

# 企业业务核心系统之核心数据模型 设计文档

Liu Baogang

[liubaogang@outlook.com](mailto:liubaogang@outlook.com)

[liubaogang@open-care.com](mailto:liubaogang@open-care.com)

(+86) 13910123818

益先科技（北京）有限公司

2021 年 10 月

## 更改历史

主 题							
保密说明							
更改历史							
版 本	章 节	类 型	更改日期	更改人	复核日期	复核人	更改说明
0.1		C	2017. 04. 06 ~ 2017. 06. 19	Liu Baogang			原《生命汇系统蓝图设计.docx》中与数据模型相关的内容，涵盖：产品、订单/销售策略/卡券、医疗数据。
0.2		U	2019. 08. 03	Liu Baogang			原《伸远业务核心系统模块需求说明.docx》中与数据模型相关的内容，是对原《生命汇系统蓝图设计.docx》中与数据模型相关的内容的微调。
0.3		A	2021. 07. 01	Liu Baogang			调整整体文档结构，并添加数据模型： “9、财务账户数据模型” （拷贝自《2020-CRM 系统中财务账户数据模型设计.docx》）
0.4		A	2021. 07. 15	Liu Baogang			添加 “6、人与组织数据模型”
0.5		A	2021. 07. 25	Liu Baogang			添加 “8.3、销售订单数据模型”
0.6		A	2021. 07. 29	Liu Baogang			添加 “8.2、销售过程管理数据模型”
0.7		A	2021. 10. 19	Liu Baogang			添加 “7.3、维度 ‘产品多级具化管理’ ”

说明：类型—创建（C）、修改（U）、删除（D）、增加（A）。



## 目 录

1	摘要.....	8
1.1	文档目的.....	8
1.2	预期读者.....	8
1.3	参考资料.....	8
1.3.1	分析模式相关资料.....	8
1.3.2	医疗数据模型相关资料.....	8
1.3.3	益先科技内部文档.....	9
1.3.4	数据模型设计相关技术的相关资料.....	9
1.4	图例说明.....	9
1.5	TODO.....	9
2	术语及定义.....	10
2.1	软件设计技术相关术语.....	10
2.2	业务术语.....	10
2.2.1	简要业务术语定义.....	10
2.2.2	重点业务概念释义：产品（Product）、资源（Resource）、耗材（Consumptive Material）.....	12
2.2.2.1	概念：物料（Material）、服务（Service）、信息（Information） 12	
2.2.2.1.1	物料.....	12
2.2.2.1.2	服务.....	12
2.2.2.1.3	信息.....	12
2.2.2.2	概念：产品、资源、耗材.....	13
2.2.2.3	产品的属性（Attribute）.....	13
2.2.2.3.1	基本属性.....	13
2.2.2.3.2	技术属性.....	14
2.2.2.3.3	供应属性.....	14
2.2.2.4	物料的属性（Attribute）.....	15
2.2.2.5	资源的属性（Attribute）.....	15
2.3	财务账户相关术语.....	15
2.4	其他业务术语.....	16
3	数据模型设计常用技术.....	16
3.1	数据模型设计者所需具有的预备知识.....	16
3.2	多种实现技术/范式（Paradigm）.....	16
3.3	独立于实现技术的特征/变化点（需求）建模.....	16
3.4	不同泛化级别的数据模型/模式设计.....	17
3.5	节选：Open-Care hpaPaaS 中的 Model-Driven.....	18
3.5.1	Model-Driven-Engineering.....	18
3.5.1.1	Model-Driven 所带来的好处.....	18
3.5.1.2	Model-Driven 中的“多级具化”.....	19
3.5.1.3	动态模型静态化.....	19
3.5.1.3.1	动态模型与静态模型的优缺点.....	19

3.5.1.3.2	动态模型静态化的方式	20
3.6	“RefDoc 1、《数据模型资源手册 卷 1》” 中的部分关键点	20
3.7	所谓“有难度 (Difficulty)”	21
4	核心数据模型涵盖的业务范围	22
4.1	核心数据模型需要支持企业业务核心系统的系统特性	22
4.2	核心数据模型所需涵盖的业务概念/实体	23
4.3	典型的复杂业务场景	24
4.3.1	产品相关业务场景	24
4.3.2	销售相关业务场景	25
5	核心数据模型摘要	26
5.1	产品核心数据模型简介	26
5.1.1	产品打包树数据模型	26
5.1.2	‘产品模板(Product Template)’与‘产品变体(Product Variant)’	27
5.2	销售核心数据模型简介	27
5.3	医疗数据核心数据模型简介	27
5.4	规则的‘具化’过程	28
6	人与组织数据模型	28
6.1	Party、Role、Capability、Relationship	28
6.2	Party 在 Party 之外的业务实体中扮演的 Role（非在 Party 与 Party 之间的 Relationship 上扮演的 Role）	32
7	产品数据模型	33
7.1	产品数据模型变化点的四个维度	34
7.2	产品模板(Product Template)——应对维度‘原子产品配置’、‘产品组合’，涉及维度‘产品多级具化管理’	34
7.2.1	BOM 模板 (BOM Template)	35
7.2.1.1	Product Template/BOM Template 建模的非唯一性	37
7.2.2	服务流程模板 (Process Template)	37
7.2.2.1	Process Template 所用到的 API Service	39
7.2.2.2	Process Template 里所引用到的资源 (Resource)	39
7.2.2.3	与 Process Template 相关的预约	39
7.2.2.4	多流程节点合并预约和自由预约	40
7.2.3	医疗数据模型 (Medical/Health Data Model)	41
7.2.4	健康报告模板 (Health Report Template)	42
7.2.5	技术属性、供应属性的固化和变化规则集	43
7.2.6	自定义特性集 (Characteristic)	43
7.2.7	所属产品分类 (Category) (一个) 与标签 (Tag) (可多个)	44
7.2.8	约束 (Constraints)	44
7.2.9	计算 (Calculations)	44
7.3	维度‘产品多级具化管理’	45
7.3.1	‘产品多级具化’概念说明	45
7.3.1.1	产品变体 (Product Variant)	45
7.3.1.2	Configuration Profile——对产品模板进行‘多级具化’	45
7.3.1.3	物料模板 (Material Template) 与物料变体 (Material Variant)	

7.3.1.4	Configuration Profile—对物料模板进行‘多级具化’	45
7.3.2	‘产品多级具化’的具体应用场景——产品阶段	45
7.3.3	产品阶段数据模型	46
7.3.4	产品阶段相关注意事项	46
7.3.4.1	每个阶段的产品的含义——类还是实例	46
7.3.4.2	上下游阶段的产品的对应关系	46
7.3.4.3	‘规格/规则/具化’可被放于的位置：产品、销售策略、某种单、等	46
7.3.4.4	如何表达‘属于多个产品阶段’的产品	47
7.4	维度‘产品版本管理’	48
8	销售业务数据模型	48
8.1	销售业务概貌数据模型	48
8.2	销售过程管理数据模型	49
8.2.1	销售过程相关的两类 Entity	49
8.2.1.1	销售过程 Pipeline 主线 Entity	49
8.2.1.2	销售过程中的内容 Entity	50
8.2.2	可能的需求变化点	50
8.2.2.1	Object 的创建时机	50
8.2.2.2	Object 的审批时机及并行性	51
8.2.2.3	审批流程	51
8.3	销售订单数据模型	51
8.3.1	订单数据模型设计中需要着重考虑的问题	51
8.3.2	处理“订单/订单条目的增改退换”——订单-Composition	52
8.3.3	处理对已购买的产品/套餐产品的使用和“部分使用”	53
8.3.3.1	预备知识——产品使用过程中对产品个体的区分度	54
8.3.3.2	根据产品树结构和产品/子产品的使用来变换 OCOrderAccountItem	54
8.3.4	处理多重多对多关系	57
8.3.5	处理多场景之间的耦合所进一步带来的复杂性	57
9	财务账户数据模型	57
9.1	账户的基本概念	57
9.1.1	账户（Account）的抽象概念	57
9.1.1.1	Account	57
9.1.1.2	Summary Account 和 Derived Account	58
9.1.1.3	Memo Account	58
9.1.1.4	Posting Rule	58
9.1.2	财务记账中的账户/科目概念	60
9.1.3	本系统中针对财务数据的记账方式	60
9.2	账户的概念和使用场景	60
9.2.1	典型场景中的账户的需求模型	61
9.2.2	账户定义	61
9.2.3	财务记账情形举例	62
9.2.3.1	销售订单的状态	62
9.2.3.2	现货转账销售（S1→S4）	62

---

9.2.3.3	先发货后收款 (S1→S2→S4) .....	62
9.2.3.4	先收款再发货 (S1→S3_1→S3_2→S4) .....	63
9.2.3.5	预收部分货款销售 (S1→S3_1→S3_2→S4) .....	63
9.3	账户数据模型设计 .....	64
9.3.1	Design Model—Accounts .....	64
9.3.2	针对财务这个特定领域的 Account 的模型设计 .....	65
9.3.3	Design Model—Posting Rule .....	65
10	医疗数据的数据模型 .....	66
11	附录：图表目录 .....	66
11.1	图目录 .....	66
11.2	表目录 .....	67

# 1 摘要

## 1.1 文档目的

本文档目的在于描述企业业务核心系统（下称“本系统”）中的核心数据模型。

## 1.2 预期读者

本文档的使用者包括但不限于：

- 需求分析人员
- 项目管理人员
- 软件设计人员
- 软件质量控制人员

## 1.3 参考资料

### 1.3.1 分析模式相关资料

如下材料，可从 <https://www.jianguoyun.com/p/DeHSPE4Q7dfiBhjo6fwD> 获得。

RefDoc 1、《数据模型资源手册 卷 1》

RefDoc 2、《数据模型资源手册 卷 2》

RefDoc 3、《数据模型资源手册 卷 3》

RefDoc 4、OFBiz Datamodel Book

RefDoc 5、《Analysis Patterns》，Chapter 6 "Inventory and Accounting", Chapter 7 "Using the Accounting Model"

RefDoc 6、财务系统分析模式相关的一些论文

<https://www.jianguoyun.com/p/Ddkp0h8Q7dfiBhiXyKkC>

RefDoc 7、David C Hay 在数据模型模式方面的一些著作

### 1.3.2 医疗数据模型相关资料

RefDoc 8、OpenEHR 医疗数据规范

<http://www.openehr.org>



RefDoc 9、《The openEHR Reference Model--Demographic Information Model》

属于 OpenEHR 医疗数据规范的一部分。本文所引用的是 OpenEHR Release 1.0.2 中的《The openEHR Reference Model--Demographic Information Model, Revision 2.0.2》, 2018.08.04

### 1.3.3 益先科技内部文档

RefDoc 10、《生命汇系统蓝图设计.docx》, 2017

RefDoc 11、《伸远业务核心系统模块需求说明.docx》, 2019

RefDoc 12、《OC--健康 IT 开放平台--Health IT Open hpaPaaS--方案介绍--202x.xx.pptx》

RefDoc 13、《Open-Care hpaPaaS 系统设计思路汇编》, 2018

RefDoc 14、《OC--健康险之健康服务运营管理系统--方案介绍--202x.xx.pptx》

RefDoc 15、《OC--高端健检中心综合 IT 系统解决方案（CRM、体检系统、HIS）--简介--202x.xx.pptx》

### 1.3.4 数据模型设计相关技术的相关资料

RefDoc 16、Domain Engineering（领域工程），尤其是其中的 Feature Modeling（特征建模）

参见 <https://www.jianguoyun.com/p/DRVO7egQ7dfiBhiZsUg>（访问密码：4hobeD）中所列举的与 Domain Engineering 相关的材料。

## 1.4 图例说明

本文中的一些图是类图（Class Diagram）。

这些类图中，有的是本文作者用 UML 工具（比如 Visual Paradigm）所画的，这些图符合标准规范比如 UML 2.0 和/或 SysML 等。

还有些类图拷贝自某参考资料，这些类图中的符号、图形的含义，则需参考出处资料中对图例的说明。例如“RefDoc 1、《数据模型资源手册 卷 1》”中的章节“1.7 本书中采用的约定和标准”。

## 1.5 TODO

本节汇总列举本文中的 TODO 事项。

<TODO 2-1、将“4.2、核心数据模型所需涵盖的业务概念/实体”中的术语补充到本章中作为一个独立的小节>.....	10
<TODO 7-1、添加“产品阶段数据模型”相关内容，尤其是类图>.....	46
<TODO 7-2、添加“维度‘产品版本管理’”相关内容>.....	48

<TODO 8-1、添加 UML 图来描述 Party、线索、商机、协议/合同、报价单、销售策略、套餐等 Entity 之间的关系>.....	49
<TODO 8-2、如下“一般性的一致性问题的”详情待补充>.....	52
<TODO 8-3、修正“图 8-2、订单-Composition”来良好表达多重多对多关系>.....	53
<TODO 8-4、用对象图描述如下方式三中的各个操作后的对象状态>.....	56
<TODO 8-5、描述针对“8.3.1、订单数据模型设计中需要着重考虑的问题”的问题(OP3)的解决方案> ..	57
<TODO 8-6、描述针对“8.3.1、订单数据模型设计中需要着重考虑的问题”的问题(OP4)的解决方案> ..	57
<TODO 9-1、账户科目设置，需要根据实际业务进行补充完善>.....	61

## 2 术语及定义

<TODO 2-1、将“4.2、核心数据模型所需涵盖的业务概念/实体”中的术语补充到本章中作为一个独立的小节>

### 2.1 软件设计技术相关术语

下表列举部分易被混淆的技术术语。

表 2-1、软件技术术语表

序号	术语 / 缩略语	全称和解释
1	Entity / 实体	含义类似于“Class / 类”
2	Type / 类型	本文不严格区分 Type 与 Class/Entity。可以将 Type 近似理解为 Class/Entity
3	EntityHierarchy / ClassHierarchy	指代类树（由有继承层次关系的多个类组成）
4	Instance / 实例	含义同“Object / 对象”

### 2.2 业务术语

#### 2.2.1 简要业务术语定义

本小节列举本文档所需用到的术语。对于概念简单的术语，本小节直接给出术语的释义。对于概念复杂的术语，后续章节专门进行释义，本小节给出对那些章节的引用。

表 2-2、业务术语表

序号	术语 / 缩略语	全称和解释
1	用户/系统用户	使用业务系统进行业务操作的操作人员。

2	客户	购买甲方产品的团体或个人，包括团体客户（如法人客户）和个人客户。
3	客人	使用甲方产品（实物型产品或服务型产品）的个人，包括团体客户里的个人以及个人客户。
4	销售策略	销售策略起到的是一个关于产品销售的“规则集合”的作用。详情请参见章节“8、销售业务数据模型”。
5	销售订单/订单	销售订单用于表达一次购买。销售订单里可以包含多个产品。销售订单简称订单。
6	服务单	服务单用于表达对服务型产品的“一次使用”。“一次使用”包含的步骤可能包括：登记、收费、各科室流转检查、出具一份完整健康报告。
7	产品	如不加区分，“产品”可能指的是“产品模板”，也可能指的是“产品变体”。
8	产 品 模 板 (Product Template)	参见章节“7.2、产品模板(Product Template)”。
9	产 品 变 体 (Product Variant)	指基于 Product Template 生成的具体的产品，类似于 SKU。参见章节“7.3.1.1、产品变体 (Product Variant)”。
10	原子产品	原子产品是最小售卖单位或服务单位，也可能是最小需单独核算的物料。如不加区分，原子产品可能指的是“原子产品模板”，也可能指的是“原子产品变体”。
11	组合产品/套餐	“组合产品/套餐”是将原子产品和/或套餐组合形成的套件。也就是说，套餐既可包含原子产品也可包含套餐。
12	具化	指的是在“抽象规则（集）”（如销售策略或产品模板）上施加一个或多个约束后导致生成具体事物”（销售订单或产品变体）的过程。参见章节“5.4、规则的‘具化’过程”。
13	数据字典	将 key-value 类的配置数据统称为数据字典。比如产品来源。
14	字典内容	指字典的具体项目（item），比如产品来源下的“自有”、“代理机构”。
15	机构	<p>“机构”指甲方下辖的或与甲方提供的产品服务相关的“相对独立”的实体（或子公司、分公司），其类型可包括：总部、健管中心、诊所、代理机构、合作机构、医院，等等。</p> <p>机构会有“所属区域”。</p> <p>比如，甲方总部，机构类型为“总部”，机构所属区域为“全国”；甲方北京诊所，机构类型为“诊所”，机构所属区域为“北京”。</p>
16	部门	部门必须隶属于某机构下，其主要作用是组织结构、角色组管理。
17	系统内置用户	系统的顶级授权管理系统用户，即超级管理员；
18	系统管理员	具有业务权限分配和授权权限分配的系统用户。

## 2.2.2 重点业务概念释义：产品（Product）、资源（Resource）、耗材（Consumptive Material）

本小节对产品、资源、耗材等基本概念进行释义，这些概念是后续数据模型的定义（尤其是章节“7、产品数据模型”）的基础。本小节里的产品概念，尚未区分“产品模板（Product Template）”和“产品变体（Product Variant）（具体产品/SKU）”。

本小节里的概念里的一部分，会是本系统所需支持的概念，比如产品、物料、资源以及它们的属性。其他概念，用于便于本文读者理解概念上下文。

### 2.2.2.1 概念：物料（Material）、服务（Service）、信息（Information）

经济交易对象	具体性	库存可能	销售
物料	作为具体的物体而存在	能库存	所有权转移
服务	提供具体资源	不能库存	允许使用权（占有）
信息	只有抽象“意义”（需要借助媒体）	不能库存	允许使用权（不占有=即使交给他人，本人依然持有）

图 2-1、物料、服务、信息

#### 2.2.2.1.1

物料

药品；

物料是具体的事物（实物），它能和其他事物区别，数量可数可量，还能占有和保管。此外，物料的所有权本身可以买卖。

#### 2.2.2.1.2 服务

可买卖 - 洗牙

服务一般指提供某种资源（人力资源、设备资源）。比如提供医疗服务、租赁机械设备、租住宾馆客房等都属于服务。因此，服务的等价报酬通常指根据资源单价计算占用时间的金额。销售的对象是使用权（分时占用或独占权），所以服务的特征是不能“库存”。

#### 2.2.2.1.3

信息

？

例如，研发、设计工作的产物是信息，信息可以用纸质或电子媒介来记录、传播。信息可以被复制、共享（收费或免费），也就是不能“库存”、没有“占用”的概念、也无法“转交所有权”。

医生

看病需要房间, 打印机  
文档

2.2.2.2 概念：产品、资源、耗材  
药 有病

甲方提供给客户的产品，事实上是物料、服务和信息的组合。其中“服务”使用到了医生、健管师、房间、设备等“资源”，而物料则是产品的“耗材”。

对于本系统的设计，“产品”可以描述为：

- 物料可以被当作产品，无论物料是否可被单独售卖给客户，还是必须与其他物料或服务组合来售卖给客户。“耗材”未必只能被挂在某“原子产品”（见下）上，“耗材”也可被挂在某“套餐”（见下）上。
- 服务可以是产品。但服务所用到的资源，在系统里被记录为资源，而非产品。服务“提供”资源，但并非服务“是”资源。
- 物料与服务的组合可以是产品。
- 对于信息，目前甲方的某些产品的“服务履行”中可能含有“提供某某信息给客户”的步骤，此类步骤将被系统里的“流程/通知”所表达，此“流程/通知”里所用到的“信息”在系统里与“流程/通知”挂钩，并不与系统里的“产品”直接挂钩。

产品包含“原子产品”和“套餐”。

原子产品是最小售卖单位或服务单位，也可能是最小需单独核算的物料。

“组合产品/套餐”是将原子产品和/或套餐组合形成的套件。也就是说，套餐既可包含原子产品也可包含套餐。

2.2.2.3 产品的属性 (Attribute)

产品的属性包括基本属性、技术属性、供应属性。

2.2.2.3.1 基本属性

产品的基本属性包括：名称、编号、类别(category)、标签(tag)。

2.2.2.3.1.1 产品的类别 (Category) 与标签 (Tag)

产品的类别本身是一棵树。根是“所有产品”，其下可配置各种子类、子子类、等等。

产品上可打上标签。

标签与类别的区别在于：

- 类别是树形结构
  - 可以有多棵类别树，每棵树用于不同的目的，比如用于“客户可见分类”或“财务科目统计分类”
  - 一个产品可以挂在某个类别树里的多个类别下，一般其中一个类别为此产品在此类别树中的主类别

- 对于套餐来说，如果套餐里包含的多个原子产品分属某颗类别树中的不同类别，一般来说此套餐就不能属于某个原子产品所属的类别，而是属于一个叫做“混合类别”的类别，或者另外命名的一个类别
- 标签可以是平面结构也可以是树形结构。一个产品上可以被打上多个标签

### 2.2.2.3.2 技术属性

产品的技术属性包括：

- 度量单位
  - 对服务型产品，度量单位可以是“次数”、“每次时长”。（其实“次数”可以用“组合产品/套餐”来表达）
  - 对实物（物料）型产品，度量单位可以是：克、千克、毫升、盒、等等。某些单位之间可以有换算关系。
- 其他规格。比如对实物型产品，会有形状（长/宽/高）
- 功能描述
- 品质描述

### 2.2.2.3.3 供应属性

产品的供应属性包括：

- 产品所属企业/机构
  - 产品可能属于甲方下属的某企业/机构
  - 产品所属企业的类型（一般纳税人/小规模）会影响产品的“各种价格”属性里的“税率”
- 各种价格
- 是否可单独售卖
- 是否可单独服务
- 所需耗材
- 所需资源（“所需资源”属性实际上隶属于产品的服务流程的某些流程节点，而非直接隶属于产品）
- 医疗数据规格和医疗报告格式规格。比如对血液检验、癌症筛查产品

“各种价格”包括：

- 市场价（吊牌价）。对原子产品和套餐，均可直接标注市场价，亦可设置为 0。套餐的市场价未必等于其内含的所有产品的市场价的总和。
- 直接加总市场价。原子产品的直接加总市场价等于市场价。套餐的直接加总市场价等于其内含的所有产品的市场价的总和。
- 递归加总市场价。原子产品的递归加总市场价等于市场价。套餐的递归加总市场价等于其内所含的所有产品的递归加总市场价的总和。
- 建议销售价。对原子产品和套餐，均可直接标注建议销售价。
- 最低销售价。对原子产品和套餐，均可直接标注最低销售价。
- 成本价、直接加总成本价、递归加总成本价。这三个价格名词之间的关系，类似于如上的市场价、直接加总市场价、递归加总市场价之间的关系，但还需要额外计算产品（原子产品与套餐）上所挂接的“耗材”与“资源”的成本，如果在“耗材”与“资源”上标注了成本的话。（实际上，不只“耗材”与“资源”上会有成本，在“服务流程/工艺流程”上也会有成本，只是本系统不管理那些成本）

- 注意，如上每个价格名目里，尤其是销售价，需要进一步区分出“税率、税额、价税合计、营收（不含税）/预收（不含税）”（实际上这四个名词里，操作人员只需要填写其中两个，系统即可计算出另外两个）。

#### 2.2.2.4 物料的属性 (Attribute)

物料的属性，实际上就是：从产品的属性里扣除掉与如下内容相关的属性：

- 与服务型产品相关的，比如服务时长
- 与销售相关的，比如各种销售/市场价、是否可单独售卖、是否可单独服务

需要注意的是，物料也有其所属的 Category 和 Tag，类似于产品的 Category 和 Tag。但物料的 Category/Tag 与产品的 Category/Tag 是互相独立的。

#### 2.2.2.5 资源的属性 (Attribute)

资源具有“基本属性”，类似于产品的基本属性，其中包含 Category 和 Tag。但物料的 Category/Tag、产品的 Category/Tag 以及资源的 Category/Tag 之间，都是互相独立的。

资源具有“供应属性”，包括：

- 单位时长成本
- 可连续工作的时间长度
- 通常可工作的时间段

资源的损耗和维修，不在本系统的管理范围内。

### 2.3 财务账户相关术语

如下列举本文中用到的术语，这些术语的含义在下文“9、财务账户数据模型”中都有介绍。本小节暂不详述。

- Account (账户)
- Summary Account、Derived Account
- Memo Account
- Balance (余额)
- Entry
- Transaction
- Posting Rule

如下术语来自于财务会计知识：

- debit/credit (借/贷)
- account or account title or account caption (科目)

## 2.4 其他业务术语

还有一些业务术语在本文中被用到，但是本节中尚未对这些业务术语进行明确说明。请先参见章节“4.2、核心数据模型所需涵盖的业务概念/实体”。

# 3 数据模型设计常用技术

## 3.1 数据模型设计者所需具有的预备知识

数据模型设计者需要具有如下方面的基础知识：

- OOA/OOD
  - OO 基础
  - OO 原则
  - 设计模式
  - 分析模式
- Domain Engineering（领域工程）中的 Feature Modeling（特征建模）

## 3.2 多种实现技术/范式（Paradigm）

有多种实现技术/范式（Paradigm）用于对软件设计与实现（Software Design and Implementation）进行建模，如：

- OO
  - 组合、子类化（单继承或多继承）、类型多态、原则、Pattern（模式）、等
- Mixin
- Generics
- AOP
- Rule Engine/Process Engine
- DSL
- Domain Engineering（领域工程）
- Model Driven Engineering
- etc

这些技术里，OO 是基础，也是最常用的。

## 3.3 独立于实现技术的特征/变化点（需求）建模

在应用之前提到的这些技术对软件设计与实现进行建模之前，需要对更早阶段的内容——即（数据模型方面的）需求——进行建模。如上所列举的技术也可一定程度上用于对需求进行建模，但 Domain Engineering（领域工程）中的 Feature Modeling（特征建模）技术更适合“抛开具体实现技术方面的局限性和倾向性”来对需求进行建模。



参见参考资料“RefDoc 16、Domain Engineering (领域工程), 尤其是其中的 Feature Modeling (特征建模)”。

不过, 对于那些对其所从事领域非常熟悉、抽象能力也很强的(数据模型)设计者, 可以跨过 Feature Modeling (特征建模), 或做一个较为粗糙的 Feature Modeling (特征建模), 即可进入针对软件设计与实现 (Software Design and Implementation) 的数据模型设计。此类人的特征之一是, 对“1.3.1、分析模式相关资料”和“1.3.2、医疗数据模型相关资料”中的资料, 能快速领会其要点和比较其优缺点。

### 3.4 不同泛化级别的数据模型/模式设计

“RefDoc 3、《数据模型资源手册 卷3》”中的章节“1.6 不同级别的模式”粗略描述了所谓的 1~4 级的模式, 从固化到泛化。其后续各个章节使用了这个设计思路。

最固化的模式形式, 是使用一个/多个具体类和/或一个/具体类中的一个/多个具体属性来表达一个特性。

此文中提到的泛化常用手段包括:



- 抽取抽象基类
  - 对子类/子集合的表达, 具体类型的多个子类是最直接、最具体 (非灵活化) 的表达形式
- 把固定和变化的特性 (指的是从职责上和 type 角度上的固化和变化, 而非具体值的变化) 区分开, 分成两个/两组 entity/entityHierarchy, 其中第二个/第二组 entity/entityHierarchy 通常会用到动态属性/属性集技术
- 使用 entity 来表达关系 (通常是表达多对多关系)
- 对 entity/entityHierarchy (无论这个 entity/entityHierarchy 是表达 thing 的还是表达关系的) 再用 (实际是关联) categoryEntity/categoryEntityHierarchy 来进行分类, categoryEntity/categoryEntityHierarchy 可以是多重的, 以应对不同分类方式
  - (此思路与本文章节 “2.2.2.3.1.1、产品的类别 (Category) 与标签 (Tag)” 所提到的 category/tag 的设计类似)
- categoryEntity/categoryEntityHierarchy 还可以再被其他的 categoryEntity/categoryEntityHierarchy 来分类。

此外, 如上描述里体现出了, 对事物进行分类 (分集合),

- 可采用多个类或多个子类 (这里不对 type、class、entiy 的含义作严格区分),
- 也可采用与 entity 关联的 categoryEntity/categoryEntityHierarchy,
- 也可同时使用。

用通俗的话来讲, 类实际上更强调一个类 (或一个集合) 里的所有实例/元素具有相同的结构 (即属性字段列表和每个属性字段的类型/class/type)。而 categoryEntity/categoryEntityHierarchy 更灵活。

当然, 结合了动态属性/属性集技术的 class/entity/entityHierarchy, 可以部分地模拟 categoryEntity/categoryEntityHierarchy 的灵活性。基础的常用的动态属性设计技术 (不引入额外的 DSL、Model Driven Engineering) 包括:

- 预留属性 (预留字段)
- 使用表达 Attribute、AttributeSet、AttributeValueOption 的类(Class)来表达一个 “被动态定义的数据类型”
  - 如上类的实例用于表达一个 “被动态定义的数据类型” 的元数据(Meta-data)
  - AttributeValueOption 的实例 (以及 Java 中的具体类型的值) 亦被用于表达被选用的 “此被

动态定义的数据类型”的值

- 此设计可被看作对传统的“关系数据库上用‘列表’表达动态字段的设计模式”的增强
- 使用 JSON 和 JSON Schema(或用其他方式为 JSON 定义 Schema)
- OpenEHR 中的 ItemStructure 及可自定义的 Archetype
  - (其实这也属于 DSL 或 Model Driven Engineering, 只是这不用我们来定义了, OpenEHR 已经定义好了)
- GraphQL
  - (GraphQL 通常被用来定义 API 中的数据模型, 而非像如上那些技术那样主要被用来定义持久化数据模型)

categoryEntity 的退化形式, 是在被 categoryEntity 所标注的 entity 上引入一个属性字段“typeName”, 然后删除 categoryEntity。

## 3.5 节选: Open-Care hpaPaaS 中的 Model-Driven

本节如下内容摘自参考资料“RefDoc 13、《Open-Care hpaPaaS 系统设计思路汇编》, 2018”中的章节“2.1 Model-Driven-Engineering”, 未作更改。故其中的文字“本文”指的是“RefDoc 13、《Open-Care hpaPaaS 系统设计思路汇编》, 2018”一文。

此内容可被数据模型设计工作所借鉴。

### 3.5.1 Model-Driven-Engineering

本文中的“Model-Driven(模型驱动)”, 强调“把具体实例应用抽象成知识并复用到其他类似的实例应用场景里”, 核心是“抽象成知识/模型”和“知识/模型复用”。而“对抽象知识/模型的表达方式”, 可以是 Model/Meta-Model, 也可以是“某个 Pattern”、“某种自定义编程语言”、“知识图谱”等。

#### 3.5.1.1 Model-Driven 所带来的好处

现有的各种程序开发语言/模型, 比如属于 3GL 语言 Java/C++等, 已经提供了“数据建模、逻辑控制等能力”来支持“几乎可以开发任何应用程序”。那么 Open-Care hpaPaaS 所需要支持的“更高阶的建模能力”, 带来的优势是什么呢?

任何一个程序开发语言/模型, 针对其所将要被应用到的业务领域, 以对数据模型的描述能力为例, 所需具有的特性是:

- 特性 A、能描述此领域内可能出现的各种具体数据模型。对“具体数据模型”的描述, 包括:
  - 特性 A1、描述此“具体数据模型”所能具有的能力(如结构上的变化/灵活性的空间等)
  - 特性 A2、描述此“具体数据模型”所需满足的约束(比如类型约束、结构约束, 比如 OCL(Object Constraint Language)所能表达的约束)
- 特性 B、能方便、直观地进行描述
  - 例子: 比如 Java 里引入了 Lambda 表达式(可以理解为把 Lambda 作为 Java 语言的 First Class Citizen)来简化对特定场景的描述
  - 例子: 再比如 C++的某些模板库(如 Loki)可以直接支持一些 Design Pattern(设计模式)

如上特性 A 对应的是“灵活性”，特性 B 指的是“方便性”并蕴含了“可复用性”。我们口语中的“灵活性”，有时包含了特性 A 和特性 B。

泛泛地来讲，从具体的应用场景抽象出模型/知识，或者从现有的某种模型/知识抽象出更高阶的模型/知识，所带来的好处是，针对某特定领域，在保持了特性 A（其中有时是增强了特性 A2）的同时，大大增强了特性 B。当然，如果设计不良，带来的可能的缺点是，丢失了一些“细节逻辑控制能力”（此能力事实上隶属于特性 A）。下文称此缺点为“缺点 AX”。

### 3.5.1.2 Model-Driven 中的“多级具化”

从“多个应用实例”中抽象出“知识/模型”，再把“知识/模型”复用（即具化）到其他“类似应用实例”的过程中，是可以有各种“具化/复用中间阶段”的，而不是非得“一步到位的具化”。

比如，基于知识/模型形成一些 Template，这些 Template 是“从知识/模型具化到实例”的过程中的一个中间状态，还遗留了一些“灵活配置的空间”，允许后续“再进一步（或多步）地具体化到实例”。

此外，这些 Template 之间亦可支持“Template 组装”。此类例子包括“Product Template”和“UI Template”。

### 3.5.1.3 动态模型静态化

#### 3.5.1.3.1 动态模型与静态模型的优缺点

本节以 OpenEHR 医疗数据规范为例来讨论“动态模型的静态化”。

OpenEHR 支持双层模型，类似于 Meta-Model 层和 Model 层。OpenEHR 规范里的 Class “ItemStructure”里的结构是灵活的。当使用某个具体的 Archetype 文件来对 ItemStructure 里的结构进行约束，就会形成一个“较为具体化的结构”。这个较为具体化的结构（下称前者），比起“纯用 Java 写的 Entity Classes 所表达的结构（下称后者）”：

- 前者的优点：前者相对来说“更动态一些”，即如果想修改这个结构定义，只需要修改 Archetype 文件即可，（基本）不用重新编译 Java 源代码。
- 前者的缺点：当“A、其他手写的 Java 源代码”需要调用“B、表达如上的较为具体化的结构的 Java 源代码时”，从 Java/IDE 的角度来看，源代码 B 没有强类型信息。由此，由于相对动态的结构，在 ORM/存储性能方面，也会受到一定影响。

前者的优缺点，反过来，即是后者的优缺点。

更进一步地，以 OpenEHR 规范的“Java 参考实现”里的如下两个 Class 为具体例子：

- `org.openehr.rm.demographic.Person`
- `org.openehr.rm.demographic.Party`

Person 继承自 Party。Party 里有个成员变量“`private ItemStructure details;`”，此成员变量是可被“Archetype”所描述的灵活结构。此灵活结构，可以允许表达多种多样的属性。

但是，对于软件系统比如 CRM 系统，通常在设计 CRM 时，软件设计者能够（基本）确定 Class “Person” 里所应该具有的各种常规属性，比如“性别”、“生日”等，而不必把这些常规属性用“Archetype”来描述和定义。如果把“性别”、“生日”等属性直接写在 Class “Person” 或者其 Sub Class 里，能带来“Java/IDE 所能识别的强类型”的好处，也能带来 ORM/存储性能提升（相对于动态结构）。

### 3.5.1.3.2 动态模型静态化的方式

基于上一节所描述的动态模型与静态模型的优缺点，在某些场合下，应该更适合“动静模型混用”或“动态模型静态化”。

以上一小节提到的 Class “Person” 为例。此 Class 持有（继承而来）的成员变量“ItemStructure details;”。

在复用 OpenEHR 的 Classes 的前提下，为某 CRM 系统，可以设计一个 Class 名如“ConcretePerson”，此“ConcretePerson”继承自 Class “Person”，并添加属性/成员变量如：

- private String sex; (或者用 Enum Type 而非 String Type)
- private Date birthday;

此 Class “ConcretePerson” 相对 Class “Person” 来说，更静态了一些。

另一种设计更静态一些的 Class 的方式是，添加一个新 Class 名为“PersonAttribute”，此 Class 继承自“ItemStructure”，此 Class 里含有成员变量：

- private String sex; (或者用 Enum Type 而非 String Type)
- private Date birthday;
- private ItemStructure nextDetails; //可被 Archetype 描述和定义

“PersonAttribute”继承自“ItemStructure”的目的是，“PersonAttribute”的 instance 可以被赋值给“Person”的成员变量“details”（Person.details）。“PersonAttribute”的如上成员变量，使得“PersonAttribute”能够表达静态属性和动态属性。

进一步地，针对某种特定场合，如果某个 Archetype 描述/定义了“Person.details”的结构，那么在此场合，可以基于此 Archetype 来自动地在编译时生成 Class “ConcretePerson” 或 Class “PersonAttribute” 的定义（静态 Java Class）。

## 3.6 “RefDoc 1、《数据模型资源手册 卷 1》” 中的部分关键点

（注意：本小节中的第 x 章指的是“RefDoc 1、《数据模型资源手册 卷 1》” 中的章节。）

1、书中大量使用了交叉关联实体来表达多对多关系，此交叉关联实体上具有一些属性。参见其章节‘1.7.4-5’以及其他章节。比较典型地，是如下实体之间的多重交叉多对多关系：产品、订单/订单条目、协议 (agreement)/协议条目 (类似于合同/合同条目)、报价/报价条目、订购请求/请求条目、装运(shipment)/装运条目 (装运项目)、工作计划/计划条目、出入库/库存条目、发票/发票条目、支付/支付方式/支付方、账户、预算/预算条目、等。以其中的子集---支付、发票、订单、装运---为例，根据实际业务场景，发票/发票条目可以和订单/订单条目进行多对多对应，也可以和装运/装运条目进行多对多对应，而订单/订单条目也是可

以和装运/装运条目进行多对多对应的;支付与发票和订单也有类似的三角/多角关系。这些三角/多角关系,需要在多重多对多关系之间的传递一致性上,由应用程序作额外的处理、校验。书中未显示描述对多重多对多关系的传递一致性的处理与校验。业务中还会涉及到对各种实体对象的增改退换,相应的多重多对多关系的变更及变更后的一致性的处理与校验,会进一步增大软件设计开发的难度。

2、“第2章 人与组织”及“第9章 人力资源模型”与“RefDoc 9、《The openEHR Reference Model--Demographic Information Model》”的设计思想很接近,在 party、role、capability、relationship 等的表达上。

3、“第3章 产品模型”及后续章节对产品模型的应用中,没有将产品和特征取值综合成一个实体/对象,这导致复用性不充分的问题,而本文中引入了“产品阶段的概念”,可以解决此问题,并引入“多种抽象具化方式比如基于规则的”,而不是仅仅支持基于对特征的可选值的选择。另外,本文还引入了产品版本(含 BOM 版本中的偏序处理)概念。

4、“第4章 产品订购模型”未显式描述对订单/订单条目的增改退换。本文中需要支持此点,通过比如引进“订单 Account、订单 Entry”的概念,或使用“订单-Composition”模式。文中给出了协议(类似于合同)与订单之间的关系,不过本文还需处理协议之间关系,比如框架协议与具体合同之间的关系。此文中还描述了需求、订购请求、报价等。

5、“第5章 装运模型”和“第6章 工作计划模型”,可与本文中的服务单功能进行对比,借鉴前者设计来增强本文中的设计。比如,“第6章 工作计划模型”中,工作计划/计划条目可与固定资产和库存条目关联,这里的固定资产和库存条目分别类似于本文中的产品/服务单所需使用的资源和耗材。另,本文中产品/产品 BOM 中对产品服务流程及流程聚合的设计,在此局部的能力上是超过了其“第6章 工作计划模型”的。

6、“第7章 发票模型”中,值得注意:发票条目与订单条目计费/装运项目计费/工作计划计费/时间条目计费的多重关联关系,发票条目自身的递归关系,发票与支付/账户之间的关联关系。参见 7.9 中的图 7-10,以及此章节中的其他小节。

7、“第8章 会计和预算模型”中对账户的设计,比起“RefDoc 5、《Analysis Patterns》, Chapter 6 "Inventory and Accounting", Chapter 7 "Using the Accounting Model"”(章节“第6章 库存与账务模式”和“第7章 使用账务模型”),有所欠缺。比如后者在表达账户组合之外,清晰地表达了 Derived 账户,并对账户组合中的会计分录(Entry,如借/贷,归属于会计事务 Transaction。本书中称作“会计事务明细”)的重复计算问题做了明确说明。

### 3.7 所谓“有难度 (Difficulty)”

当人们提到某个事情比较“有难度 (Difficulty)”,通常人们指的是如下含义之一:

- 深度复杂 (Complexity)
  - 比如需要复杂算法、需要深刻的认知、需要艰深的抽象
- 量大 (Volume)
  - 比如每种场景用几个简单的“if ... else ...”就能表达,但是会有成千上万中场景(如果不对场景采用归类等抽象手段的话)
- 不确定性 (Uncertainty)
  - 未来可能会有些需求变化或者意外,它们要么是不可被预测的,要么是可被预测但是项目组成员不愿意花太多时间去做良好预测分析

通过进行良好的软件设计，

- 是可以在一定程度上将“量大”转化为“深度复杂”的。转化后的效果是：
  - 项目成本降低。比如原来需要 100 个普通开发人员，转化后需要 10 个高级开发人员和 20 个普通开发人员，而 10 个高级开发人员的薪酬一般会小于 80 (= 100 - 20) 个普通开发人员
  - 项目后期维护成本（比如应对需求变更）降低
- 是可以在一定程度上产生一个良好的副作用，即能应对一定程度的“不确定性”。效果是，
  - 项目未来维护成本降低

“进行良好的软件设计”的重要方法之一就是“抽象”。关于“抽象”的具体技术手段，部分地可参见章节“3.2、多种实现技术/范式（Paradigm）”和“3.3、独立于实现技术的特征/变化点（需求）建模”。

顺便提一句，有些方法如 ATAM（架构权衡分析方法，Architectures Trade-off Analysis Method），可以对软件架构（及软件设计）的质量进行评估，“可更改性”被评估的质量内容之一。

## 4 核心数据模型涵盖的业务范围

### 4.1 核心数据模型需要支持企业业务核心系统的系统特性

根据益先科技发展策略，益先科技所开发的企业业务核心系统需要支持如下系统特性：

- 支持“前中后全链条业务功能（面客/销售管理、运营、供应商管理）”。对全链条业务功能进行扩展、定制，即可适合其他行业企业的运营业务功能需求
  - 关于全链条业务功能的一个示意，参见“图 4-1、前中后全链条业务功能（面客/销售管理、运营、供应商管理）”。此图摘自“RefDoc 14、《OC--健康险之健康服务运营管理系统--方案介绍--202x.xx.pptx》”
- 强有力地支持业务复杂性/多业态服务生态，和业务多变性/业务创新
  - 多业态服务生态：管理集团化、多业态、多渠道、多供应商的服务业务，为多业态服务生态体系、服务闭环夯实基础
  - 服务业务中台：建设具有强大灵活性的平台系统，支持业务应变。技术驱动管理和服务创新，提升体验。利用现代化的技术架构提高效能与伸缩性。进行良好的业务领域拆分与复用，形成服务业务中台

核心数据模型需要对如上这些系统特性进行支持。



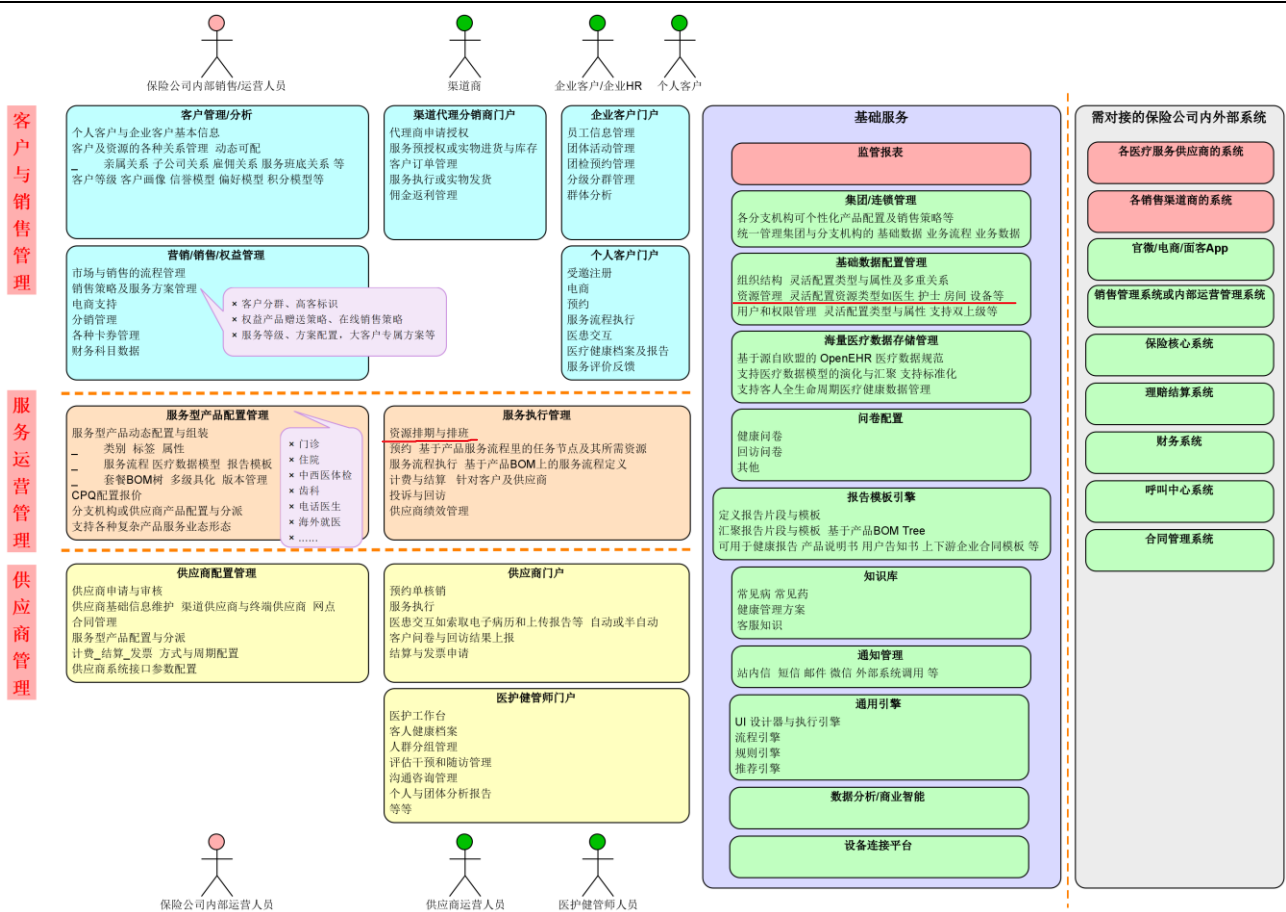


图 4-1、前中后全链条业务功能（面客/销售管理、运营、供应商管理）

## 4.2 核心数据模型所需涵盖的业务概念/实体

核心数据模型所需涵盖的业务概念/实体：

- 产品
- 订单/订单条目(Order/OrderItem)
- 协议(Agreement)/协议条目(类似于合同/合同条目)
- 报价(Quota)(或称报价单)/报价条目
- 订购请求(Purchase Request)/请求条目
- 预约(Appointment)(或称预约单)/预约条目
  - 通常为针对服务型产品的服务时间的预约
- 交付(Delivery)(或称交付单)/交付条目，或称服务单/服务条目
  - 对实物型产品，即货品，交付/交付条目退化或装运(Shipment)/装运条目(装运项目)
  - 对服务型产品，交付/交付条目会关联服务流程或工作计划/计划条目
- 库存/库存条目(Inventory Item)
- 入库单(Godown Entry)/入库条目，出库单(Delivery Order)/出库条目(Item Issuance)
- 发票(Invoice)/发票条目
  - 包括系统主体企业（指使用企业业务核心系统的主体企业）开给客户的发票，和供应商开给主体企业的发票
- 支付(Payment)/支付条目/支付方式/支付方
- 账户(Account)/Transaction/Entry
- 预算(Budget)/预算条目


- 等

核心数据模型需要支持如上多种实体之间的多重交叉多对多关系。“4.3.2、销售相关业务场景”给出了部分例子。


## 4.3 典型的复杂业务场景

### 4.3.1 产品相关业务场景

如下数个图摘自“RefDoc 12、《OC--健康 IT 开放平台--Health IT Open hpaPaaS--方案介绍--202x.xx.pptx》”中的“03.服务型产品体系配置管理”。



### 服务型(及实物型)产品配置管理 (1/2)



#### 原子产品配置

- 产品数据模型配置
  - 技术属性
    - 如度量单位、物理规格、功能描述、品质描述、等
  - 供应属性
    - 所属机构/服务商(可多个)、各种价格(如市场价/建议销售价/渠道结算价/成本价/税率)、是否可单独售卖、是否可单独服务、所需耗材、所需资源(如房间/设备/车辆/医生)、等
  - 医疗数据模型(基于OpenEHR双层医疗数据模型规范)
  - 平面/树状特性(动态配置的属性)集
- 服务流程配置
  - 服务型产品具有复杂的服务流程
  - 可动态配置服务流程、流程片段、任务
- 报告模板配置
  - 服务型产品在服务执行后通常需要出具服务结果报告。报告模板可被动态配置
- 类别与标签(Category and Tag)配置
  - 类别是树/森林结构。标签是多维正交结构，也可具有树/森林结构。类别和标签可被动态定义和关联

#### 产品组合(套餐产品/BOM tree)

- 动态配置套餐
  - 某套餐可包含其他套餐或原子产品
  - 套餐可混合包含实物型产品和服务型产品
    - 套餐产品在被执行时，系统需处理其内的实物型产品的库存/物流和服务型产品的排期/预约
- 套餐数据模型自动组合汇聚
  - 套餐产品的数据模型可由其所包含的产品的数据模型自动递归汇聚而来
    - 价格加总、耗材/资源加总
    - 医疗数据模型汇聚
    - 流程汇聚
    - 报告模板汇聚，等
  - 可根据自定义公式进行汇总计算
- 套餐的配置/销售场景
  - 常规套餐
  - 季节性/地域性促销套餐
  - 针对某团单/个单/人群的个性化套餐

40

图 4-2、服务型(及实物型)产品配置管理 (1/2)



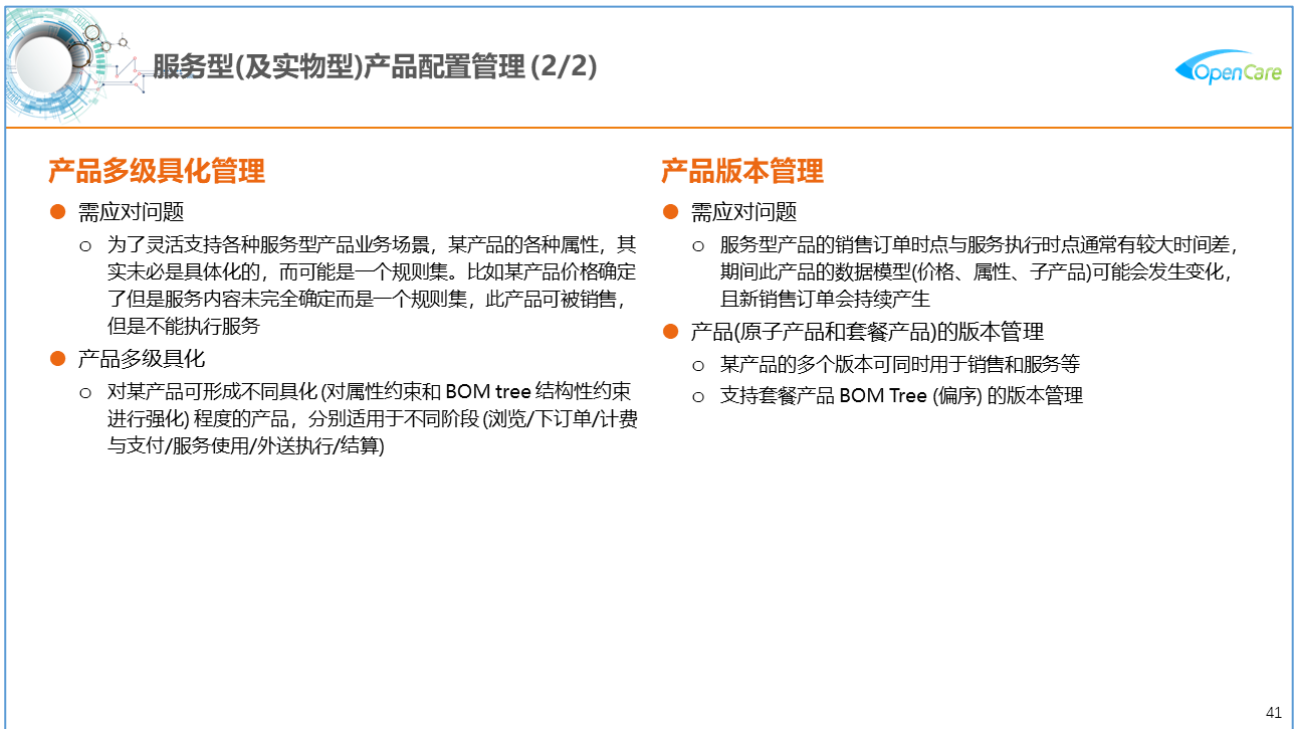


图 4-3、服务型(及实物型)产品配置管理 (2/2)



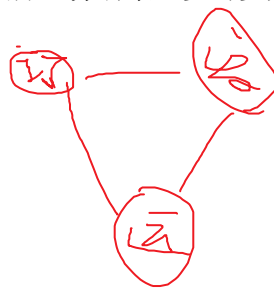
图 4-4、产品服务履行管理(自有及第三方供应商)

### 4.3.2 销售相关业务场景

“4.2、核心数据模型所需涵盖的业务概念”所描述的多种实体之间存在多重交叉多对多关系。

以数据模型的子集---支付、发票、订单、装运---为例，根据实际业务场景，多对多关系会发生于如下组合情况中：

- 三角关系 1
  - 订单/订单条目 <—> 发票/发票条目
  - 发票/发票条目 <—> 装运/装运条目
  - 订单/订单条目 <—> 装运/装运条目
- 三角关系 2
  - （背景信息）
    - 一次支付中会包含多种支付方式。例如，针对某个订单，体检中心前台护士在客人的某一次支付中共收取 300 元，其中现金 100、支付宝 60、储值卡 40、企业挂账 100
    - 一次支付中会包含多个支付方。比如，如上例子中的现金、支付宝是属于客人个人出钱，企业挂账是企业出钱（挂账表示实际上是后付费），而储值卡根据其所属的主人可能是个人储值卡或企业储值卡
  - 订单/订单条目 <—> 发票/发票条目（/发票所属方）
  - 订单/订单条目 <—> 支付/支付方式/支付方
  - 发票/发票条目 <—> 支付/支付方式/支付方



这些三角/多角关系，需要在多重多对多关系之间的传递一致性上，由应用程序作额外的处理、校验。业务中还会涉及到对各种实体对象的增改退换，相应的多重多对多关系的变更及变更后的一致性，也需要处理与校验。

一般来说，可以使用交叉关联实体来表达多对多关系，此交叉关联实体上具有一些属性。关于此种设计，可参考“RefDoc 1、《数据模型资源手册 卷 1》”和“RefDoc 2、《数据模型资源手册 卷 2》”。

## 5 核心数据模型摘要

### 5.1 产品核心数据模型简介

本节简介与产品相关的核心数据模型。详情请参见章节“7、产品数据模型”。

#### 5.1.1 产品打包树数据模型

支持产品分类树、产品打包树。

产品打包树上的数据模型，包括：

- 在产品上赋予与服务流程相关的属性，即使得产品与服务流程发生关联。
  - 关联——每个产品关联“一个或多个服务流程”，或关联一个“关于服务流程的规则”。
  - 聚合——产品打包树上的某产品（组合产品/套餐）的服务流程，可以根据“服务流程的规则”里的有关“如何聚合其下级产品上的服务流程”的规则，来对其下级产品的服务流程进行聚合而得到。

- 被 Timer 触发来 scan ‘SomeSource’
- Trigger 功能，是
  - 触发所关联的 PostingRule

PostingRule 的功能是创建一些 Output，这里的 SomeOutput 可能是：

- 针对某些 Accounts 的一些 Transaction 以及 Entry
- 针对某些 Memo Account 的一些 Entry

PostingRule 的具体实现，可以是：

- 一个具体的子类
- Strategy Pattern
- Rule Engine

上图中，某些 SomeOutput 其实可能也是 SomeSource，所以对 Posting Rule 的触发执行可能是级联的。

## 10 医疗数据的数据模型

关于医疗数据核心模型简介，参见章节“5.3、医疗数据核心数据模型简介”。

医疗数据核心模型内容比较复杂，但是它是标准的（基于 OpenEHR 规范），故本文不再赘述。

关于产品模板/产品变体与医疗数据模型之间的对应关系，参见章节“7.2.3、医疗数据模型（Medical/Health Data Model）”。

## 11 附录：图表目录

### 11.1 图目录

图 2-1、物料、服务、信息 .....	12
图 4-1、前中后全链条业务功能（面客/销售管理、运营、供应商管理） .....	23
图 4-2、服务型(及实物型)产品配置管理 (1/2).....	24
图 4-3、服务型(及实物型)产品配置管理 (2/2).....	25
图 4-4、产品服务履行管理(自有及第三方供应商).....	25
图 6-1、OpenEHR rm.demographic Package.....	29
图 6-2、Person Demographic Information .....	30
图 6-3、Group Demographic—1 .....	31
图 6-4、Group Demographic—2 .....	32
图 6-5、Patient Relationship with Roles and Credentials.....	32
图 6-6、混合上下文角色模式 .....	33
图 7-1、BOM Template 的 Class Diagram（类图）（简化版示意图） .....	35
图 7-2、Object Diagram: BOM Template 生成 Product Variant 的例子 .....	36
图 7-3、基于 BPMN(Business Process Model and Notation)规范的一个流程例子 .....	38

图 8-1、销售核心数据模型 .....	48
图 8-2、订单-Composition.....	53
图 9-1、帐目和计入规则的更深刻的图形 .....	59
图 9-2、典型场景中的账户的需求模型 .....	61
图 9-3、Design Model—Accounts Class Diagram.....	64
图 9-4、Design(Concept) Model—Posting Rule.....	65

## 11.2 表目录

表 2-1、软件技术术语表 .....	10
表 2-2、业务术语表 .....	10
表 9-1、财务记账中的账户/科目 .....	60
表 9-2、一级账户（此处只列目前业务用到的账户） .....	61
表 9-3、二级账户 .....	61