适用于离岸开发的 UML 应用指南

V3.0

2010 年 12 月

特定非营利活动法人 UML 建模技术推广协会

离岸软件开发部会

目　录

1. 目的 3
(1) （面向离岸开发全过程中）本指南的地位 3
(2) 本指南的适用对象 4
2. 离岸开发现状的问题点和对策 9
(1) 问卷调查概要 9
(2) 问卷调查结果（部分摘录） 10
(3) 听证调查 14
3. UML 建模 15
3.1 UML 的特征 15
3.2 UML 建模的前提技能 18
4. UML 的适用范围和开发诀窍（HINTS & TIPS） 20
【要点 01：明确作业范围和作业分工】 32
【要点 02：选定需要使用的 UML 图】 35
【要点 03：不是非 UML 不可】 36
【要点 04：力争参加上游阶段】 38
【要点 05：定义非功能需求】 39
【要点 06：使用分析模型理解业务】 40
【要点 07：制作术语字典】 43
【要点 08：建立命名规则】 44
【要点 09：建立建模规则】 45
【要点 10：明确通用功能】 46
【要点 11：架构模型】 47
【要点 12：制作架构说明的成果物】 48
【要点 13：灵活运用模式】 49
【要点 14：指定设计书的描述级别和格式】 50
【要点 15：制作详细设计指南】 52
【要点 16：明确式样尚未决定的部分】 53
【要点 17：离岸方实施对文档的审核】 54
【要点 18：日方的反复检查】 55
【要点 19：确认(VALIDATION)和验证(VERIFICATION)】 56
【要点 20：用模型对详细设计进行审核】 58
【要点 21：各种 UML 图之间的一致性】 59
【要点 22：使用代码生成工具产生代码并进行调试】 64
【补充：在日本本土系统开发的特点】 65
附录 A. 使用要点的样本事例的说明 67
附录 B. 成果物样本 74

1. 目的

(1) （面向离岸开发全过程中）本指南的定位

离岸的系统开发，与非离岸的日本国内开发同样、在需求/设计/架构/测试整个开发生命周期的各阶段中，有关技术观点的方面可以采用开发方法论、有关流程和成本的方面可以采用项目管理手法、有关成果物的方面可以采用质量管理手法。

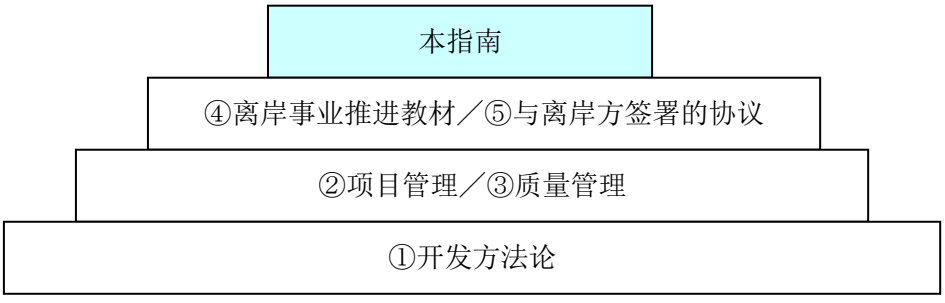
本指南把焦点对准“采用 UML 的离岸软件开发”。从实践经验中总结了离岸开发所特有的、并且是采用 UML 来开发的常见问题，汇总成各针对策略(Hints & Tips)。

下面说明一下本指南在面向离岸开发全过程中的地位。

离岸开发全过程有以下要素：

① 开发方法论（开发流程、开发步骤、开发技法）
② 项目管理（基于项目管理计划等的管理）
③ 质量管理（基于质量管理计划等的管理）
④ 推进离岸事业的教材（离岸事业推进的事例集锦（有售）、离岸开发邮件杂志、各公司的秘诀等。）
⑤ 与离岸方签署的协议（和离岸方谈妥的作业范围、日程、交货内容、价格等）

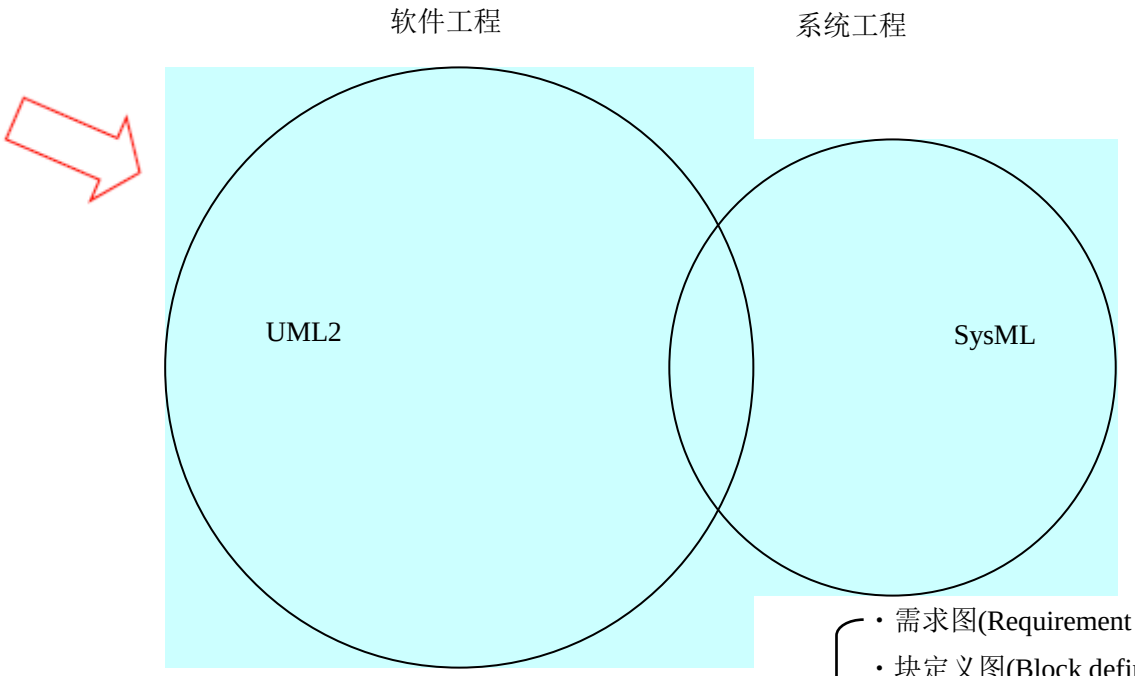
进行系统开发涉及要素①～③，进行离岸开发还涉及要素④⑤。①～⑤是在离岸开发进行之前各公司必需要准备好的。本指南解说了以建模法为中心的通用开发方法、开发步骤及其成果物。也是要素①～⑤的补充说明(也就是秘诀)。前提是各公司已经准备好了要素①～⑤，在此基础上附加使用本指南、以期达到离岸开发的高效率化。所以，本指南中不涉及各公司自行决定的作业内容和作业步骤等内容。



(2) 本指南的适用对象

本指南将以下范围作为适用对象。

首先从广义的“系统开发”来考虑，可以分为主要采用 UML2 等的软件工程、和另外还使用 SysML 等的系统工程两类。本指南记载的是针对于软件工程的研讨成果。



- 类图(Class diagram)
- 包图(Package diagram)
- 交互图(Communication diagram)
- 对象图(Object diagram)
- 构件图(Component diagram)
- 复合结构图(composite structure diagram)
- 部署图(deployment diagram)
- 交互纵览图(interaction overview diagram)
- 时序图(Timing diagram)

- 状态机图（state machine diagram）
- 顺序图（sequence diagram）
- 用例图（use case diagram）
- 活动图（activity diagram）

- 需求图(Requirement diagram)
- 块定义图(Block definition diagram)
- 内部的块定义图(Internal block diagram)
- 包图(Package diagram)
- 参数图(Parametric diagram)

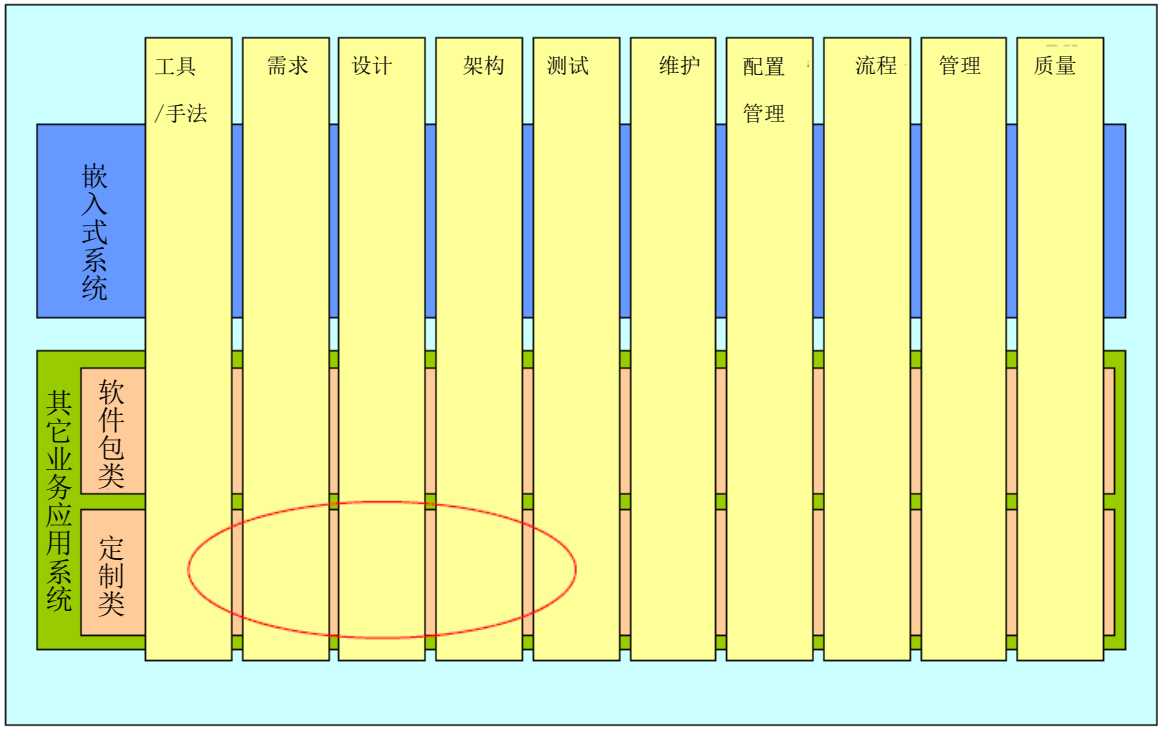
接下来，我们来详细看一下软件工程。

软件工程大致分为：

- 需求～设计～架构～测试～维护整个开发周期中各工程阶段的开发技术要素
- 工具、开发方法论、配置管理方法等支持成果物制作的要素
- 有关开发流程、成本的项目管理要素
- 有关设计书、程序等作业成果物的质量管理要素

作为工程对象的软件又分为 1）以嵌入式为前提的软件；2）业务应用软件的两类。业务应用软件进一步可分为 2a)将软件产品做成软件包，在其可定制范围内进行设计・架构；2b)难于构成应用软件包、但自由度很大、重新设计・架构部分很多的定制型开发的两种情况。

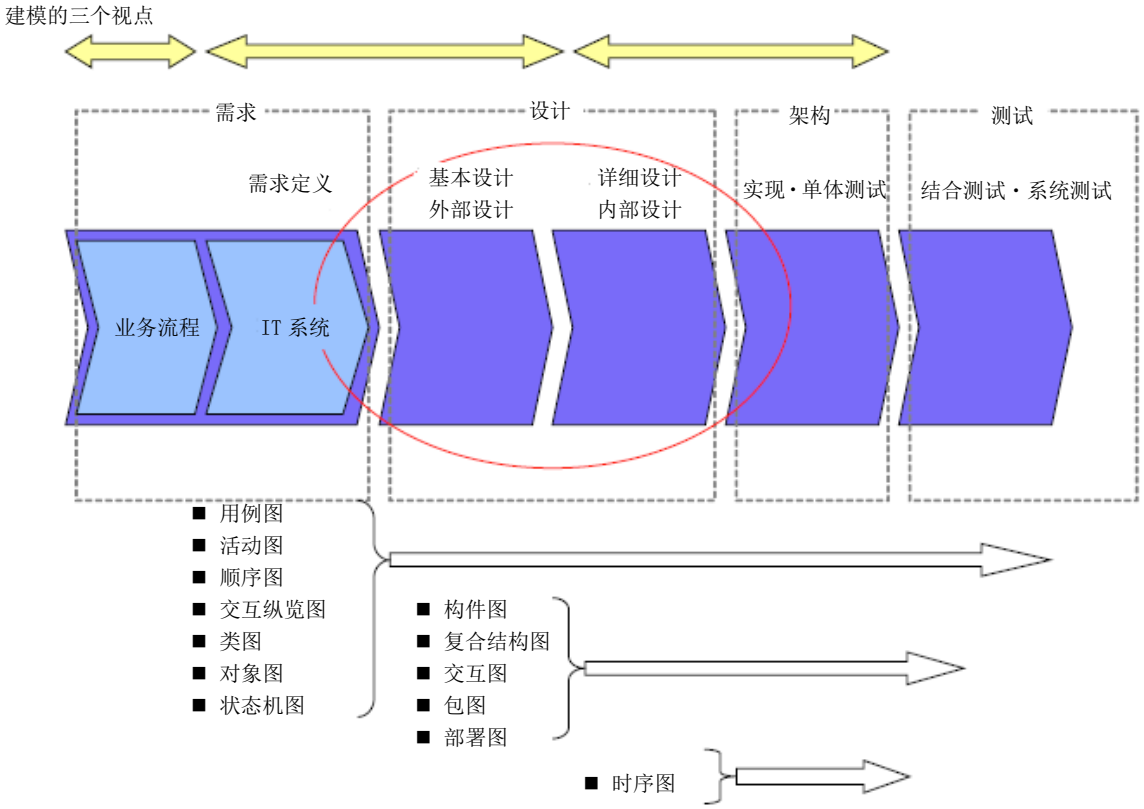
本指南以非软件包类的、定制型业务应用软件开发周期各工程阶段的开发技术要素为焦点，给出了要点和提示(Hints & Tips)。



现在让我们进一步详细分析一下开发周期中的各个阶段。

开发周期从需求阶段后半的需求定义开始、经过设计、直至架构和测试。如果从“UML”的角度出发来看，从实现业务流程的 IT 系统的需求定义开始，到开始编程的架构阶段，都是 UML 的应用对象。具体就是：需求用 UML 来描述、设计式样书用 UML 进行描述。然后把式样传达给离岸开发人员、基于 UML 来进行实现的这样一系列场面。

本指南着眼于在需求定义、外部、内部设计各阶段中如何使用 UML 的来记述设计式样。但从建模的观点看，属于设计建模和实现建模的范畴。



请注意，上图中所使用的需求定义、外部设计、内部设计、实现等名词，在后述的本指南中分别被按下述分类及术语所取代。

- 业务分析
- 需求分析
- 系统分析
- 架构设计
- 详细设计
- 实现、单体测试
- 结合测试

最后，让我们来看一下在上述与 UML 关联的阶段中，用离岸的角度如何加以整理。

在本指南中，离岸被定义成外部设计（架构设计）结束后、从内部设计（详细设计）到实现阶段的工作。从开发者的角度来看，可以列出：

「发给离岸人员的设计书，用 UML 怎么来写、需要写些什么？」

「离岸开发特有的注意点是什么？」

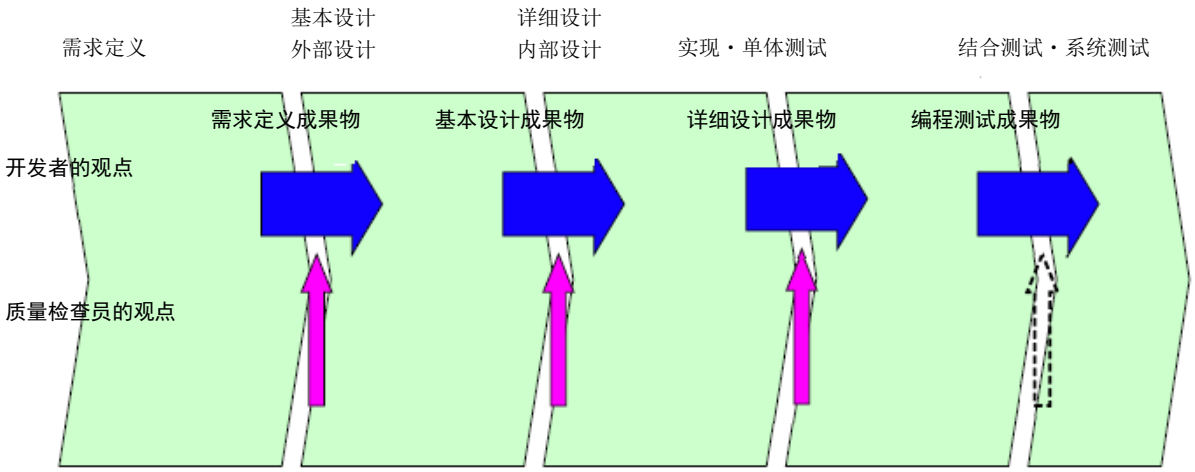
另一方面，从检查离岸成果的质量的立场来看

「离岸开发需要注意哪些要点来检查？」

「离岸开发需要满足什么标准？」

等关心事项。

本指南从上述观点总结了其中的秘诀。



		需求定义→基本设计	基本设计→详细设计	详细设计→ 实现	实现→ 结合测试・系统测试
开发者 的观点	日方	如何使用和制作	交付什么样的东西合适	--	--
	离岸方	--	想要接受什么样的东西	--	--
质量检查员 的观点	日方	检查什么样的内容	写些什么样的内容， 达到什么样的质量水准	检查什么样的内容	--
	离岸方	--	想要接受什么样的东西	检查什么样的内容	--

上图表示了本指南的对象范围。

我们假定在离岸开发中典型的模式是从「软件内部设计到架构」，由日方整理需求・设计，并将其传达给离岸人员；离岸方和日方双方共同检查离岸成果物。这些就是本指南的对象内容。

接下来再从另外一个角度来补充说明一下对象范围。

对需求进行分析、然后根据需求进行设计系统、再把设计式样要求传达给离岸方、根据传达的结果来实现系统、最后对成果进行质量检验。在这样的一个开发过程中、存在

● 日本国内开发的共性、离岸开发的特性

单单仅着眼于离岸开发特点，就可以举出下列开发中所用到的要素：

● 使用 UML 语言的技能和秘诀

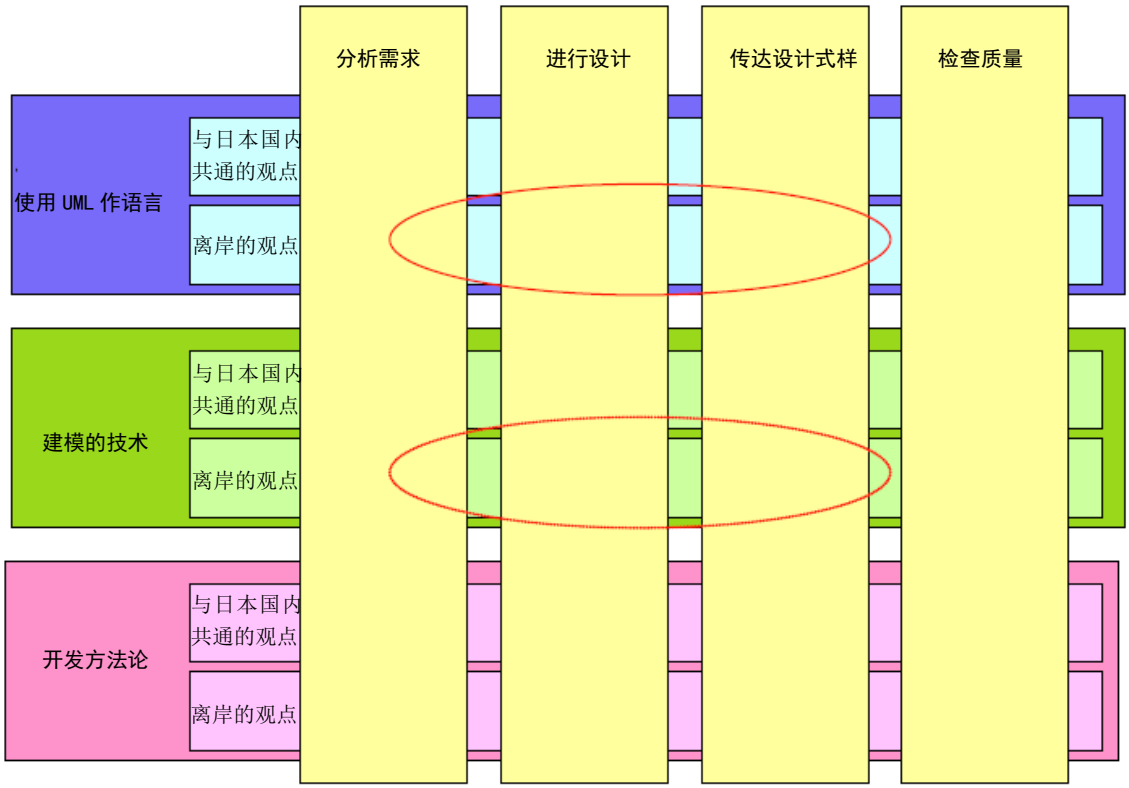
● 进行建模时的技能和秘诀

● 基于经验积累形成的知识资产的开发方法论技术和秘诀

等等。

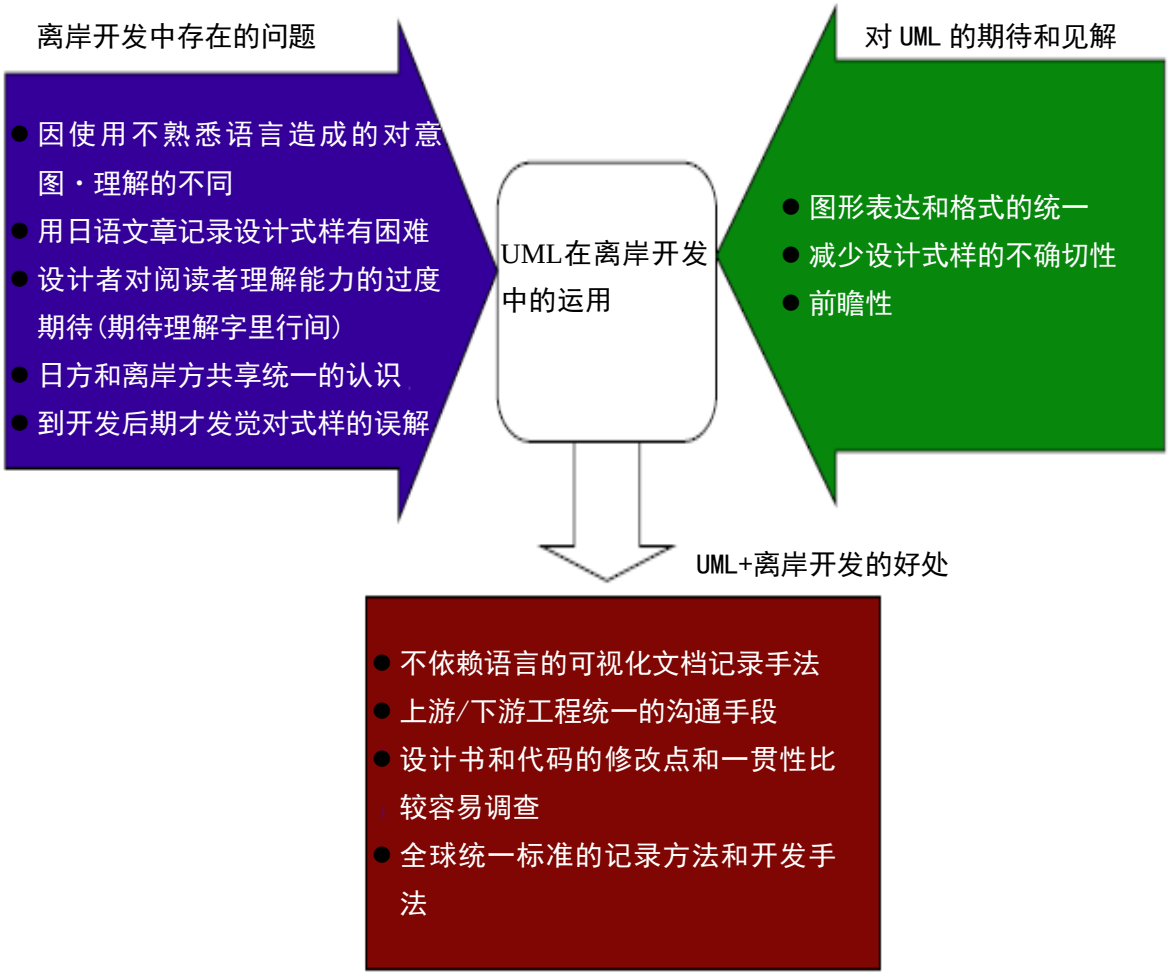
但是，本指南重点描述「在采用 UML 的离岸开发中，如何记述需求与设计式样、传达设计式样、检查成果物」。本指南中不包括对 UML 本身及其使用、建模、具体例子以及开发方法论的解说。

综上所述，本指南的读者，应该掌握 UML、建模、开发方法论、软件开发技术要素等方面的知识。希望读者以这些知识为基础、将本指南与前提知识边学边关联。（对「非离岸开发所特有的 UML、建模、开发方法论、项目管理的所有知识进行解说、并且对软件离岸的秘诀进行详细说明」不是本指南的目的。）



2. 离岸开发现状的问题点和对策

为了掌握离岸开发中的 UML 运用现状，我们分别对离岸开发的发包方（日方）和接包方（离岸方）实施了问卷调查和听证调查。调查结果的要点总结如下图。发包方和接包方均对 UML 期待很高、认为采用 UML 能够解决现在存在的大部分问题。



(1) 问卷调查概要

〈发包方（日方）的调查问卷〉

调查对象：UMTP 加盟企业及相关企业

调查时期：2006 年 11 月

有效回答数：70（主要是 PM/PL 的回答 平均开发经验 16.2 年 离岸开发经验 2.2 年）

〈接包方（离岸方）的调查问卷〉

调查对象：UMTP 加盟企业及相关企业的离岸中国企业

调查时期：2007 年 3 月

有效回答数：91（主要是 PM/PL 的回答 平均开发经验 4.4 年）

注：本次的调查对象因考虑到中国是日本目前离岸最多的接包方，所以选择了中国。

(2) 问卷调查结果（部分摘录）

对离岸开发中的课题、对 UML 的期待・见解以及在离岸开发中应用 UML 的好处等方面，根据调查结果摘录如下。

离岸开发中的课题

表 2.1.1 离岸开发中存在的课题及其严重程度 – 发包方（日方）

		极其严重	非常严重	严重	比较严重	完全不严重	严重程度高所占的比例（%）
1	因语言与文化之差别而造成误解	4	19	14	13	10	38
2	把内容正确传达给离岸企业需花费大量时间	1	15	16	18	10	27
3	由于设计书不确切造成误解	1	11	14	22	13	20
4	设计书与离岸企业交付的源代码有偏离	4	17	21	14	4	35
5	设计书有遗漏	2	13	19	16	10	25
6	为对应离岸企业来的大量提问造成对业务的挤压	6	19	19	13	2	42
7	大量问题在测试阶段之后才曝露出来	3	14	11	26	6	28
8	为了把设计书写详细，负担增大	3	14	22	14	8	28
9	离岸企业方的测试不到位，不够	5	13	14	22	6	30
10	离岸企业担当人员的辞职，对业务的延续产生影响	12	22	13	9	3	58
11	离岸企业接包维护的情况下，修改 BUG 时交货期长	11	23	16	7	2	58
12	不能与离岸企业在同一形象下共享有效信息	7	27	17	6	2	58

表 2.1.2 离岸开发中存在的课题及其严重程度 – 接包方（离岸方）

（注：由于部分调查问卷的问题不同，所以编号不连续）

		极其严重	非常严重	严重	比较严重	完全不严重	严重程度高所占的比例（%）
1	因语言与文化之差别而造成误解	6	7	11	31	36	14
2	把内容正确传达给发包企业(日方)需花费大量时间	2	6	14	32	37	9
3	由于设计书不确切造成误解	9	28	24	28	2	41
4	设计书与离岸企业交付的源代码有偏离	13	12	10	27	28	28
5	设计书有遗漏	12	13	33	19	4	38
12	不能与发包企业(日方)在同一形象下共享有效信息	7	27	17	6	2	58
13	项目组成员之间的沟通不顺畅	4	15	16	23	32	21

发包方（日方）・接包方（离岸方）都认识到与对方「12.不能在同一形象下共享有效信息」是严重的问题。接包方认为「3.由于设计书不确切造成误解」问题严重、而制作设计书的发包方却不那么认为。但是可以推测到，实际上由于设计书不确切，造成了发包方「6.为对应离岸企业来的大量提问造成对业务的挤压”这样一种格局。

离岸开发课题的解决方案

针对上述「12.不能在同一形象下共享有效信息」这一课题，如表 2.2 所示、有 74%的在设计书以及审核中使用 UML、或其他统一的图表形式。大多数人在实际运用图表并认识到了它的重要性。

表 2.2 设计书以及审核中使用图表及 UML 的比例 – 发包方（日本）

	情况	平均比例（%）	
（a）	使用 UML（即便只使用某个图表）	19%	
（b）	非 UML，但组织内部对图表的画法有统一标准并加以利用	38%	
（c）	非（a）（b），但个人对图表的画法有统一的标准并使用	17%	74%
（d）	使用图表，但画法没有统一的标准	14%	
（e）	用文字说明，说明中不使用图表	5%	
（f）	口头说明	3%	
（g）	其他的方法	4%	

并且如以下调查结果所显示，可以看到相对于一般的图表形式、人们期待 UML 可以收到更明确的效果。

表 2.3.1 对 UML 的期待和见解——发包方（日本侧）

	意见	完全反对	偏向反对	无所谓	有些赞成	非常赞成	赞成度（%）
1	UML 只要从事上游阶段的人员了解就可以了	29	28	8	2	0	3
2	即便使用 UML，成本减少的效果也不明显	3	9	28	21	6	40
3	周围几乎没有使用 UML 的案例	10	17	9	22	9	46
4	UML 培训难、培训成本高	4	10	17	31	5	54
5	使用 UML 能够提高生产性	1	4	38	19	5	36
6	使用 UML 能减少下游阶段的返工	2	5	30	24	6	45
7	使用 UML 能够提高质量	2	4	27	29	5	51
8	UML 太专业、太难	9	22	24	10	2	18
9	使用 UML 会加大报价的难度	5	20	32	8	2	15
10	使用 UML 会加大进度管理的难度	6	20	37	4	0	6
11	在设计书中使用 UML 可以减轻不确定性	1	2	18	37	9	69
12	在设计书中使用UML能够掌握设计书变更的影响范围	2	6	17	32	10	63
13	在审核中使用 UML 能提高沟通的效果	2	4	15	35	11	69
14	使用 UML 会导致开发产生混乱	12	29	20	6	0	9
15	在上游阶段中使用 UML 还不如使用专用工具	5	15	36	9	2	16
16	使用 UML 能促进发包方和接包方之间的沟通	2	3	30	23	9	48
17	使用 UML 能减少维护管理的总成本	1	13	36	12	5	25
18	使用 UML 能有效提高开发的自动化程度	3	15	26	18	5	34
19	与国内开发相比较、UML 更多的被应用于离岸开发中	3	6	42	12	3	23
20	与国内开发相比较、UML 更适用于离岸开发	4	6	29	24	3	41

表 2.3.2 对 UML 的期待和见解——接包方（离岸方）

（注：由于部分调查问卷的问题不同，所以编号不连续）

	意见	完全反对	偏向反对	无所谓	有些赞成	非常赞成	赞成度（%）
1	UML 只要发包方（日本）理解就可以了	51	22	4	4	1	6
2	即便使用 UML，成本减少的效果也不明显	7	23	35	15	2	21
3	周围几乎没有使用 UML 的案例	27	19	23	9	4	16
4	学习 UML 成本花费很高	4	18	26	27	6	41
5	使用 UML 能够提高生产性	0	0	14	47	21	83
6	使用 UML 能减少发包方（日本）的返工	1	6	21	33	18	65
7	使用 UML 能够提高质量	0	1	26	36	19	67
8	UML 太专业、太难	10	21	20	23	8	38
9	使用 UML 会加大报价的难度	4	32	39	7	0	9
10	使用 UML 会加大进度管理的难度	6	45	27	3	1	5
11	在设计书中使用 UML 可以减轻不确定性	1	3	12	45	21	80
12	在设计书中使用UML能够掌握设计书变更的影响范围	0	3	18	37	24	74
13	在审核中使用 UML 能提高沟通的效果	1	4	19	35	23	71
14	使用 UML 会导致开发混乱	24	37	15	5	1	7
16	使用 UML 能促进接包方和发包方（日本）之间的沟通	0	3	21	40	17	70
18	使用 UML 能有效提高开发的自动化程度	1	3	23	43	12	67
19	与其他开发相比，UML 更多的被使应用离岸开发中	2	12	44	20	4	29
20	UML 更适用于离岸开发	4	4	43	27	6	39
21	有关 UML 的书籍非常多	0	13	16	30	23	65
22	掌握 UML 对升迁和跳槽有利	1	4	25	28	24	63

发包方（日方）・接包方（离岸方）共同认识到 UML 具有「11.可以减轻设计不确定性」、「12.掌握设计变更的影响范围」、「13.在审核中使用 UML 能提高沟通的效果」的作用。由于双方之间「16.能促进彼此间的沟通」从而达到「7.提高质量」。接包方尤其对 UML 寄予较高的期待。但在另一方面，发包方多数认为存在「4.培训难、培训成本高」的课题。

使用 UML 的好处

表 2.4.1 使用 UML 的好处——发包方（日本方）

	好处	完全不重要	某种程度上重要	非常重要	加权合计
1	文档书写可使用全球通用的标准	13	26	22	131
2	可使用可视化的文档记录方法	10	26	24	134
3	可以使用让用户企业也能理解的文档记录方法	13	27	20	127
4	不用看源代码就能够从文档上检查出作过修改的地方、易于维护	14	31	15	121
5	开发的上游阶段和下游阶段无缝连接	16	27	18	124
6	离岸开发和国内开发可以使用相同的开发支持工具	9	21	30	141
7	上游阶段和下游阶段具有统一的沟通手段	8	26	26	138
8	可以使用全球通用标准的开发方法	10	27	24	136
9	能重用已开发的部件	15	28	17	122
10	能尽可能地使用不依赖于自然语言的表记方法	13	30	17	124
11	能够确认设计书和代码的一致性	8	31	21	133
12	能使用接包国的语言（例如：中文）进行沟通	24	22	14	110
13	能确立质量指标，所以质量有保证	13	21	26	133

表 2.4.2 开发中使用 UML 的好处——接包方（离岸方）

	好处	完全不重要	某种程度上重要	非常重要	加权合计
1	文档书写可使用全球通用的标准	7	41	43	218
2	可使用可视化的文档记录方法	5	44	42	219
3	可以使用让用户企业也能理解的文档记录方法	3	43	45	224
4	不用看源代码就能够从文档上检查出作过修改的地方、易于维护	5	43	43	220
5	开发的上游阶段和下游阶段无缝连接	2	48	41	221
6	离岸开发和国内开发可以使用相同的开发支持工具	17	42	32	197
7	上游阶段和下游阶段具有统一的沟通手段	7	39	58	259
8	可以使用全球通用标准的开发方法	5	46	40	217
9	能重用已开发的部件	5	36	50	227
10	能尽可能地使用不依赖于自然语言的表记方法	10	53	28	200
11	能够确认设计书和代码的一致性	5	26	60	237
12	能使用发包国的语言（例如：日文）进行沟通	5	35	51	228
13	能确立质量指标，所以质量有保证	5	20	66	243

在离岸开发中，与离岸方的沟通尤其变得重要。因此「12.能使用对方国家的语言进行沟通」非常重要。日中离岸的情况下，大多数场合中方都理解日语。从发包方来看，使用 UML 的效果并不

怎么明显，但是接包方（离岸方）感到非常有用。一般来说，上游阶段由发包方（日方）承担、下游阶段由接包方（离岸方）承担比较常见，双方对「7.上游阶段和下游阶段具有统一的沟通手段」这一优点均一致认同。

另外，UML 的特征如「2.可视化的文档记录方法」「1.全球通用的标准」「8.全球通用标准的开发方法」等都是离岸软件开发的要点。

再者，从维护的观点来看，「11.能够确认设计书和代码的一致性」等的支持率也很高。

(3) 听证调查

为了完善上述调查结果，我们进行了以下听证调查，确认本指南的内容同样得到接包方（离岸方）的认同。本指南不仅适用于中国，也能适用于印度，具备通用性。

<对中国的 PM/PL 实施的听证调查（一部分是书面的调查问题）>

目的：为了掌握定量调查问卷难于把握的实况细节

时间：2007 年 11 月

<对印度架构人员的听证调查>

目的：为了把握在离岸开发方面走在中日前头的美印之间的 UML 使用状况

时间：2008 年 3 月

3. UML 建模

3.1 UML 的特征

(1) 什么是 UML

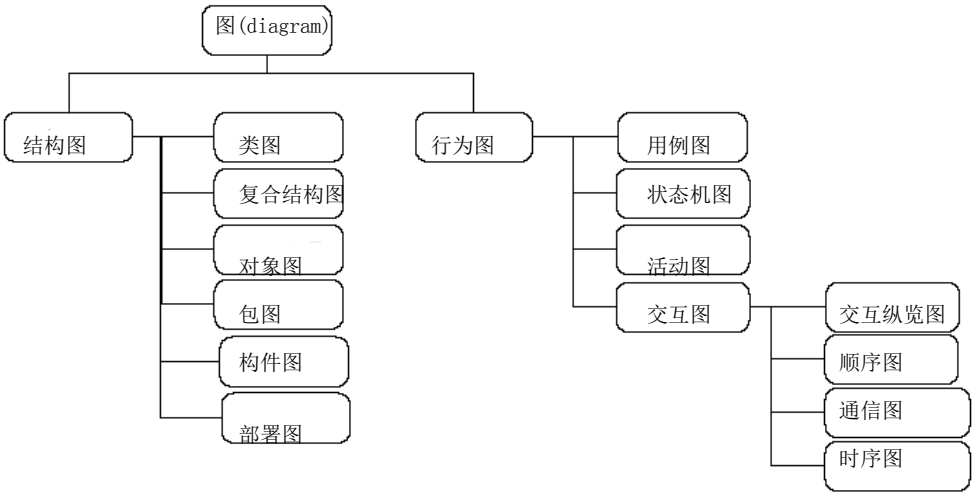
UML(Unified Modeling Language)是一种由 OMG(Object Management Group)制定标准的、用来设计、图式化描述软件开发成果物时的表达方法。

(2) UML 的特征

UML 的特征如下：

- 世界标准
  - 易于在不同的国度、不同的项目之间进行沟通。
- 表现力强而且理解容易
  - 使用 UML 提供的各种图（Diagram），可以从多个视点对分析・设计对象的进行表达。而且无论哪种图示均直观易懂。
- 可以贯穿整个开发过程的各个阶段加以使用
  - 能够确保各阶段成果物的可追踪性。在各阶段中开发者之间容易沟通。

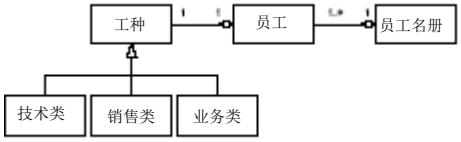
(3) UML 的图(diagram)



●结构图

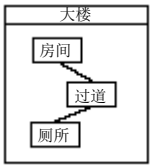
- 类图

  从概念上把握分析・设计领域的事和物。把它们用类和类之间的关系静态地表示出来。



- 复合结构图

  用层次关系表示出类和组件的内部结构。

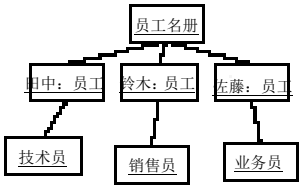


- 包图

  表示包含 UML 要素的包与包之间的关系。

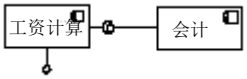
- 对象图

  用对象和对象之间的关系表示出系统某一瞬间的状态。



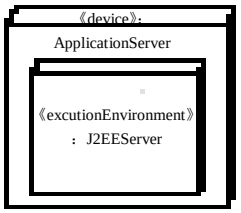
- 构件图

  表示软件组件（可重复利用的部件）的构成。



- 部署图

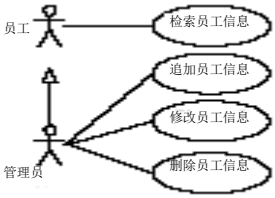
  表示系统的运行环境。



●行为图

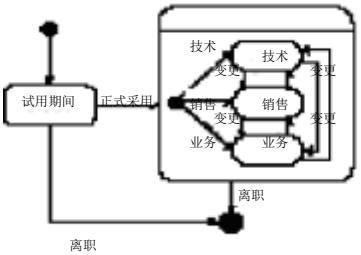
- 用例图

表示系统提供的功能及其与外部的关系。



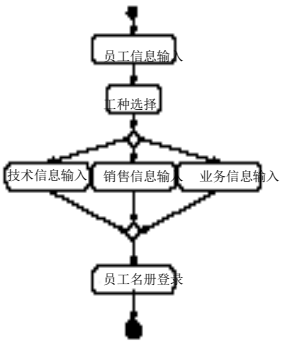
- 状态机图

表示一个对象随时间推移的状态变化。



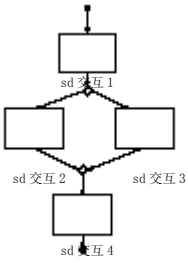
- 活动图

表示业务流、事件流、算法流（处理顺序）。



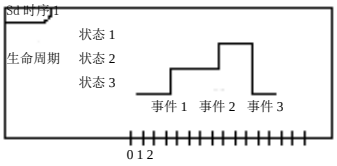
- 交互纵览图

表示多个交互图的概要关系。



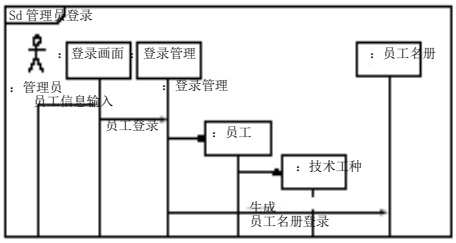
- 时序图

表示随时间推移而变化的生命期状态的变化、信息收发。



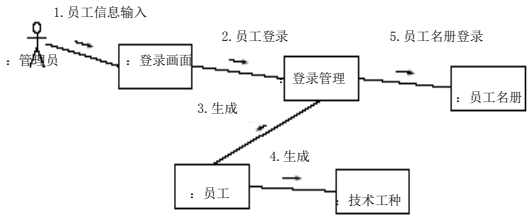
- 顺序图

表示相互作用（分析・设计对象中存在的物或人的交互）的时序。



- 通信图

将相互作用（分析・设计对象存在的物或人的交互）以物或人为中心、表示他们之间的的连接关系。



3.2 UML 建模的前提技能

为了有效地推进利用建模的离岸开发，发包方、接包方都要具备一定的 UML 建模技能。如果不具备这些技能的话，应事先组织学习和培训。根据分析员、设计员、实现人员、架构员、项目领导、项目经理等的立场不同，所需要的 UML 建模技能也有区别。

在此定义的技能等级，仅指建模能力，指作为分析员、设计员、实现人员、架构员、项目领导、项目经理等所必需的一般能力和知识。这些一般能力和知识在此不作定义。

【目的】

离岸开发的发包方和接包方之间使用 UML 交接作业成果物时，双方如果具备一定的 UML 建模能力的话，只需要交换必需最少量的成果物就能正确传达各自的意图。如果开发要员 UML 建模能力不足，那么不仅不能正确地传达成果物的意图、设计担当人员还必须承担对 UML 及其建模的说明，就会发生工数的浪费。

【详细和补充】

○ 等级的测定（确认）方法

UMTP 把 UML 建模能力分为 L1～L4 四个等级。定义如下：

级别	建模能力	说明
L4	能够基于实践指导建模	具有 L3 的能力，且在开发项目中具备一定数量或者期间的建模实践经验
L3	对实际业务能够实践建模	能够从易扩充性、易修改性的观点出发定义模型
		具备业务建模、分析、架构设计、嵌入式开发的专业知识（业务范畴可以选择）
L2	能够顺利地对 UML 模型进行读写（具备建模读写能力）	承担一部分的开发范围，能够建模。
		能够理解他人所建模型的含义
L1	能够理解简单的 UML 模型的含义	具备最基础的 UML 建模知识

UMTP 实施 L1～L2 级别的认证考试（L3 从 2008 年 4 月开始实施）。通过这些考试，就能够评价能力级别。

其中 L1 级别还分为 L1-T1 和 L1-T2 两门考试。

L1-T1：掌握 UML 的表达方法。

L1-T2：能够读懂用 UML 书写的模型并判断优劣。

※注：在中国实施 L1-T2 和 L2 两门考试。

在实际开发中、对各种角色的最低级别要求分别如下：

实现人员：能够看懂 UML 模型，编写源代码。→L1-T2 级别。

一般分析员：根据高级分析员的指示，能独立完成 UML 分析模型。→L2 级别。

设计员：能够根据系统架构独立完成设计模型。→L2 级别。

高级分析员：能够设计初期分析模型。→L3 级别。

架构员：能够设计架构模型。→L3 级别。

项目领导：能够审核分析模型・设计方针，并能够决策。→L3 级别。

项目经理：项目的最终负责人。→L1～L2 级别。

	L1-T1	L1-T2	L2	L3
实现人员	○	○	－	－
一般分析员	○	○	○	－
高级分析员	○	○	○	○
设计员	○	○	△	－
架构员	○	○	○	○
项目领导	○	○	○	○
项目经理	○	○	△	－

○ 能力提高

如果想要掌握 UML 的（L1-T1）知识、或者是初级（L1-T2）建模能力，去买 1 本公开出版的书籍来自学就足够了。但是想要获得 UML 建模能力(L2)的话，仅仅靠看书自学是很困难的。首先需要通过培训、从师来把握建模的要领，其次要通过大量的阅读和试写模型来掌握。

更进一步的高级建模能力（L3），必须在实际业务中积累经验。

○ 参考信息

● UMTP 标准遵循

公开出版的书籍中，有些标有 UMTP 标准遵循。这是指经审核符合【建模术语集 UML 篇 第 2 版】的内容的意思。

UMTP 标准遵循的书籍如下：

http://www.umtp-japan.org/modules/data4/index.php?id=7

● 培训教材的认证

教材的认证除了要审核是否符合【建模术语集 UML 篇 第 2 版】的内容之外，还要审核是否对建模所必备的知识点进行了说明。

认证教材如下：

http://www.umtp-japan.org/modules/data4/index.php?id=8

【相关信息】

无。

4. UML 的适用范围和开发诀窍（Hints & Tips）

从第 2 章问卷调查的结果可以看出，大多数的人都认识到了离岸开发中使用 UML 的好处。但是并非离岸开发一旦使用 UML 就能马上见效的，重要的是在开发的哪个阶段、如何来加以利用。本章介绍为了保障离岸开发的成功、在各个阶段中怎么使用 UML 的诀窍。说明以 UML 为中心进行，但也包含非 UML 独有的东西。

图 4.1 给出开发流程，表 4.1 表示各阶段的目的、作业内容、作业人员、输入成果物和输出成果物。在图 4.1 中只列出了主要成果物，详细成果物请参照表 4.1。开发各阶段需要注意的事项（诀窍），用要点号码标记在图 4.1 中。按照开发各阶段的要点号码以如下形式进行说明：

- 要点号与标题
- 内容
- 目的
- 详细和补充
- 相关信息

有些诀窍，虽然在条件满足的情况下予以实施可以收到明显效果、但在离岸开发的起步阶段无需勉强实施的要点，作为可选项（option）、可根据情况选择是否实施。

图 4.1 开发流程

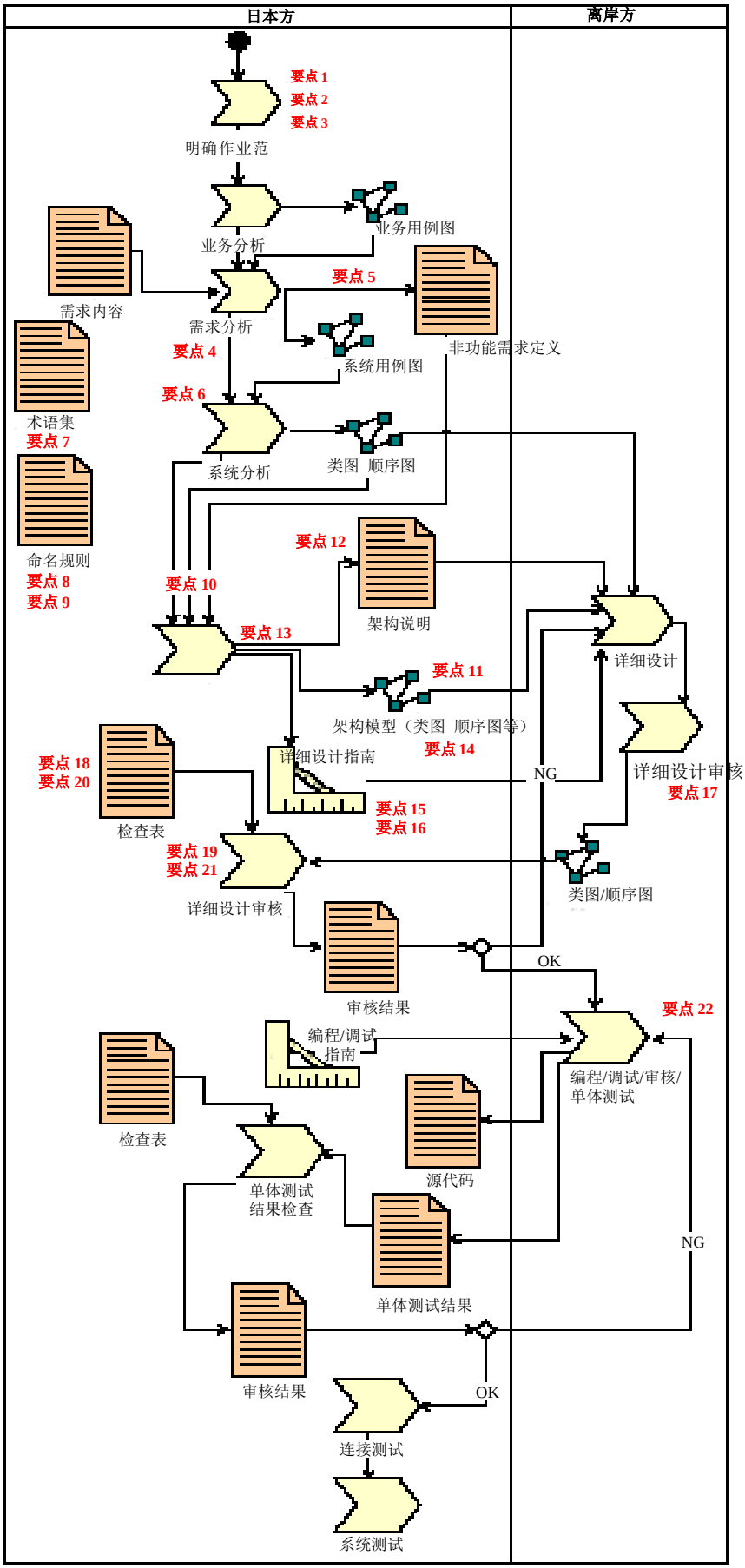

日本方
离岸方
要点 1
要点 2
要点 3
明确作业范
业务分析
业务用例图
需求内容
需求分析
要点 5
非功能需求定义
要点 4
系统用例图
术语集
要点 7
要点 6
系统分析
类图 顺序图
命名规则
要点 8
要点 9
要点 12
要点 10
架构说明
要点 13
详细设计
要点 11
架构模型（类图 顺序图等）
详细设计指南
要点 14
NG
详细设计审核
要点 17
要点 18
要点 20
检查表
要点 15
要点 16
要点 19
要点 21
类图/顺序图
详细设计审核
审核结果
OK
要点 22
编程/调试
指南
编程/调试/审核/
单体测试
检查表
源代码
单体测试
结果检查
NG
单体测试结果
审核结果
OK
连接测试
系统测试

表 4.1 各阶段的说明

阶段名：业务分析	
目的（日方）： 理解并明确本次系统的业务内容，使得需求分析阶段要将哪部分业务进行系统化的工作变得明确和简单。	目的（离岸方）： 理解本次对象系统的业务内容，为后继阶段的作业顺利进行作准备。
作业内容（日方）： 理解并明确业务内容，根据需要整理业务流程。	作业内容（离岸方）： 理解本次对象系统的业务内容。（如果参加上游阶段的话）
作业人员（日方）： PM 分析员	作业人员（离岸方）： PM 系统分析员（如果参加上游阶段的话）
输入成果物： ・业务流程	输入成果物： ・业务用例图 ・业务流程 ・业务用例描述 ・概念类图
输出成果物： ・业务用例图 ・业务流程 ・业务用例描述 ・概念类图	输出成果物：
相关信息： ・术语集 ・命名规则	相关信息： ・术语集 ・命名规则

阶段名：需求分析	
目的：（日方） 明确系统的功能需求和非功能需求。	目的：（离岸方） 从详细设计阶段开始接包时，并不直接利用本阶段的成果物。 通过本阶段的成果物，理解本次开发对象系统的功能需求。 理解了系统的功能需求，就能理解为什么需要这样设计，为详细设计以后阶段的沟通打下基础。
作业内容：（日方） 明确顾客对系统的需求。	作业内容：（离岸方） 参照该阶段的输出成果物，理解对象系统的功能需求。（如果参加上游阶段的话）
作业人员：（日方） 需求分析员	作业人员：（离岸方） 系统分析员（如果参加上游阶段的话） 详细设计员（如果参加上游阶段的话）
输入成果物： ・业务用例图 ・业务流程 ・业务用例描述 ・需求内容	输入成果物： ・系统用例图 ・系统用例图一览表 ・业务规则定义 ・系统顺序图 ・系统用例图描述 ・画面迁移图 ・画面・帐票设计书 ・非功能需求定义书
输出成果物： ・系统用例图 ・系统用例图一览表 ・业务规则定义 ・系统顺序图 ・系统用例图描述 ・画面迁移图 ・画面・帐票设计书 ・非功能需求定义书	输出成果物：
相关信息： ・术语集 ・命名规则	相关信息： ・术语集 ・命名规则

阶段名：系统分析	
目的：（日方） 用自然的形式分析系统开发对象领域的知识。 通过建立不依赖于开发、运行环境的模型，提高重用性和维护性。	目的：（离岸方） 从详细设计阶段开始接包时，并不直接利用本阶段的成果物。 通过本阶段的成果物，理解本次开发对象系统的功能需求。 理解了系统的功能需求，就能理解为什么需要这样设计，为详细设计以后阶段的沟通打下基础。
作业内容：（日方） 定义系统的上下文，表达概要设计级别下的系统构造和关联。然后整理出把握系统地理分布情况和运用的复杂性的概要，分析系统的行为。	作业内容：（离岸方） 参照该阶段的输出成果物，理解对象系统的功能需求。（如果参加上游阶段的话）
作业人员：（日方） 系统分析员	作业人员：（离岸方） 系统分析员（如果参加上游阶段的话） 详细设计员（如果参加上游阶段的话）
输入成果物： ・系统用例图 ・系统用例图一览表 ・业务规则定义 ・系统顺序图 ・系统用例图描述 ・画面迁移图 ・画面・帐票设计书	输入成果物： ・分析类图 ・分析顺序图 ・对象图 （・通信图） （・状态机图）
输出成果物： ・分析类图 ・分析顺序图 ・对象图 （・通信图） （・状态机图）	输出成果物：
相关信息： ・术语集 ・命名规则	相关信息： ・术语集 ・命名规则

<table>
<tr><td colspan="2">阶段名：架构设计</td></tr>
<tr><td>目的：（日方）
明确系统的基盘架构。</td><td>目的：（离岸方）
作为详细设计的输入。</td></tr>
<tr><td>作业内容：（日方）
定义候补的架构，根据分析的结果、提取能够重用的模型元素、优化定义的架构。同时进行组件的设计、数据库的设计，详细研究架构、识别类/子系统以及子系统之间的接口。</td><td>作业内容：（离岸方）
参照该阶段的输出成果物，理解系统的架构。（如果参加上游阶段的话）</td></tr>
<tr><td>作业人员：（日方）
架构员</td><td>作业人员：（离岸方）
架构员（如果参加上游阶段的话）
详细设计员（如果参加上游阶段的话）</td></tr>
<tr><td>输入成果物：
・分析类图
・分析顺序图
・对象图
（・通信图）
（・状态机图）
・非功能需求定义书</td><td>输入成果物：
・设计类图　・分析类图
・设计顺序图　・分析顺序图
・对象图
・包图
・构件图
・部署图
・架构图说明
・详细设计指南
・详细设计检查表
・数据模型图（E-R 图）</td></tr>
<tr><td>输出成果物：
・设计类图
・设计顺序图
・对象图
・包图
・构件图
・部署图
・架构说明
・详细设计指南
・详细设计检查表
・数据模型图（E-R 图）</td><td>输出成果物：</td></tr>
<tr><td>相关信息：
・术语集
・命名规则</td><td>相关信息：
・术语集
・命名规则</td></tr>
</table>

阶段名：详细设计	
目的：（日方） 把作业工数多的部分，委托给中方作业。	目的：（离岸方） 将模型细化、直到能够进行编程/调试作业为止。
作业内容：（日方） 检查中方的作业是否与架构有偏差。	作业内容：（离岸方） 以架构模型和架构说明书为输入，边作子系统间的调整，边细化制作编程/调试模型。
作业人员：（日方） 审核员	作业人员：（离岸方） 架构员（如果参加上游阶段的话） 详细设计员
输入成果物： ・详细设计类图 ・详细设计顺序图 （・对象图） ・详细设计审核结果（离岸方）	输入成果物： ・设计类图　・分析类图 ・设计顺序图　・分析顺序图 ・对象图 ・包图 ・构件图 ・部署图 ・架构说明 ・详细设计指南 ・详细设计检查表
输出成果物： ・详细设计审核结果（日方）	输出成果物： ・详细设计类图 ・详细设计顺序图 （・对象图） ・详细设计审核结果（离岸方）
相关信息： ・术语集 ・命名规则	相关信息： ・术语集 ・命名规则

阶段名：编程/调试・审核・单体测试	
目的：（日方） 　　继续把作业工数多的部分，委托给中方作业。	目的：（离岸方） 　　编写能使系统运行的源代码。
作业内容：（日方） 　　根据需要检查中方的作业。	作业内容：（离岸方） 　　以详细设计的成果物为输入，进行编程/调试和单体测试。
作业人员：（日方） 　　审核员	作业人员：（离岸方） 　　程序员
输入成果物： 　　・单体测试结果	输入成果物： 　　・详细设计类图 　　・详细设计顺序图 　　（・对象图）
输出成果物： 　　・检查结果	输出成果物： 　　・源代码 　　・单体测试结果
相关信息： 　　・术语集 　　・命名规则 　　・编程/调试指南 　　・检查表	相关信息： 　　・术语集 　　・命名规则 　　・编程/调试指南 　　・检查表

阶段名：单体测试结果检查	
目的：（日方） 检查质量是否符合标准。	目的：（离岸方） 进行质量的最终确认。
作业内容：（日方） 检查单体测试的结果。	作业内容：（离岸方） 根据需要修改单体测试的结果。
作业人员：（日方） 审核员	作业人员：（离岸方） 程序员
输入成果物： ・单体测试结果	输入成果物： ・审核结果
输出成果物： ・审核结果	输出成果物： ・单体测试结果
相关信息： ・术语集 ・命名规则 ・编程/调试指南 ・检查表	相关信息： ・术语集 ・命名规则 ・编程/调试指南 ・检查表

阶段名：连接测试	
目的：（日方） 用经过单体测试后的源代码进行连接测试。	目的：（离岸方） 进一步提高源代码的质量。
作业内容：（日方） 进行结合测试。	作业内容：（离岸方） 修正有关详细设计、编程/调试中发现的错误之处。
作业人员：（日方） 测试员	作业人员：（离岸方） 详细设计员 程序员
输入成果物： ・源代码 ・单体测试结果	输入成果物： ・详细设计类图 ・详细设计顺序图 ・源代码 ・错误报告
输出成果物： ・结合测试结果	输出成果物： ・详细设计类图 ・详细设计顺序图 ・源代码 ・错误报告 ・检查表
相关信息： ・术语集 ・命名规则	相关信息： ・术语集 ・命名规则

表 4.2 各阶段的主要成果物

阶段	成果物集	成果物	UML 的图
业务分析	业务流程模型	业务用例图	用例图
		业务流程	活动图
		业务用例描述	
	领域模型	概念类图	类图
			对象图
需求分析	用例模型	系统用例图	用例图
		系统用例一览	
		业务规则定义	
		系统顺序图	顺序图
			交互纵览图
		系统用例描述	有时需要制作活动图
	用户接口模型	画面迁移图	状态机图
		画面・帐票设计书	
	非功能需求	非功能需求定义	
	术语集	术语集	
系统分析	分析模型	鲁棒分析图	
		分析类图	类图
			对象图
		数据模型图（E-R 图）	
		相互作用图	顺序图
			通信图
		状态图	状态机图
架构设计/详细设计	架构模型	架构说明	
		软件配置	包图
			构件图
			复合结构图
		硬件配置	部署图
		基本处理方式	顺序图
	设计模型	设计类图/详细设计类图	类图
			对象图
		数据模型图（E-R 图）	
		相互作用图	顺序图
			通信图
		状态图	状态机图
			时序图
		类设计书	
编程/调试	程序集	程序代码	
单体测试/连接测试	测试成果物集	测试计划书	
		测试设计书	
		成果物	UML 的图

表 4.3 开发诀窍（Hints & Tips）一览表

阶段	要点号码	诀窍
明确作业范围	01	明确作业范围和作业分工 (*)
	02	选定需要使用的 UML 图
	03	不是非 UML 不可 (*)
业务分析/需求分析	04	力争参加上游阶段（可选）
	05	定义非功能需求
	06	使用分析模型理解业务 (*)
	07	制作术语字典
	08	建立命名规则
	09	建立建模规则
系统分析/架构设计	10	明确通用功能
	11	架构模型
	12	制作架构说明的成果物
	13	灵活运用模式
	14	指定设计书的描述级别和格式 (*)
	15	制作详细设计指南
	16	明确式样尚未决定的部分
详细设计/编程/调试	17	离岸方实施对成果物的审核
	18	日方的反复检查
	19	确认(Validation)和验证(Verification)
	20	用模型对详细设计进行审核
	21	各种 UML 图之间的一致性 (*)
	22	代码使用工具的代码生成功能（可选）
---	补充	日本本土的系统开发特点

(Option)：指那些在条件完全具备的情况下实施效果明显的诀窍，但没有必要离岸开发的初期阶段勉强实施。

（*)；备有样本。附录 A 是这些样本的说明。

【要点 01：明确作业范围和作业分工】

明确各阶段的作业分工。决定各阶段的成果物（UML 图以及 UML 图以外的东西）。明确离岸方要的担当开发领域、决定建模范围和测试范围。

【目的】

系统开发过程中，要推进多个企业（国内、海外）的项目进程。

因为各个企业、各个阶段的作业内容和成果物都不同，所以项目开始时要明确各个阶段的作业内容，避免发生作业阶段的偏差。明确发包方／接包方的成果物，减少交付时的纠纷。

【详细和补充】

明确各个阶段的离岸担当作业，提高作业效率。

例如、由于开发环境方面等带来的限制，关于运用、障害对策、安全需求等方面的内容由离岸方来对应确有困难。应事先安排好日方担当的领域和离岸方担当的领域。

<全体>

- 描述各阶段使用的成果物。
- 明确各阶段的输入和输出。
- 明确规定成果物的格式和内容记录的粒度。
- 对于互相关联的成果物，为了防止成果物之间的不一致，要明确规定的确认方法。

<架构设计、详细设计（建模）>

- 明确由基盘、其他开发部分提供的地方。

<编程/调试・单体测试（制造・制作）>

- 开发环境（服务器、软件种类、版本等）的状态
- 提供部件的日程表（预定提供时期）
- 提供部件（清单、外部设计书）
- 使用的开发工具

<测试>

- 考虑开发环境的状态
- 是否提供测试数据（提供数据样本能加大理解度）
  如提供测试数据、尽可能及早提供。

离岸型的软件开发特别要注意、在项目的一开始的时候就要把以下几点考虑进去：

- 离岸方能够准备的开发环境・测试环境的制约
  例：能否使用日语字符/中文字符、能否准备大型机、能否共享文件
- 有无离岸方难于设计・实现的有关非功能需求部分的制约
  例：需要依赖综合认证功能的测试、依赖现有测试设备的性能的测试、非实际运行环境引起

的制约。

- 离岸方难于准备的测试数据的制约

  例：受安全规定制约的顾客数据。依赖于日语/中文的字符的测试数据。

此外，在离岸开发中沟通尤其重要。必须明确沟通计划。例如谁跟谁、多少时间（比如每周一次）、用什么方法进行沟通等都应事先决定。

不仅作业范围／作业分配如此，离岸方的开发体制也有必要在需求中明确记载。

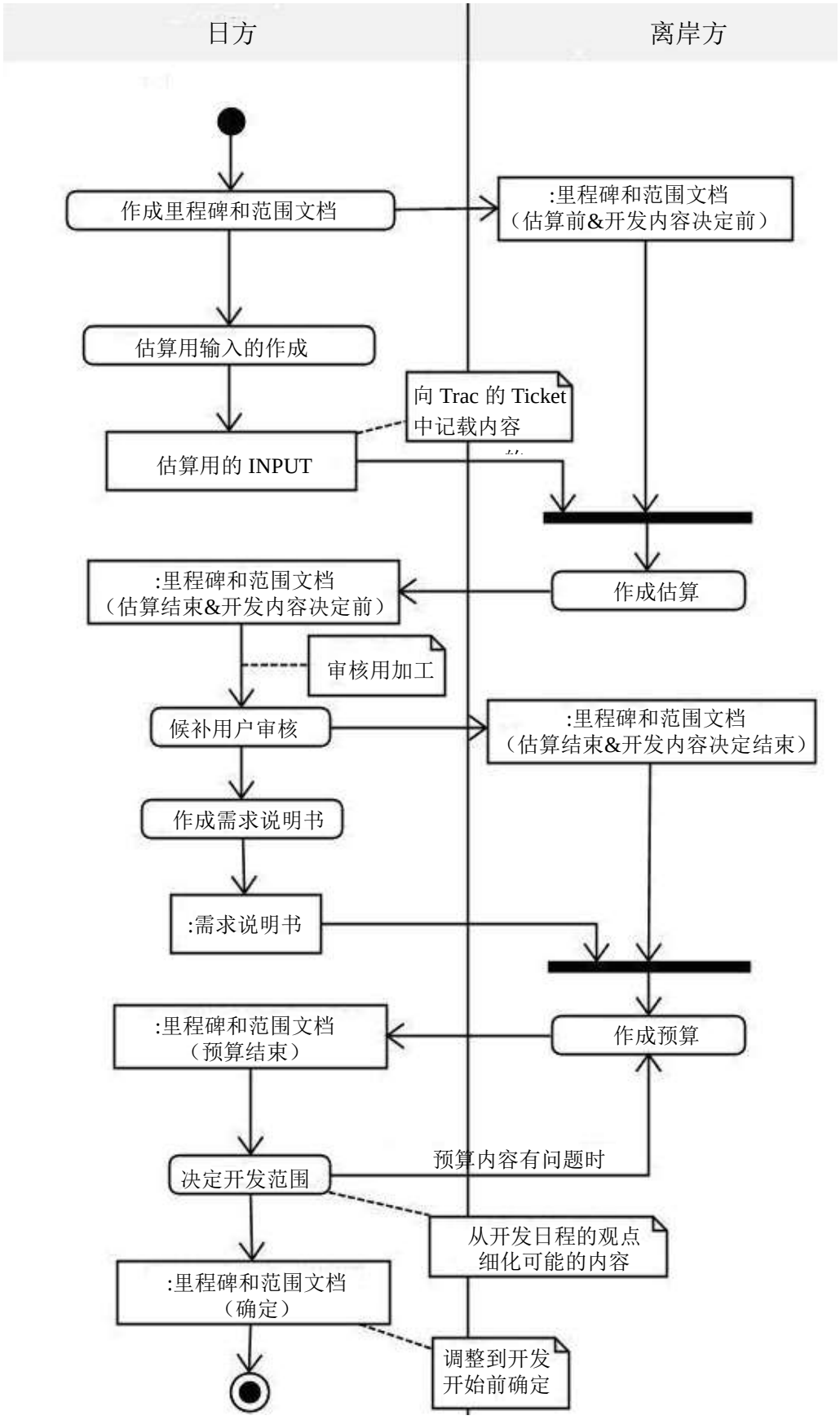
【相关信息】

图 4.1 开发的流程

表 4.1 各阶段的说明

【用活动图明确记载作业分组的例子】

采用活动图的泳道来区分日方和离岸方的作业



【要点 02：选定需要使用的 UML 图】

明确各阶段所要使用的 UML 图和辅助资料的种类。

【目的】

在开发的各个阶段所用的 UML 图并不是一成不变的。因根据开发的内容来选择和取舍。项目开始时，预先决定各个阶段所用的 UML 图、提高作业的效率。

【详细和补充】

明确各个阶段所用的 UML 图，可以避免各个阶段开始作业时发生的浪费。根据必需･可选对 UML 图进行等级分类(在说明建模依据时需要)。在业务分析～详细设计的各个阶段，可以考虑使用下述 UML 图（UML2.x）：

<业务分析>	
・用例图	△（根据情况可选）
・活动图	○
・类图	△（根据情况可选）
・对象图	△（根据情况可选）

<需求分析>	
・用例图	○
・活动图	△（根据情况可选）
・顺序图	△（根据情况可选）
・交互纵览图	△（根据情况可选）
・状态机图	△（根据情况可选）

<系统分析>	
・类图	○
・对象图	○
・顺序图	○
・通讯图	△（根据情况可选）
・状态机图	△（根据情况可选）

<架构设计・详细设计>	
（包括对其前阶段制作的图进行修改和细化）	
・构件图	△（根据情况可选）
・合成架构图	△（根据情况可选）
・类图	○
・通信图	△（根据情况可选）
・状态机图	△（根据情况可选）
・顺序图	○
・对象图	○
・包图	○
・部署图	△（根据情况可选）
・时序图	△（根据情况可选）

【相关信息】

表 4.2 各阶段的主要成果物

【要点 03：不是非 UML 不可】

仅用 UML 来制作系统开发的所有成果物是不可能的。还应根据需要使用 UML 图之外的东西。

【目的】

在开发中，不必限制仅用 UML 定义的图来写设计书。如何将开发对象的内容描述得更为易懂、更为准确才是设计书本来的目的。不可迷失本来目的。

【详细和补充】

把握 UML 图的特点，选择能将信息描述清楚的最合适的图。

在适宜用 UML 的地方使用最合适的 UML 图。对那些 UML 不支持的成果物，还是应当使用 UML 图以外的图来描述。例如：有些信息整理成表格就能一目了然，如果硬是要套用 UML 的模型来描述，结果只能是适得其反。

所以，系统开发中切勿拘泥于试图用 UML 图来描述所有细节。防止陷入文档的泥潭。

除了 UML 以外，可以考虑使用以下的图形：

- 画面迁移图
- 画面・帐票设计书
- 数据模型图（E-R 图）
- 数据流图（DFD）
- 鲁棒分析图（UML 图的扩充）
- 网络图
- 一览表(EXCEL 等)

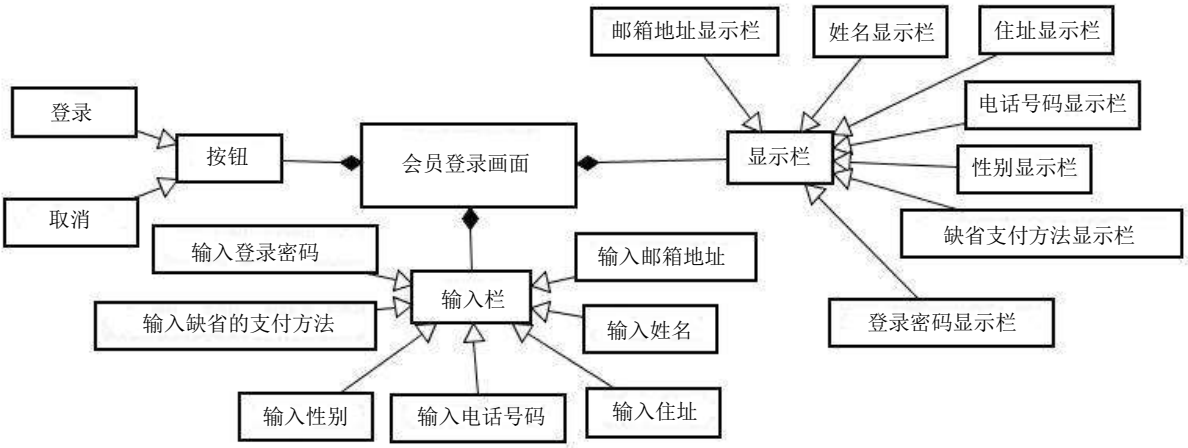
【相关信息】

表 4.2 各阶段的主要成果物

【最好不使用 UML 图的例子】

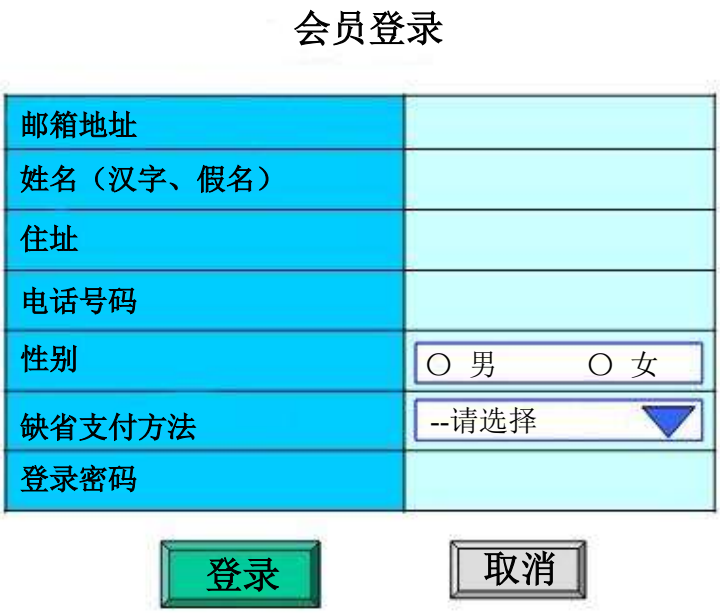
在设计会员登录的时候，使用 UML 作成的画面如例 1 所示，用画面开发工具、EXCEL 等作成的画面如例 2 所示。在进行画面格式设计、确认和审核的时候，例 2 明显比例 1 容易使人明白。所以，在画面、帐票的设计书里附上例 2 的图，最好不使用 UML。

<例 1>



输入住址

<例 2>



【要点 04：力争参加上游阶段】

进行离岸开发时，如果离岸方拥有日语能力强、设计经验丰富的 SE，则最好应当让他们参加到在日本国内开发的上游阶段中来。

【目的】

将详细设计以后的阶段进行离岸开发时，这样做能够有效地推进详细设计以后阶段的开发。

【详细和补充】

上游阶段全部由日方企业承担、详细设计以后的阶段由离岸方承担时，即使成果物制作得非常完备，要让离岸方完全理解系统仍然是有困难的。

因此，通过让离岸方的成员参与上游阶段的工作，理解所开发系统的目的、内容，就能减少自详细设计阶段之后的错误。并且由于在离岸方存在理解系统整体概念的成员，就有可能可以将一部分的结合测试工作委托给离岸方。参加人数视系统的规模而定，一般为几个人左右。应当在对离岸方是否拥有能够胜任上游阶段工作的 SE 做出准确判断的基础上而加以实施，逐步扩大离岸方的担当领域。

【相关信息】

要点 01：明确作业范围和作业分工

要点 06：使用分析模型理解业务

要点 11：架构模型

【要点 05：定义非功能需求】

要将非功能需求整理清楚。所谓非功能需求，是指决定系统功能应当如何来实现（系统的性质）的需求。

【目的】

非功能需求是架构设计时的重要需求，必须在架构设计之前加以明确化。非功能需求将成为架构设计时的输入。

【详细和补充】

系统需求分为功能需求和非功能需求。

在使用用例图明确功能需求的同时，定义非功能需求。非功能需求一般有如下几个方面：

- 开发／执行环境（硬件、软件）
- 性能需求
- 可靠性需求
- 成本（短期、长期）
- 安全性
- 维护需求
- 运营需求

这些需求不仅仅存在于离岸开发领域、它们就是系统所要求的非功能需求。离岸开发的情况下，会因“貌似常识其实并非常识”造成诸多的误解・问题，所以明确定义包括非功能需求在内的成果目标非常重要。离岸方如果认为非功能需求不明确时，应及早向发包方确认。

【相关信息】

要点 11：架构模型

【要点 06：使用分析模型理解业务】

制作分析模型、能够将业务内容可视化、便于理解。

【目的】

在系统分析中使用模型，能够将业务内容的整体框架在视觉上加以理解。模型由精通需求的日方制作。模型的描述语言为日语，建议不要写很长的文章。通过使用模型，即使离岸方项目组成员的日语水平不怎么高、也能够做到牢靠地传达信息。

【详细和补充】

分析模型以类图为中心制作、旨在表达业务的内容。分析模型不描述实现的方法。内容不允许有矛盾，但无必要收罗详细细节。只需要将能够达成共识的信息加以建模便可。这个阶段的不确切问题点的排除，有助于减少来自后继阶段的返工。如果离岸方不能理解分析模型的内容或有疑问，应当尽早提出。

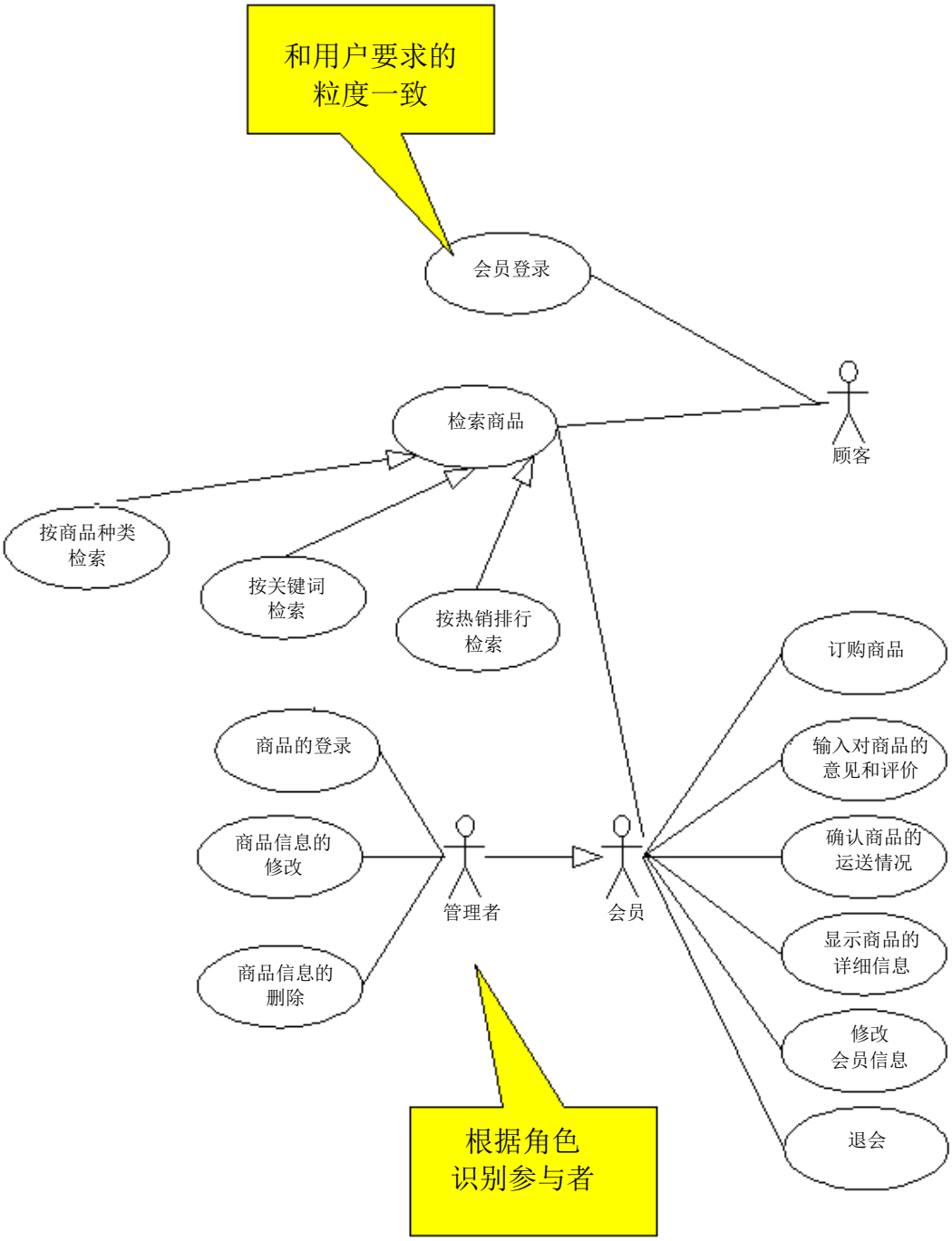
【相关信息】

附录 成果物样本（1）各阶段的成果物 ③阶段：系统分析

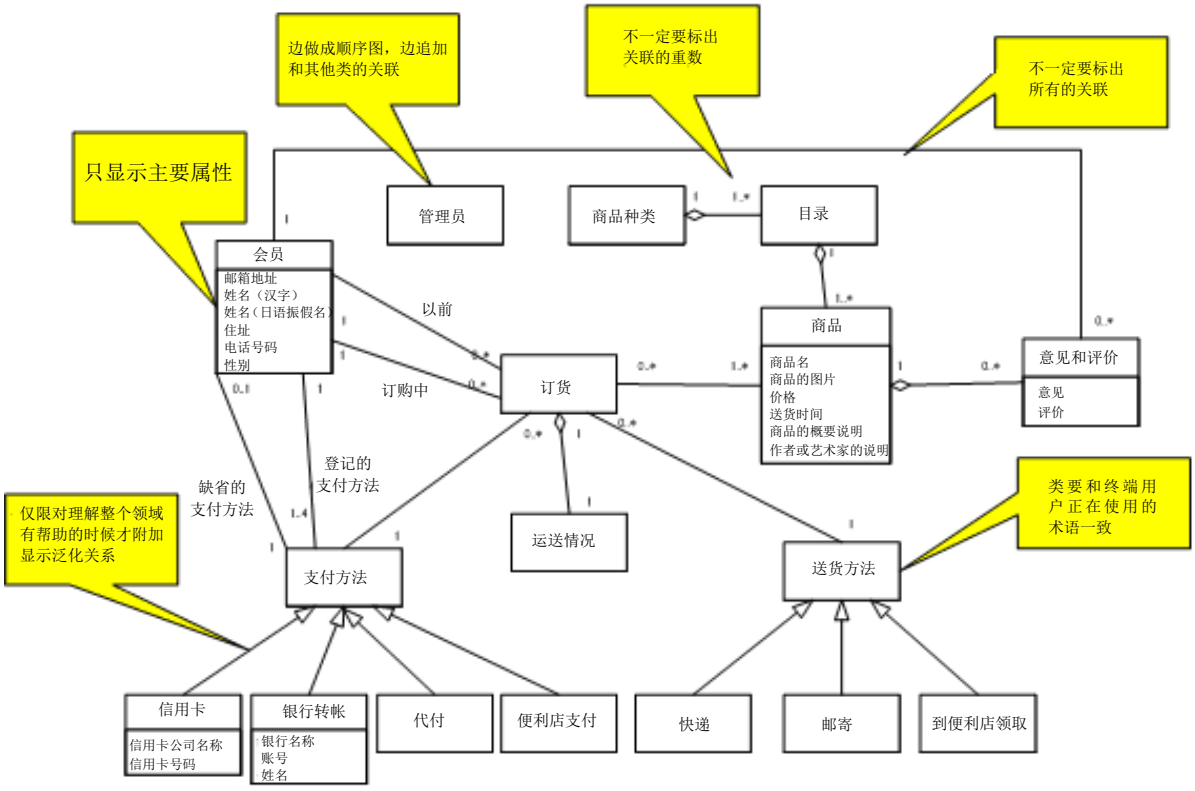
【用于理解业务的分析模型的例子】

使用用例图、分析类图来理解业务。

用例图的例子



类图的例子



<注意事项>

- 以终端用户（业务）的视点作成，不要太详细

用例名就是实际的业务名称，属性名是实际业务中使用的术语。

【要点 07：制作术语字典】

制作术语词典。业务术语和通用术语词典分开建立、能收到明显效果。

术语词典不仅要有日语、最好还要有离岸方的语言（离岸方是中国的话，中文或者英语）。特别是业务术语，有离岸方语言的话、更能正确向离岸方项目组成员传递信息。

另外，不要使用日语片假名，用英语为好。

【目的】

目的是为了提高对业务内容的理解和统一认识。

对于离岸方的项目组成员而言，明确命名规则，以图达到下列效果。

- 提高对日方项目组提供的设计书的理解度
- 提高离岸方项目组的设计书的制作效率
- 可以用于交货时的检查

【详细和补充】

记录术语和内容。用于提高对业务的理解度。

要明确术语词典版本升级时的联络方法。

应尽可能地实时做到保持日本项目组和离岸方项目组的词典版本的统一。

【相关信息】

要点 08：建立命名规则

UMTP 建模术语集：

http://www.umtp-japan.org/modules/data4/index.php?id=2

【要点 08：建立命名规则】

建立命名规约(Naming rule)。至少要在系统分析阶段进行命名规约的标准化 。

【目的】

将开发中要使用的各种名称的命名规则标准化，目的是加强需求分析／系统分系阶段的成果物的式样理解和提高后继阶段成果物的制作效率。

这有益于离岸方的项目组成员读懂非母语写的设计书和制作后继阶段的设计书。

并且对于离岸方的项目组成员而言，明确命名规约(Naming rule)，有以下好处：

- 提高对日方项目组提供的设计书的理解度
- 提高离岸方项目组的设计书制作质量
- 可以用于交货时检查

对日方的项目组而言，也能帮助理解和审核在日语用词有错误、说明不到位的设计书。

【详细和补充】

采用英语还是日语、长度为几位、各位怎样来区分使用等，应事先统一。另外，日语的词语后面注上英语效果很好。

决定命名规约(Naming rule)的项目有以下候补项目：

- 子系统名/组件名
- 包名
- 用例名
- 类名
- 属性名
- 操作名/方法名
- 事件名
- 消息名
- 状态名
- 画面 ID、画面迁移 ID

等等

【相关信息】

要点 07：制作术语字典

【要点 09：建立建模规则】

与建立源代码的编程规则，建立建模时的规则(rule)。

【目的】

建模时，通过建立建模规则(rule)，能够使得项目组成员制作的模型具有统一的标准，从而提高维护性。

【详细和补充】

通过对 UML 的各种要素(类、包、消息等)分别建立独自的规则、分类，可以使得各个 UML 要素的含义更为明确。

例如、在系统分析、架构设计、详细设计中，通常把类(class)分为 BCE(Boundary，Control，Entity)。Boundary 是画面和其他系统的接口类；Control 是流程控制类；Entity 是实体类。

此外，还有诸如「数据传递类」,「数据库存取类」和「有时限的消息类」等，由开发项目组或者企业单位设置自己的规则，方便项目组成员之间的意识沟通。

另外，最好预先定义模型的修改权限。例如：离岸方只能追加属性为 private 的操作，而要追加 public 属性的操作就必须得到日方的认可等等。

根据需要设计这些建模的制作规则时，也可体考虑使用 UML 的扩充功能。在 UML 中可以使用 UML profile。通过 UML profile 扩充的 UML 功能有：构造型（stereotype）、制约(constraint)和元属性(meta-propties)。

【相关信息】

要点 07：制作术语字典

要点 08：建立命名规则

【要点 10：明确通用功能】

研讨 UML 建模阶段时的功能共通化。

对于设计/编程/调试时离岸方的项目组独自的共通化，日方项目组也应在架构上事先予以确立。

【目的】

提高质量和开发效率。

具体是为了防止离岸方项目组成员的下列倾向：

- 反复拷贝粘贴所提供的样本或者初期开发成果物，稍作修改就作为成果物提交。
- 反复拷贝粘贴设计/编程/调试的结果，造成可以作为通用组件、通用功能、通用模块的类似功能重复存在。

【详细和补充】

离岸方的项目组为了提高内部的开发效率，以样本为模板独自进行共通化、制定步骤、进行拷贝展开的情况比较常见。

这种情况最好事先由日方提供架构和指南。日方要在早期阶段进行分析和审核，以确认功能分割是否恰当合适、共通功能是否被完全抽出。

为了使相同的处理逻辑，不要分散在各个功能组件中，我们应该：

- 在安排作业阶段计划时，要安排研讨组件共通化的作业项目。
- 在审核的内容里增加审核组件是否已共通化的审核项目。

【相关信息】

无

【要点 11：架构模型】

「架构模型」、是指硬件和软件的基本架构的设计。架构模型要坚固、内容要共享。

【目的】

为了判断硬件和软件的架构是否满足客户的业务需求，必须充分理解作为对象的业务内容。因此，架构模型要由日方制作。并且架构的变更，对系统而言会导致大规模的设计变更・式样变更，必须在用长远的眼光下、考虑到系统的将来扩充和变更的因素来进行设计。

架构设计委托离岸方的话，会导致将来维护或者系统扩充时、发包方（日本）对架构理解肤浅、无法进行合适的变更・扩充的后果。（基于这点，可以考虑选定特定的企业包括维护在内长期离岸。即使是这种情况，日方也有必要充分进行审核、把握内容。）

【详细和补充】

作为开发对象的系统的硬件、软件架构，应该由精通需求的日方制作。此时需要考虑的事项有：

- 这个架构在功能需求上和非功能需求上是否都妥当。
- 如何实现客户要求的性能和容量。
- 如何解决来自正式环境/测试环境的制约事项。
- 如何满足包括故障对应的运营需求。
- 如何满足安全需求。
- 如何解决开发成员的能力问题。

等等。

架构模型由日方研究、设计后，作为成果传达给离岸方的项目组成员。根据需要也可以制作诸如「基于硬件、软件架构开发的步骤」等指南。

【相关信息】

要点 12：制作架构说明的成果物

附录 成果物样本 （1）各阶段成果物 ④阶段：架构设计

【要点 12：制作架构说明的成果物】

制作说明系统的软件架构的成果物。

【目的】

在离岸软件开发中，在开发主体由日方的需求分析～架构设计向离岸方的详细设计发生转移时，日方和离岸方双方的项目组成员都必须对开发对象系统的软件架构拥有相同的正确理解。日方项目组成员如何有效地将他们是根据怎样的观点和构架来对开发对象系统进行设计的？正确传达给离岸方的项目组成员，这是本要点的目的。

【详细和补充】

使用补充设计书作为本流程的输入，研讨软件架构，输出研讨结果，也就是对架构的说明书。从系统分析到架构设计阶段的类图和顺序图、以及对此作补充说明的资料群等，构成架构说明的成果物。

主要由功能需求方面出发，将开发对象系统设计的硬件／软件架构，为何如此设计的理由、判断要点，架构瞄准的目标、用各种图给予表达。根据需要还可以使用构件图、包图、部署图。与使用日语说面资料相比、通过使用模型图，能够更为正确地向离岸方的项目组成员传达信息。

综上所述，制作架构模型的流程有下列相关成果物：

- 作为输入的成果物：　非功能需求定义书（补充设计书）
- 作为输出的成果物：　架构说明
  详细设计指南

架构模型，架构说明文档等不要作成以后成为摆设，要确认是否能够实际使用。

【相关信息】

要点 11：架构模型

附录 成果物样本 （2）架构设计书

【要点 13：灵活运用模式】

在架构设计、或者详细设计中，应灵活采用架构与设计上众所周知的各种模式。

【目的】

要想传达架构设计中架构设计者的意图、或者是详细设计中组件与类的职责时，采用项目组成员间彼此熟知的设计模式，能够促进双方达成共识。因为即使不加说明，也能收到增加了这些部分的目的・职责等信息同等的效果。

【详细和补充】

软件开发中的设计模式，经常用 UML 组合来说明。各种设计模式包含了各自的设计诀窍，当它渗透到开发者中时，不仅能通过选用・重用这些设计模式达到确保质量的目的，还能在设计和审核的交流沟通过程中，通过设计模式达成共识的场合非常多。架构设计中架构级别的设计模式有、MVC(Model，View，Control)等。应用这些模式能够容易地将架构意图无偏差地传达给负责详细设计的项目组成员。详细设计中设计级别的模式、例如使用 GoF(Gang of Four)等设计模式，在审核时就能很方便地想象各设计单元的职责，使得精力聚焦到应该检查的重点、令检查很得要领。使用这类模式将会非常有效地促进项目组成员之间统一认识。

离岸开发不一定非得要学会使用模式，但是为了得到上述效果，可以根据开发对象与项目组成员的能力，作为保障沟通的方法之一，应该考虑掌握的一个候补。

【相关信息】

要点 06：使用分析模型理解业务

要点 11：架构模型

要点 12：制作架构说明的成果物

【要点 14：指定设计书的描述级别和格式】

对离岸方成果物的各类设计书，指定 UML 描述的级别和文档格式。

【目的】

不仅是 UML，离岸开发经常被要求提供成果物的样板。日方的指示或指南如果不充分，就会造成离岸方各项目组成员之间的记述详细程度有差别、文档格式有差别问题的发生。这也是离岸方项目组不太愿意行动起来统一全员的记录详细程度和文档格式的背景原因。

本要点还是要根据离岸方交付的成果物是要提交给客户，还是日方项目组成员检查用，来考虑如何运用为好。

【详细和补充】

关于包含在成果物中的 UML，应根据实际的书写格式，按日方要求的记述详细程度和模型的粒度，做成离岸方项目组成员能够参考的样本、作为指南的一部分提供给离岸方。这时，最好能够明白需要在 UML 中表达什么、表示出什么就可以了，能够定义模型的目的。通过这些，才能够容易做到 UML 详细度/正确度/粒度记述的整齐。

关于例外处理／出错处理，也有必要按照想要记录的级别给出适当的样本。（想当然肯定会被追加而不写进样本的话，往往会造成最终成果物里也漏写的后果。）

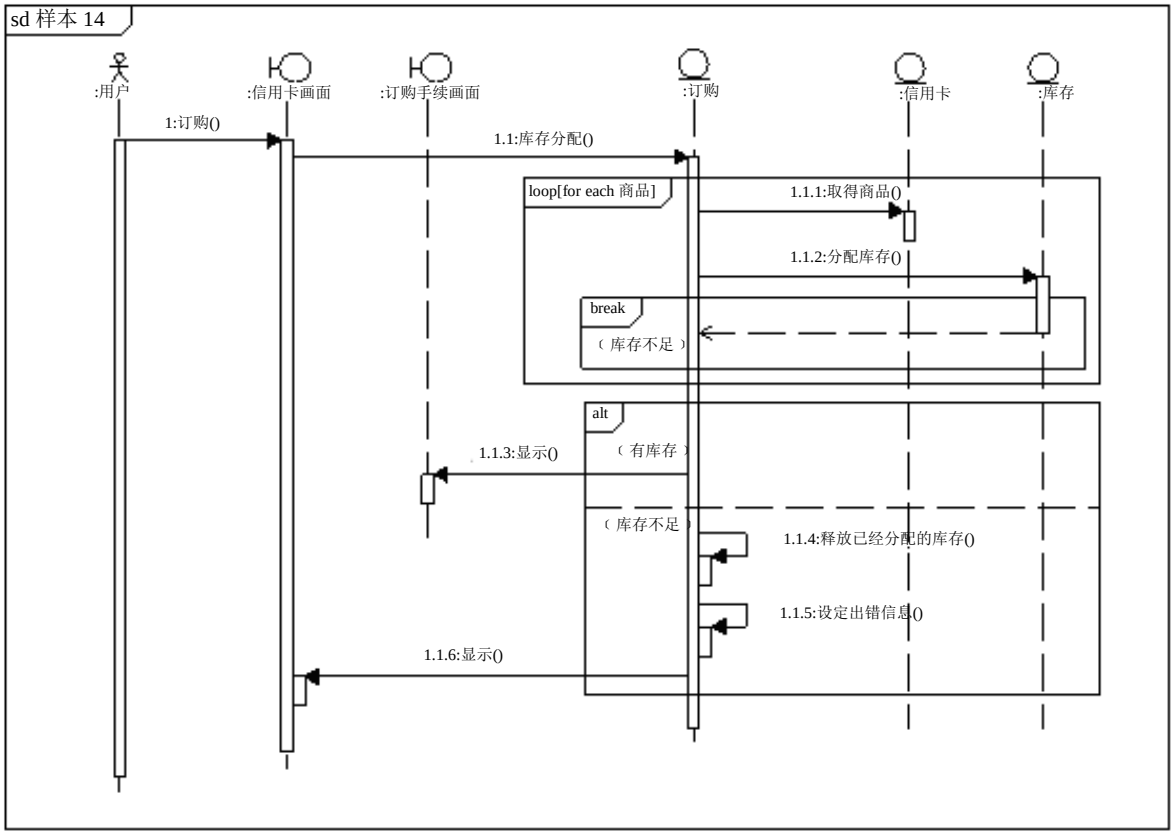
此外，作为记述成果物时的注意点「避免模棱两可的描述、要使用具体的表达」、「使用术语要统一」、「片假名词语用英语表示」，最好也要一并予以指示・指南。

【相关信息】

无

【例外处理/错误处理的例子】

在本系统中，将商品放入购物车的时候不作库存分配，按下“订购”按钮时作库存分配。购物车里的物品全部有库存时进入办理订购手续的画面，即使只有一个物品库存不够，也要放弃所有其他物品的库存分配，回到原先的画面（购物车画面），显示错误信息。



注意事项；

① 往往只描述满足特定条件时的处理（本例中“有库存”），不描述不满足条件时的处理（本例中“库存不够”）。如果两种情况都要描述，本例明确表示了两者都必须描述。

② 错误时的处理（本例中“解除已经分派的库存”）如果不描述的话，很容易被忽略，不予实现。

③ 错误时的处理要具体描述要作什么处理（本例中“回到原先的画面（购物车画面），显示错误信息。”）。

④ 还要补充描述依存于运行环境、实现方式等的处理（明确出错信息 ID，出错信息发送到哪个对象等）。

【要点 15：制作详细设计指南】

根据架构，制作进行详细设计的指南。

【目的】

架构设计里考虑到了维护性、效率、性能等方面。

详细设计中，只要沿袭架构的思想，就能维持系统全体的维护性、效率和性能。

为了使详细设计作业不偏离架构的方向，指南制作是非常重要的。

【详细和补充】

在架构说明中使用类图、顺序图等模型以及自然语言来说明系统的架构。

在详细设计中，有必要和架构设计同样，采用它来描述系统（模型）的其他部分（功能）。

架构的说明中记述的只是架构的思路。指南就要具体说明如何使用建模工具、在哪里制作类图、顺序图；类、对象部署在什么地方等。

把架构说明看作整体方针的话，指南就是它的具体细分。

【相关信息】

附录 成果物样本 （3）详细设计指南

【要点 16：明确式样尚未决定的部分】

要把成果物中式样尚未决定的部分明确告诉离岸方。离岸方如果发现说明不明确的地方，不要随意判断，应尽早联系发包方进行确认。

【目的】

由于开发期间短等的原因，在需求分析～系统分析～架构设计阶段，日方式样尚未决定部分仍然很多的情况下就委托给离岸方，这种情况并不少见。没有注明式样尚未决定部分的成果物，其内容缺乏一致性，会导致离岸方的项目组成员产生许多疑问、提出许多问题。

【详细和补充】

在递交给离岸方项目组成员的成果物中，明确注明式样尚未决定的部分（尚未决定或者变化可能性大），可以抑制吸收式样未定部分的遗漏。

对于式样尚未决定的部分（尚未决定或者变化可能性大），可以在成果物 UML 中用备注或者注释等明确标明：

- 哪些范围的式样尚未决定，影响波及的范围
- 预计现在的式样将会怎样变化（是今后被进一步细化、还是内容会被替换为完全不同的内容）

日方项目组成员通过对这些项目的管理，可以防止式样决定・式样变更发生时离岸方开发部分遗漏这些部分的吸收、减少式样变更的工数。在与离岸方的进度状况确认会议上，可以根据这些未决定部分的清单，确认反映情况的更新、确认式样已决事项。

【相关信息】

无

【要点 17：离岸方实施对文档的审核】

让离岸方项目组成员确实实施对成果物的审核。

【目的】

如果离岸方得知日方也要对成果物的内容进行审核，离岸方有可能会将完全没有审核过的成果物交付给日方。其背景是因为存在「因为日方要审核，离岸方也审核的话、作业重复浪费工数。」这样一种观点。

【详细和补充】

对于所委托的成果物，双方应经过协商并确定在离岸方审核对象的清单、审核的检查要点、审核的合格标准。此时要让离岸方明确理解离岸方审核的目的和日方审核的目的区别，让离岸方切实实施对成果物的审核。离岸方也必须做好审核报告。日方可以根据需要对审核报告进行检查。

基本的审核要点一般有：

- 是否完全覆盖了预定开发功能清单中的所有功能
- 设计书是否对前阶段的所有设计内容都进行了记述。
- 是否遵守了记述级别和格式的规定
- 术语是否使用统一的正确术语
- 预定开发功能的静态结构是否正确传达并被设计和记述了
- 预定开发功能的动态表现・举动是否正确传达并被设计和记述了
- 预定开发功能物理部署是否正确传达并被设计和记述了
- 类的职责有没有混乱
- 各种图之间是否具有一致性
- 式样变更有没有反映进去

等等。

【相关信息】

要点 18：日方的反复检查

【要点 18：日方的反复检查】

日方要在离岸方作业开始后马上进行检查

注：本指南中把对对成果物的严格的确认・承认称作「审核」；而把抽样方式的确认、不光是成果物还包含认识上・理解程度的确认称作「检查」。

【目的】

对于离岸方的项目组成员来说，使用的是不太熟悉的语言来进行交流沟通，所以：

- （日方认为）很难把握离岸方项目组成员的理解程度
- （离岸方认为）在错误的理解上越陷越深。

本要点的目的就是要防止这种互相不好好确认相关的内容就分头作业、不能消除误解的情况的发生。

如果将项目委托离岸方后不闻不问，很容易发生无法完成成果物、质量明显很差的问题。所以，必须切实地执行这些检查。日方通过合适的检查，有助于提高质量、降低开发工数。日方必须确保检查所需的工数和人才。

【详细和补充】

在离岸方作业开始后日方也应采用遍历（Walk-though）方式频繁实施检查。检查要点如下：

a)成果物的内容记述是否合适

b)成果物的格式是否遵照标准

c)对需求和对设计内容的理解有没有偏差

日方应该从开发的早期阶段就介入，不断地检查离岸方的审核内容。日方不是检查所有的成果物，而是在合适的时间、挑选合适的成果物进行检查。

如果在开发过程中，离岸方项目组成员养成互相确认自己的理解和成果内容的习惯就更好了。

【相关信息】

要点 17：离岸方实施对成果物的审核

【要点 19：确认(Validation)和验证(Verification)】

要从是否达成日方目的的 Validation(妥当性确认)和确认离岸方实现内容的 Verification(验证)这两个观点出发来进行审核。

【目的】

离岸开发的作业分工明确：日方负责业务分析～系统分析、主要关注于客户需求的实现。离岸方负责详细设计～单体测试、主要关注于日方设计的实现。由于关心视点的不同，就会产生遗漏测试范围、遗漏需求功能实现的问题。为了防止这种遗漏，不仅要对照日方的设计要求验证离岸方的实现成果，还要对成果是否满足了客户需求的妥当性进行验证。

【详细和补充】

离岸方通常通过单体测试和结合测试，对照详细设计或者架构设计和程序的执行结果来验证(Verification)是否按照承接的设计书正确实现。日方却经常需要确认(Validation)该实现是否还正确满足客户业务的需求。

注意这两个观点的不同，要在离岸方实施的以 Verification 为主的审核之基础之上再做到：

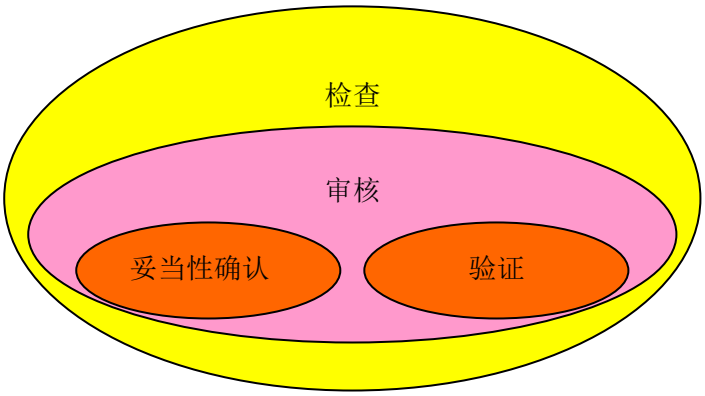
- 希望离岸方实施什么样的测试，尽早明确地提供 Validation 观点的具体测试要求。
- 从离岸方收到成果物之后，日方要同步准备好可以实施 Validation 观点的审核和测试。

例如可追踪性矩阵，这是一种通过把需求和实现相关联的方式，可以确认各个需求都被哪些实现所满足的有效确认手法。如果学会使用，将是很有效的。

在本指南中，前面出现的「审核」、「检查」，和本节中出现的「妥当性确认」、「验证」词语，他们之间的关系和用法如下图所示。

换句话说，以承认成果物为目的审核，需要从妥当性确认和验证两个视点来展开审核。除了已经定义的成果物审核之外，还应进行圆满无误地推进远程开发的各种广范围的检查。

。



【相关信息】

要点 01：明确作业范围和作业分工

要点 17：离岸方实施对成果物的审核

要点 18：日方的反复检查

【要点 20：用模型对详细设计进行审核】

详细设计中，要对类图、顺序图、对象图和包图进行审核。

【目的】

在进行编程/调试前要根据模型确认方向性。没有必要去读源代码来分析流程，可以用模型（顺序图等）以大家都能够容易看懂方式切实地实施审核。

日方通过模型对详细设计进行审核，可以提高编程/调试的质量。

【详细和补充】

与常规开发相比较，离岸开发更有必要切实地进行审核。但是如果检查项目过于细致，就会导致日方项目组成员的审核工数的膨胀。所以，使用模型来审核是非常有效的。

根据日方项目组制作的架构说明、架构指南等，离岸方项目组进行详细设计，制作顺序图、类图、对象图、包图等成果物。

日方项目组审核离岸方项目组制作的顺序图、类图、对象图和包图。

审核时，参照架构设计书、检查离岸方项目组制作的制作顺序图、类图、对象图和包图是否与架构设计书有偏差。

【相关信息】

要点 09：建立建模规则

要点 12：架构模型说明成果物的作成

要点 15：制作详细设计指南

要点 21：各种 UML 图之间的一致性

【要点 21：各种 UML 图之间的一致性】

要注意离岸方项目组成员设计的各种 UML 图之间的一致性。

【目的】

不仅是 UML，离岸开发中，经常发生成果物内容互相之间缺乏一致性的情况。

日方提供成果物的一致性缺乏，会引起离岸方项目组成员的许多质疑和提问。日方接受成果物时的一致性缺乏，会造成日方项目组成员理解内容困难、使得双方都无法审核。在日方发出的设计书中，必须将所需信息全部写明白，这点非常重要。

【详细和补充】

动态表现方面

- 类图／顺序图／交互图之间
- 顺序图／状态机图／活动图之间

静态／物理结构方面

- 类图／顺序图／对象图／构件图之间

等等，相关图之间应该有互相对应部分存在的图的一致性，应该是日方审核详细设计时必须检查的对象。

另外在各个图中，相同的东西有没有使用不同的术语来表达，从一致性的角度来看必须加以检查。

- 明明写着在某个顺序中工作的对象，在类图中却看不到
- 引起某个状态迁移的触发器，在顺序图、活动图中却看不到
- 部署在某个节点上的组件，在类图、构件图中却看不到

这样的不一致，会使编程/调试出错。

【相关信息】

无。

【各种 UML 图之间的一致性的例子】

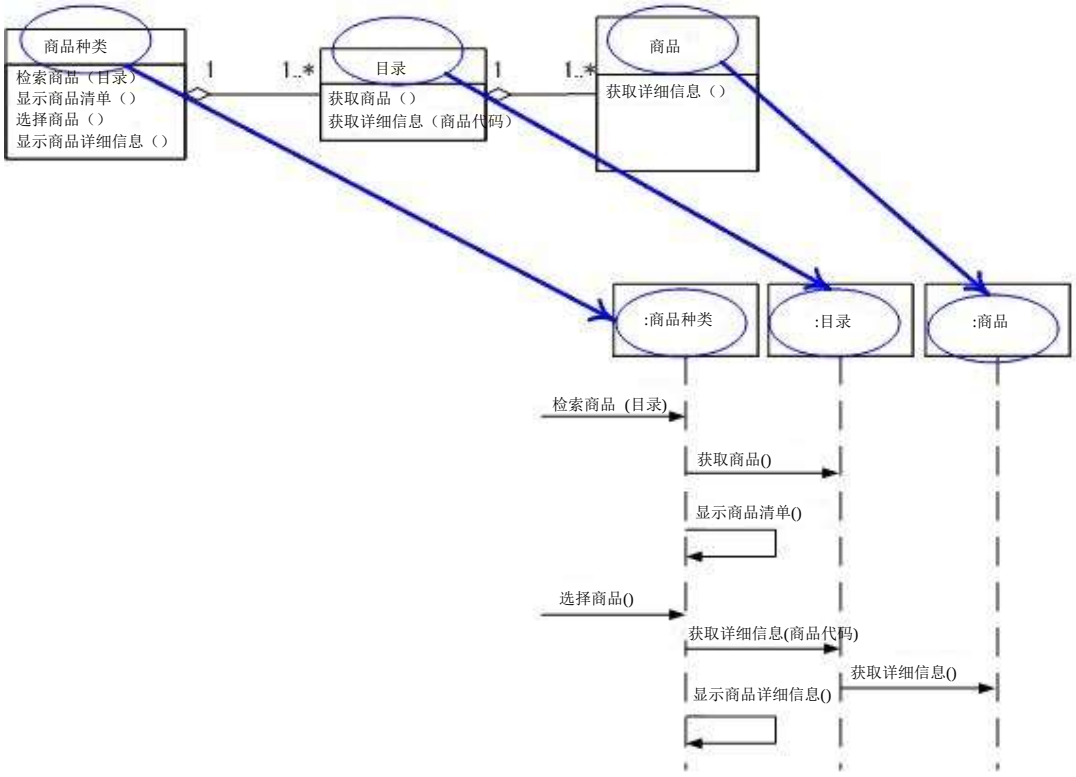
○ 顺序图和类图的对应

顺序图和类图主要留意如下 3 处的一致性。

例：用商品种别检索商品

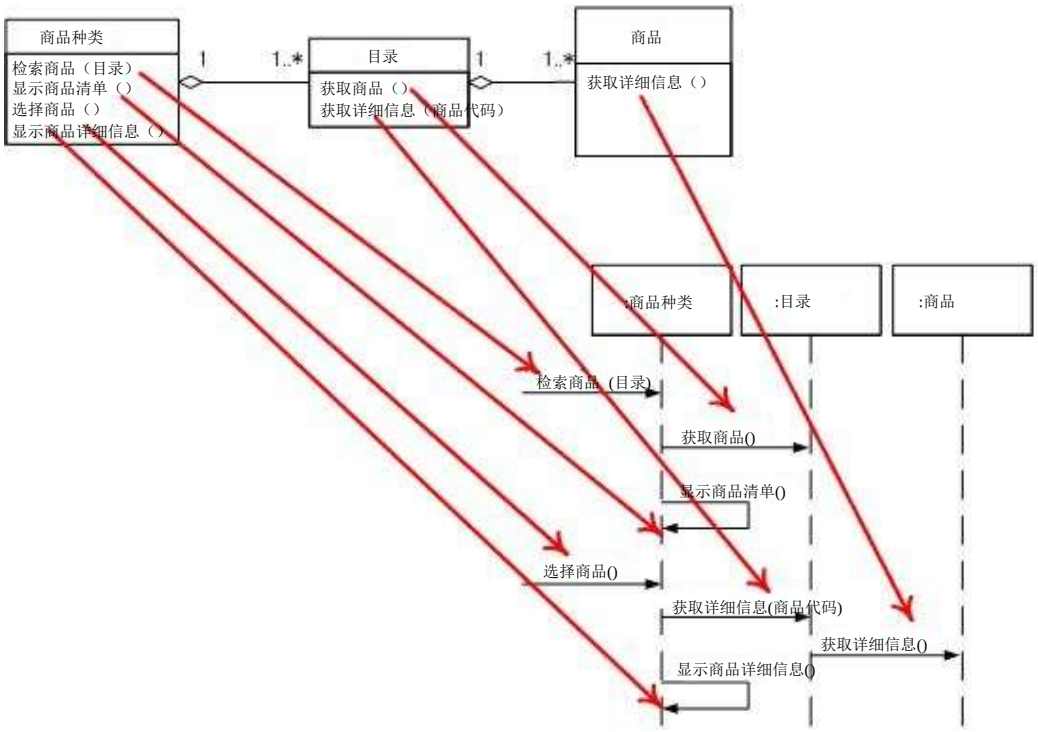
- 生命线和类

  顺序图的生命线必须是类图中定义的类。



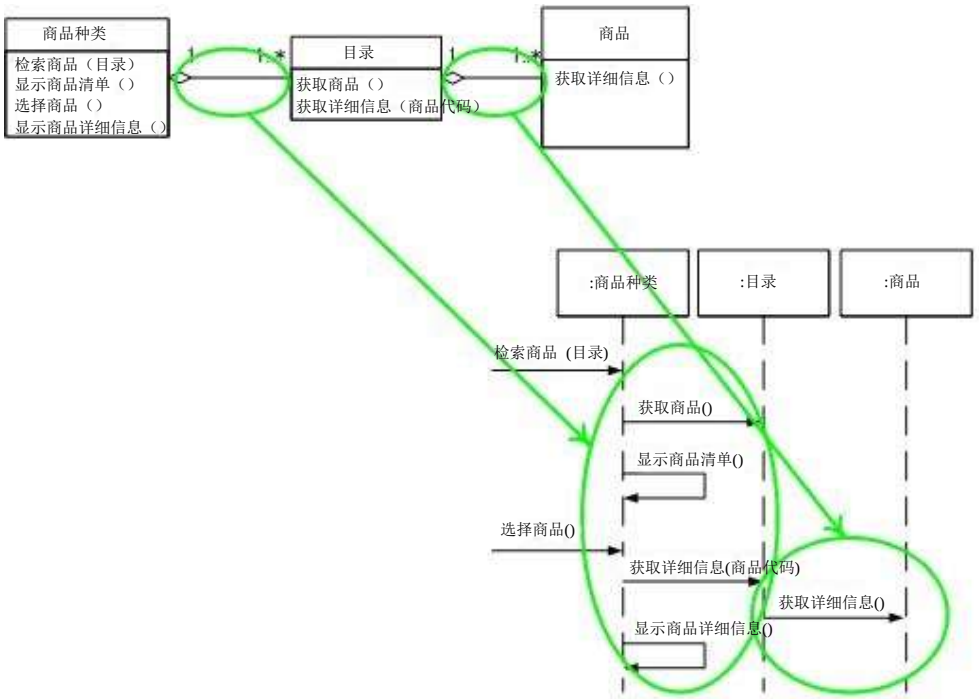
- 信息和操作

  顺序图中被调用信息的生命线必须是类图中定义的调用该信息的操作的类。调用信息是通过调用操作实现的。

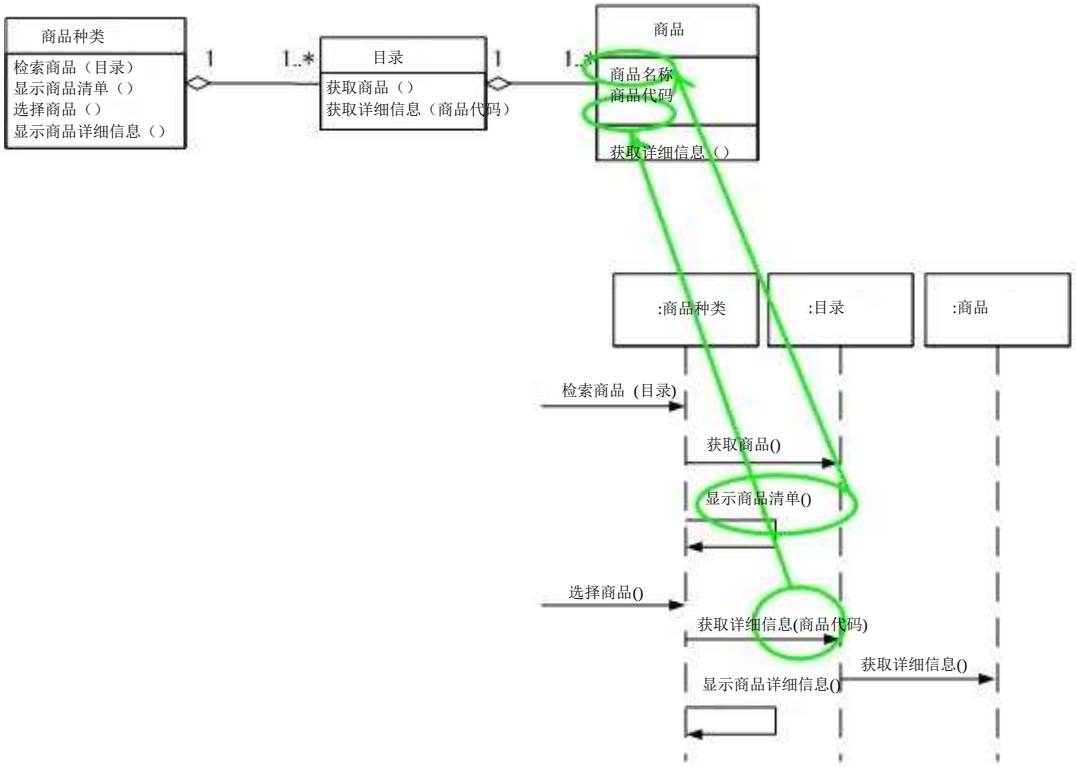


- 信息和操作

  顺序图中调用和被调用信息的生命线之间的调用关系必须是类图中定义的调用和被调用该信息的操作的类之间的关系。



（注）另外，顺序图中数据的存取也要和类图中定义的属性一致。



【要点 22：使用代码生成工具产生代码并进行调试】

如果有能使用的建模工具，编程/调试时使用工具的代码生成功能，生成代码模板，在这个代码模板上进行编程。虽然并不一定非要使用工具不可，但是，使用工具能够有效地开发可靠性高的代码。

【目的】

在详细设计中，日方项目组要审核离岸方项目组制作的类图和顺序图等成果物。

对于审核后被确认为模型设计上没有问题的（可靠的）模型，用建模工具的代码自动生成功能产生代码，可以提高代码的可靠性。

【详细和补充】

对类而言，用建模工具的代码自动生成功能可以产生源代码的骨架(source code skeleton)。

根据源代码框架，能将类名、属性名、操作名等输出到源代码中，可以对照模型（类图）确认源代码的内容。

对处理流程（主要是顺序图）而言，一般的建模工具不对其生成代码，需要手工检查（有些建模工具能够把代码执行的结果逆向输入，生成顺序图）。

利用建模工具的代码自动生成功能，通过对可视性好易于理解的模型的检查，可以提高源代码的可靠性。

【相关信息】

无。

【补充：在日本本土系统开发的特点】

明确发包方（日方）的系统开发特征，消除和离岸方（离岸方）的差距。

【目的】

开发日本的系统，具有下列倾向：

- 设计变更被认为是理所当然的，项目推进以此为前提。
- 对测试的覆盖度和管理方法等要求非常严格

明确了日方的系统开发特征，可以减少发包方／离岸方双方由于认识不同而造成的项目纠纷。

【设计篇】

顾客、开发方（日方和离岸方）必须共同认识到，设计变更是造成项目质量下降的最大因素。

【详细和补充】

发生设计变更的主要原因如下：

①在顾客所在的行业里的常识、习惯、特殊条件下的举动等在设计书里没有描述，往往到了测试结束后交付顾客确认时才被发觉。

②日本的顾客往往会对操作方法、画面布置等细节提出要求。

③日本的顾客对即便互相达成合意了的需求，也会提出变更要求。

④在需求还没有完全确定的阶段就已经要求离岸方开始所担当的作业（详细设计）。

【编程/调试、测试篇】

【详细和补充】

有关测试的主要要求事项如下：

①测试需要遍历所有路径

条件的变化和组合，全部的画面迁移、全部代码（覆盖率 100%）等等。

②一般需要提交测试结果的证据（evidence）

- 为了明确处理模式，需要提交测试证明。
- 为了证明开发过程中的质量，需要提交测试证据。

③测试指标的基准值是有规定的。

- 根据代码行数、画面的项目数、数据库 DB 的表的个数设定测试项目数。
- 设定正常、异常、边界的各种测试用例的比例。

④测试中的错误件数是有规定的

- 正常、异常、边界的错误个数如果不满足各自的基准值，要求重新测试。
- 错误个数如果超过各自的基准值，要求重新检查。

⑤测试中需要提交多种管理资料

- 错误记录票、问题课题表、错误发生情况的曲线、测试阶段质量管理图等等。

附录 A. 使用要点的样本事例的说明

网络购物

A 公司在研究上网销售的系统

○　商品的检索方法

检索商品有如下三种方法

①　根据商品种类检索

②　根据关键字检索

③　根据热销排行检索

①　根据商品种类检索

商品可以按照种类（书、杂志、CD\、DVD）划分。商品可以按各自所属的种类编制目录，分类整理。

书类
・动漫
・医学
・娱乐
・音乐
・科技
・参考书
・乐谱
・教育
・日常生活
・经济
・游戏
・工程
・语言
・少儿书
・计算机
・资格考试
・词典
・人像
・地理
・商务
・历史

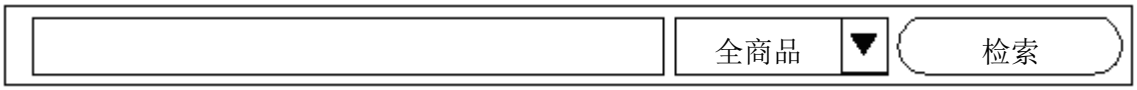
杂志类
・动漫
・音乐
・游戏
・婚庆
・保健
・漫画
・计算机
・信息
・汽车
・摩托车
・商务
・文艺
・生活方式
・餐饮
・旅游

CD类
・流行音乐
・动漫音乐
・初级听力
・演歌
・古典音乐
・爵士乐
・流行音乐
・摇滚乐
・游戏音乐
・外国音乐

DVD类
・外国电影
・日本电影
・电视剧
・音乐
・动漫
・曲艺
・综艺
・趣味
・体育
・偶像
・成人

② 根据关键字检索

选择要检索的对象，输入关键字，然后按检索按钮，显示相应的商品清单。



检索的对象可以是：

- 全商品
- 书
- 杂志
- CD
- DVD

③ 根据热销排行检索

选择要检索的对象，按热销排行按钮，显示相应的热销排行榜商品清单。

检索的对象可以是：

- 全商品
- 书
- 杂志
- CD
- DVD

○ 商品的显示

选择商品，按显示商品信息按钮，显示下述商品信息。,。

这些是全商品目录通用的信息。每个目录可以显示各自不同的信息。：

- 商品名
- 商品的图片
- 价格
- 送货日期
- 商品的概要说明
- 作者或者艺术家的说明
- 客户的意见和评价

○ 会员的登录

要订购商品、发表意见和评价，首先需要作为会员登录。会员登录时，需要在会员登录画面上输入如下会员信息，然后按下“登录按钮”。：按下“登录按钮”后，出于临时登录状态。系统向所输入的邮箱地址发送邮件。邮件内有 URL。打开这个 URL，在其显示的画面上按“确认”按钮，完成正式登录。

- 邮箱地址（登录号）
- 姓名（日语汉字和假名）

- 住址
- 电话号码
- 性别
- 缺省支付方法（信用卡支付、银行转帐、便利店支付）
- 登录密码

正式登录后，用登录号和登录密码就可以登录。登录之后，可以使用下列功能：

- 订购商品
- 对商品发表意见和评价
- 确认商品的运送情况
- 查询所购商品的详细信息
- 修改会员信息
- 腿会

○ 商品的订购

购入商品，需要事先登录。

检索商品，显示。按下“放入购物车”。可以选择多个商品放入购物车。

按下“查看购物车”按钮，显示购物车中所有的商品清单。可以删除不需要的商品。

按下“订货”按钮，进入订货程序。在订货程序中，选择送货方式（快递、邮寄、到便利店领取）。选择支付方法（信用卡、银行转帐、代扣、便利店支付）。显示会员登录时选定的缺省支付方法。可以选择新的支付方法，只是需要输入新的支付方法相关的信息。输入的信息被保存，以便后用。最后确认所购商品的明细核合计，按下“确认”按钮。交易成立。这时显示订购号码，并且通过邮件把订购号码发送给订购者。

在便利店接收的时候，把购买番号告诉对方。

○ 对商品发表意见和评价

会员在登录后，可以输入对商品发表意见和评价

不限于自己已购商品。

意见可以自由输入，最长不超过 300 个字。

评价可以从 1-5 打分。数字越大评价越高。

○ 确认商品的运送情况

会员可以确认所购商品的运送情况。

登录后显示商品运送情况画面。输入需要确认的商品的订购号码。可以看到相应商品的运送情况。

- 准备中，在送往取货典途中，到达取货店，已经签收。
- 准备中，运送途中，已经签收。

○ 查询所购商品的详细信息

可以查询所购商品的详细信息。包括已经签收的非本次订购的商品。

登录后显示所购商品明细画面。选择需要显示详细内容的商品的订购号码，就能显示相应商品的详细信息。

○ 修改会员信息

可以修改已经登录的会员信息。

登录后显示修改会员信息画面。根据需要修改相应的信息。按下“更新”按钮保存修改后的信息。

○ 退会

登录后，选择退会，显示确认画面。按下“退会”按钮实现退会。

退会后会员信息完全被删除，不能再次登录。会员的所有操作都不能使用。

○ 管理员的登录

可以在非公开的画面上，登录管理员。登录管理员时，要输入管理员的代码和密码。

管理员除了能够进行会员的所有操作，还能进行如下操作：。

- 商品的登录。
- 商品信息的修改
- 商品信息的删除。

○ 商品的登录

管理员可以在非公开的画面上登录。进行商品的登录。

选择商品目录后，输入如下信息。不同的商品目录可能会有如下信息之外的不同信息。根据需要补充输入。

- 商品名
- 商品的图片
- 价格
- 送货日期
- 商品的概要说明
- 作者或者艺术家的说明
- 客户的意见和评价

○ 商品信息的修改

管理员可以在非公开的画面上登录。进行商品信息的修改。

修改商品信息，首先，选择商品。商品的选择有如下方法：

- 选择商品目录，然后在商品一览中选择所需的商品
- 通过检索画面检索商品，选择所需的商品

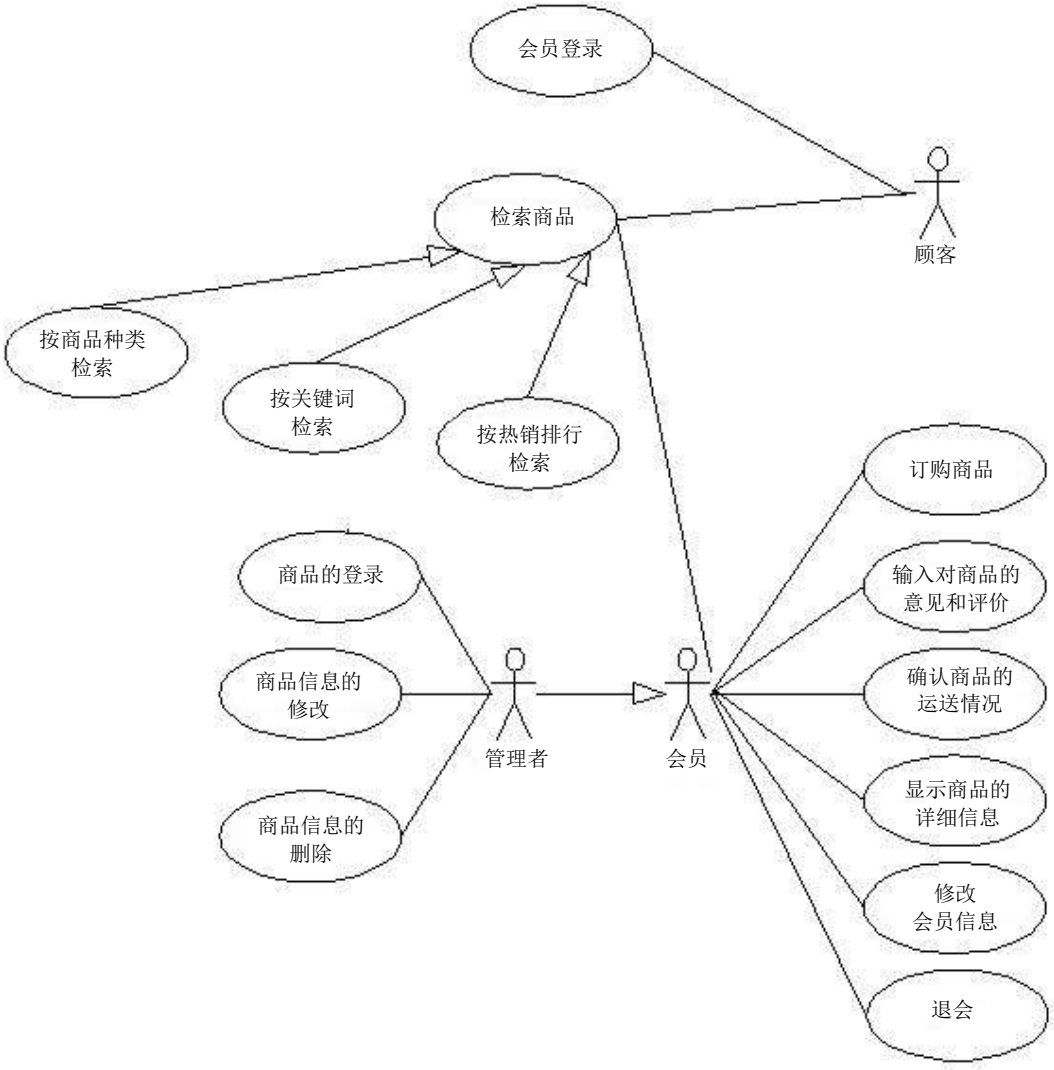
显示已经登录的商品信息。根据需要修改后，按“更新”按钮保存修改后的内容。

○ 商品信息的删除

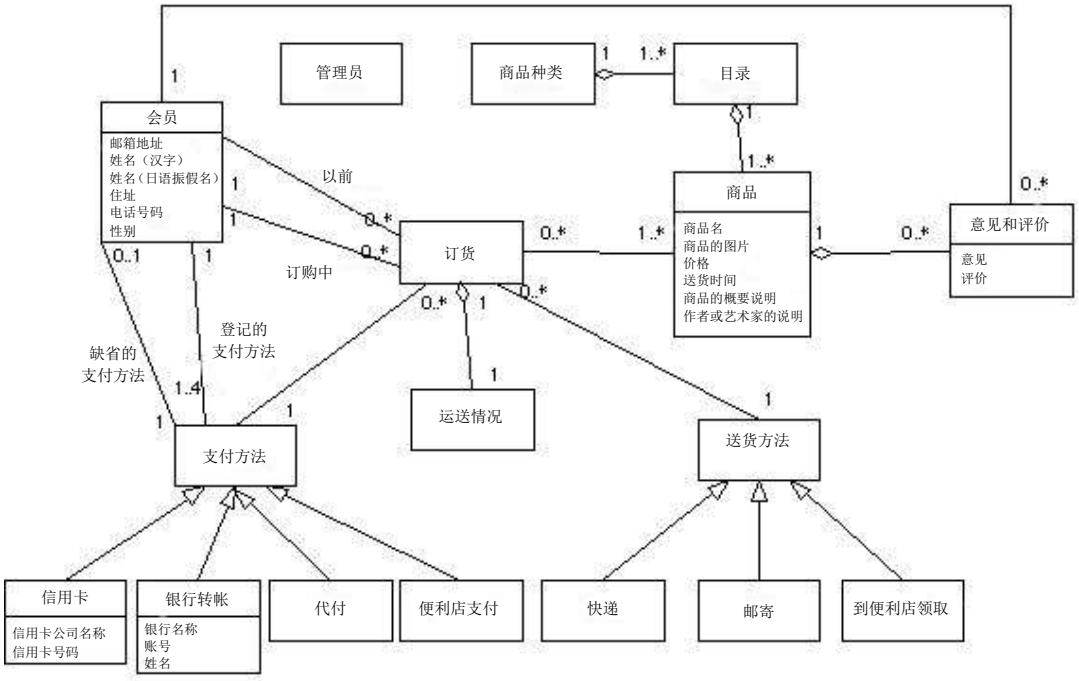
管理员可以在非公开的画面上登录。进行商品信息的删除。

和商品信息的修改一样，选择商品后。按“删除”按钮删除商品信息。：

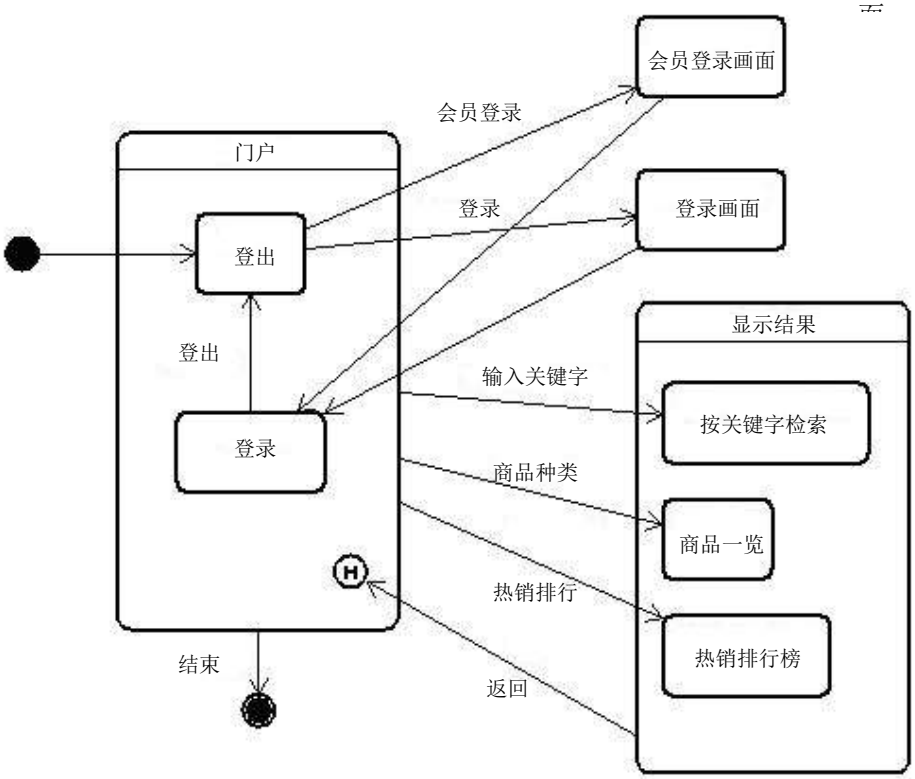
系统用例图



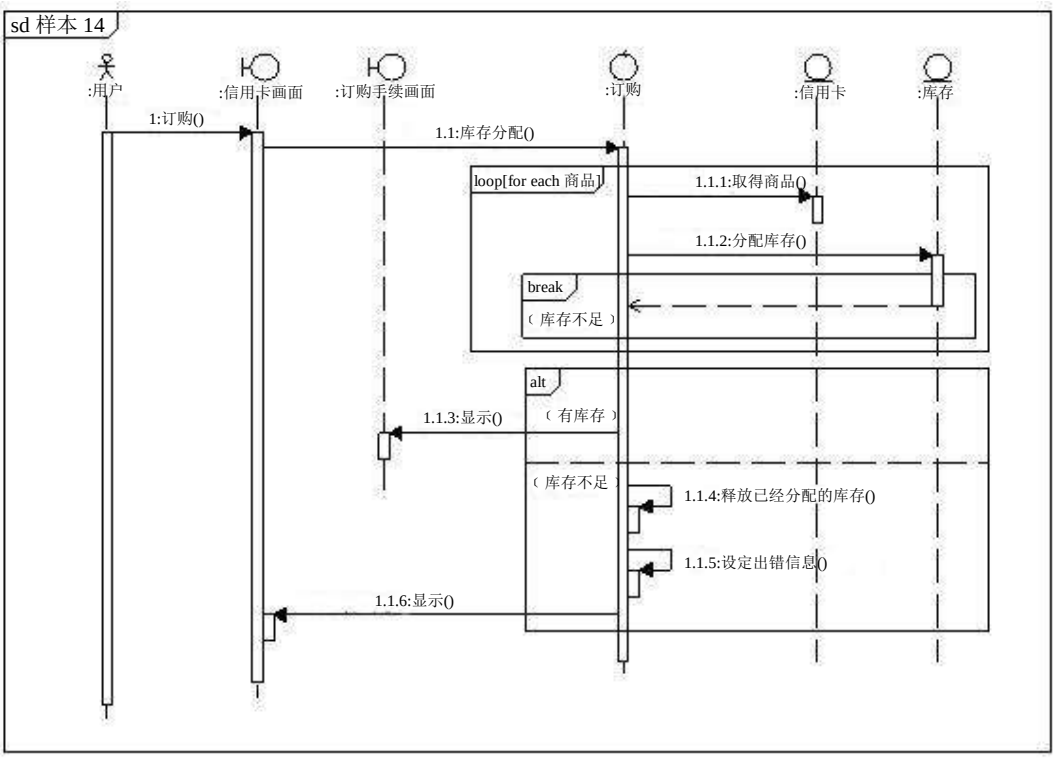
类图



状态机图（画面转移）　　*面向顾客和会员的部分画面



顺序图


sd 样本 14
:用户
:信用卡画面
:订购手续画面
:订购
:信用卡
:库存
1:订购()
1.1:库存分配()
loop[for each 商品]
1.1.1:取得商品()
1.1.2:分配库存()
break
（ 库存不足 ）
alt
（ 有库存 ）
1.1.3:显示()
（ 库存不足 ）
1.1.4:释放已经分配的库存()
1.1.5:设定出错信息()
1.1.6:显示()

附录 B. 成果物样本

(1)各阶段的成果物（部分样本）

①需求定义阶段：

A 公司正研讨开发一个职员信息检索系统。

输入职员的部门，年龄，地址等关键字，就能显示符合要求的职员信息。只要是公司的职员均可以使用这个功能。

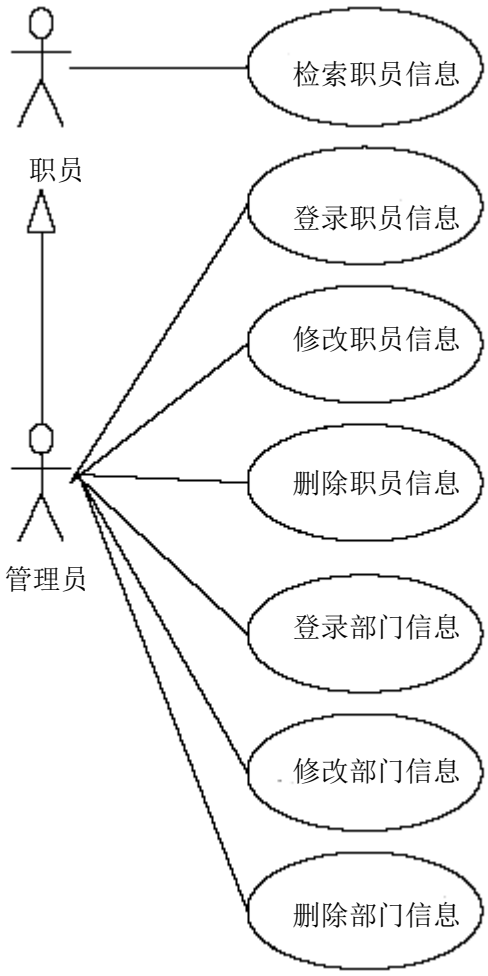
这个公司职员的职位分为技术，销售和事务三类。公司里有许多部门，职员分别属于不同的部门。职员所属的部门可能不止一个，有些职员会在好几个部门兼职。

职员的信息由管理员预先登录到系统中。可以修改和删除。

职员所属的部门也是由管理员预先登录到系统中的，可以修改和删除。

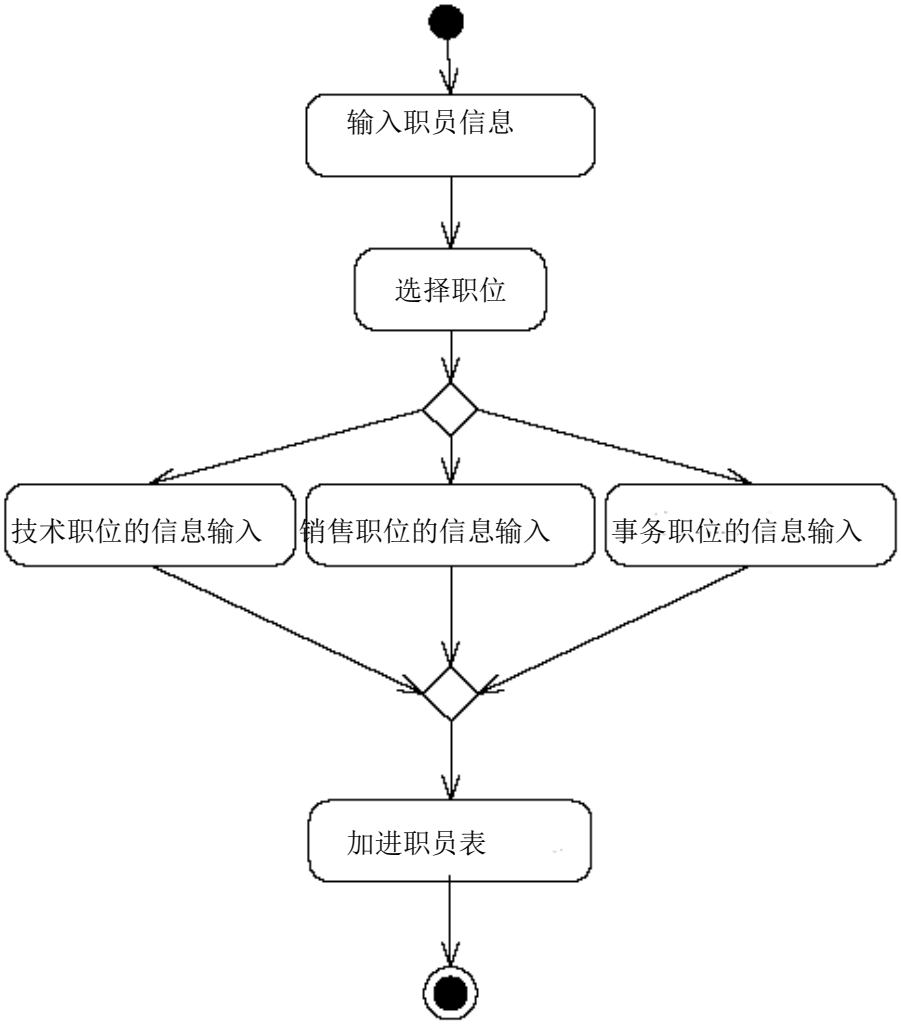
②阶段：需求分析（日方→离岸方）

●用例图

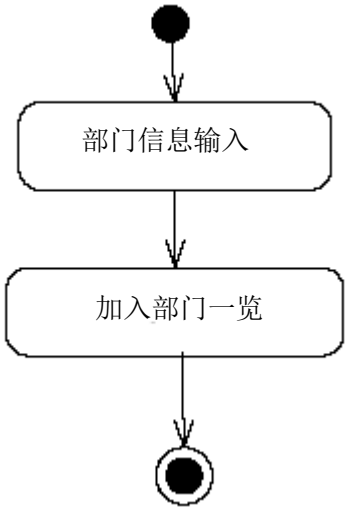


●用例的活动图(事件流)

用例「登录职员信息」的活动图(事件流)



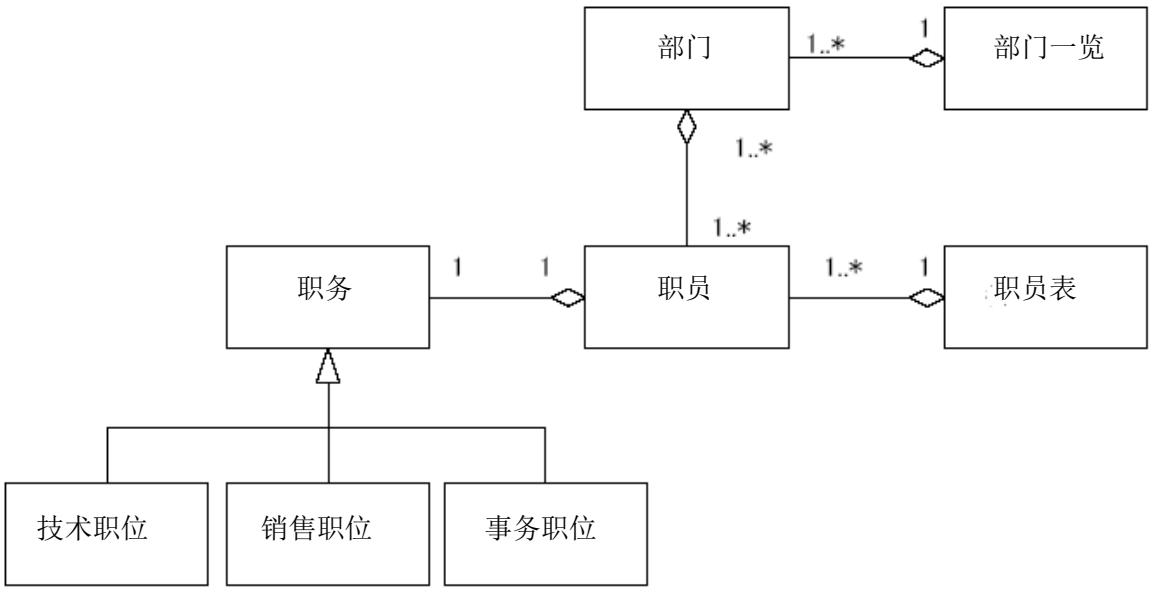
用例「登录部门的信息」的活动图(事件流)



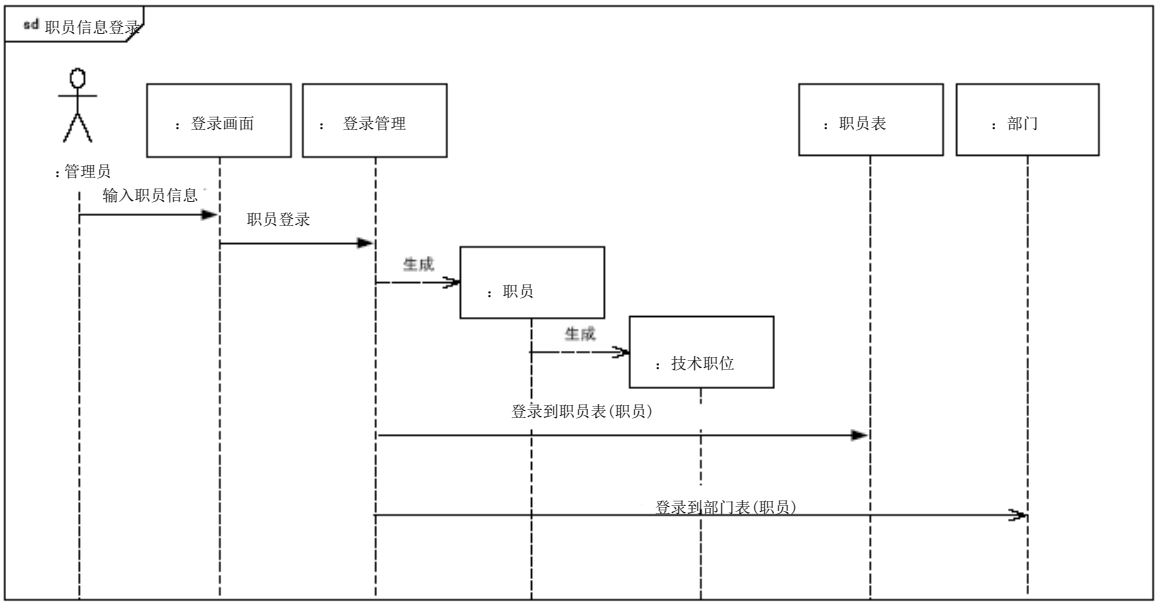
注)：对于其它所有用例，都要写出活动图(事件流)

③阶段：系统分析（日方→离岸方）

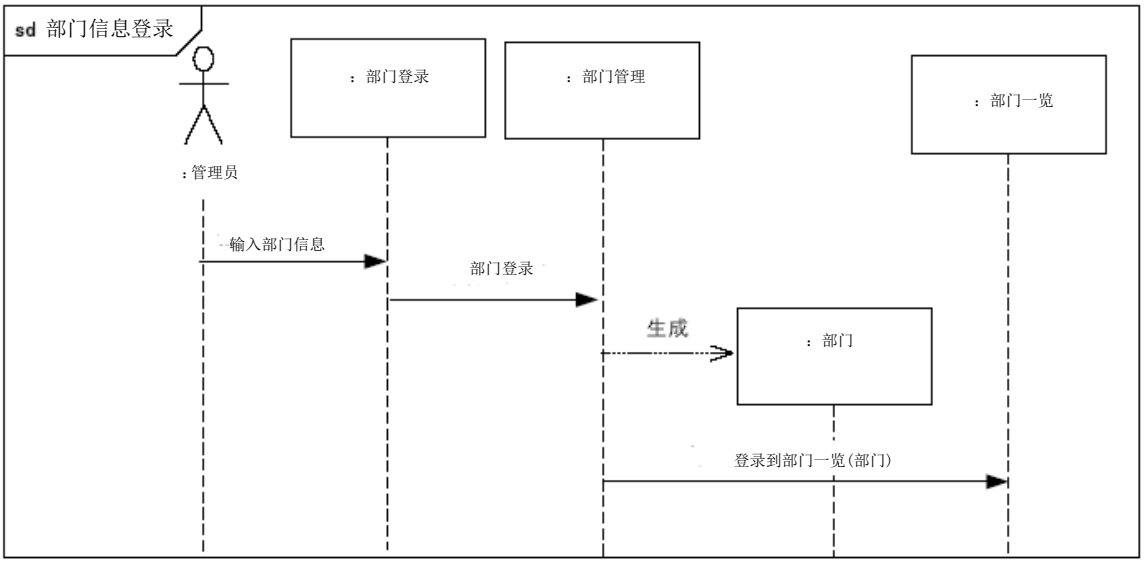
●类图



●登录职员信息的顺序图

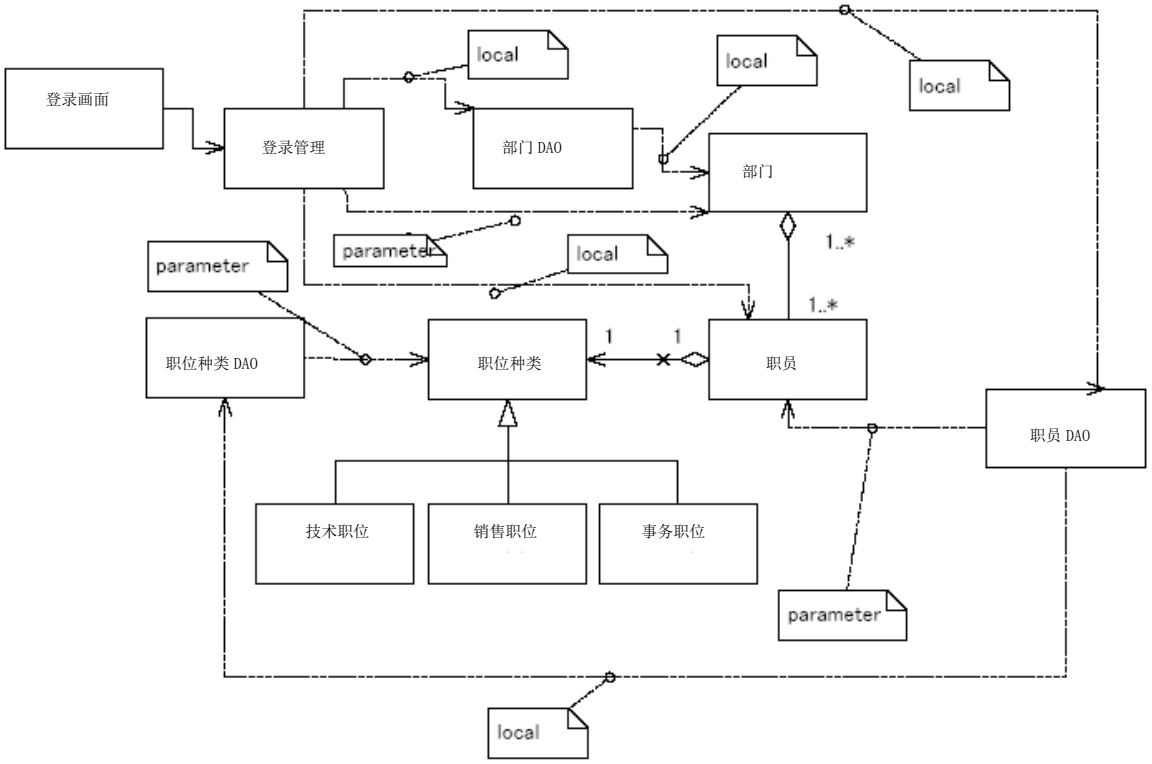


●登录部门信息的顺序图

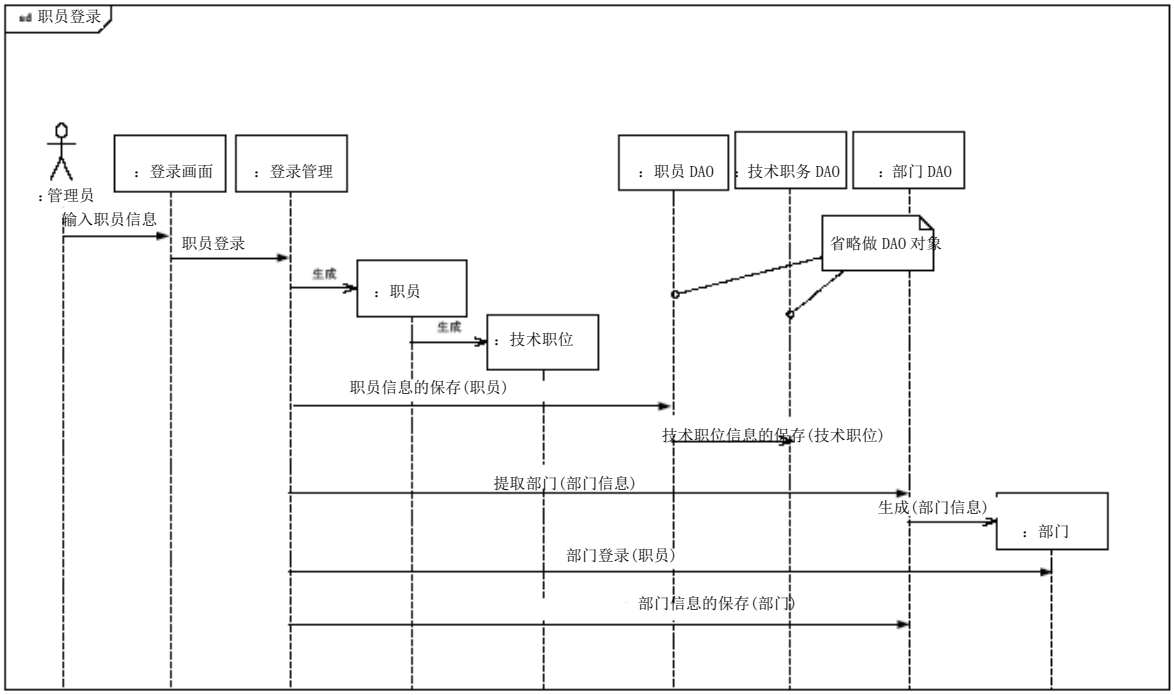


④阶段：架构设计（日方→离岸方）

●类图

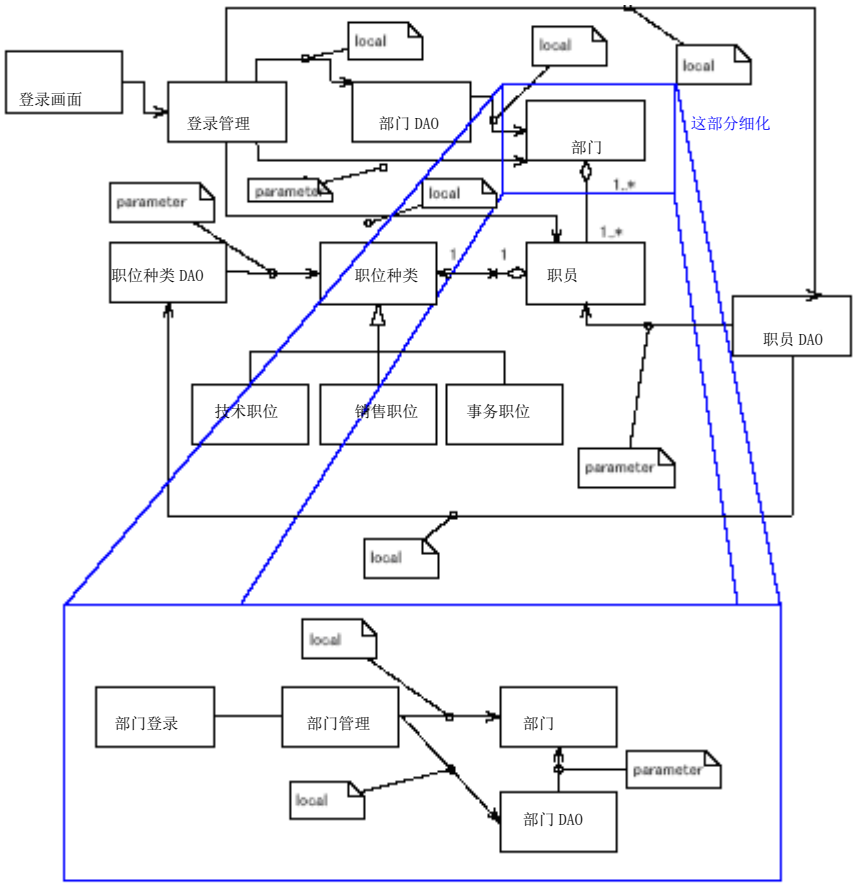


●登录职员信息的顺序图

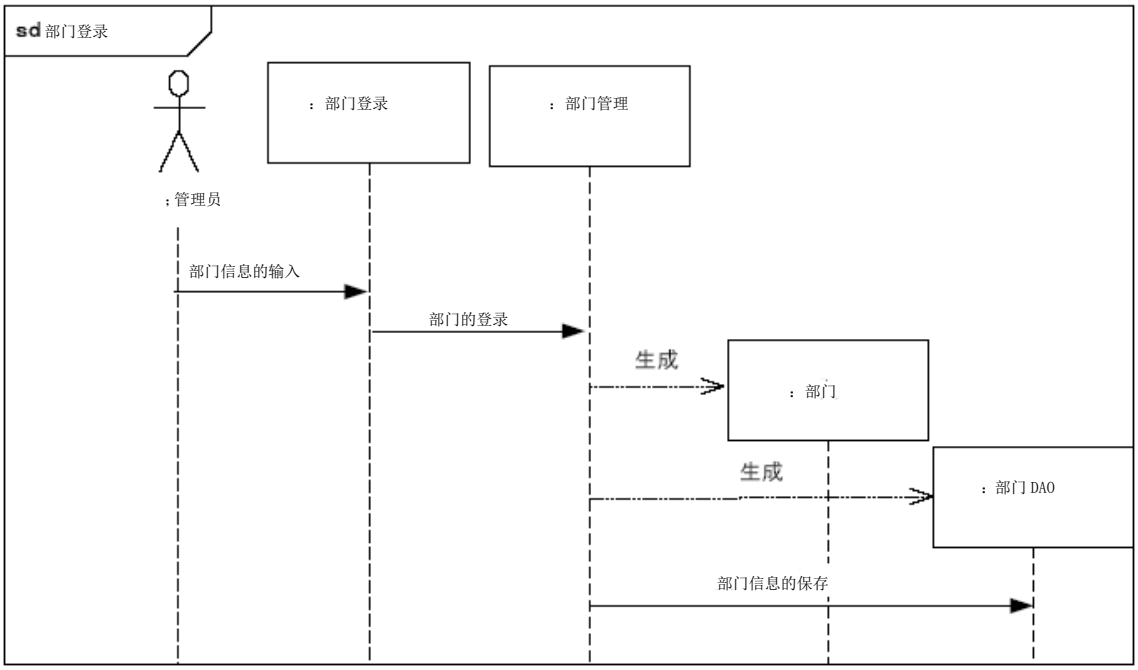

职员登录
：管理员
：登录画面
：登录管理
：职员 DAO
：技术职务 DAO
：部门 DAO
输入职员信息
职员登录
生成
：职员
省略做 DAO 对象
生成
：技术职位
职员信息的保存(职员)
技术职位信息的保存(技术职位)
提取部门(部门信息)
生成(部门信息)
：部门
部门登录(职员)
部门信息的保存(部门)

⑤阶段：详细设计（离岸方→日方）

●类图



●顺序图



(2)架构设计书

版本 0.1

目录
概论
架构的表示
架构的目标和制约
用例视图
过程视图
部署视图
编程/调试视图
规模和性能
质量
概论

○　关于永久化的问题（用例视图和逻辑视图的部分）

● Data Access Object (DAO) 模式的运用

目的：

将业务逻辑和永久化的实现分离开来，吸收永久化的方法、数据库的不同。

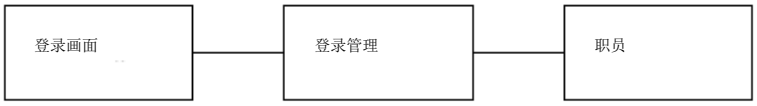
DAO 的作用：

DAO 接收数据的 CRUD。

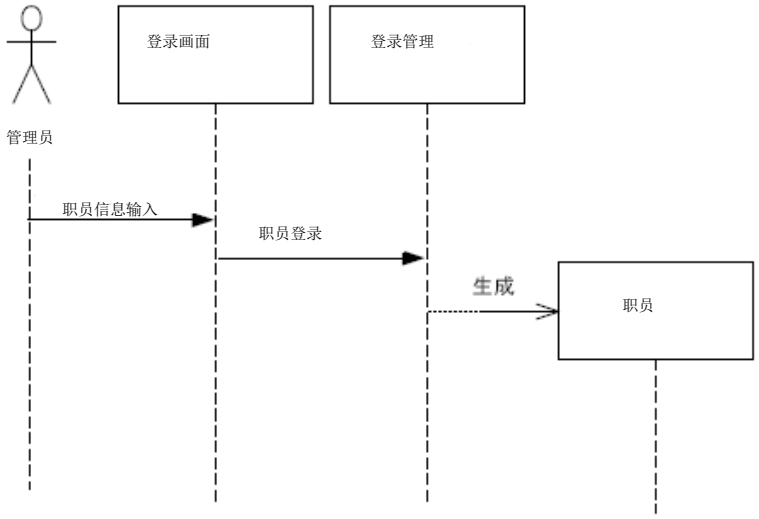
DAO 负责数据库连接等的处理。

例：简单的例子（职员登录）在系统分析模型中导入 DAO 模式

系统分析的类图

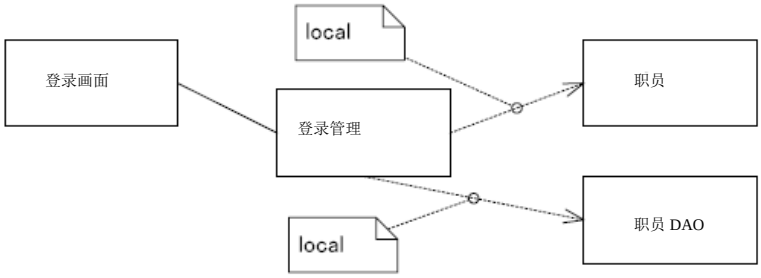


系统分析的顺序图：



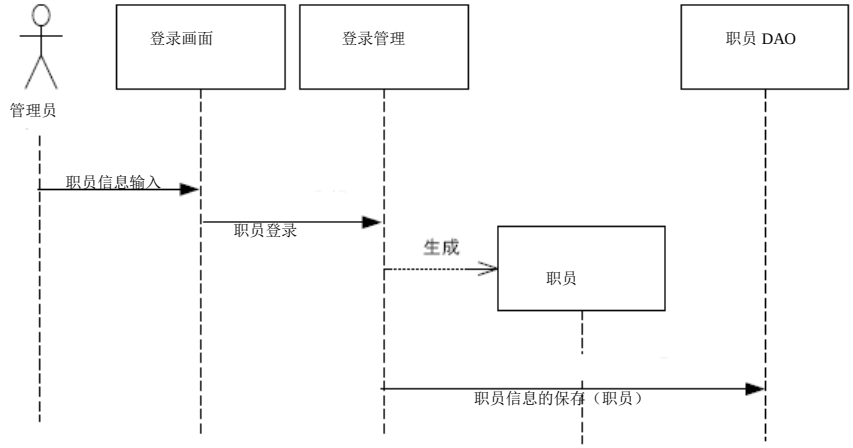
导入 DAO 模式后的设计模型：

类图



※Local 是指在上级依赖关系源的类拥有的操作中、生成下级依赖关系类对象的操作。

顺序图



「职员 DAO」类，负责向数据库存储和读取「职员类」的数据。

(3)详细设计指南（部分样本）

追加设计类

- 永久化 Data Access Object (DAO) 模式

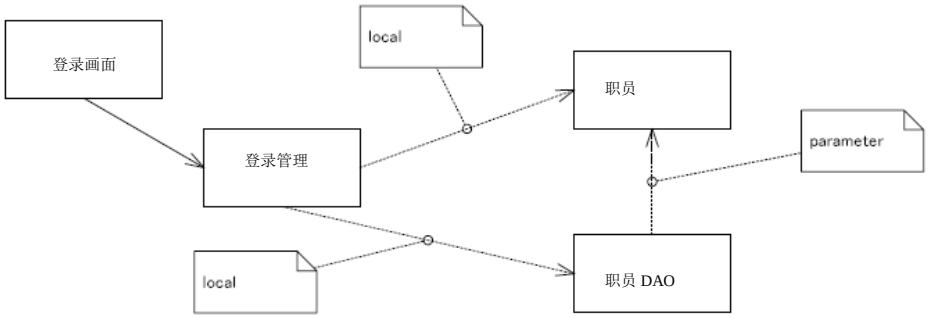
  对于需要永久化的一个 entity 类，制作一个 DAO 类。

  DAO 类的命名规定为：entity 类的类名+”DAO”。

- 类图

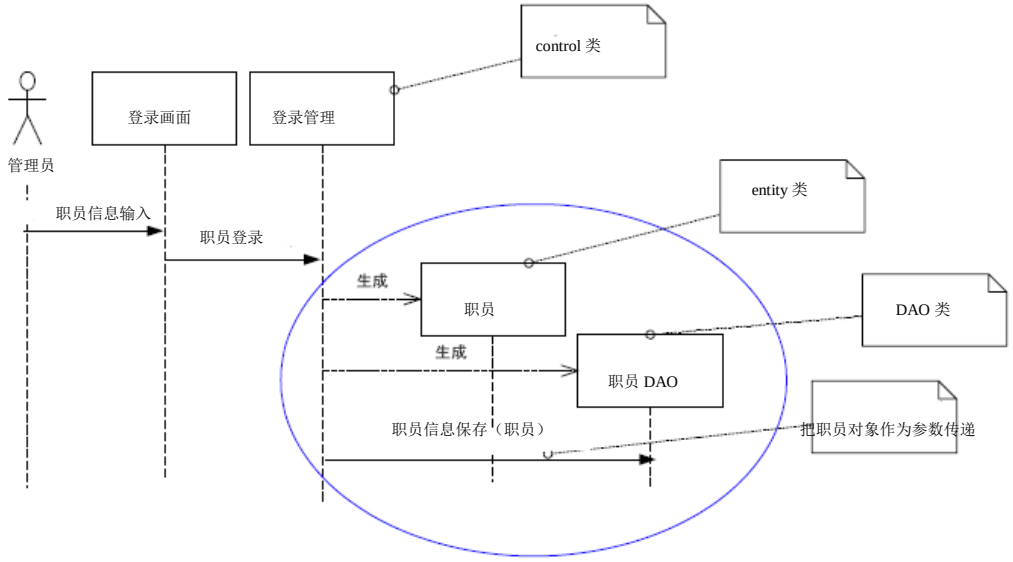
  Control 类，在 local 中生成新的 DAO 类的对象，同时也会在 local 中生成新的 entity 类的对象。所以把它们画上依赖关系，写上“local”的注释。

  DAO 类将 entity 类作为参数接收，所以把它们画上依赖关系、写上“parameter”的注释。



● 顺序图

DAO 类的对象在 local 中生成 Control 类并调用。保存时将 entity 类作为参数传递。



-
-
-

适用于离岸开发的 UML 应用指南

特定非营利活动法人 UML 建模技术推广协会
离岸软件开发部会

本指南的制作成员

Masaru Adachi	足立　勝	Nikon Systems Inc.
Chun Zheng Wan	王　春正	Oriental Standard Japan Co.， Ltd.
Miho Oshita	大下　美穂	Bab-Hitachi Software K.K
Junichi Oka	岡　純一	NIPPON INFORMATION CO.,LTD.
Hiroshi Ogura	小倉　博	Open-Works Inc.
Taro Kamioka	神岡　太郎	Graduate School of Commerce and Management, Hitotsubashi University
Akira Kamoshita	鴨下　明	Japan Systems Co., Ltd.
Yan Xu	許　炎	TONGHUA TECHNOLOGY(DALIAN) Co., Ltd.
Ryuichi Konno	今野　隆一	Japan Systems Co., Ltd.
Hajime Saito	斉藤　肇	Hyron Software Co., Ltd.
Maria Sutherland	サザーランド　真理亜	Japan International Cooperation Agency
Rui Shouda	正田　塁	Osaka Gas Information System Research Institute Co., Ltd
Kazushi Takahashi	高橋　和司	Ad-Sol Nissin Corporation.
Akitoshi Takemasa	竹政　昭利	Osaka Gas Information System Research Institute Co., Ltd
Shigeru Tanaka	田中　繁	Open-Works Inc.
Hiroshi Nakajima	中島　拓	IT Engineering Ltd.
Toshimasa Nakahara	中原　俊政	Bab-Hitachi Software K.K
Yohei Haba	幅　洋平	Bab-Hitachi Software K.K
Tsunahiro Hidaka	日高　綱弘	Nippon Information Co., Ltd.
Masahiko Hiroi	広井　政彦	Nikon Systems Inc.
Hiroyuki Fujino	藤野　博之	NEC Nexsolutions, Ltd.
Narutoshi Hosaka	保坂　成利	Chiyoda Corporation
Hiroshi Makino	牧野　宏	Open Resource Corporation
Toshitaka Mori	森　稔貴	Nippon Information Co.,Ltd.
Shoji Yamaguchi	山口　昭二	Sakura Information Systems Co., Ltd.
Ryo yoshida	吉田　亮	IBM Japan
Ryouji Wakabayashi	若林　良二	NEC Nexsolutions, Ltd.