

**软件是组织的零件**

业务建模的目的是从组织的角度来定位系统应该提供的价值，所以说“业务建模”这个名字其实起得不好，应该更名为“**组织建模**”。出于对过去叫法的尊重，本书依然称为“业务建模”。做一做以下题目，可能会发现答案出乎你的意料。

很多时候我们把自己在开发的系统（现在流行叫××平台了）看得太重了，感觉没有我们开发的系统，组织就玩不转了。其实，我们那牛×哄哄的系统也只是组织的一个零件，和组织厕所里的马桶没有本质区别。组织里面还有很多系统，其中最值钱的是千百年来一直在使用，现在依然是最复杂的系统——人肉系统，它由“父母公司”开发，“老师公司”不断升级，组织以每月每人几千上万的租金租用。**要改进组织的价值，不一定要开发软件，有时换人（也就是说，不是更换软件系统，而是更换人肉系统）**更管用。和一套软件系统的价格相比，也许雇用一名高级员工花费更多。

开发人员有时会有意无意把自己的系统想得太重要。有一次我到北京某公司讲课，开发人员在写某信贷风险系统的愿景时写道：本系统的目标是，银行风险部能够对贷款做风险评估。我问道：难道银行以前不能做风险评估吗？他认真地回答：不能啊，有我们的系统才行！其实想想就知道，银行已经成立多少年，贵公司才成立几年？所以，为了抵消开发人员这种“致命的自负”，有时我会故意将“系统”称为“马桶”——你做这个马桶是干什么的？

为什么需求经常“容易变化”？根源之一是它们的来路不正，一开始的时候是拍脑袋得来的，没有把系统当作一个零件放在组织中来看，得到的系统当然和组织的其他零件[格格不入](https://www.baidu.com/s?wd=%E6%A0%BC%E6%A0%BC%E4%B8%8D%E5%85%A5&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd" \t "https://www.cnblogs.com/lcword/p/_blank)，系统上马磨合后发现问题，自然要改。“需求变化剧烈”只是一个假象，许多需求的变化是假的变化，真正的需求并没有变，只不过开发人员一开始捕获的需求是假的。如果能正确运用业务建模技能，大多数假的需求变化会消于无形。遗憾的是，不少开发团队在改进的时候给自己开错了药方，以为应该通过提升设计的弹性来应变。设计是应对真正需求变化的手段，假的需求变化应该通过改进业务建模来应对。

如果东西拿到客户面前，客户说“好呀，正是我想要的！”，过了半年，客户又说“形势变了，这个东西要改一下。”这是需求变了。如果东西拿到客户面前，客户说“这不是我想要的！”，这时硬要说“需求变了”，脸皮有点厚了。

说起业务建模，很多人会联想到大企业、ERP、工作流….其实，所有类型的系统都可以使用业务建模技术来得到有价值的需求。后文会看到各种领域的具体例子。

**3.2【业务建模步骤1-1】：选定要改进的组织**

业务建模既然是研究组织，那么研究哪个组织呢？理论上来说，先研究整个世界，再慢慢缩小范围，一直到能定位我们要开发的的系统，但这显然太浪费了。最佳的研究范围就是愿景波及的、需要改进的组织，它可能是一个公司、一个政府单位、一个部门、一间办公室、一个家庭、一群人……

现在的许多系统，改进的组织是一个人群（如车友）而不是正式组织（如某公司）。人群也是组织，组织也就是人群，业务建模的方法是一样的。某公司是一帮人，车友也是一帮人，它们之间的区别仅在于有没有组织机构代码证而已。某公司里有总经理、部门经理、普通职员……车友里有骨灰车友、发烧车友、入门车友……

如何圈定这个最合适的研究范围，没有标准答案，可能要试很多次。如果发现组织的绝大多数流程和改进不相干，说明选的范围可能太大了；如果发现很多要改进的地方未涉及，说明选的范围可能太小了。

以下是可以参考的几点思路：

画一个圈，把大多数责任可能被（部分/全部）替换的系统（人肉系统/电脑系统…）包在里面。

例如：要开发一个企业内部员工训练平台，替换的责任大多属于人力资源部人肉系统的工作，这时可以把人力资源部作为研究对象：

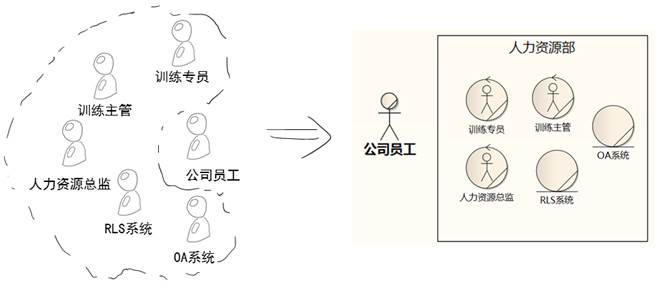


图3-1 把大多数系统圈在里面

请注意上面的用词，“大多数责任可能被替换的系统”，而不是“大多数系统用户”。因为此时待开发系统的边界尚未确定，不能[先入为主](https://www.baidu.com/s?wd=%E5%85%88%E5%85%A5%E4%B8%BA%E4%B8%BB&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd" \t "https://www.cnblogs.com/lcword/p/_blank)地把某些人肉系统称为用户（而且前面我们说了，不是“用户”，而是“执行者”）。系统要做的改进和训练专员的工作有关，不代表训练专员一定是该系统的执行者，也许人力资源总监会觉得取消所有的训练专员可能是更好的方案。

得到的结果其实就是老大所代表的那个组织，所以这个范围大小和老大的职权范围是相关的，如果老大只是人力资源总监，把整个公司作为研究对象就太大了；如果老大只是大学里面某个学院的院长，把整个大学作为研究对象就太大了。

互联网网站项目如何选择业务组织

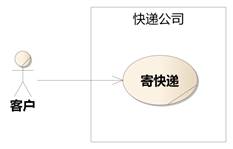
如果苍井老师购买了一套软件安装在她的电脑里，她会觉得这套软件是她的，但是，如果苍井老师觉得新浪微博很好用，她的认识就有些微妙的变化，会觉得新浪微博是新浪公司的。开发人员在为探索互联网网站的需求而做业务建模时，也常会发生类似错误——要开发下一代互联网网站挑战新浪微博，所以把新浪公司作为研究对象？更正确的做法是把明星人群作为研究对象，思考如何做出更牛的网站来向大众辐射苍井老师们的魅力。

面向大众的互联网网站只不过是软件的另一种表现形式——不再是购买光盘或下载安装包到本地安装，而是免费或收费“租用”系统（前面也说了，“免费”并非真的免费，我们付出了最宝贵的时间）。苍井老师碰到 “新浪微博”，她只关心这个系统的功能和性能就够了，她会关心这个系统是新浪公司还是搜狐公司开发的？路上捡到的？还是外星人开发的？业务建模时应该研究的组织是该网站的目标人群，并非负责开发和维护这个网站的开发组织。

如果要改进的是运营网站的组织的内部运营工作，把网站组织作为研究对象就是合理的。总之，病在谁身上，就给谁做检查。

实际工作中，经常会出现这样的现象：该研究目标人群时，建模人员却害怕去研究，[千方百计](https://www.baidu.com/s?wd=%E5%8D%83%E6%96%B9%E7%99%BE%E8%AE%A1&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd" \t "https://www.cnblogs.com/lcword/p/_blank)要退回来研究网站组织。例如，要思考微博需要添加什么新功能，应该去观察艺人的生活和工作，但这个太麻烦了，还是观察自己公司内部的运营吧！这就相当于在黑暗的地方丢了钥匙，却到明亮的地方找，因为这里亮堂，好找。

对松散的"目标人群"做业务建模，是创业很好的入手点。例如，宅男找餐馆叫外卖这样一件事情，一端是“宅男”组织，一端是“餐馆”组织。如何改进一家餐馆的流程，可能已经有很多人做过严肃的建模；如何让宅男的生活更美好，严肃建模的人就少得多，许多改进点只靠拍脑袋发现。选定组织后，我们需要从两方面来研究它。从外部看，组织是一些价值的集合，我们可以用业务用例图表示；从内部看，组织是一些系统的集合，我们可以用业务序列图来表示。



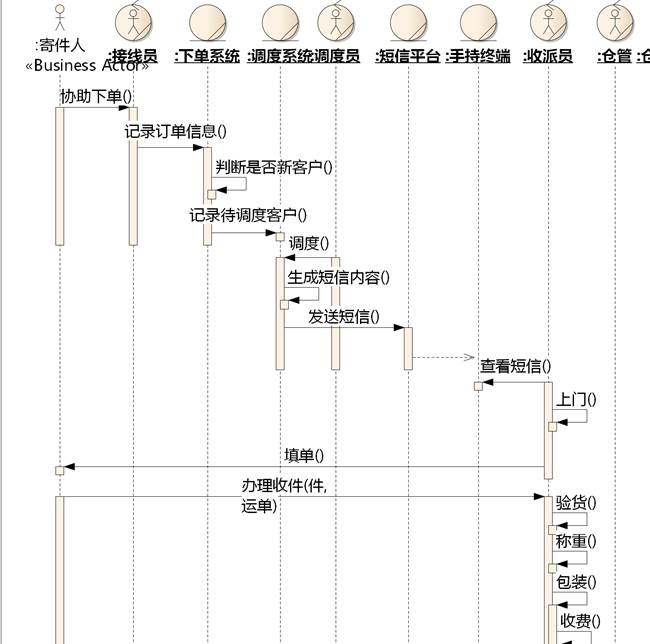


图3-2 组织的外观和内观

**3.3 【业务建模步骤1-2】：组织的业务用例图**

**3.3.1 业务执行者**

**首先来寻找组织的执行者，也就是业务执行者（Business Actor）。业务执行者的定义是：在组织之外和组织交互的人群或组织。**

以一家商业银行为研究对象，谁在外面和它打交道？储户来存钱，企业来贷款，人民银行要对它作监管…。这些就是该商业银行的执行者。

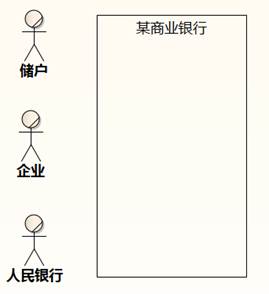


图3-3 业务执行者

业务执行者的图标是一个小人，头上有一道斜杠，这个带斜杠的小人实际上是一个执行者的构造型<<Business Actor>>的图形表示，Rational工具和EA里都有。如果您使用的UML工具没有这个图形，可以用执行者的小人图标加上文本构造型<<Business Actor>>取代，或者不加构造型也无所谓，因为系统边界框已经表明了研究对象是一个组织，它的执行者自然是业务执行者。

**3.3.2 业务工人和业务实体**

在寻找业务执行者时，有时会和组织内的人混淆，例如银行里面的营业员。营业员在组织里面，不是业务执行者，我们称其为业务工人（Business Worker）——组织内的人肉系统。**业务执行者和业务工人的区别是，一个在组织外面，一个在组织里面，一个是组织不可替换的，一个是组织可以替换的零件。**

业务工人可能会被新的业务工人替换，但更多的可能是被新的业务实体（Business Entity）替换。业务实体就是组织中的非人系统，例如银行的取款机、点钞机、营业系统。

在没有点钞机的时代，储户拿着一叠钞票到银行存钱，营业员需要凭着手感（界面层）一张张数，数的时候大脑（核心领域层）还要很快判断钞票的真伪。点钞、数数的逻辑封装在营业员的大脑里，营业员非常累。有了点钞机，营业员只需要把整叠钞票放进点钞机过一下，点钞机会负责辨认假钞和计数。“验钞”、“数数”的责任和逻辑从人脑转移到了点钞机的大脑。营业员轻松了，当然，银行也就不需要那么多营业员了。

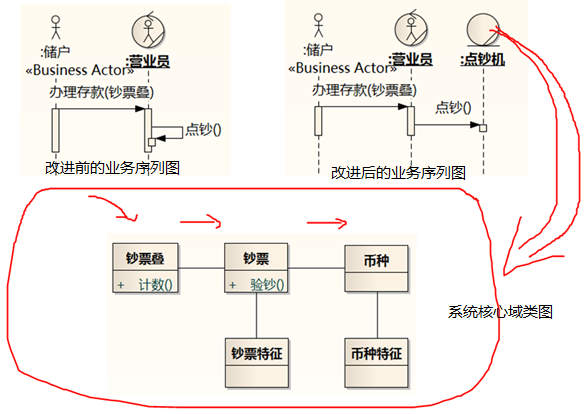


图3-4 逻辑从营业员的大脑转到点钞机的大脑

责任转移的思想对识别待开发系统的需求很有帮助。开发人员说，**“我在开发一个新系统”，其实说的就是“我在开发一个新的业务实体，取代现有业务工人或业务实体的一些责任”**。这样，探索需求的思路就出来了。**我们画好现状的业务序列图，然后把待开发的系统作为一个新的业务实体空降到序列图中，寻找改进点，得到该业务实体的责任，就可以直接映射到待开发系统的用例。**

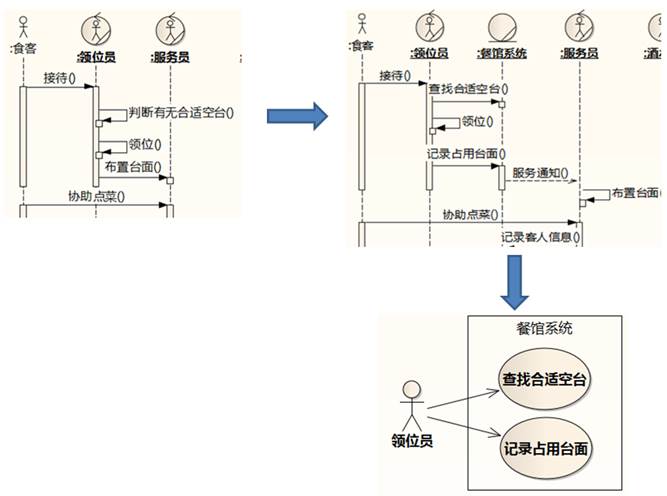


图3-5 改进业务序列图，映射系统用例

业务工人和业务实体不在业务用例图中出现，因为它们不是组织的价值，而是成本。业务工人和业务实体放在名为“业务对象”的包里，作为类（Class）的一个构造型。**在识别业务执行者时，不需要画业务工人和业务实体，在接下来画业务用例的实现——业务序列图的时候加上就可以。**

和业务执行者一样，如果您使用的工具没有<<Business Worker>>和<<Business Entity>>构造型，可以自己造，或者干脆不要构造型直接用类表示。背后的思想是一样的：类之间通过协作实现用例。业务工人和业务实体协作完成业务用例，系统类协作完成系统用例。

**业务执行者是一个组织（或人群），而不是系统**。因为研究对象是组织，和它对应的概念也应该是组织。下面的图是错的：

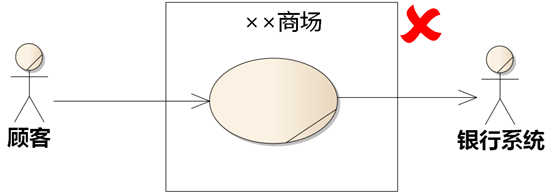


图3-6 错误：把系统作为业务执行者

应该把银行系统看作业务实体，在业务建模的类图、序列图里出现，正确的业务用例图如下：

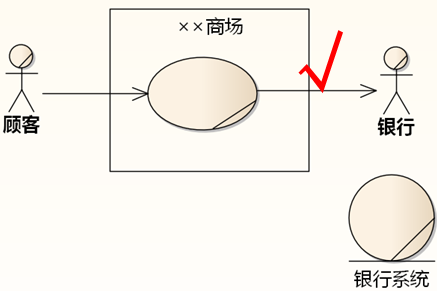


图3-7 正确：组织为组织提供服务

**3.3.3 识别业务执行者**

识别业务执行者时，可以这样思考：拿着摄像机对准组织的边界，谁上门找它办事？谁打电话给它？谁发邮件给它？它出门找谁办事？打电话给谁？发邮件给谁…可能就会找到它的供应商、客户……如果研究的组织是一个部门，那么其他部门就有可能成为它的执行者。

**3.3.4 业务用例**

**业务用例指业务执行者希望通过和组织交互达到的，而且组织能提供的价值**。以上面提到的商业银行为例，我们可以这样思考，储户到银行的目的是什么？可能是存款、取款、转账，得到银行针对储户的用例如下：

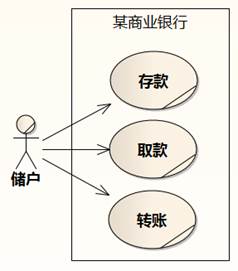


图3-8 某商业银行针对储户的用例

如果穿越回到三百年前，图3-8所示用例图依然适用。业务用例代表组织的本质价值，很难变化，改变的是业务用例的实现。三百年前，银行要实现“储户→存款”的用例，需要许多人肉系统（业务工人）一起协作，现在则变成了**少数人肉系统（业务工人）和许多电脑系统（业务实体）**之间的协作。

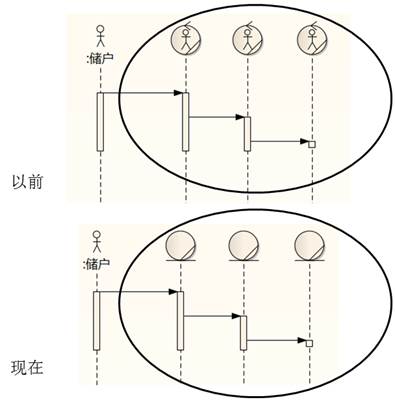


图3-9 用例没变，实现用例的零件变了

我们做一些题来理解业务用例的概念：

特别要说一下第2题。选项A，患者确实是医院的执行者，但挂号不算医院针对患者的用例，因为挂号不是患者对医院的期望和医院对患者的承诺的平衡点。患者到医院来，挂到了号，医院的服务到此为止，患者到医院来时心中的期望并没有得到满足（如果说满足？那么执行者就不叫患者，叫号贩子）。或者说，医院这个研究对象能承诺向患者提供的价值不只是挂号，而是看病。选项C才是正确的。  
可以这样思考：医院能这样叫卖自己吗？“来呀来呀，我这家医院能挂号呀！”，患者一听，“哇，真棒耶，这医院能挂号耶，我赶紧去！”其实患者巴不得不挂号也能看病，只不过人太多了，需要拿号排队而已。

如果研究对象是挂号室，A就是对的。患者对挂号室的期望就是能挂号，他不会因为挂号室没帮他看病就破口大骂，挂号室对他的承诺也就是能给他号。

如果研究对象是医院信息系统，又不一样了，患者无所谓挂号室人员是怎样帮他挂号的，医院信息系统的价值是让挂号室人员通过它来（为患者）办理挂号。图3-10列出了以上提到的三种正确的情况。

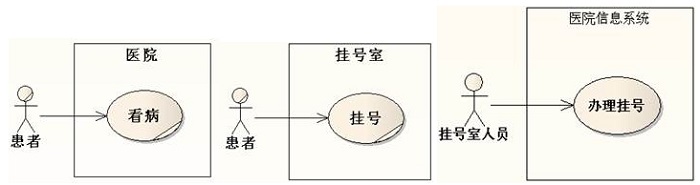


图3-10 期望和承诺的平衡

答案B和D比较容易看出来是错的，因为医生和收费人员是医院内的业务工人。但经常会有人认为，既然B和D不正确，表达成E和F倒是可以的，但是E和F也是错的。业务用例是为业务执行者服务，不是为业务工人服务。或者说，业务用例表达组织的本质价值。

可能有人又会想：难道医院不要收费吗？收费不是医院的一个本质吗？这里反映了开发人员常见的一个问题：分不清问题和问题的解决方案。医院确实要收费，但是选项E说的不是“收费”，而是“收费人员收费”，也就是说人肉系统坐在那里收钱，这是很容易替换的解决方案，背后的问题是医院的老板要通过医院来赚钱，至于钱怎么收上来的，是收费人员这个业务工人坐在那里收钞票，还是银行系统内部转账，无所谓。

同理，“医生诊治”也只是解决方案。病人要的是把病治好，治疗的过程短，不痛苦，没有后遗症，收费还便宜，并不在意他的病是医生动手术治好的，还是有个很牛的仪器给照好的。医院老板巴不得不用请那么多医生，照样为病人看病赚钱，只不过做不到而已。

像收费人员这样的人肉零件，以现在的IT技术替换掉没有问题，不过像医生这样读了很多年书，经过许多年专业训练的人肉零件，也许50年以内都替换不掉。在训练班上做这个题目时，一个有趣的现象是经常有学员不选择E，但选择F，大概是意识上认为医院里总应该有医生吧。所以职场中人，要尽量做组织中不容易替换的零件。

即使现在医生的地位还比较稳固，他的责任也已经被替换了一部分。过去去看病，说：“医生我咳嗽。”，医生会让我们伸出舌头看一看，听诊器放胸口听一听，躺在床上按一按。现在呢，医生抬头看一眼，啪啪就开单，“去照个××吧”，把检查的责任转移给仪器了。

最后需要说明的是，题目的第一句是“以医院为研究对象”。**在讨论“有哪些用例”的时候，必须说清楚研究对象，否则讨论是没有意义的**（同理，不说清楚执行者是谁，讨论用例也是没有意义的）。如果工具箱里有系统边界框，加上一个边界框来标明当前的研究对象。有的建模工具（像老版本的Rose）没有提供这个框，也要用一个Note注明研究对象：

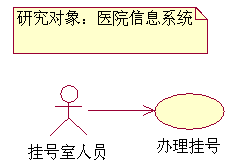


图3-11 没有边界框，用例图也要用Note注明研究对象

业务用例刷新了业务流程的概念。我们把业务流程看作是业务用例的实现，将其组织在业务用例的下面。组织内部之所以有业务流程，是因为要实现业务用例。组织里发生的一切都是为了给业务执行者提供价值。

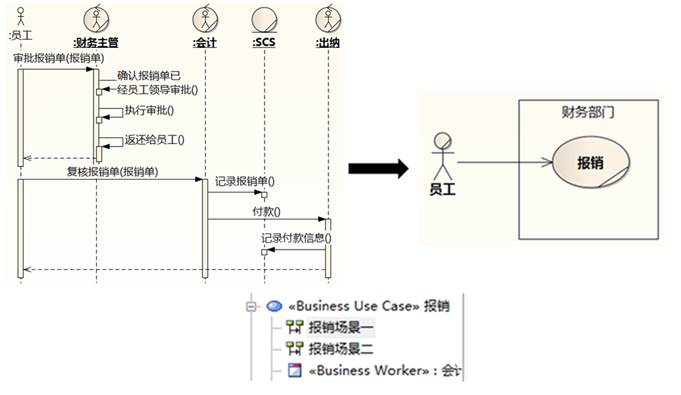


图3-12 业务流程是业务用例的实现

这种思路对改进业务流程有非常大的帮助：先归纳出组织对外提供什么价值，再思考如何更好地优化组织内部流程来实现这些价值。

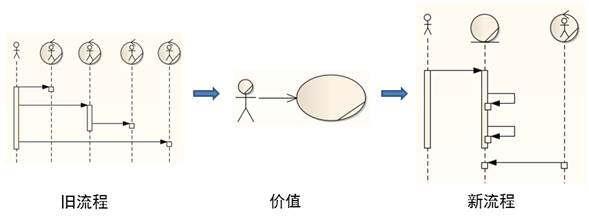


图3-13 从价值出发重新构造业务流程

识别业务用例有两条路线：一条是从业务执行者开始，思考业务执行者和组织打交道的目的；另一条是通过观察组织的内部活动，一直问为什么，向外推到组织外部的某个业务执行者。

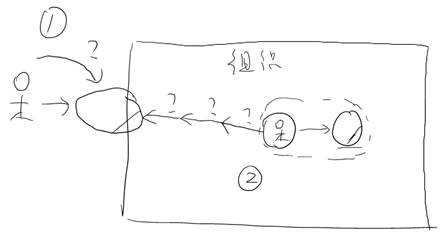


图3-14 识别业务用例的两条思路

第一条路线是主要的，思考的焦点是“执行者对组织的期望和组织对执行者的承诺”的平衡点，注意不要出现“患者到医院挂号”的用例。第二条路线用于补漏，找到之前可能会遗漏的执行者和用例。

采用第二条路线思考时，会出现的一个错误是直接将内部活动作为业务用例。只要了解基本道理，稍加注意就可以避免这个错误。这条路线还引出一个值得商讨的话题：管理型业务用例。

例如，公交公司里有调度员负责制订运营计划，计划哪条线路配备多少台车，配备多少司机，每天多少个运行班次，每班的发车时间。如果以公交公司为研究对象，调度员是业务工人，所以“调度员→制订运营计划”不是业务用例，那么这个工作归属于哪个业务用例呢？通过思考调度员为什么要“制订运营计划”，一直向外推，可能会得到类似“公司董事会：提高公交线路效率”这样的用例，即管理型业务用例。

以前我也支持这样的做法，但是发现这样的“管理型业务用例”很容易被滥用，而且容易和愿景混淆。现在我更倾向于将这样的流程归纳到类似于公交公司的“市民→乘车”这样的用例下面，而调度流程是乘车的支撑流程。业务用例是组织为组织服务，在不同的场景中，两个组织的各种系统形成不同的交互。图3-15中的场景④可以看作是支撑流程。

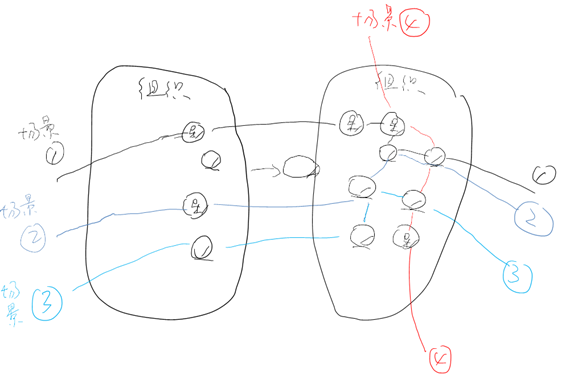


图3-15 实现业务用例的不同场景

“管理型业务用例”还隐含另一个问题：也许现在挑选的研究范围并不合适。如果调度流程是需要改进的核心流程，说明选择公交公司为研究范围不合适，可以考虑缩小为公交公司调度室。当然，如果不存在这样一个专门的调度室，仍然保持原来的研究范围也没关系。

**3.4 案例**

UMLChina业务系统改进的组织是UMLChina组织。我们把摄像机对准UMLChina的边界，可以得到图3-16所示的业务执行者：

猜谜06

长假七天陪着亲友连轴玩。

打UML术语一。谜底在本书内找。

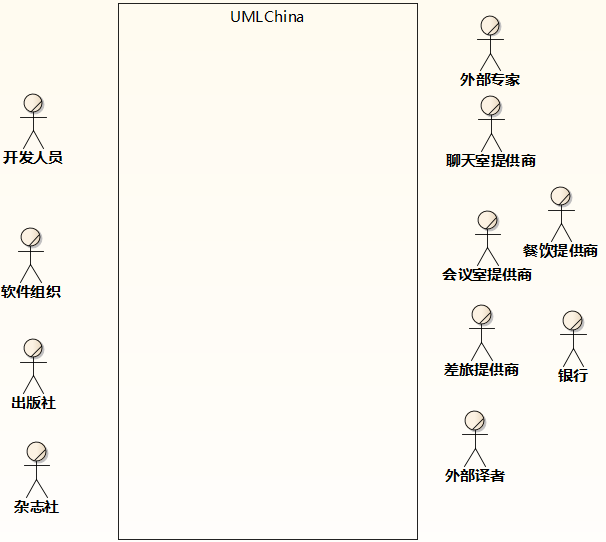


图3-16 UMLChina的执行者

再思考执行者和UMLChina打交道的目的，可以得到图3-17所示的用例：

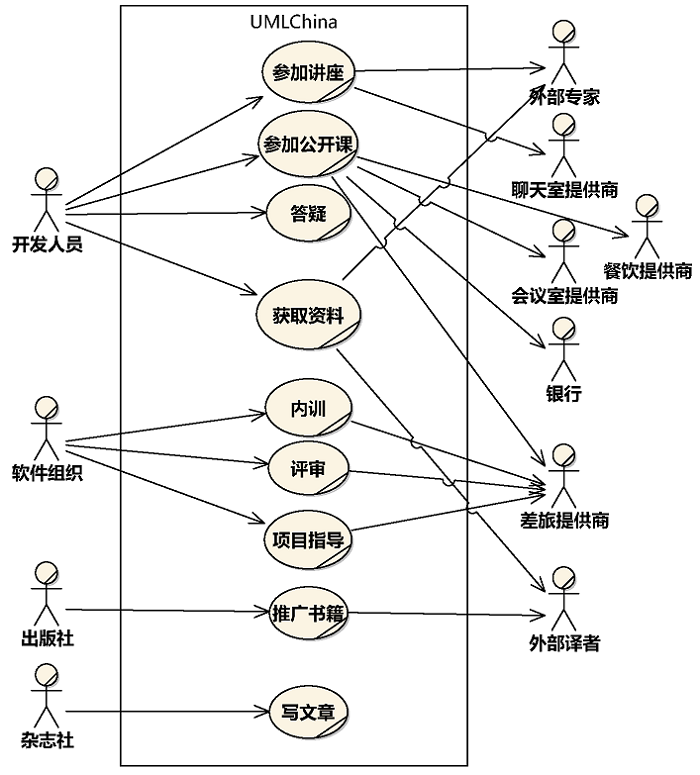


图3-17 UMLChina的用例

图3-17中，有箭头从执行者指向用例，也有箭头从用例指向执行者。前一种执行者称为用例的主执行者，后一种执行者称为用例的辅助执行者。

“出版社→推广书籍→外部译者”可以这样解读：为了给出版社提供推广书籍的服务，UMLChina靠自己的力量不足以完成，需要找外部译者帮忙。或者这样解读：出版社向UMLChina“购买”服务，UMLChina向外部译者“购买”服务。

UMLChina还有没有别的用例呢？当然有的，例如，所有公司都逃不掉一个用例：政府要向它征税。不过，从愿景判断，我们的改进和UMLChina如何纳税的流程无关，所以不需要把不打算改进其实现的用例画出来。注意：虽然不画出来，但它们是存在的。

**3.5 工具操作**

【步骤1】展开1-业务建模包下的业务用例包，双击业务用例用例图，

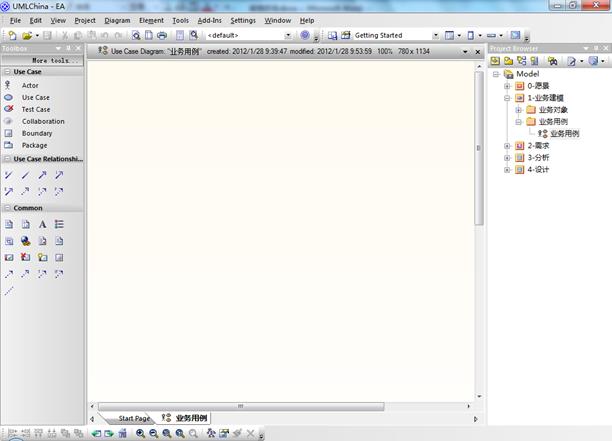
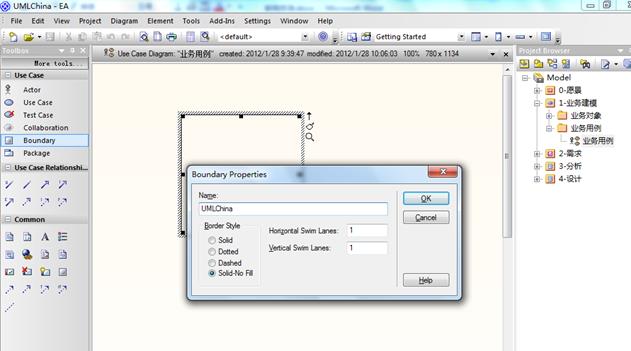


图3-18 空白的用例图

【步骤2】点击工具箱中的，再点击图的顶部中间，在弹出的属性框中输入UMLChina，点击OK。拖动边界框的边调整到合适的大小。



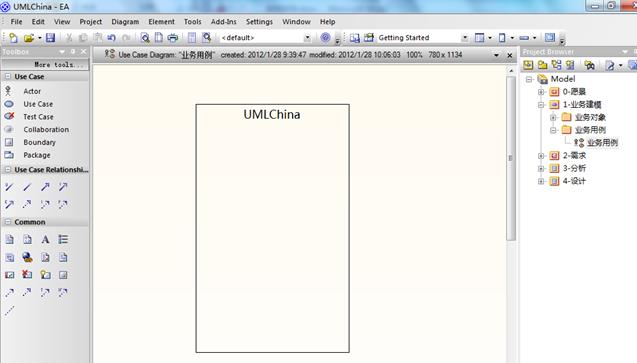


图3-19 放置边界框，确定研究对象

【步骤3】点击工具箱中的，点击边界框的左侧，在弹出属性框的Name栏输入开发人员，Stereotype选择business actor，点击确定。

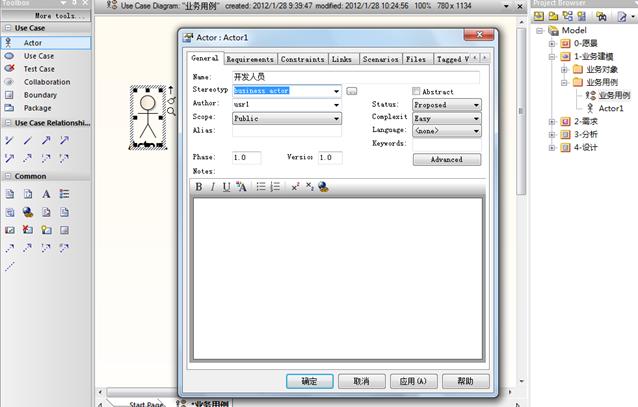




图3-20 添加业务执行者

【步骤4】用同样的方法，继续添加其他执行者：软件组织、出版社、杂志社、会议室提供商、聊天室提供商、差旅提供商、餐饮提供商、银行、外部专家、外部译者。

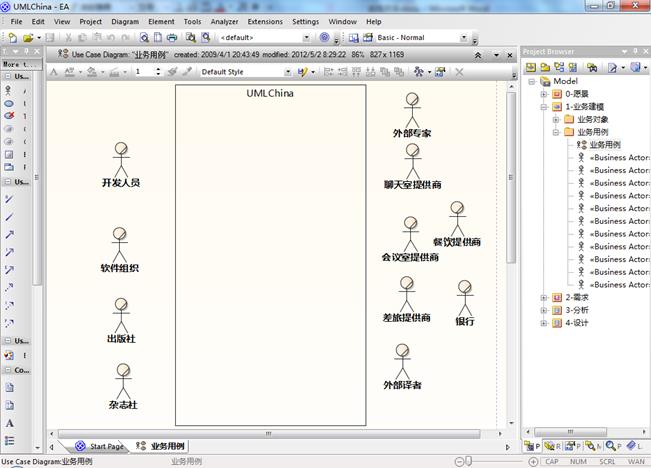
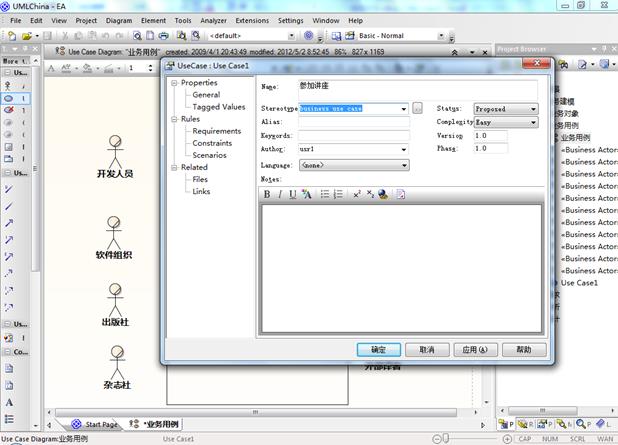


图3-21 UMLChina的业务执行者

【步骤5】点击工具箱中的，点击边界框内，在弹出属性框中的Name栏输入参加讲座，Stereotype选择business use case，点击确定。



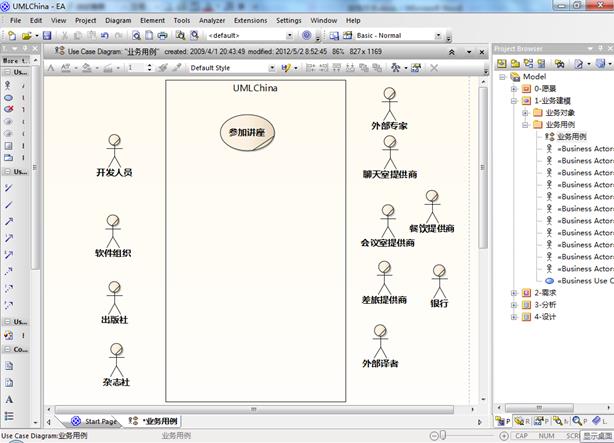
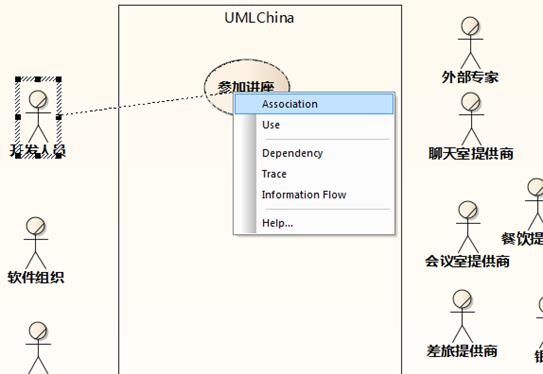


图3-22 放置业务用例

【步骤6】按住开发人员执行者右侧的小箭头（Quick Link），拖到参加讲座用例上，松开鼠标按键，从快捷菜单中选择Association。



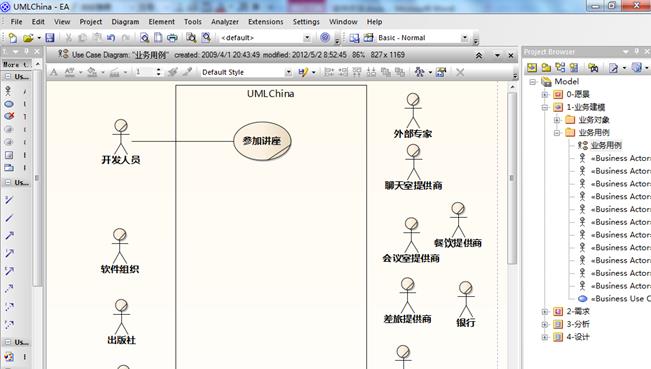
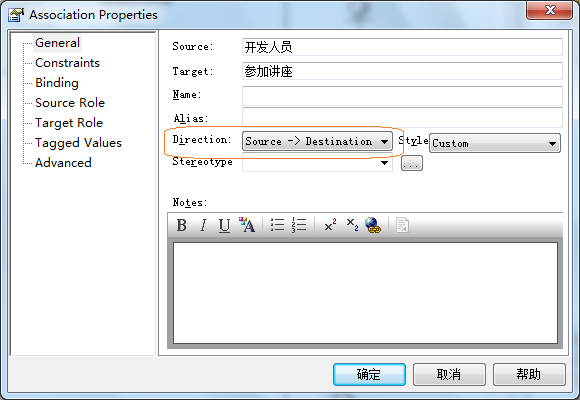


图3-23 建立执行者和用例之间的关联

【步骤7】双击执行者和用例之间的关联线，在弹出属性框的Direction选择框中选择Source→Destination。



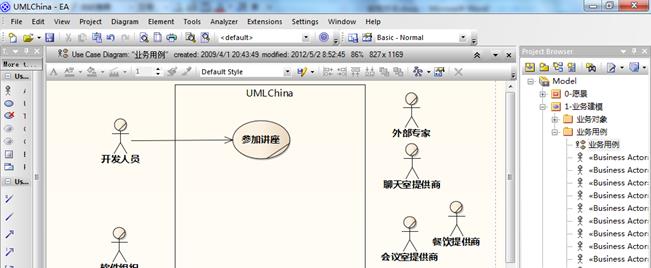


图3-24 设置关联线的方向

【步骤8】同样，在参加讲座用例和执行者聊天室提供商和外部专家之间建立关联，并设置关联方向为参加讲座用例指向执行者聊天室提供商和外部专家。

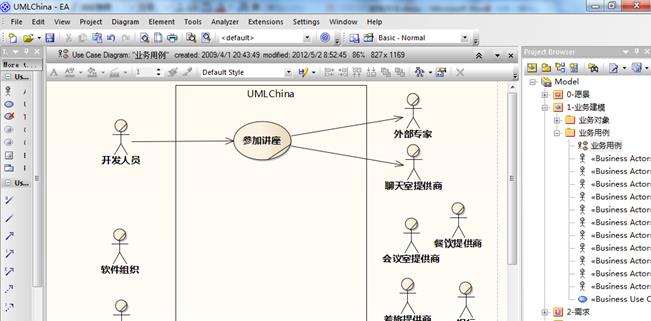


图3-25 添加用例和辅执行者之间的关联

【步骤9】用同样的方法，继续添加其他用例，得到业务用例图：

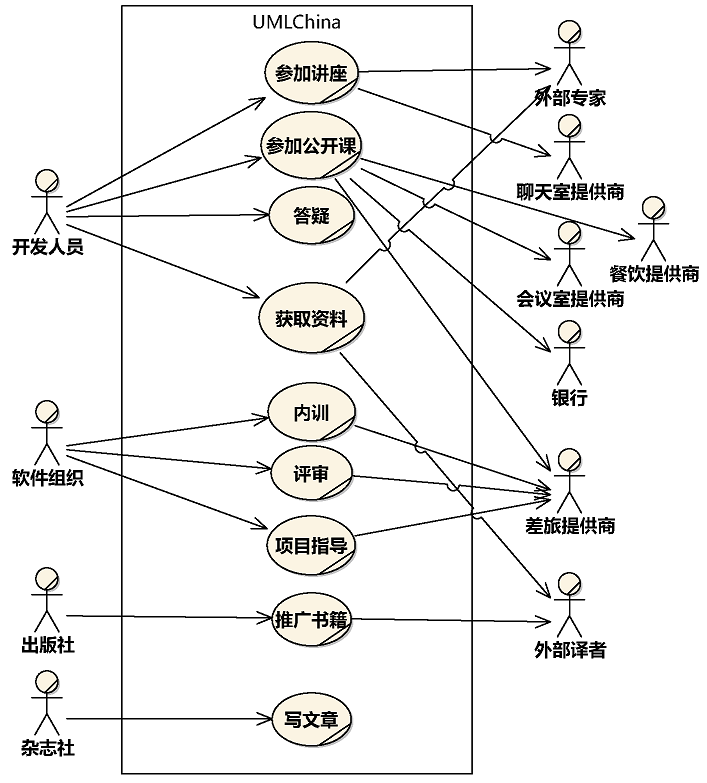


图3-26 完成业务用例图

**3.6 总结**

业务用例是组织的、而不是组织内某个系统的用例。组织的用例不会因为某个人肉系统或电脑系统的存在或消失而改变。所以，“这个系统的业务用例是什么”这样的说法是错误的。您可以注意到：前面画的业务用例图，研究对象都是组织。

既然如此，我们在开发软件系统时，为什么要研究组织的用例呢？因为我们想要把系统的价值和组织的价值挂上钩，给组织一个购买系统的理由。也就是说，业务用例不是思考系统提供什么“功能”，而是思考组织购买了这个系统，对组织本来就有的哪些“功能”会带来一点点帮助。

一个组织，甚至组织的一条流程都涉及许许多多的系统。在开发不同的系统时，研究业务用例和业务流程，发现得到的结果和开发另一个系统时的研究结果差不多，这是很正常的。开发人员不必因此感到惊慌，更不要因为“业务用例太少”、“业务用例太简单了”不自觉地改变研究对象，把待开发系统的用例搬上来。

如果一时之间难以确定要改进的业务组织，难以思考组织的用例，可以先不画业务用例图，先画要重点改进的流程的业务序列图，再从中归纳出合适的待研究组织和业务用例。