O	O	O	N/A	N/A	N/A	M	N/A	N/A
发货地 where_from	O	O	O	O	N/A	N/A	N/A	O	N/A	N/A
发送地 where_to	O	O	O	O	N/A	N/A	N/A	O	N/A	N/A
负责人 whom_incharge	O	O	O	O	N/A	N/A	N/A	O	N/A	N/A
选择option	O	O	O	O	N/A	N/A	N/A	O	N/A	N/A
注：M：强制，O：任意，N/A：不适用　“何时完成”和　“何时何处”不包括在内。										

7.3.1 请求消息

C子系统在P子系统中下订单，使用此消息类型做为对请求的一个响应，以等待P子系统发出响应。这个“请求”类的消息可以在一个消息中包含多个订单。在执行订单期间，按订单来衡量进度。当对话事件成为C: 宣布完成、C：取消、P：取消、P：拒绝时，进度管理将终止。

——verbtype="performative"；

——verb="Request"；

——clause = (("sender","Cxxxx"),("receiver","Pyyyy"),...,("language","zzzz"))；

——order ="Buy"。

订单值与能力专规中定义的活动标识符相对应。生产系统的所有者（Owner）明确订单值和能力专规中所定义的活动标识符。附录B中显示了常用订单值的示例。

例如，订单值“Make”表示订单项目已根据订单规格在所需的日期之前完成。订单值“Ship”表示订单项目已在所需日期之前从指定的仓库发送至指定的收货人。

——ordereditems=(("oid","#0004"),("who","CUST01"),("what","screw"),...,("when_by","2017-10-02"),("whom_incharge","ABC"),(("option","option is ..."))；

——能力专规定义了应在该消息的订购项目中包含哪种商品。名为“什么（what）”是要求将项目的标识符作为其值。；

——checker=(("number-of-records",1), ("check-sum",13))。

该示例中显示记录数和校验值已得到确认。任意项目可以省略。默认值的解释由应用系统自行确定。

7.3.2 接受消息

当P子系统从C子系统接收到请求消息，并判断该项目的规格、编号、交货日期等已达到可接受水平时，即会返回“接受（P：Promise）”消息。这将导致F1中的状态2转换到状态3。P子系统是否会在此消息中返回订单估价和/或预计交货日期则将视整个系统而分别做出决定。

接受（Promise）消息的订购项目包含订单标识以及仅仅那些不同于请求（Request）消息的订购项目。如果所有接受（Promise）消息中的订单项目（ordereditem）与相对应的请求（Request）消息中的订单项目(ordereditem)相同,则接受(Promise)消息中的订单ID (O-id)只有订单项目(ordereditem)和订单标识符。

——verbtype="performative"；

——verb="Promise"；

——clause = (("sender","Cxxxx"),("receiver","Pyyyy"),...,("language","zzzz"))；

——order="Buy"；

——ordereditems=(("o-id","#0004"),)。

它显示了请求（Request）消息的订单项目与接受（Promise）消息的所有相应订单项目相同。

checker=(("number-of-records",1), ("check-sum",13))。

此示例中显示了已确认的记录数字和校验和值。

7.3.3 返回消息

尽管C子系统的请求消息已被P子系统所接收，但规格、货物数目或种类、交货日期等仍可能会超出所允许的范围。这时的P子系统就可以向C子系统提出返回。这将导致图1中的状态2转换到状态6。

——verbtype="performative"；

——verb="Counter"；

——clause = (("sender","Cxxxx"),("receiver","Pyyyy"),...,("language","zzzz"))；

——order ="Buy"；

——ordereditems=(("o-id","#0004"),("when_by","2017-10-10"))。

它显示了P子系统提出返回以更改交货日期的情况。

checker=(("number-of-records",1), ("check-sum",13))。

7.3.4 接收消息

当C子系统接受P子系统返回的返回（Counter）消息时，C子系统将接收（Accept）消息返回给P子系统。这将导致图1中从状态2’转换到状态3。

verbtype="performative"

verb="Accept"

clause = (("sender","Cxxxx"),("receiver","Pyyyy"),...,("language","zzzz"))

order ="Buy"

ordereditems=(("o-id","#0004"))

checker=(("number-of-records",1), ("check-sum",13))

7.3.5 取消消息（反对返回）

当C子系统不接受P子系统返回的返回（Counter）消息时，C子系统将取消消息返回给P子系统。 这将导致图1中的状态2’转换到状态2”。

消息是PerformativeMessage的实例，标头是PerformativeHeader的实例。以下是PerformativeHeader实例的属性值。

verbtype="performative"

verb="Cancel"

clause = (("sender","Cxxxx"),("receiver","Pyyyy"),...,("language","zzzz"))

下列是PerformativeContent实例的属性值。

message-content拥有PerformativeMessageContent的实例。

下列是PerformativeMessageContent实例的属性值。

order ="Buy"

ordereditems=(("o-id","#0004"))

7.3.6 返回消息

当C子系统无法接受P子系统发送的返回(P: Counter)时，C子系统对P子系统返回(P: Counter)的答复或是取消(C: Cancel)请求或是反向返回(C: Counter)。当C子系统确定有机会能够与P子系统签约时，C子系统就会对反向返回(C: Counter)作出响应。C子系统的返回(C: Counter)是针对P子系统反向返回中任何一个（或全部）项目、规格、交货日期、数量进行修改。这将导致图1中的状态2’转换到状态2。在此期间，请求状态为待定。

verbtype=“performative”

verb=“Counter”

clause= (("sender","Cxxxx"),("receiver","Pyyyy"),...,("language","zzzz"))

order ="Buy"

ordereditems=(("o-id","#0004"),("when_by","2017-10-09"))

它显示了P子系统提出反向还盘以更改交货日期的情况。

checker=(("number-of-records",1), ("check-sum",13))

7.3.7 拒绝消息

当P子系统拒绝C子系统的C：请求或C：还盘时，P子系统返回拒绝消息。这将导致图1中的状态2转换到状态3。

消息是PerformativeMessage的实例，标头是PerformativeHeader的实例。以下是PerformativeHeader实例的属性值。

verbtype="performative"

verb="Decline"

clause = (("sender","Cxxxx"),("receiver","Pyyyy"),...,("language","zzzz"))

下列是PerformativeContent实例的属性值。

Message-content拥有PerformativeMessageContent的实例。

下列是PerformativeMessageContent实例的属性值。

order="Buy"

ordereditems=((“o-id","#0004"))

7.3.8 取消消息（P：Cancel）

当出于某些原因而中途停止已承诺的订单时,P子系统基本上会发送“报告完成”(ReportCompletion)消息。P子系统还可以发送P;取消（“P：Cancel”）的消息。这只限于那些由于单方面及特殊情况的发生而中断了与C子系统的对话。这将导致图1中的状态3转换到状态3’。

消息是PerformativeMessage的实例，标头是PerformativeHeader的实例。以下是PerformativeHeader实例的属性值。

verbtype="performative"

verb="Cancel"

clause = (("sender","Cxxxx"),("receiver","Pyyyy"),...,("language","zzzz"))

下列是PerformativeContent实例的属性值。

message-content拥有PerformativeMessageContent的实例。

下列是PerformativeMessageContent实例的属性值。

order="Buy"

ordereditems=((“o-id","#0004"))

7.3.9 取消消息（C：Cancel）

当由于C子系统出现状况而停止进行到一半的订单或取消已完成的订单时，C子系统会向P子系统发送取消（Cancel）消息。如何处理已完成的半产品则由系统外部决定。这将会导致图1中的状态3转换到状态9或状态4转换到状态9。

消息是PerformativeMessage的实例，标头是PerformativeHeader的实例。以下是PerformativeHeader实例的属性值。

verbtype="performative"

verb="Cancel"

clause = (("sender","Cxxxx"),("receiver","Pyyyy")...,("language","zzzz"))

下列是PerformativeContent实例的属性值。

message-content拥有PerformativeMessageContent的实例。

下列是PerformativeMessageContent实例的属性值。

order="Buy"

ordereditems=(("o-id","#0004"))

7.3.10 报告完成消息

当P子系统完成或停止请求消息所指意的工作时,P子系统会将报告完成消息发送给该任务的承包商C子系统。这将导致图1中的状态3转换到状态4。如果P子系统决定终止的话，那么即使结果无法满足数量、交货日期甚至规格，P子系统也会发送报告完成（RepotCmpletion）消息，而满足与否则由C子系统来决定。

verbtype="performative"

verb="ReportCompletion"

clause = (("sender","Cxxxx"),("receiver","Pyyyy")...,("language","zzzz"))

order="Buy"

ordereditems=(("o-id","#0004"),...,("how_many","1000"),("how_much","$200,000")

,("when_by","2017-10-09"))

订单项目务必要带有“o-id”、“how_many”，“how_much”和“when_by”这些名称。

checker=(("number-of-records",1), ("check-sum",13))

7.3.11 声明完成消息

C子系统对P子系统发送的“报告完成”（ReportCompletion）消息决定是否要返回一个接受消息（DeclareComplete Message）还是一个拒绝消息（DeclineReport Message）。即使数量不足或交货日期延迟，C子系统也会向C子系统返回一个表明接受的声明完成（DeclareComplete）消息。这将导致图1中的状态4转换到状态5。

消息是PerformativeMessage的实例，标头是PerformativeHeader的实例。以下是PerformativeHeader实例的属性值。

verbtype="performative"

verb="DeclareComplete"

clause = (("sender","Cxxxx"),("receiver","Pyyyy"),...,("language","zzzz"))

下列是PerformativeContent实例的属性值。

message-content拥有PerformativeMessageContent实例。

下列是PerformativeMessageContent实例的属性值。

order="Buy"

ordereditems=(("o-id","#0004"))

7.3.12 拒绝报告消息

如果C子系统拒绝报告完成(ReportComplete)消息,则会向P子系统发送拒绝报告(DeclineReport)消息。这将导致图1中的状态3转换到状态4。决定是否需要在系统外恢复工作（商业判断等）则要根据系统外部的情况来进行判断（例如商业决策）。

消息是PerformativeMessage的实例，标头是PerformativeHeader的实例。以下是PerformativeHeader实例的属性值。

verbtype="performative"

verb="DeclineReport"

clause = (("sender","Cxxxx"),("receiver","Pyyyy"),...,("language","zzzz"))

下列是PerformativeContent实例的属性值。

message-content拥有PerformativeMessageContent的实例。

下列是PerformativeMessageContent实例的属性值。

order="Buy"

ordereditems=(("o-id","#0004"))

8 C子系统与P子系统之间的通信协议

8.1 概述

第6章和第7章描述了C子系统与P子系统之间的对话功能得以实现。附录G概述了实现的示例。在解释该示例之前，本章节描述了C子系统与P子系统之间的接口、所采用的消息传递方法以及消息通信方法。

8.2 介于C子系统与P子系统之间的接口

第6章和第7章阐述到，对话消息的方向是由消息类型来决定的。C子系统的方向是从P子系统入站，而P子系统的方向是从C子系统出站。C子系统和P子系统之间的交互消息可归类为以下其一：

a) P子系统请求C子系统的响应（入站，出站）；

b) P子系统仅仅发送到C子系统的消息（入站）；

c) P子系统刚刚才发送到C子系统的消息（出站）。

对于C子系统发送给P子系统的通知（Notify）消息，P子系统拥有一个通知（Notify）接口以接受来自C子系统的请求，而其他子系统则拥有接受P子系统请求的接口。P子系统发送到C子系统的这种操作是“推”，而P子系统请求来自C子系统响应的行为则是“拉”，如通知消息。推拉接口与（接口除外的）C子系统和P子系统之间的关系如图8所示。附录D中给出了示例。

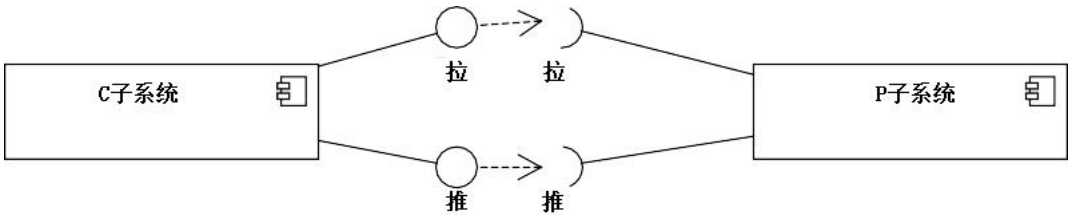


图8 不含通知接口的C子系统与P子系统之间的接口

8.3 C子系统与P子系统之间对话中的消息通信

HTTP1.1被采纳为消息传输协议，其原因在于RESTfull的架构风格深入人心，且便于实现。当对话消息通过C子系统与P子系统之间的接口时，此消息所通过的路径称为通道。例如，通过提拉接口的消息路径是一个推送通道。对话中消息是异步交换的，因此，通过提拉通道的消息被C子系统接收，之后提拉通道即被阻断。由于没有针对此消息的响应消息通过该路径，所以会有一个C子系统进行响应的回调接口被添加进来。

表5显示了接口名称、通道名称（即通过该接口传递消息的路由）与能力专规的通道（Channel）标签属性值类型之间的对应关系。对话消息的方向与通道之间的关系是，入站消息（inbound message）通过推通道(push channel）或拉通道(pull channel），而出站消息则通过通知（notify）通道或回调(callbak)通道。表6显示了消息类型与行为和通道之间的关系。

表5 接口名称与通道名称之间的对应关系

接口名称	通道名称	通道标签类型
推（Push）	push channel	push

接口名称	通道名称	通道标签类型
提（Pull）	pull channel	pull
通知（Notify）	notify channel	notify
回调（Callback）	callback channel	callback

表 6　消息类型与通道之间的关系

消息类型	行动	出站	入站
通知消息 Notify message	Notify	Notify channel	
询问响应消息 Ask Response message	AskResponse		pull channel
请求消息 Request message	C: Request	callback channel	
承诺消息 Promise message	P: Promise		push channel
还盘消息 Counter message	P: Counter		pull channel
	C: Counter	callback channel	
接收消息 Accept message	C: Accept	callback channel	
拒绝消息 Decline message	P: Decline		push channel
取消消息 Cancel message	C: Cancel	callback channel	
	P: Cancel		push channel
报告完成消息 Report Completion message	P: Report Completion		pull channel
声明完成消息 Declare Complete message	C: Declare Complete	callback channel	
拒绝报告消息 Decline Report message	C: Decline Report	callback channel	

附 录 A
（资料性）
JSON 模式中的消息定义

本附录以文本形式描述了消息格式。消息格式由Json（JavaScript对象表示法）模式来表示。消息顺序的规则在第7章中进行了描述。本文档假定制造软件单元（MSU）交换那些由Json模式所定义的消息。

```
{
// Json Schema of messages in this document
   "$schema": "http://json-schema.org/draft-04/schema#",


   "definitions": {
      // simple types
      "integertype" : { "type" : "integer" },
      "numbertype" : { "type" : "number" },
      "datetype" : { " type" : "string",
               "format": "date",
               "description": "ISO8601/RFC3339 formats for date",
      },
"timestamptype" : {" type" : "string",
               "format": "date-time",
               "description": "ISO8601/RFC3339 formats for date-time ",
      },
       // composite types
       "clause": {
        "type": "object",
        "properties": {
          "name": { "enum": [ "sender", "receiver","time-stamp",
 "message-id","in-reply-to","language"} ,
         "value":      {
          "anyOf" : [
              {"$ref" : "#/definitions/integertype" },
              {"$ref" : "#/definitions/numbertype" },
                    {"$ref" : "#/definitions/datetype" },
                    {"$ref" : "#/definitions/timestamptype" },
                    {"type" : "string"},
                  ],
               },
              },
              "required": ["name", "value"]
            },
            //
```

```
          "clauses" : { "type" : "array",
          "items" : {"$ref" : "#/definitions/clause"                    },
            },


          "NumberOfpieces": { "type": "object", "properties": {
              "name": {"type":"string" } ,
              "value": {"$ref" : "#/definitions/integertype" },
            },
            "required": ["name", "value"],
          },
          "Quantity": { "type": "object", "properties": {
              "name":{"type": "string" } ,
            "value": {"type" : "object",
                       "properties" : {
                         "value" :     {"$ref" : "#/definitions/numbertype"},
                         "measure" : { "type" : "string" }
                       },
                       "required" : ["value","measure"],
              },
              "descriptions" : "       ",
        },
        "required": ["name", "value"],
      },
      "stringValues": { "type":
          "object",
          "properties": {
           "name": {"type": "string" },
           "value":    {
             "anyOf" : [
               {"$ref" : "#/definitions/datetype" },
               {"$ref" : "#/definitions/timestamptype" },
               {"$ref" : "#/definitions/stringtype" },
             ],
           },
           "descriptions": "this object includes not only string value, but also
ISO8601/RFC3339 formats for date, time",
        },
        "required": ["name", "value"],
      },
      "CollectionValues": { "type": "object",
      "properties": {
          "name":{"type": "string" } ,
```

```
        "value": { "type": "array", "items": {
            "$ref" : "#/definitions/nameValuePair",
            },
        },
      },
      "required": ["name", "value"],
   },


 //
         "nameValuePair": { "type": "object", "properties": {
                "anyOf": {
                  "$ref":"#/definitions/NumberOfpieces",
                  "$ref" : "#/definitions/Quantity",
                  "$ref" : "#/definitions/stringValues" ,
                  "$ref" : "#/definitions/CollectionValues",
                 },
              },
         },
         "ordereditem": {
           "$ref": "#/definitions/nameValuePair"
         },
         // component "checker": {
           "type": "object", "properties": {
             "name":{  "enum":  ["number-of-records",  "check-sum"]  }  ,  "value":
                          { "type": "integer" },
            },
           "required": ["name", "value"], "minitems":1,
         },
         "PerformativeMessageBody" : { "properties" :{
                "order" : {"type" : "string"},
                "ordereditems" : { "type" : "array",
                    "items" : { "type" : "array", "items" : {
                             "$ref" : "#/definitions/ordereditem"
                           },
},
             },
        },
        "required" : ["order","ordereditems"],
      },
   },
```

```
"type": "object",
"properties": {
  // control messages
  "NofifyMessage":{       "type" : "object",
    "properties" :{
     "controlVerb" : {"enum": [ "Notify"]},
     "clauses": {"$ref" : "#/definitions/clauses"},
      "message-content": {"type" : "string" },
     },
      "required": ["controlVerb","clauses","message-content"],
      "additionalProperties": false
  },
  "AskResponseMessage": {         "type" : "object", "properties" :{
     "controlVerb" : {"enum": [ "AskResponse"]},
     "clauses":{"$ref":"#/definitions/clauses"},
     "message-content":       {"type" : "string",
        "pattern" : "^select[\\s]+[\\(]\\*[\\)][\\s]+where.+$" },
     },
      "required": ["controlVerb","clauses","message-content"],
      "additionalProperties": false
  },
  "WarnMessage": {       "type" : "object", "properties" :{
     "controlVerb" : {"enum": [ "Warn"]},
     "clauses": {"$ref" : "#/definitions/clauses"},
     "message-content":  {"type" : "string"},
             },
           "required": ["controlVerb","clauses","message-content"],
           "additionalProperties": false
       },
       // performative messages
       "RequestMessage": { "type" : "object",
           "properties" : {
             "performative" : {"enum": [ "Request"]},
             "clauses":       {"$ref":"#/definitions/clauses"},
           "message-content":  {"$ref":"#/definitions/PerformativeMessageBody"},
            "checkers" : { "type" : "array",
                  "items" : {"$ref" :         "#/definitions/checker" },
             },
           },
           "required":                 ["performative","clauses","message-content"],
           "additionalProperties": false
      },
       "PromiseMessage": { "type" : "object",
```

```
          "properties" : {
           "performative" : {"enum": [ "Promise"]},
           "clauses":      {"$ref":"#/definitions/clauses"},
         "message-content":  {"$ref":"#/definitions/PerformativeMessageBody"},
          "checkers" : { "type" : "array",
                "items" : {"$ref" :         "#/definitions/checker" },
           },
         },
         "required": ["performative","clauses","message-content"],
         "additionalProperties": false
    },


   "CounterMessage": { "type" : "object",
          "properties" : {
           "performative" : {"enum": [ "Counter"]},
     "clauses":      {"$ref":"#/definitions/clauses"},
   "message-content": {"$ref":"#/definitions/PerformativeMessageBody"},
    "checkers" : { "type" : "array",
          "items" : {"$ref" :         "#/definitions/checker" },
     },
   },
   "required":["performative","clauses","message-content"], "additionalProperties":
   false
},
"AcceptMessage": { "type" : "object",
"properties" : {
     "performative" : {"enum": [ "Accept"]},
     "clauses":      {"$ref":"#/definitions/clauses"},
   "message-content": {"$ref":"#/definitions/PerformativeMessageBody"},
     "checkers" : { "type" : "array",
          "items" : {"$ref" :         "#/definitions/checker" },
     },
   },
   "required":["performative","clauses","message-content"], "additionalProperties":
   false
},
"CancelMessage": { "type" : "object",
   "properties" : {
     "performative" : {"enum": [ "Cancel"]},
     "clauses":      {"$ref":"#/definitions/clauses"},
   "message-content": {"$ref":"#/definitions/PerformativeMessageBody"},
     "checkers" : { "type" : "array",
```

```
            "items" : {"$ref" :         "#/definitions/checker" },
          },
        },
        "required":["performative","clauses","message-content"], "additionalProperties":
        false
},
        "DeclineMessage": { "type" : "object",
        "properties" : {
            "performative" : {"enum": [ "Decline"]},
            "clauses":      {"$ref":"#/definitions/clauses"},
          "message-content":  {"$ref":"#/definitions/PerformativeMessageBody"},
            "checkers" : { "type" : "array",
                "items" : {"$ref" :         "#/definitions/checker" },
            },
          },
          "required": ["performative","clauses","message-content"],
          "additionalProperties": false
        },
        "ReportCompletionMessage": { "type" : "object", "properties" : {
            "performative" : {"enum": [ "ReportCompletion"]},
            "clauses":      {"$ref":"#/definitions/clauses"},
          "message-content":  {"$ref":"#/definitions/PerformativeMessageBody"},
            "checkers" : { "type" : "array",
                "items" : {"$ref" :         "#/definitions/checker" },
            },
          },
          "required": ["performative","clauses","message-content"],
          "additionalProperties": false
        },
        "DeclareCompleMessage": { "type" : "object",
          "properties" : {
            "performative" : {"enum": [ "DeclareComplete"]},
            "clauses":      {"$ref":"#/definitions/clauses"},
          "message-content":  {"$ref":"#/definitions/PerformativeMessageBody"},
            "checkers" : { "type" : "array",
                "items" : {"$ref" :         "#/definitions/checker" },
            },
          },
        "required":["performative","clauses","message-content"], "additionalProperties":
        false
    },
    "DeclineReportMessage": { "type" : "object",
        "properties" : {
```

```
      "performative" : {"enum": [ "DeclineReport"]},
      "clauses":      {"$ref":"#/definitions/clauses"},
    "message-content": {"$ref":"#/definitions/PerformativeMessageBody"},
      "checkers" : { "type" : "array",
           "items" : {"$ref" :         "#/definitions/checker" },
      },
    },
    "required":["performative","clauses","message-content"], "additionalProperties":
    false
  },
},
```

附 录 B
（资料性）
管理层

B.1 生产管理系统层

生产管理系统由子系统组成。本文件中假设的生产管理系统的子系统如图B.1所示。此外，本标准的子系统可以与IEC 62264定义的0层到4层的功能层次结构相对应。

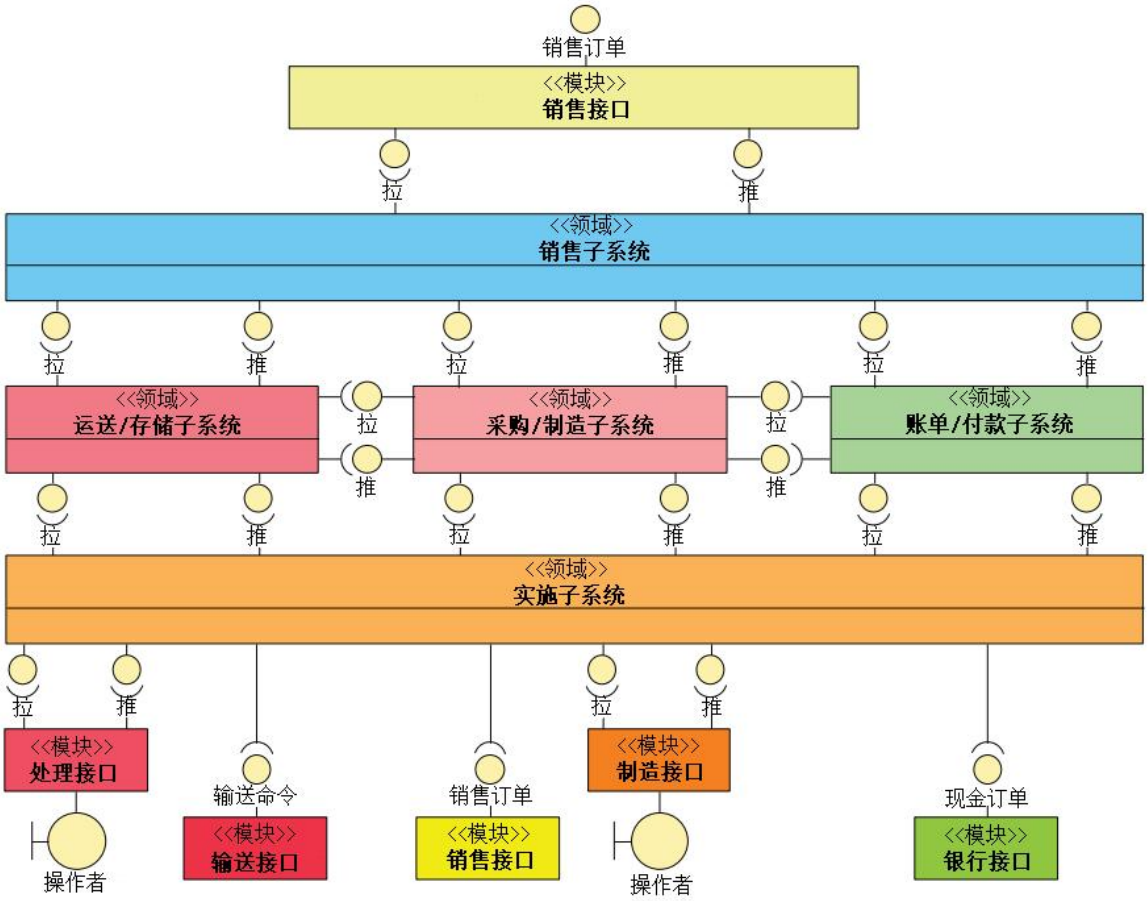


图 B.1 生产管理系统总体结构

B.2 子系统

a) 销售子系统：销售子系统负责从确认订单的接收到订单的完成（请求或存款）这个生命周期。该子系统对应于 IEC 62264 功能层次结构的第 4 层。对订单进行采购还是制造则取决于商品本身。

b) 采购/制造子系统：采购/制造子系统作为供应商来规划销售子系统发出的采购或生产订单（采购/制造）。该子系统对应于 IEC 62264 功能层次结构的第 4 层。

c) 运送/存储子系统：运送/存储子系统规划物流工作，例如接受和交付在“购买/制造”子系统中生产或购买的零件。该子系统对应于 IEC 62264 功能层次结构的第 4 层。

d) 索赔/付款子系统：索赔/付款子系统制定结账和购买的付款计划，并按照销售子系统发出的订单进行购买。该子系统对应于 IEC 62264 功能层次结构的第 4 层。
e) 实施子系统：实施子系统具体实现物流订单、制造和采购订单、存款订单及上级子系统规划订单，以及控制物流或制造设备并记录结果。

B.3 请求消息中订单示例

订单值取决于子系统。表B.1中列出了常用订单值的示例。

表 B.1 常用订单值

C子系统	订单	描述	P子系统	谁	什么	多少	多少	何时完成	来处	去处
销售接口	销售	销售商品	销售	消费者	商品	个数/数量	付款	预定日期	—	目的地
销售	运输	运输商品	运输/存储	销售商	商品	个数/数量	—	运输日期	运输来源	目的地
销售	采购	采购商品	采购/制造	销售商	商品	个数/数量	付款	到达日期	购买者	仓库
销售	制造	制造商品	采购/制造	销售商	商品	个数/数量	—	完成日期	—	仓库
销售	存储	接收商品	运输/存储	销售商	商品	个数/数量	个数/数量	到达日期	运输来源	仓库
销售	索赔	将会支付到账户	索赔/付款	销售商	账户	—	付款	收付日期	—	付账目的地
采购/制造	采购	采购部件	采购/制造	工厂	项目	个数/数量	—	预定日期	购买者	工厂
采购/制造	制造	制造部件	采购/制造	工厂	项目	个数/数量	—	完成日期	工厂	工厂
采购/制造	存储	接收部件	运输/存储	工厂	项目	个数/数量	—	到达日期	运输来源	仓库
采购/制造	运输	分配产品	运输/存储	工厂[2]	项目	个数/数量	—	分配日期	工厂	仓库
采购/制造	存储	接收产品	运输/存储	工厂	项目	个数/数量	—	收到日期	工厂	仓库
采购/制造	付款	将会支付到账户	索赔/付款	购买	账户	—	付款	收付日期	—	收款人
采购/制造	实施	实施制造	实施（制造）	工厂	—	个数/数量	—	—	—	—
运输/存储	实施	实施运输	实施（运输）	仓库	—	—	—	—	仓库	转运公司
运输/存储	实施	实施接受	实施（存储）	仓库	—	—	—	—	转运公司	仓库
索赔/付款	实施	索赔金钱	实施（索赔）	销售商	账户	—	付款	付账日期	—	付账目的地
索赔/付款	实施	付款并相应开发票	实施（付款）	购买	账户	账户	付款	付款日期	—	收款人

B.4 C 子系统、P 子系统和服务提供商的协作

UML2.4中的协作图对交互功能进行了描述。MSU之间的交互使用UML来表达一种协作性的发生。MSU之间的交互如图B.2所示，以解释子系统之间的连接。进行交互的制造软件单元发生了C子系统与P子系统的对话。图B.2中A子系统的MSU和B子系统的MSU通过C-P对话的发生来显示一种交互行为。图B.2中的矩形表示MSU,椭圆形表示功能。紧挨着（MSU与功能连接线的文本显示C-P对话功能的生命线。在A和B之间的对话中，图B.2显示C-P对话的C子系统的角色分配给了A，而C-P对话的P子系统的角色则分配给了B。

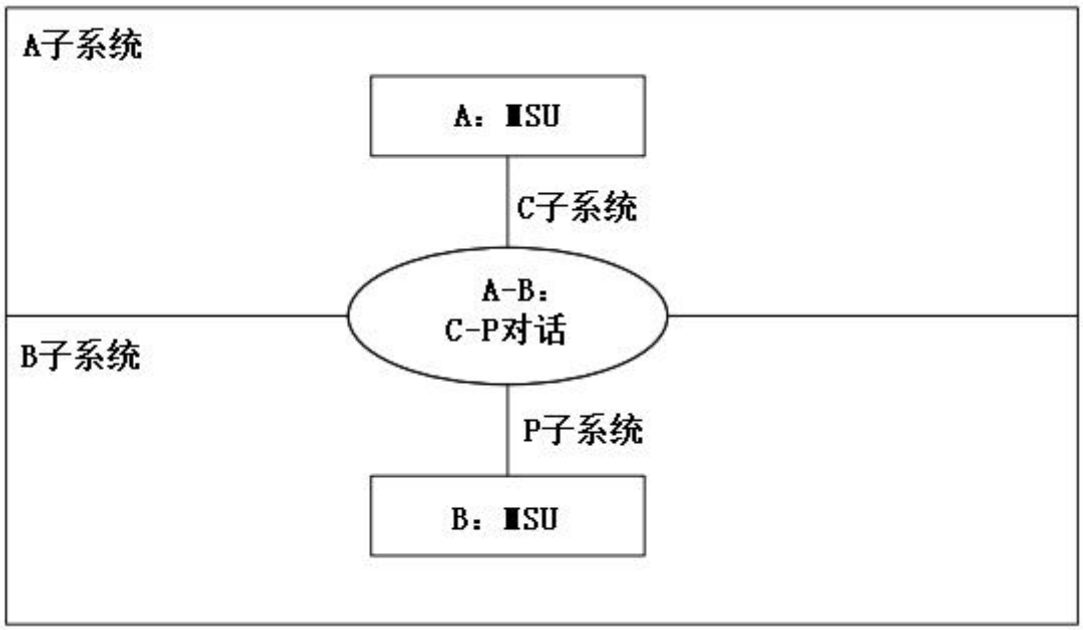


图 B.2 A 子系统与 B 子系统之间 C 与 P 对话的实例

图B.3显示了销售子系统与存储子系统的MSU是如何交互的。

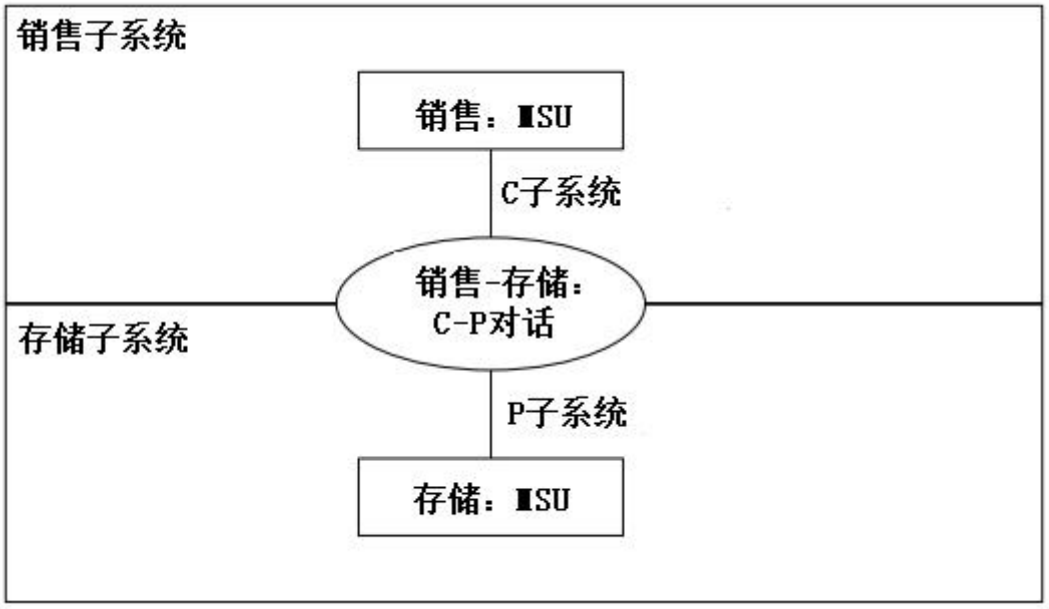


图 B.3 销售子系统与存储子系统之间的 C-P 对话实例

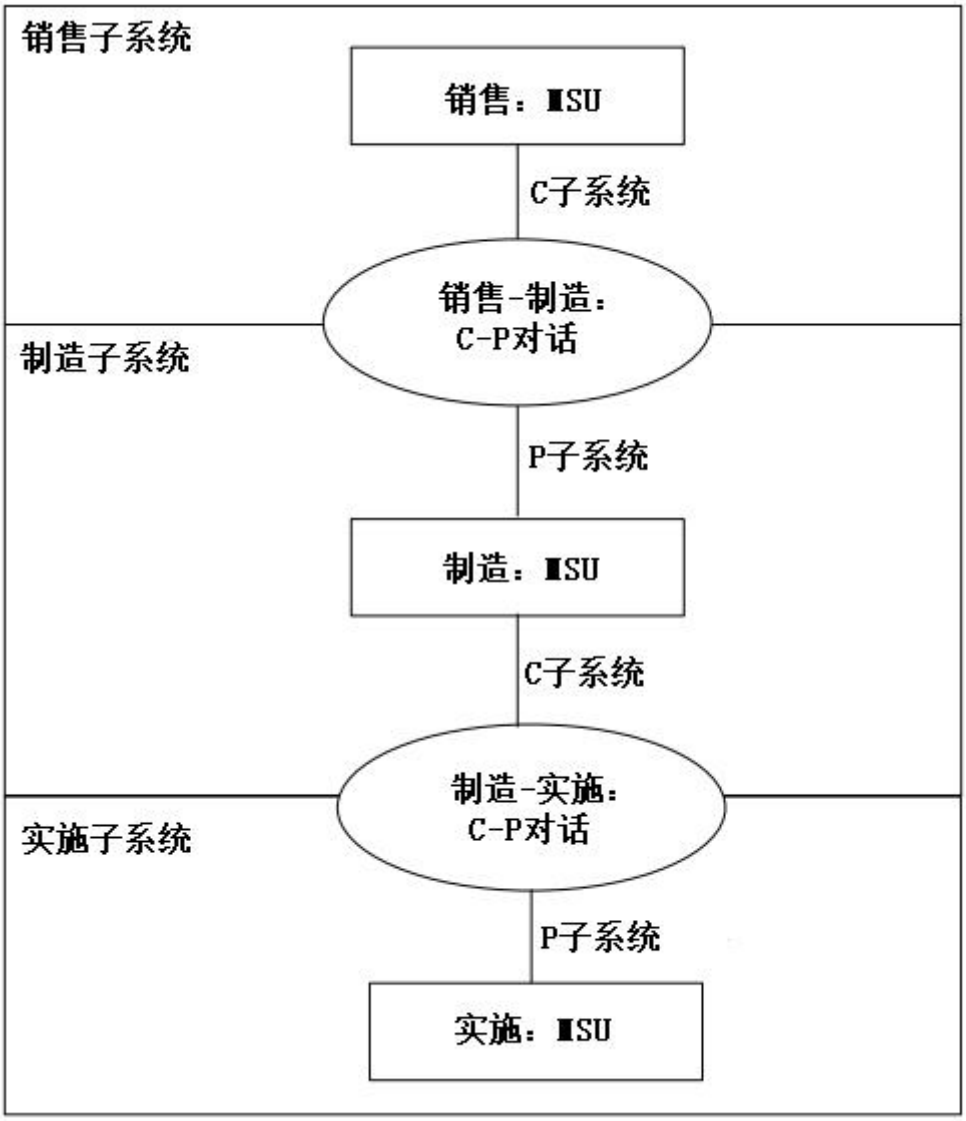
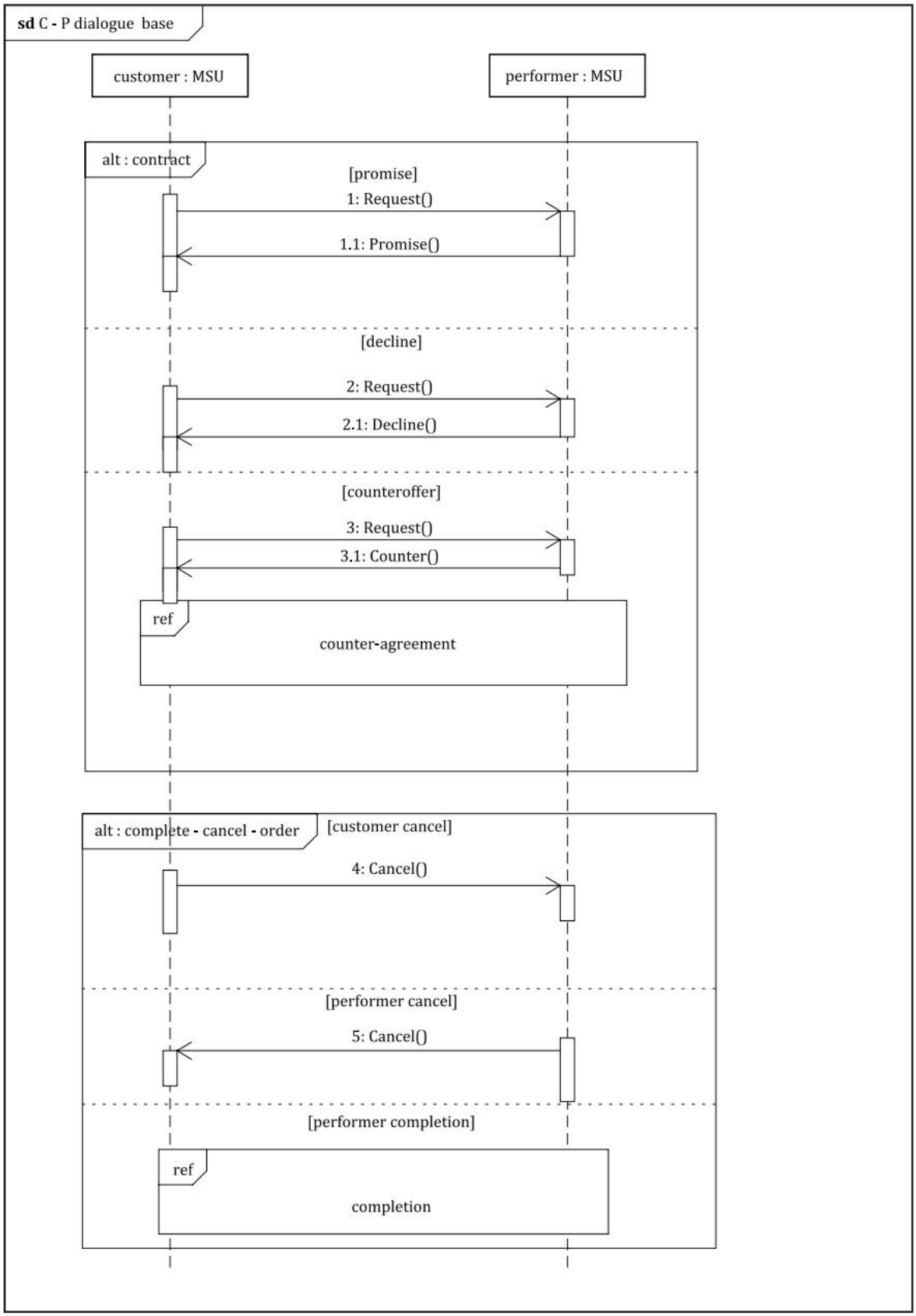


图 B.4 销售子系统与制造子系统之间和制造子系统与实施子系统之间的 C-P 对话实例

图B.4显示制造子系统与实施子系统交互，而且一个MSU具有两种角色（P子系统和C子系统）。销售子系统充当P子系统的角色，而实施子系统充当C子系统的角色。

B.5 C-P 对话基础顺序图



图B.5 C-P对话基础顺序图

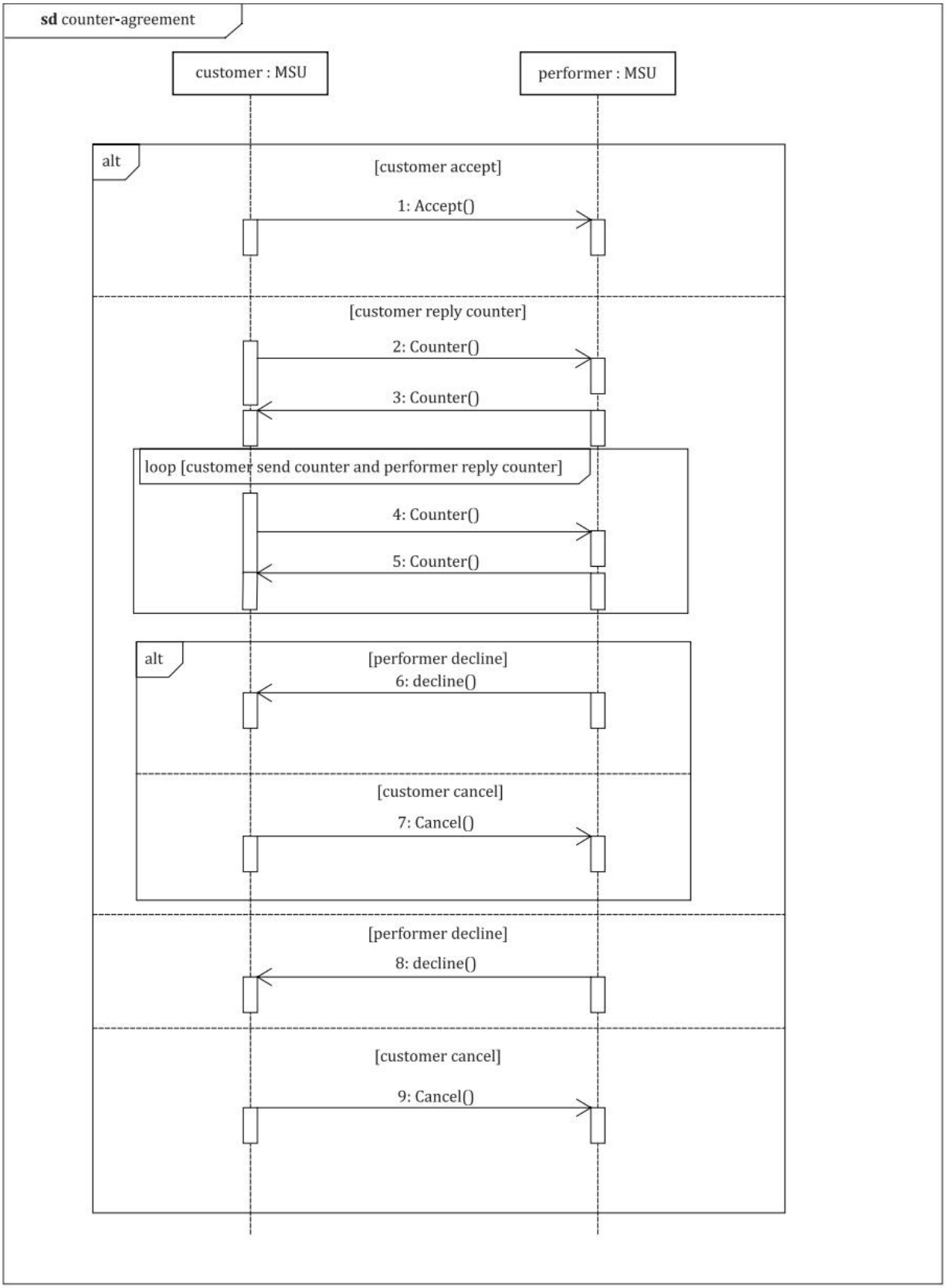

sd counter-agreement
customer : MSU
performer : MSU
alt
[customer accept]
1: Accept()
[customer reply counter]
2: Counter()
3: Counter()
loop [customer send counter and performer reply counter]
4: Counter()
5: Counter()
alt
[performer decline]
6: decline()
[customer cancel]
7: Cancel()
[performer decline]
8: decline()
[customer cancel]
9: Cancel()

图 B.6 返回协议顺序图

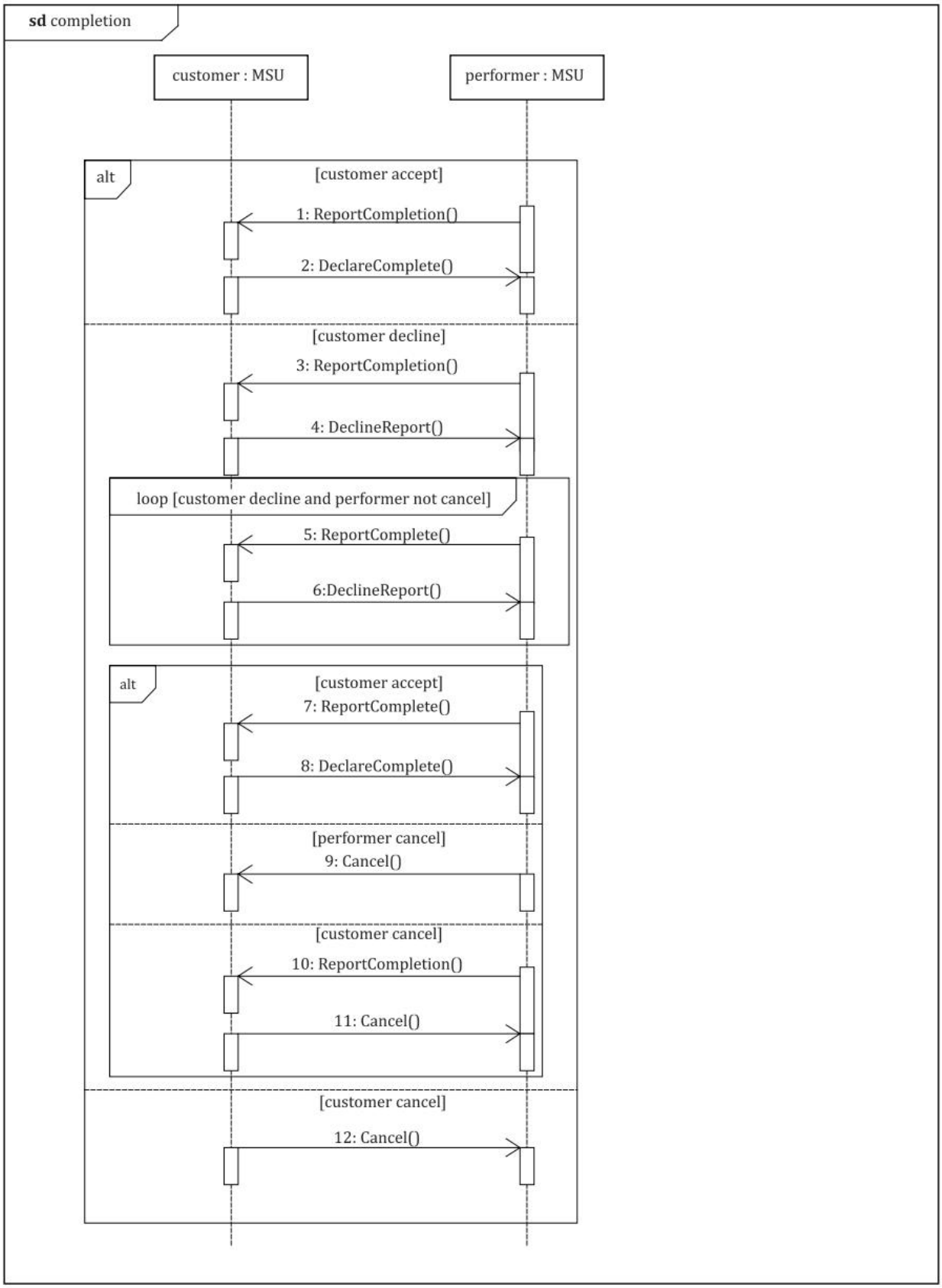

sd completion
customer : MSU
performer : MSU
alt
[customer accept]
1: ReportCompletion()
2: DeclareComplete()
[customer decline]
3: ReportCompletion()
4: DeclineReport()
loop [customer decline and performer not cancel]
5: ReportComplete()
6:DeclineReport()
alt
[customer accept]
7: ReportComplete()
8: DeclareComplete()
[performer cancel]
9: Cancel()
[customer cancel]
10: ReportCompletion()
11: Cancel()
[customer cancel]
12: Cancel()

图 B.7　完成顺序图

附 录 C
（资料性）
客户-实施者模式

客户与实施者之间的对话如图1的对话转换图所示，状态的转换通常从1、2、3、4到5。以下的描述使用了图1中对话转换图中所示的状态。图C.1显示了图1中呈现的普通客户和实施者之间的信息流与客户和实施者的状态是相对应的。

实施者可以在状态2时拒绝客户的订单，并在状态3时取消客户订单。在拒绝或撤销订单后，客户与实施者之间的对话则结束。

客户可以在状态2’、状态3和状态4中自行取消订单。订单取消之后，客户与实施者之间的对话则结束。

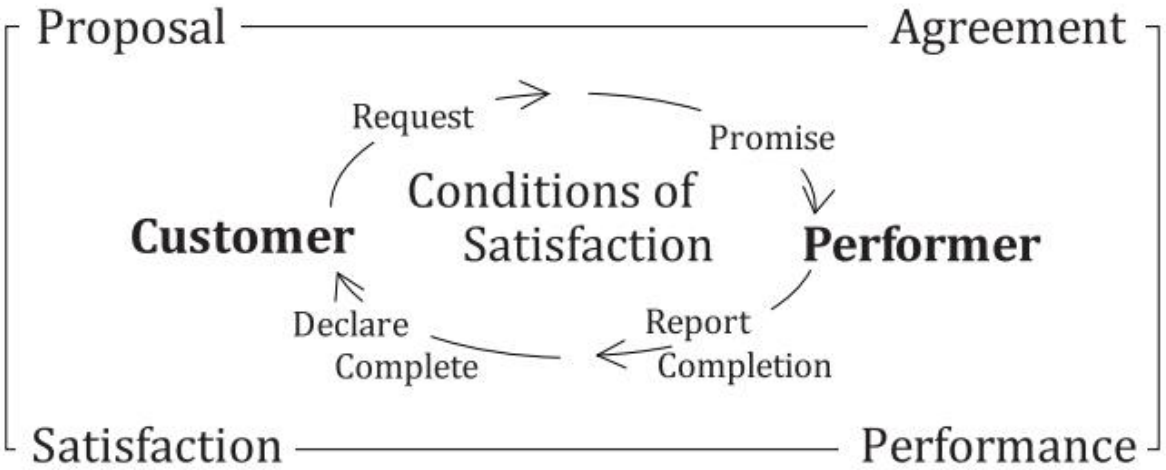


图 C.1 客户-实施者模式

实施者可以对客户订单内容进行协商。

实施者针对订单内容向客户还盘，并转换到状态2’。

客户选择以下其中一个行动：

接受实施者还盘，并从状态2’转换到状态3。

拒绝实施者的还盘并结束对话。

针对实施者的还盘发送一个反向还盘，然后过渡到状态2。

当客户向实施者返回还盘时，实施者可选择以下操作之一：

拒绝并结束对话。

接受并向客户返回承诺消息，并转换到状态3。

对客户的对策作出回应，并返回到状态2。

附 录 D
（资料性）
消息示例

第7章中定义的消息示例使用了简单的场景：

此场景是客户和实施者之间的正常对话，如下所示：

1.C子系统向P子系统发送一个通知（Notify）消息。

2.P子系统将一个要求响应（Ask Response）消息发送到C子系统。

3.C子系统向P子系统发送请求消息。

4.P子系统向C子系统发送承诺（Promise）消息。

5.P子系统将报告完成消息（Report Completion message）消息发送到C子系统。

6.C子系统向P子系统发送声明完成消息（Declare Complete message）。

在此对话示例中，实施者收到客户的两个订单，然后针对这两个订单发送承诺（Promise）消息。实施者发送承诺（Promise）消息后，对话即以订单完成报告和完成声明而结束。本示例使用“C”作为C子系统的标识符，使用“P”作为P子系统的标识符并应用以下规则：

订单的标识符以 "#”开头，然后紧跟5位数字。

“盒子”的规格以宽度（W）、高度（H）和深度（D）描述。

示例在附录A中以Json格式定义。

Notify message例子：。

Notify message 例子：

```
{
    "NotifyMessage":
    { "controlVerb" :
    "Notify", "clauses" :
      [{"name" : "sender","value":"C"},
       {"name" : "receiver", "value" : "P"},
       {"name" : "time-stamp","value" : "2015-10-10T10:15:30"},
       {"name" : "message-id","value" : "3781209846309"},
       {"name" : "language" , "value" : "zzzz"}
      ],
     "message-content" : "",
     }
}
```

AskResponse message 例子：

```
{
    " AskResponseMessage":
    { "controlVerb" :
    "AskResponse", "clauses" :
     [{"name" : "sender","value":"P"},
      {"name" : "receiver", "value" : "C"},
      {"name" : "time-stamp","value" : "2015-10-10T10:15:45"},
```

```
      {"name" : "message-id","value" : "3781209848823"},
      {"name" : "language" , "value" : "zzzz"}
       ],
 "message-content":"select(*),where(time-stamp>\"2015-10-10,09:12:28\")",
    }
}
```

Request message 例子：

```
{
 "RequestMessage":{
      "performative" : "Request",
      "clauses" :
      [{"name" : "sender","value":"C"},
       {"name" : "receiver", "value" : "P"},
       {"name" : "time-stamp","value" : "2015-10-10T10:15:45"},
       {"name" : "message-id","value" : "37812309854398"},
       {"name" : "language" , "value" : "zzzz"}
      ],
       "message-content" : {
          "order" : "Buy",
          "ordereditems" :
           [
            [{"name":"o-id","value" : "#00004"},
             {"name" : "what", "value":"Box"},
                   {"name" : "spec", "VALUE" : [{"name" : "W" , "value" :100},
      {"name" : "H" , "value" :20},{"name" : "D" , "value" :30}]},
                   {"name" : "how_many", "value":10},
                   {"name" : "when_by", "value":"2015-10-20"},
                ],[
                 {"name":"o-id","value" : "#00005"},
               {"name" : "what", "value":"Box"},
                 {"name" : "spec", "value" : [{"name" : "W" , "value" :110},
      {"name" : "H" , "value" :30},{"name" : "D" , "value" :30}]},
                 {"name" : "how_many", "value":15},
                 {"name" : "when_by", "value":"2015-10-20"},
                 ]
               ]
            },
           "checkers" :
            [
             {"name" : "number-of-records","value" : 2},
             {"name" : "check-sum","value" : 13},
```

```
            ]
          }
        }
```

Promise message 例子：

```
{
"PromiseMessage":{
      "performative" : "Promise",
      "clauses" :
      [{"name" : "sender","value":"P"},
       {"name" : "receiver", "value" : "C"},
       {"name" : "time-stamp","value" : "2015-10-10T10:20:03"},
       {"name" : "message-id","value" :"37812311059863"},
       {"name" : "language" , "value" : "zzzz"}
      ],
 "message-content" : {
    "order" : "Buy",
    "ordereditems" :
     [
                  [{"name":"o-id","value" : "#00004"},
                   {"name":"o-id","value" : "#00005"}
           ]
      ],
 },
  }
}
```

ReportCompletion 消息例子：

```
{
 "ReportCompletionMessage":{
      "performative" : "ReportCompletion",
      "clauses" :
      [{"name" : "sender","value":"P"},
       {"name" : "receiver", "value" : "C"},
       {"name" : "time-stamp","value" : "2015-10-13T14:16:54"},
       {"name" : "message-id","value" :"37812312234458"},
       {"name" : "language" , "value" : "zzzz"}
      ],
       "message-content" : {
          "order" : "Buy",
          "ordereditems" :
```

```
            [
             [{"name":"o-id","value" : "#00004"},
             {"name" : "what", "value":"Box"},
             {"name" : "spec", "VALUE" : [{"name" : "W" , "value" :100},
{"name" : "H" , "value" :20},{"name" : "D" , "value" :30}]},
             {"name" : "how_many", "value":10},
             {"name" : "when_by", "value":"2015-10-20"},
                ]
                 ],
              },
         }
      }
```

DeclareCompleMessage 例子：

```
      {
       "DeclareCompleMessage":{
            "performative" : "DeclareComplete",
            "clauses" :
            [{"name" : "sender","value":"C"},
             {"name" : "receiver", "value" : "P"},
             {"name" : "time-stamp","value" : "2015-10-13T14:20:32"},
             {"name" : "message-id","value" :"378123123487633"},
             {"name" : "language" , "value" : "zzzz"}
            ],
             "message-content" : {
                "order" : "Buy",
                "ordereditems" :
                 [
                  [{"name":"o-id","value" : "#00004"},
                  ]
                 ]
              },
         }
      }
```

附 录 E
（资料性）
定制对话状态转换图示例

E.1 C 子系统与 P 子系统之间对话的个性化定制

此示例假定其符合日本的商业惯例，并且是对图1进行个性化定制的一个示例。

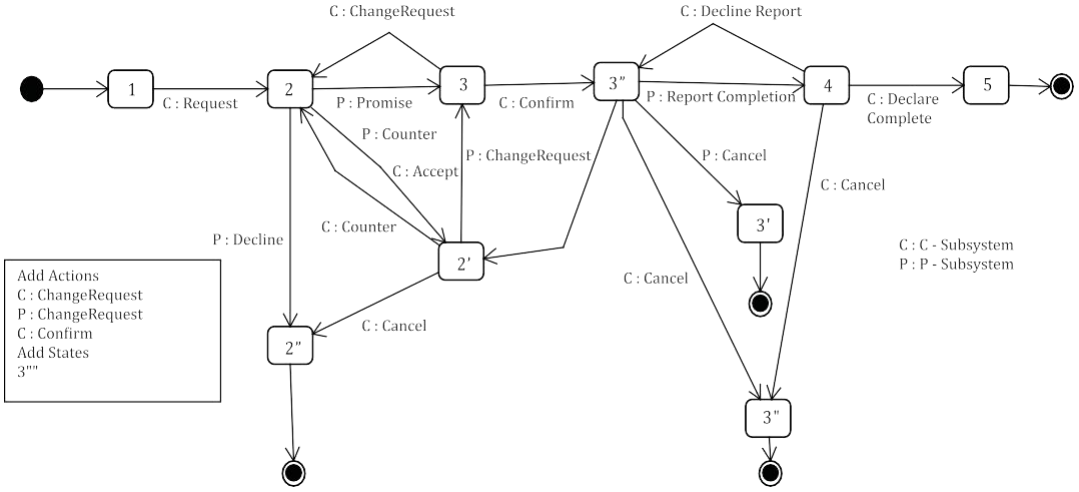


图 E.1 对话定制状态转换图

表 E.1 Extra PerformativeMessage 分类类型

E.1 消息类型	行动	描述
更改请求	C: ChangeRequest P: ChangeRequest	C子系统向P子系统发送一个ChangeRequest消息。 P子系统向C子系统发送一个ChangeRequest消息。
确认	C: Confirm	C子系统向P子系统发送一个Confirm消息。

E.2 附加消息

添加消息和PerformativeMessage的原因如下：

对来自C子系统的请求（Request）消息进行响应的承诺（ Promise）消息所针对的是一个临时订单的消息，且订单由C：Confirm消息确定。

在订单确定之前，C子系统可以更改订单。

由于意外情况，P子系统可以要求C子系统对启动之前所承诺的订单进行一次更改。

更改请求消息 (C: ChangeRequest)

在P子系统对C子系统承诺订单后，C子系统可以更改订单内容。可以更改的内容是“什么（what)”，“规格（spec）”，“序列号（serno）”，“多少（how_many）”，“多少 （how_much）”，“何时完成（when_by）”，“出处（where_from）”，“去处（where_to）”和“负责人（whom_incharge)”。这将导致图E.1从状态3转换到状态2。

— verbtype="performative"

— verb="ChangeRequest"

— clause = (("sender","Cxxxx"),("receiver","Pyyyy"),...,("language","zzzz"))
— order="Buy"
— ordereditems=(("o-id","#0004"),("when_by","2017-10-10"))
它显示了C子系统更改交货日期的情况。
— checker=(("number-of-records",1), ("check-sum",13))
确认消息 （Confirm message）
订单内容落实到P子系统之后，C子系统发出不能更改订单内容的通知。这将导致图E.1中的状态3转换到状态 3’’’’ 。
— verbtype="performative"
— verb="Confirm"
— clause = (("sender","Cxxxx"),("receiver","Pyyyy"),...,("language","zzzz"))
— order="Buy"
— ordereditems=(("o-id","#0004"))
— checker=(("number-of-records",1), ("check-sum",13))
更改请求消息 (P: ChangeRequest)
这是一个P子系统发送到C子系统的消息,询问之前有关由于不可预见情况需对承诺的订单内容进行变更的事宜。如果C子系统有可能向P子系统进行重新订购，那么C子系统最好借此重新订购。可以更改的内容是“什么（what）”、“规格（spec）”、“序列号（serno）”、“多少（how_many）”“何时完成（when_by）”、“出处（ where_from）”，“去处（where_to）”及“负责人（whom_incharge)”。这将导致图E.1中的状态 3’’’’ 转换到状态2’。客户的行动或是C：接受、C：反对或是C：取消。
— verbtype="performative"
— verb="ChangeRequest"
— clause = (("sender","Pxxxx"),("receiver","Cyyyy"),...,("language","zzzz"))
— order= ="Buy"
— ordereditems=(("o-id","#0004"),("when_by","2017-10-10"))
它显示了C子系统更改交货日期的情况。
— checker=(("number-of-records",1), ("check-sum",13))

E.3 附录 A 中定义的消息扩展

```
{
"$schema": "http://json-schema.org/draft-07/schema#",
"definitions": {
……
    "undefinedtype":       {"enum": [ "UNDEFINE"],
        "descriptions": "Undefine ",},

    "subrangeWithAlternatetype" : { "type" :"string",
        "pattern":"^[0-9]+(\\|+[0-9]+)+(\\.\\.[0-9]+(\\|+[0-9]+)*)+$",  "descriptions":
        "mixed pattern of subrange and alternate, subrange:
10..20 in string means from 10 to 20            ",},
```

```
    "alternatetype" : { "type" :"string",
         "pattern" : "^[0-9]+(\\|+[0-9]+)+$",
        "descriptions" : " alternate ex. 10|15 in string means 10 or15",}, "wildcardtype" :
    {                    "type" : "string",
         "pattern" :
         "^(\\?|\\*|[0-9A-Za-z_\\s])+$",
         "descriptions" : "wild card ",},

"nameValuePair": {
       "type": "object",
        "properties": {
            "anyOf": {
              "$ref":"#/definitions/NumberOfpieces",
              "$ref" : "#/definitions/Quantity",
              "$ref" : "#/definitions/stringValues" ,
              "$ref" : "#/definitions/CollectionValues",
              // local extention
              "$ref" : "#/definitions/localNameValuePair"},
          },
     },
// local extension of messages
"localNameValuePair" : {
      "type" : "object",
       "properties" : {
           "name" : {"type" : "string" },
                "value" : {
                  "anyOf" : [
                     {"$ref" : "#/definitions/undefinedtype"},
                     {"$ref" : "#/definitions/alternatetype"},
                     {"$ref" : "#/definitions/subrangeWithAlternatetype"},
                     {"$ref" : "#/definitions/wildcardtype"}
                  ]
               },
```

```
        "measure" : { "type" : "string",
           "descriptions" : "optional measure "},
    },
     "required" : ["name","value"],
},
```

附 录 F
（资料性）
处理对话状态转换、消息和能力专规的解决方案

F.1 概述

本附录说明了制造软件单元（MSU）之间对话功能的实施概要。实现的软件是前端组件，负责制造软件单元（MSU）之间的交互，该组件称为“采用器”。作为对话功能实现的软件组件是采用器和采用器与MSU主体间的接口。本附录仅限于那个实现C子系统与P子系统之间对话中涉及到的制造软件单元（MSU）间对话通信的采用器。采用器实现C子系统的对话部分和P子系统的对话部分。当被用在制造软件单元（MSU）的前端时，采用器（Adopter）可以执行C子系统的功能或P子系统的功能。因此，当一个制造软件单元（MSU）担当C子系统和P子系统的角色时，前端会有两个采用器（Adopters）。一个充当C子系统角色，另一个则充当P子系统角色。图F.1显示了MSU之间第8章中所描述的通道。图F.2显示的示例是，在C子系统的制造软件单元（MSU）与多个P子系统交互时，一个P子系统的制造软件单元（MSU）拥有一个P子系统角色和一个C子系统角色。图F.2还显示了多个来自制造软件单元（MSU）的消息可以通过采用器（Adopter）的通道。采用器可在任何时候访问会话伙伴的能力专规，并获取能力专规中本身为会话伙伴的制造软件单元（MSU）采用器的URI。至此之后，交互消息将被简称为消息。

F.2 设计策略和实施约束

a) 当与另一个制造软件单元（MSU）进行交互时，制造软件单元（MSU）通过采用器进行交互。
b) 采用器已经彼此获取了会话伙伴的制造软件单元（MSU）的能力专规。
c) 当在制造软件单元（MSU）自身上创建 RequestMessage 时，要将订单标识符的值，即发出的 o-id 属性值分派给 PerformativeMessage。
d) 采用器不转换消息，而是由制造软件单元（MSU）主体的组件，即消息转换器来负责转换。
e) 订单标识符。o-id 的值在制造软件单元（MSU）中是独有的。
f) 消息编号在采用者中内部是独有的。

F.3 采用器的主要功能

a) 当把一个消息发送到另一个制造软件单元（MSU）时，采用器将制造软件单元（MSU）接收到的数据转换到一个消息中去，并将通信所需的项目添加到该消息中去，以及将消息发送到其他制造软件单元（MSU）的统一资源标识符（URI）那里。需要添加的项目是消息号、发送日期，校验和等。
b) 当接收到来自另一制造软件单元（MSU）发送的消息时，采用器将从接收到的消息中排除那些仅用于消息通信的项目，只有制造软件单元（MSU）应用程序所请求的数据才会被传递到制造软件单元（MSU）的主体上去。
c) 当检测到通信中发生了 7.2.3 中所描述的错误时，采用器将输出系统中共享的日志，然后发送一个错误信息并停止处理过程。C 子系统的错误会被发送到 C 子系统，而 P 子系统的错误则被发送到 C 子系统。
d) 输出日志中包括发送消息、接收消息及以上 c）中提及的错误。

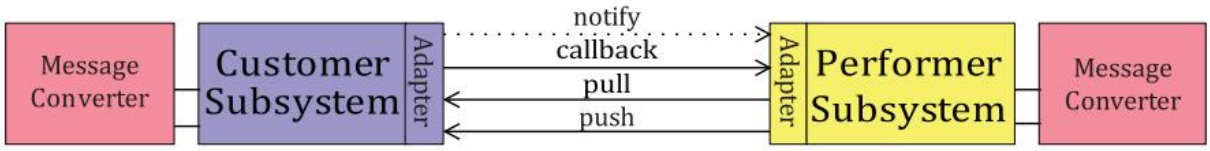


图 F.1 C 子系统采用器与 P 子系统采用器之间的通道

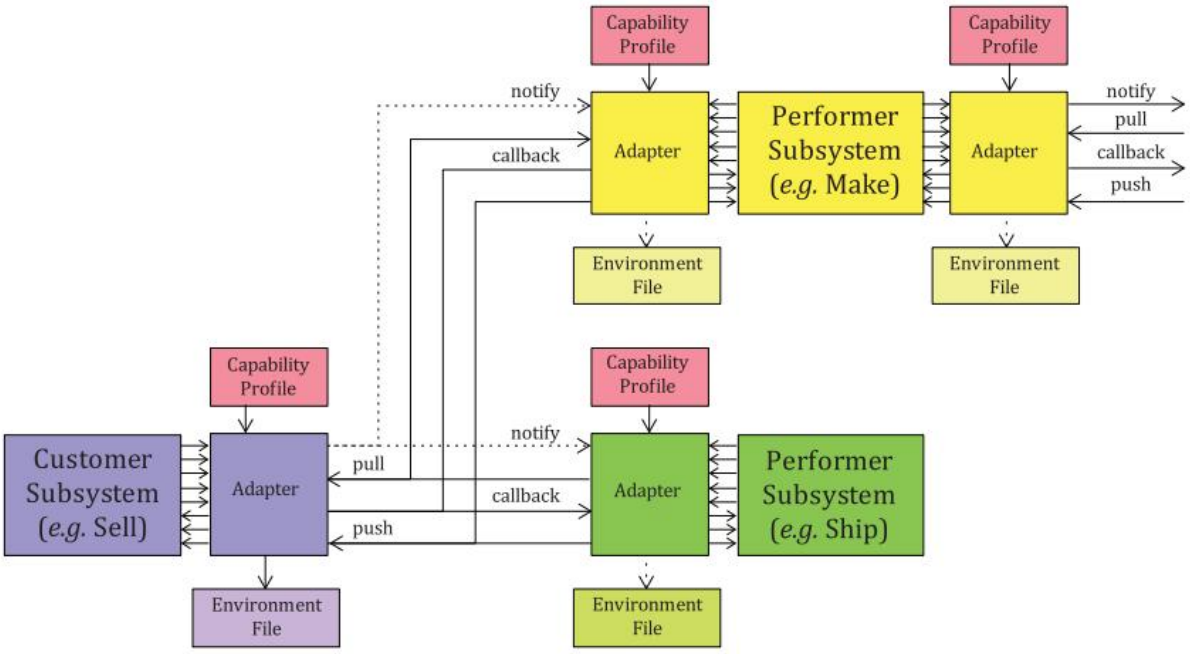


图 F.2 众多采用器的连接

附 录 G
（资料性）
OAGiS 与本文件之间的动词映射表

OAGiS的消息格式包含的信息远远超出本文件中的信息，且OAGiS的消息范围比本文件范围更广。本文件对话中的消息是异步交换，而在OAGiS的对话中，消息则既是同步也是异步进行交换的。因此，映射表显示的OAGiS与本文件之间的关系仅限于动词。

表 G.1 OAGiS 与本文件之间的动词映射表

	本文件的消息类型	
动词(OAGiS)	控制消息动词	执行消息动词
Cancel	—	Cancel
Change	—	Counter C to P
Process	—	Request
Notify	Notify	ReportCompletion
Get	Ask Response	—
Acknowledge	—	Promise ,Accept, DeclareComplete
ConfirmBOD	—	—
Respond	Warn	Counter P to C , Decline, DeclineReport
Show	—	—
Sync	—	—

注：C到P表示C子系统到P子系统，P到C表示P子系统到C子系统。

参 考 文 献

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

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

[3] ISO/IEC 19505-2, Information technology — Object Management Group Unified Modelling Language (OMG UML) — Part 2: Superstructure

[4] IEC 62264-1, Enterprise-control system integration — Part 1: Models and terminology

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

[6] IEC 62264-5, Enterprise-control system integration — Part 5: Business to manufacturing transactions

[7] OAGIS 10.4 Standard Edition XML http://www.datypic.com/sc/oagis10/ss.html 2018.6.5

[8] IETF STD 90 The JavaScript Object Notation (JSON) Data Interchange Format https://tools.ietf.org/html/std90 2018.6.5

[9] JSON Schema draft-07 http://json-schema.org/ 2018.6.5