中台之上

业务架构设计

BUSINESS ARCHITECTURE DESIGN

Geekbang. 极客邦科技 InfoQ

扫码了解更多内容

CONTENTS / 目录

本书主编：杜小芳　　内容投稿：editors@geekbang.com

流程编辑：丁晓昀　　业务合作：hezuo@geekbang.org

发行人：　霍泰稳　　反馈投诉：feedback@geekbang.org

作者序

作者简介

付晓岩，原国有大行资深业务架构师，负责业务架构设计、项目管理，热衷新技术探索与实践，具有丰富的银行业务经验和企业级项目业务架构设计经验，曾主导客户关系、金融市场、同业、资管、养老金等多个领域核心系统的业务架构设计。公众号：晓谈岩说。

有人说，将来的企业都是科技公司，虽然就目前来讲，这句话还为时尚早，但是，很多传统行业已经被科技大大改变了。大家都知道 BATJ 是科技公司，其实星巴克也已经是科技公司了，在美国的星巴克门店，有将近 16% 的收入来自手机客户端；星巴克自己的 App 有将近 1300 万的活跃用户，星巴克内部已经将网页、手机、社交媒体、网络营销、StarbucksCard 、电子商务、Wi-Fi、星巴克数字网络和新兴的店内消费技术等等，统一作为数字业务战略。今年在风口浪尖上的滴滴，2016 年时就已经日产生数据超过 50TB（相当于 5 万部电影），每天规划 90 亿次路径；2017 年据称全年累计提供出行服务 74.3 亿次。美团公司也是人工智能的玩家，只不过他们没搞个钢铁侠之类的，而是更接地气的、服务快递小哥的语音助手，支持骑手全程通过语音和系统进行沟通、确认，避免了手动操作，提高了效率和安全性。以派单为例，语音系统会说：“派单，从哪里到哪里，收到回复”，骑手只需要说：“收到”，系统就可以确认派单。

到了用户家附近的时候，骑手亦可以通过语音关键词回复，直接拨打电话，从而避免了掏出手机这个动作。在电量过低的时候，系统会提醒骑手，骑行速度过快的时候也会提醒骑手放慢速度，到达顾客附近时，自动提示顾客的地址。这个语音助手不仅方便了小哥，也让快递过程更加安全，减少事故发生。

倒退回 15 年前，恐怕没有多少人真的相信零售、餐饮、出租车、外卖这些行业会跟科技如此紧密相关，甚至直接成为了科技企业，而他们用的技术已经是大多数普通业务人员无法理解的，这不仅仅指技术原理无法理解，连应用方式都无法理解，这是一个真实的“数字鸿沟”。其实技术人员也被这道鸿沟困扰，成天喊着找场景、找场景，说到底，没场景要么是这项技术无用，要么就是没法让业务人员真正理解，导致无法与业务结合。

大家都清楚地认识到了科技的力量，心里都明白要应对技术推动的跨界竞争，但是，要怎么做呢？高薪聘请一些技术人员？买买大厂的科技产品？这些也是需要的，但正如交给你一把狙击步枪，不代表你已经成为了一名合格的狙击手，你还需要自身的转变。这种转变才是最终促成数字化转型的关键。

转变当然不是要大家都去学技术，都当技术小能手，而是转变思维方式，架起一道跨越“数字鸿沟”的桥梁，我认为，这就是业务架构的核心作用。业务架构可以帮助业务人员整体化、结构化、模块化地思考问题，从业务和系统的整体视角，附带一些对技术的基础了解，如分层理念去认识业务和技术；也能够帮助技术人员理解、归纳业务人员的想法和目标，从而让业务和技术能够在同一个语境下，使用同一种“语言”工作。过去那种业务不用管技术怎么实现、技术听懂需求就够了的时代已经过去，以后是深度融合的时代，深度融合就代表互相深入理解，而这种理解需要首先从思维方式的转变开始，通过建立业务架构，让双方都向对方迈出一步，当然，这一步对业务人员的挑战更大，但科技是这个时代的特征，在一个信息化的时代，就得具备这个时代的思维方式，这是任何人也无法回避的问题。在构建业务架构的过程中，业务人员需要技术人员的大力协助来共同掌握这个工具，这就不仅是一个通向理解的过程，更是一个达成信任的过程。此外，我们也无法忽视一点，如果业务本身不能被很好的结构化、模块化，我们也很难做出一个具有良好架构的系统来，就算你是中台的拥趸、“死粉”，也无法解决这个问题。所以，培养业务人员的逻辑思维、架构意识，对于系统开发而言，只有好处，没有坏处。

可能有些技术人员还会觉得应该让业务人员只专注业务就好，但是，

不妨想一想，业务人员和技术人员在现实中的比例，你会发现要是业务人员也能对技术的思维方式有所了解，那将会对技术的合理应用乃至创新产生多大的推动力。打个不恰当的比方，技术人员就好比茶商，你可能想象不到，有多少现代人的喝茶习惯、茶叶知识都是拜茶商所赐，客户对茶叶了解的越多反而兴趣越浓，更愿意尝试不同的茶叶、茶具、技法，很多消费者最终在知识上远超越一般的茶商，这就是大家常说的培养客户、与客户共同成长吧。

为什么业务架构存在 20 多年，技术人员还觉得它有点虚？

业务架构这个词大家时常听到，但是能解释得清楚的却不多，撩撩度娘，你就会发现，不少人问及业务架构和应用架构的关系，聊天时，也常有人问起业务架构师和产品经理什么区别？业务架构分析和需求分析什么区别？为了思考这个问题，我把《软件工程》、《软件系统架构》、《系统分析与设计》都翻了，这些经典教材确实没讲过业务架构这件事；我把《聊聊架构》也翻了，发现其中的讨论有解释到业务、架构和技术的关系，但是也没有特别强调业务架构，所以本文就先梳理下几个较为有名的业务架构理论。

Zachman 模型

其实，业务架构这个词并不新，它隐藏在企业架构（EA）中。企业架构是上世纪 80 年代的产物，其标志就是 1987 年 Zachman 提出的企业

架构模型，该模型按照“5W1H”，即 what（数据）、how（功能）、where（网络）、who（角色）、when（时间）、why（动机）六个维度，结合目标范围、业务模型、信息系统模型、技术模型、详细展现、功能系统六个层次，将企业架构分成 36 个组成部分，描述了一个完整的企业架构要考虑的内容，详图如下：

	数据（什么？）	功能（怎样？）	网络（哪里？）	角色（谁？）	时间（何时？）	动机（为何？）
目标范围	列出对业务至关重要的元素	列出业务执行的流程	列出与业务运营有关的地域分布要求	列出对业务重要的组织部门	列出对业务重要的事件及时间周期	列出企业目标、战略
业务模型	实体关系图（包括 M: M关系、N-ary 关系、归因关系）	业务流程模型（物理数据流程图）	物流网络（节点和链接）	基于角色的组织层次图，包括相关技能规定、安全保障问题。	业务主进度表	业务计划
信息系统模型	数据模型（聚合体、完全规格化）	关键数据流程图、应用架构	分布系统架构	人机界面架构（角色、数据、入口）	相依关系图、数据实体生命历程（流程结构）	业务标准模型
技术模型	数据架构（数据库中的表格列表及属性）、遗产数据图	系统设计：结构图、伪代码	系统架构（硬件、软件类型）	用户界面（系统如何工作）、安全设计	“控制流”图（控制结构）	业务标准设计
详细展现	数据设计（反向规格化）、物理存储器设计	详细程序设计	网络架构	屏显、安全机构（不同种类数据源的开放设定）	时间、周期定义	程序逻辑的角色说明
功能系统	转化后的数据	可执行程序	通信设备	受训的人员	企业业务	强制标准

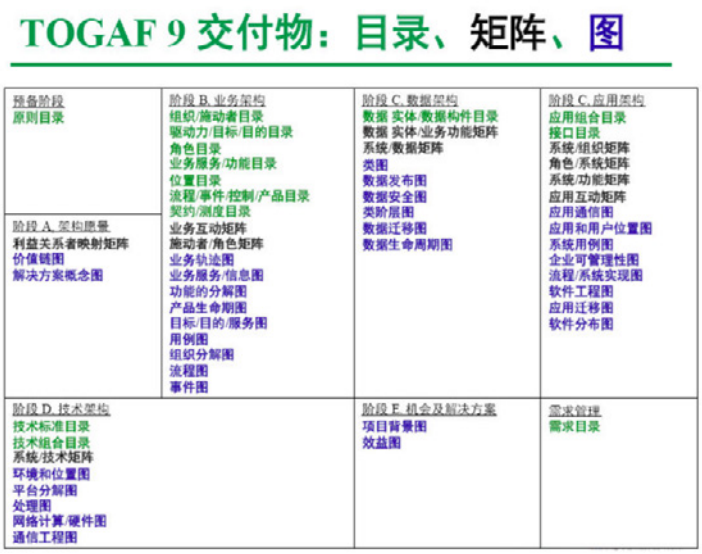
资料来源：网络

Zachman 模型虽然没有明确提出业务架构这个概念，但是已经包含了业务架构关注的一些主要内容：如流程模型、数据、角色组织等，既然没有提出业务架构概念，自然也就没有包含构建方法，所以，Zachman 模型应该算是业务架构的启蒙，同时，它也表明了这一工具或者技术的最佳使用场景——面向复杂系统构建企业架构。

TOGAF

1995 年，大名鼎鼎的 TOGAF 登场了，这个在企业架构市场中据说（2009 年统计）占了半壁江山的架构模型明确提出了业务架构的概念。TOGAF 将企业定义为有着共同目标集合的组织的聚集。例如，企业可能是政府部门、一个完整的公司、公司部门、单个处 / 科室，或通过共同拥有权连接在一起的地理上疏远的组织链。TOGAF 进一步认为企业架构分为两大部分：业务架构和 IT 架构，大部分企业架构方法都是从 IT 架构发展而来的。业务架构是把企业的业务战略转化为日常运作的渠道，业务战略决定业务架构，它包括业务的运营模式、流程体系、组织结构、地域分布等内容。TOGAF 强调基于业务导向和驱动的架构来理解、分析、设计、

构建、集成、扩展、运行和管理信息系统，复杂系统集成的关键，是基于架构（或体系）的集成，而不是基于部件（或组件）的集成。TOGAF 还提供了一个详细的架构工件模型：



资料来源：百度

其中可以明确看到业务架构阶段的交付物。相信很多对架构有兴趣的朋友都认真学习过 TOGAF 模型，此处不再赘述。

FEA 和 DODAF

TOGAF 之后，又先后诞生了 FEA（联邦企业架构）和 DODAF（美国国防部体系架构框架）。前者的体系由五个参考模型组成：绩效参考模型（PRM）、业务参考模型（BRM）、服务构件参考模型（FRM）、数据参考模型（DRM）、技术参考模型（TRM），该方法应用于美国电子政务领域，着眼于跨部门、跨机构提升业务效率，解决重复建设、信息孤岛等问题，很具有“企业级”理念，虽然没有明确的业务架构定义，但是很好地应用了业务架构的思维。后者体系挺复杂的，8 个视点 52 个模型，但是实用性不错，美国国防部和一些企业在用，详细内容如下。

其中能力视点和作战视点就是我们做企业时关注的业务部分。这两个模型网上有相关资料，感兴趣的话可以自行查阅。

为何沉闷至今？

通过寻根溯源，可以发现，即便从 TOGAF 算起，业务架构这个词也有 20 多年的历史了，但是在开发人员中，业务架构显然没有需求分析的

DODAF的核心－8个视点与52个模型

能力视点	作战视点	系统视点	服务视点
CV-1 构想模型 CV-2 能力分类模型 CV-3 能力实现时段模型 CV-4 能力依赖关系模型 CV-5 能力与机构发展映射模型 CV-6 能力与作战活动映射模型 CV-7 能力与服务映射模型	OV-1 顶层作战概念图	SV-1 系统接口表述模型	SvcV-1 服务接口表述模型
	OV-2 作战资源流表述模型	SV-2 系统资源流表述模型	SvcV-2 服务资源流表述模型
	OV-3 作战资源流矩阵	SV-3 系统－系统矩阵	SvcV-3a 服务－系统矩阵 SvcV-3b 服务－服务矩阵
	OV-4 组织关系图	SV-4 系统功能模型	SvcV-4 服务功能模型
	OV-5a 作战活动分解树 OV-5b 作战活动模型	SV-5a 系统功能与作战活动跟踪矩阵 SV-5b 系统与作战活动跟踪矩阵	SvcV-5 服务与作战活动跟踪矩阵
		SV-6 系统资源流矩阵	SvcV-6 服务资源流矩阵
		SV-7 系统度量矩阵	SvcV-7 服务度量矩阵
		SV-8 系统演变表述模型	SvcV-8 服务演变表述模型
		SV-9 系统技术和技能预测	SvcV-9 服务技术和技能预测
	OV-6a 作战规则模型 OV-6b 作战状态转换模型 OV-6c 作战事件跟踪模型	SV-10a 系统规则模型 SV-10b 系统状态转换模型 SV-10c 系统事件跟踪模型	SvcV-10a 服务规则模型 SvcV-10b 服务状态转换模型 SvcV-10c 服务事件跟踪模型
全景视点	标准视点	项目视点	数据视点
AV-1 综述和概要信息 AV-2 综合词典	StdV-1 标准概要模型 StdV-2 标准预测模型	PV-1 项目与机构关系模型 PV-2 项目实现时段模型 PV-3 项目与能力映射模型	DIV-1 概念数据模型 DIV-2 逻辑数据模型 DIV-3 物理数据模型

概念明确，业务架构师也远不如产品经理常见。作者所在单位曾经实施了一个长达数年的企业级转型项目，其中有明确的业务架构组织，但是，每每与技术人员讨论，他们也常觉得业务架构有点儿“虚”。细究其原因，可能有如下几点：

- 用的少。原有的单体式或者竖井式开发依然是大家更经常采用的项目构建方法，而这种开发基本上没有横向视角，所以无需强调业务架构，通常的产品分析或者需求分析足以满足开发需要；
- 难设计。业务架构，特别是大型企业这种错综复杂的业务架构，说起来容易做起来难，业务架构对战略的分解、业务架构自身的整合与标准化、到 IT 设计的过渡都有不少坑，业务越复杂越宽泛就越难驾驭，因此，即便做过业务架构设计的企业，也有不少将业务架构设计保持在高阶状态，有点儿“虚”；
- 易跑偏。施工期间由于客观因素可能导致实施对业务架构的偏离，这种偏离如果没有及时纠正或者调整架构，累积久了会造成业务架构的失真，会变“虚”；
- 难维护。少数扛过了业务架构落地困难期的企业，也会由于感受到维护架构的难度而心生放弃，从而降低了对业务架构的评价。

其实，业务架构从诞生之初就很清楚地定义了自己的使命：面向复杂系统构建。也就是说，业务架构同其他架构一样，目的也是要降低复杂度，更好地规划系统，因此 TOGAF 是将业务架构归属于 IT 战略部分。但是从本人的实践经验看，业务架构不仅具有上述作用，其更突出的影响是对参加过业务架构设计工作的业务人员的影响，他们的逻辑思维能力、结构化能力、企业级观念和意识都有明显的改变，所以，应当将业务架构从

IT 战略中独立出来，更多面向业务人员，以充当业务与技术之间的桥梁。当然，业务架构真正要承担起这一职责，还需要改进、简化业务架构设计方法，对业务人员更友好，并且坚持使用业务架构方法做企业级需求管控，否则，熵增一定会将已经建好架构秩序回归混沌状态。

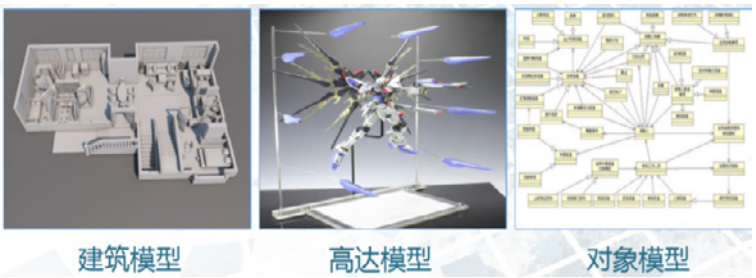
中台说到底也是一种业务架构设计结果，回顾软件设计的发展历程，中台也不是石头中蹦出来的齐天大圣，它并非一种超越了企业架构这个概念的存在，因此，想要深入理解中台设计方式，多去学习下业务架构、软件架构的发展历程还是有帮助的。

架构伴侣：业务模型

业务架构是战略、流程、组织等业务元素的结构化表达，因此，说起业务架构，自然离不开业务模型，所以，本章我们讲讲架构的伴侣——业务模型。

模型与业务模型

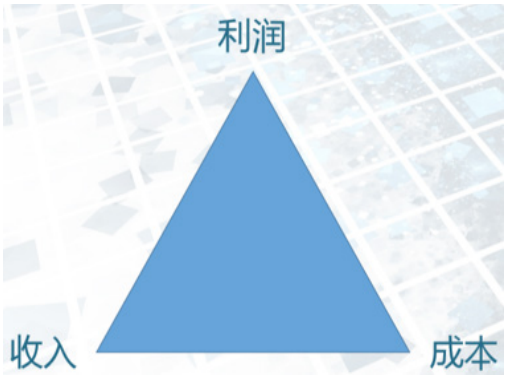
业务模型也是模型的一种，因此我们先从模型讲起。模型的概念大家可以查到很多种，不过，度娘上有一种是我觉得比较容易理解的，这个解释中说，模型是所研究的系统、过程、事物或概念的一种表达形式，也可指根据实验、图样放大或缩小而制作的样品。很多人一说起模型都喜欢说模型是抽象的东西，模型最重要的是抽象，这个说法对软件开发人员而言并无不妥，但是对于理解模型这个概念而言，还是有些狭窄了。模型可以是具象的，可以是实物，比如售楼处常见的楼盘模型，我们的老祖宗修故宫、给皇帝家造亭台楼榭时，也会先做出精巧的木制模型；模型不仅可是真实事物，也可以是虚拟的，只要脑洞开的够大，比如很流行的高达玩具模型、变形金刚等；模型当然也可以是抽象的，比如软件开发中常用的实体模型、时序图、状态图、用例图等等。例子参见下图。



模型就是一种表达形式，其实说出来的话也可以视为一种模型，它是你头脑中的想法的表达，说的过程也就是个建模过程，还遵循了一定语法

规则。所以模型不是个神秘的东西，对于业务人员而言，工作时候经常会画的业务流程图也是模型，与软件开发中用的模型相比，无非是个建模视角和抽象程度的差别。

理解了模型，我们再来看看业务模型。套用上边的概念，业务模型就是对业务的表达，至于这个业务的范围就看你的需要了，如果只是针对一个产品，那业务模型可能就是对产品的生产、销售、使用、售后管理过程的描述，其中还要包含所有参与方的目标、活动、角色、职责等等；如果针对的是一个大型企业，那业务模型的范围就可能包含多条产品线，每条产品线都有不同的业务过程，而涉及到的参与方也会更多、更复杂。所以，业务模型最主要描述的就是组织及其运作过程。企业的业务模型有一个最高阶抽象的三角形。



这个三角形可以说是一切盈利性企业的基本行为，企业为生产而投入成本，产品或服务销售后取得收入，而衡量企业业绩的最基本方法就是通过收入减去成本形成的利润。其实所有企业的行为都可以从这个三角形出发去分析，比如，一个企业基本流程就可以概括为：

企业准备向哪些人销售自己的产品或服务，这就体现了企业自身的价值定位；企业准备组织那些人生产，组织哪些人销售，在什么样的渠道上销售，为此投入什么样的资源，这就是企业的生产和销售流程；收入和成本都需要记账，这就是财务会计的流程；对利润实现情况的衡量、盈亏原因的分析等，体现在管理会计中；所有行为都会产生数据，这些数据是我们做系统设计时的必要输入，是结合业务流程做架构分析的基础。从这个最高阶的核心模型出发，我们可以演化出整个企业的过程，可以模型化地创造一个企业，这就是“大道至简，衍化致繁”吧。

建模原则与模型思维的应用

既然业务模型对业务架构、对系统设计如此重要，那么建模是否有什么诀窍呢？很遗憾，没有。这不仅是我个人的理解，不少关于建模的书中也都会提到，建模看似有很多方法、标准可以遵守，但是模型质量却十分依赖于建模者的经验，是一个“熟练工种”，“老司机”很重要。虽然没有捷径，但还是有两个原则可以时刻注意的：

- 整体性原则。做模型切忌快速上手，不要快速被业务细节吸引，

更不要被立马解决问题的冲动左右，一定要将问题域或者说建模对象放在一个更大的环境中观察，要先找到建模对象的边界，也就是上下文环境。搞不清边界，就搞不清范围，即不知道起止，也不知道思虑是否周全，甚至无从检验建模成果，容易一叶障目，不见森林。

- 合适性原则。大家可能都听说过一个比方，把世界上最美的五官凑在一起，并不会成为世界上最美丽的脸，这就是合适性原则，美丽的脸通常是五官比例好、搭配好的脸，也就是说，模型中包含的各个部分、各类元素要有机结合在一起，不能在设计时为了图新潮、赶时髦，甚至为了建模者个人的“执念”，生搬硬套，强买强卖，忽视了模型的平衡。
- 业务模型是为业务架构服务的，所以细心的读者也一定注意到了，这两条其实也是架构设计的重要原则。建模唯有不断练习，不断参与项目实践，以获得对建模成果的必要反馈，才能有所提高，设计上我们经常把不管实现的架构师比作“PPT 架构师”，其实建模也一样，不能在生产环境中得到反馈，建模者也会成“PPT 模型师”，所以，“实践是理论之源”啊。

经历过的人都知道，认认真真建模是项枯燥繁琐的事情，而且，我也提到，业务架构设计可以帮助业务人员提升逻辑思维能力，应该让业务人员多参加，那么广大业务人员也会疑虑，投入这么大精力参与这事儿，做完了项目，这技能还用得上吗？肯定用得上啊，虽然不会到处去建模，但是重要模型思维可是非常有用的，我个人总结，有这么三点是在各类工作中都值得借鉴的。

- 把握整体。这条不再赘述，应用上，我建议，对于任何领导交办给你的工作，尽可能不要第一时间就“Just do it”，而是要挤出点时间，考虑下来龙去脉，前因后果，这样你才能控制好工作的度，过犹不及啊。
- 穿透现象。浮在水面上的往往是冰山一角，透过现象看本质是我们对建模人员的基本要求，这种注意事物内在联系、本质差别的能力，有助于你拨开现象的迷雾，找到最佳的解决方案。
- 保证落地。前一阵子曾经流行过一句话“一切不为业务目的服务的技术都是耍流氓”，套用一下，“一切不考虑落地的架构设计都是耍流氓”，架构不能飘在天上，印在纸上，所以，真正了解架构本质的人，无论做的是“矮穷挫”的搬砖方案，还是“高大上”的传奇方案，

都要以落地为前提，对应到日常工作中，就是我们无论何时何地提出的工作建议都不能是“空谈”。

中台的表达方式其实也是一种模型化表现方式，毕竟当前的软件设计基本都是“模型驱动开发”，无非是模型工具的差别。关于模型的一些基础性介绍先到此为止，本文所讲的业务架构都是使用业务模型来构建的。

战略和组织结构，业务架构设计中不应被忽视的关键因素

业务架构的起点：解读企业战略

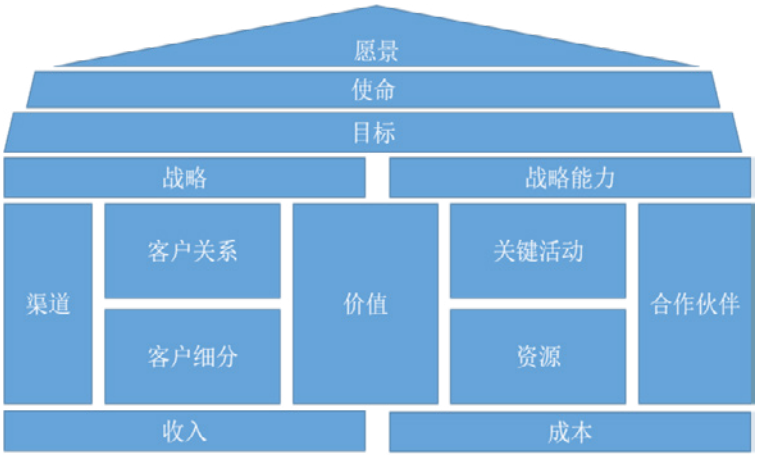
业务架构最大的特点就是要从企业整体视角出发思考问题，要有居高临下的俯视视角，时刻有一张企业整体的业务能力地图印在脑子里，而企业的业务能力是服务于业务目标的，业务目标有不同的层次，高级管理者、中层管理者、操作层都有不同的目标诉求，但是所有的目标都会聚焦在最高层次的企业目标——企业战略上，所以，企业战略也就自然成为了企业级业务架构设计的起点和检验标准。

企业战略听起来是一个非常“高大上”的词汇，也有人会觉得企业战略有点儿画大饼的嫌疑。其实企业战略没有那么神秘，也不是非得智商 180 的人中龙凤才配谈起，企业无论大小，都有自己的目标、路线、方法，企业战略就是对这些内容的提升性描述。随着研究的不断深入，企业战略也

有了更加丰富的内容和种类，但是说到底，企业战略仍是企业为达成某项目标而确定的过程与方法。关于企业战略的各类资料已经浩如烟海，我也无意在这里喧宾夺主、浪费时间，我还是介绍一个设计企业战略或者说分解战略目标的有效方法，毕竟从目标出发引导出业务架构设计才是我们关心的内容。

盖个房子

我介绍一个很容易掌握的企业战略设计模型，该模型应该是由BMGovernance 公司设计的，模型如下图所示。



该模型从企业愿景、使命这种相对宏观的概念入手，向下分解出可度量的目标，这三部分做为“屋顶”，描述的是企业为自身发展所设定的目标和成功的标准，无法衡量的目标只能是个精神口号，不能被转化为行动。这不是说不能喊口号，而是这个口号必须能够被分解成可以指挥行动的具体度量，比如，愿景定义为“让全世界都使用清洁能源”，使命就是“逐步使用风电、太阳能、水电等能源取代具有污染及潜在污染可能性的火电、核电等能源”，那目标自然就是“清洁能源使用量占世界能源使用量的百分之百”。衡量起来也容易，看统计数据就行了。

无论是世界级企业给自己定下的改变全人类的宏大誓愿，还是小企业只盼明天还能够生存的小梦想，都必须要“量化”出来，成为可执行的目标。这一点是业务架构设计人员在设计企业战略时必须特别注意的，千万不要把战略只当口号，漂亮就行，而是要能够做为一个参天大树的根，可以坚实地向上生长、开枝散叶。

愿景、使命、目标这三个经常被大家诟病为“假、大、空”的战略元素在一个可靠的业务架构设计中，被如何强调都不过分，因为它是“屋顶”，

搞错了，就成了“上梁不正下梁歪”，所以务必与企业的管理者沟通清楚，务必让所有参与方都达成一致认识。这是项目中最高层级的概念一致性，绝对不能“输在起跑线上”。不少技术人员不重视企业战略，认为跟开发人员无关，这是大错特错，如果一个企业的战略会让员工觉得跟自己无关，那只能说明这个战略本身以及对战略宣导的失败。不落实到流程中的战略是无法被执行的战略，而跟战略落地无关的流程到底是在干什么呢？创造的是什么价值呢？为这样的流程开发的系统又是在干什么呢？如果一个企业花了几百万、上千万开发的业务系统，其设计人员连企业的战略都不知道，那这个系统能支持企业的发展吗？业务与技术的融合就更无从谈起了。

屋顶之下左侧是战略、右侧是战略能力。战略是为了完成上边提到得目标所需要采取的路线、方法，这类似于银行的分行每年为了完成总行下派的经营指标所制定的各种经营策略，比如大力挖潜，激活存量客户。战略能力则是为了完成这一策略需要的能力，比如为了采取激活存量客户的行动，需要的客户分析、营销组织、渠道应用、业务处理、合作伙伴管理等。

再下一层级就涉及到了具体工作，左侧表述的是为实现战略而在客户一侧采取的行动，包括渠道、客户关系、客户细分，实际上就是指面向哪一类客户、在什么渠道上、如何为其提供服务。继续上边的例子，激活存量客户要先考虑激活哪类客户，是一般客户还是高端客户，不同的细分客户需要不同的策略。现在大家都讲求精准营销，在大数据、人工智能的加持下，从“千人千面”一路飙升到“亿人亿面”。选择了客群之后就要考虑渠道类型，是通过互联网渠道、电话渠道、手机银行渠道、微信渠道还是柜面服务。确定了渠道之后，还要考虑激活行动的消息如何送达客户、如何让客户愿意接受以及相应的售后服务，这些属于客户关系范畴。

左侧最下边的是收入，也就是说上述行动成功后，应当产生预期的收入。右侧对应的三个方块则是在企业内部还需要采取的行动，关键活动是支持激活客户战略所必备的业务处理过程，包括交易流程、积分规则调整流程、积分兑换流程等。

关键资源则是为支持促销战略需要提供的资金、人员、物品、参加活动的网点等；合作伙伴则是为了补充银行能力的不足引入的外部力量，比如为了激活一般客户所提供的交易积分兑换电影票、为了奖励高交易量客户提供的体检、高尔夫球等活动，都需要与第三方合作才能办到。右侧最下边的是成本，也就是说上述行动会带来合理的成本支出。收入与成本的差额就是收益了，这就是对激活存量客户策略的最终检验。

居中的是价值定位，企业为什么类型的客户提供什么类型的服务就是

企业的价值定位，方块左右两边描述的其实就是这个含义，企业的价值定位是否准确、可持续，也就是看左边的活动产生的收入是否能覆盖右侧的活动带来的成本，如果是，企业就能够长期发展，如果否，企业就需要重新思考价值定位了，而这种反思很可能带来“屋顶”的变化。

在这个模型中，大家可以看到上一讲中提到的三角形也包含其中，如果说那个三角形是“大道至简”，那这个模型就可以看作是“衍化致繁”了。

廉价沙盘

这种建模过程也是对战略的一次“廉价”沙盘推演，能够衡量战略的合理性、可行性，这种分析是非常有价值的，可以避免工作的盲目性。相信大家在日常工作中都遇到过“不计成本”、“不计代价”的“强硬”需求，那么今后大家可以试试通过这个模型对“强硬”需求做个全面的合理分析，也许能够帮助需求方发现战略缺陷，找到改进方法，使业务方案更符合各方的期望。否则，一个连沙盘推演都走不通的战略如何能指导业务发展呢？更别提去为此开发系统了。

经过之前建模大家已经能够看到隐藏其中的业务架构了，渠道管理、客户信息管理、客户细分、客户关系管理、金融产品组件、合作伙伴管理、财务核算、绩效考核等业务能力组件已经呼之欲出，你心心念念的“中台”也是若隐若现，就等更为细化的建模工作了。此外，大家也不妨思考一下，既然企业战略没那么神秘，那无论对于各种规模的客户，是不是都该鼓励大家享受一下战略的“乐趣”呢？

不可回避的问题：组织对架构设计的影响

前面提到过，业务模型描述的就是企业的组织和运作过程，可见组织在业务架构中的地位。有些了解过业务建模或者企业级架构理念的小伙伴可能会问，既然业务架构要横向看问题，那做企业级不是更应该要打破部门限制，实现企业级的统一吗？是的，但那是目标，不是过程，也未必真能是结果。下面我们就来谈谈组织结构对业务架构设计的影响。

组织要背“锅”吗？

说到组织结构对系统设计的作用，很多人都会想起“康威定律”。Melvin Conway 于 20 世纪 60 年代后期确定的 Conway 法则告诉我们，任意一个软件都反映出制造它的团队的组织结构，这是因为人们会以反映他们组织形式的方式工作。换句话说，分散的团队可能用分散的架构生成系统。项目团队的组织结构中的优点和弱点都将不可避免地反映在他们生成的系统中。这个规律延伸到需求方身上也一样适用：需求方的组织结构不

可避免地会影响到系统的组件结构。俗话说:“干活儿不由东，累死白搭工”，简直是对这个问题最直观的解释。

做业务模型，我之前说过，有两个原则要注意，其中之一是整体性，设计业务架构，我们当然希望能够整个企业通盘考虑，不要因为部门利益影响组件边界的划分，影响功能设计，做出来的东西，凡是公用的部分，应该照顾到所有利益相关方的需求；凡是已实现的功能都应该对新的需求方开放并支持必要的扩展，这是企业级设计应该追求的目标，但是，实现上常常很困难。企业无论大小，一旦系统的设计边界跨越了单个部门的职能范围时，都会出现部门利益问题，无非是企业规模、文化差异造成的协调难度的差别。

在企业内部，部门利益是部门需求的天然边界，即便要做企业级，大家肯定也是先要说清楚自己的需求，才能考虑别人的需求，种了别人家的田，荒了自家的地是绝对不行的。所以，大家在坐到企业级这个谈判桌上来的时候，都是先揣了自家账本的，首先要满足自己的业务诉求。这就要求，做为业务架构的设计者，你拿出来的方案最好是以一种更有效的方式满足所有相关方的需求，而不是单纯做抽象、归并，要大家你让一陇地，他少一棵树的方式搞折衷，这样实际上失去了做企业级的核心价值，因为这样的折衷即无法保证系统先进性的，也无法保证用户体验，甚至可能退步。部门利益是做企业级最大的障碍，跨越这个障碍是对业务架构师设计能力的最高挑战，当然，要客观，没有更好解决方案时，不动也是一种选择，因为，同样接受这个挑战的还有企业文化。

举个例子，银行都有积分系统，近年来大家也都做了综合积分，其实这里边的难度很大的，主要问题不在技术而在业务。理想的综合积分是企业只有一个积分系统，支持所有产品的不同积分规则，对不同客户群、不同营销方案可以进行参数化配置，最重要的是，支持以单一积分形式统一用于奖励兑换，而不是要换个包包，得分别去花信用卡积分、黄金交易积分、基金业务积分，这样客户会疯掉。但是都用一个积分，内部问题就上来了，信用卡部门为了促销，提高积分发放，这样信用卡用户就在兑换奖励时占了便宜，黄金交易就可能下去，黄金交易的管理部门一激动，也开始刷积分，结果会造成积分的营销费用提前被花光，反应慢的部门已经没机会促销了。说到这里，大家也就明白了，综合积分背后的博弈是营销费用，搞综合积分以前，这个费用其实是划分到各部门的，各部门自行支配，蛋糕先分好，如果综合积分设计不考虑清楚这个问题，就会动了大家的蛋糕。所以，如果能解决这个问题，综合积分就能真正做到一个组件里，解

决不了，就还是各自为政，那放不放在一个组件中就没有实际意义了，只是解决积分综合查询问题的话，有很多方法都好过功能迁移。

组织问题会延伸到项目上

即便设计好了上层的业务架构方案，是不是就能顺利实现企业级了呢？也未必。企业的组织结构会影响其内部沟通效率，壁垒森严的大型企业，沟通效率通常较低。大家可能知道，组织的沟通方式主要是正式和非正式两种，其中，正式沟通在大企业中最常见的方式就是开会。如果项目少可能还好些，但是大型企业通常会有多个项目同时施工，一般都是项目群、项目组合的方式，而如果下决心搞企业级转型项目，十几个、甚至几十个项目长年同时施工也是很正常的，互联网企业相信更是如此，每天都在挑灯夜战。那么由此带来的一个问题就是项目组之间为执行企业级设计蓝图，在开发过程中可能需要对组件协同问题、边界问题频繁沟通，项目经理、业务经理、技术经理这些角色甚至会成为职业开会人，如果会议效率难以保证，一个问题久拖不决，就会产生两种结果，一是项目组担心工期延误直接改变架构方案；二是采用非正式沟通方式，项目组间通过私下交流解决问题，而后者也极有可能是以改变架构方案为代价的。

这两种结果都会令企业架构很“尴尬”，导致架构失灵。而应对这种问题并没有特别好的方法，只有加强企业级架构人员的能力与数量，让企业级架构人员以 BP（合作伙伴）的方式走入项目组，在项目组间搭建起企业级架构协作网络，提升架构决策效率，才能不让企业级架构成为瓶颈。当然，有个牛人居中，在最高层领导的支持下，强行拍定各种决策，也是一种选择，但是，企业越大，尤其是业务居于主导地位的企业中，这种结构也很难形成。

企业的组织结构在业务架构的设计与实现中具有很重要的影响，其实，理想的企业级系统建设与组织结构转型是相辅相成的，一个在组织结构上高度部门化的企业是很难建成一个真正的企业级业务系统的，这一点在做业务架构设计时务必要考虑，方案与组织结构要匹配，否则很可能在落地时困难重重、面目全非。企业级转型多数是需要时间的，休克疗法、瞬间跨越很难，这一点上，业务和技术要通过业务架构设计过程互相影响、互相协作、互相改变。阿里的“中台”建设也经历了一个“砸烟囱”的过程，其实无论你砸的多卖力气，“烟囱”总还是会有的，对于企业级设计来讲，这是大家必经的“坑”。

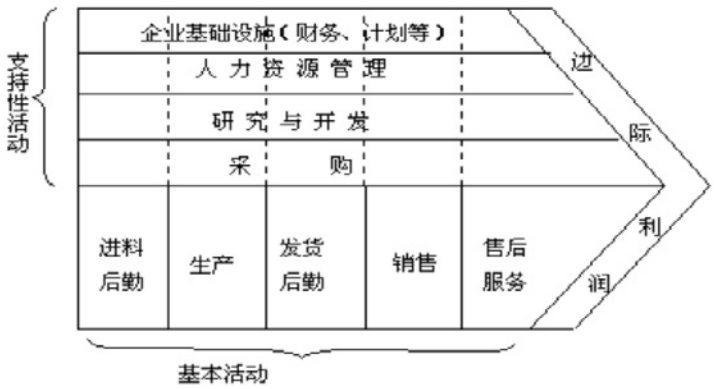
面对复杂的流程和数据，我们总结出了一个分析套路

前面的文章中我们分析了企业战略、理清了组织结构，是不是就该进入业务分析了呢？先别急，业务分析，特别是对于具有多个不同业务线的企业而言，是一种垂直式的分析，如果直接开始业务分析，那就走上了竖井式开发的老路，就算有共同的战略目标，也未必建得出企业级的业务架构和业务系统来。业务架构强调的是横向视角，强调通观整个企业的生产过程，因此，展开垂直的业务分析之前，我们必须先确立一个统一的业务分析框架做为观察各个业务线的统一方法，这样才能将企业需要的业务能力进行分类汇集，产生合理的组件结构。

价值链分析

我们首先讲一下这个用来做横向分析的方法，通常管理学上分析企业竞争力多是使用价值链模型。价值链 (value chain) 概念首先由迈克尔·波

特 (Michael E.Porter) 于 1985 年提出。最初，波特所指的价值链主要是指针对垂直一体化公司的，强调单个企业的竞争优势。随着国际外包业务的开展，波特于 1998 年进一步提出了价值体系 (value system) 的概念，将研究视角扩展到不同的公司之间，这与后来出现的全球价值链 (global value chain) 概念有一定的共通之处。之后，寇伽特 (Kogut) 也提出了价值链的概念，他的观点比波特的观点更能反映价值链的垂直分离和全球空间再配置之间的关系。2001 年，格里芬在分析全球范围内国际分工与产业联系问题时，提出了全球价值链概念。全球价值链概念提供了一种基于网络、用来分析国际性生产的地理和组织特征的分析方法，揭示了全球产业的动态性特征。具体采用哪一种价值链模型，要看企业的实际需要，比如，是否更关注上下游关系等。这种模型的建立往往不是企业自身能简单搞定的，可能需要一定的咨询或者学习过程。波特价值链如下图所示：



图：波特价值链

价值链主要包括基本活动和支持性活动，基本活动是主要生产过程，支持性活动则是对基本活动起辅助作用及维持企业基本运转的各类活动。实际使用中不必完全一模一样的照搬，因为波特价值链一看就知道偏重制造业，偏重生产类型的企业，对于服务业而言就需要适当变形，其实，价值链主要描述的是企业的价值创造过程，引入价值链分析主要是为企业横向审视自身业务能力提供分析框架，因此，价值链如何设计，完全是可以个性化的，只要确认好能够符合企业特点，覆盖其价值创造过程即可。

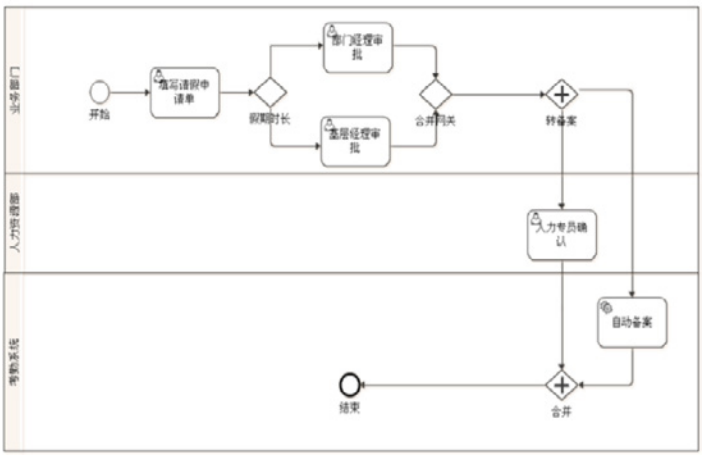
业务领域设计

做好这一“横”之后，我们就可以画出多个“竖”了。科学地讲，业务领域的划分取决于企业的战略和价值定位，企就是业打算为什么类型的客户

提供什么类型的服务或产品，比如，银行为个人客户提供金融服务，就产生了个人金融业务线，这里边存款、贷款、金融市场、非金融服务等等，会有一大堆东西，而如果觉得这样划分依然太粗，那么很有可能私人银行这类高端客户服务就会独立出来，为其设计的一些业务功能聚类成的业务组件可能不会为普通客户提供服务。划分领域其实可以有两种方式，从客户出发和从产品出发，选择哪一种，要看企业的特点以及企业更关注什么。还以银行业为例，银行业有不少产品是同时适用于个人和企业客户的，因此，从客户出发，很多产品会交叉；而从产品出发，会避免这一问题，毕竟业务系统的设计多数还是以产品为主线的，但要注意，这里指的不是具体的某一个产品，而是一组同类产品的集合，比如存款、贷款、托管、资管、投行等。选好维度之后，就有了横轴——价值链和纵轴——业务领域两个维度，接下来就可以去分析业务流程了。

业务流程分析

业务流程的分析实际上就是将一个业务领域中的所有业务处理过程按照价值链约定的分解方式分解，形成每一个价值链环节中的一个或者多个工作流，具体每一个工作流程的设计可以采用常见的 VISIO 设计工具，可以遵循 BPMN 语法标准，也可以采用其他制作工作流的语法标准，但是要注意，必须整个企业统一采用一种，不然是没法整合的。以 BPMN 语法为例，一个工作流在 BPMN 语法中称为一个活动，每个活动可能会有多个不同的角色共同参与，具体涉及到哪些角色就又涉及到企业的组织结构了。每个角色在活动中承担的职责称为任务，其实工作流分析重点在任务，活动的范围并不那么严格，甚至不是非常重要，活动之间在 BPMN 语法中是可以靠事件串接起来的，既然能够串接，那么范围或者说流程图



的长度就不是特别重要。我们甚至可以把一个业务领域中不同价值链环节下的所有活动都连接成一个特别复杂的活动，只不过这样可读性会非常差。所以，操作上，还是建议活动尽可能在每个价值链的范围内，而每个价值链内有多少个活动，可以自由些，可以多参照对业务场景的划分。业务流程的分析重点在任务，因为任务在后续设计中对功能、组件内部结构的影响比较大，这个后边章节还会陆续介绍。一个常见的 BPMN 工作流如上图所示。

综上，业务领域其实是企业确定以某类产品服务某类客户的一个业务范围，在建模上，它表现为，为实现这一价值定位，企业在整个价值链上的各种业务活动的有机结合，一个业务领域实际上就是由一组业务活动构成的，通过活动中的角色和任务，体现了所有参与到价值创造过程中的组织单元的分工协作关系。

要注意的是，这一阶段完成的模型通常是不够准确的，因为还没有经过“精炼”的过程，其对企业级设计的贡献还只是个开始。业务领域及流程的分析中，还有一点要强调的是，别在忙于细节时忘了大方向，业务架构设计是从企业战略开始的，那么做到业务领域分析时，要时刻提醒自己，业务领域内的活动是否能够有力地支持战略的实现，是否能够有效地服务客户，是否能够有效地应对行业竞争，也就是先进性的衡量，把这三个问题如同曾子三问一样看待，“日叁省乎己，则知明而行无过矣”。

上一章概述了业务流程建模的过程，流程建模其实“一言难尽”，需要不断的练习、摸索，自己总结套路，也就是之前说过的，模型质量严重依赖建模者的经验。软件设计主要研究的是行为和数据，流程模型分析了行为，数据模型当然就是要分析数据。数据模型在很多系统分析、软件工程的教材上都有介绍，所以我不去赘述三范式之类的数据建模规则，而是从企业级数据模型、业务模型与数据模型协作关系的角度，讲讲数据模型。

企业级数据模型

提起数据模型大家可能第一反应都是 ER 图，实体关系图是我们做关系型数据库设计的基础。实体图是按照对业务对象的划分，将数据属性按照实体聚类，并描述实体间的关系，从而指导程序设计和数据库设计。我们通常做 ER 图是面向单个系统构建的，而要构建企业级数据模型时，就需要横向分析所有业务领域的 ER 图，所以，我们也需要以一种总体结构先建立分析框架，比如金融类企业常用的 FSDM(financial services data model)，它是 IBM 的一个企业级数据模型，囊括了银行约 80% 的业务数据。

FSDM 将数据分为九大类，分别是参与人、合约、条件、产品、分类、时间、资源向、位置、业务方向，具体定义如下：

类名称	简称	定义
关系人	IP	银行的业务开展过程中的相关各方，个人、机构、柜员
合约	AR	参与者之间达成的 合约、合同、协议等
条件	CD	描述银行的业务正常开展，所需要的前提条件、资格标准和要求
产品	PD	产品是为客户所提供，以换取利润的产品和服务，产品也包括合作伙伴或竞争对手的产品和服务，是金融机构销售或提供的可市场化的产品、组合产品和服务
地点	LO	参与人相关的所有地址，如家庭地址、公司地址、邮政信箱、电话号码、电子地址、网址等或地理位置区域
分类	CL	适用于其它数据概念的分类或者分组
业务方向	BD	银行或参与人开展业务所在的环境和方式
事件	EV	是参与人和银行的交互，以及银行内部的业务交互，它包含最详细的行为和交易数据，例如存款、提款、付款、信用 / 借记卡年费、利息和费用、投诉、查询、网上交易等
资源项目	RI	是银行有形或无形的有价值资源项目，是银行拥有，管理，使用的，或支持特定业务目的的

通过这个框架可以将数据实体、数据属性进行归类，形成统一的企业级逻辑模型。做为企业级模型，数据实体和属性都要保证唯一，这一点在建模中好说，通过工具筛查就可以比较出名称、定义、取值重复的数据项，从而保证数据唯一性。但是重点在于生产阶段的管控，而非建模阶段。生产阶段要通过数据管控平台或工具对数据字典进行严格管理，没有进数据字典的数据项，无法生成企业唯一的数据项 ID，无法在设计时被使用，从而达到严防死守一般的控制，虽然也让生产上一顿抱怨，但这个方法很有效。企业级数据模型说起来容易，做起来难，要首先对业务数据进行全面建模，再对生产进行严格管理，并对历史数据进行处理。本人原所在单位经历了两年多的努力，成为行业内首家真正建成企业级数据模型、真正实现企业级数据管控的大型金融机构。

与流程模型的配合

流程模型与数据模型是描述业务需求的一对儿“难兄难弟”，流程模型表达的是“处理”，数据模型表达的是“输入”和“输出”，合起来就是计算机的基本工作流。数据模型和流程模型的组合，可以清楚的描述出，什么样的事件或条件可以触发一组业务活动，业务活动需要的输入有哪些，经过业务流程的处理，输出又有哪些。如果将该业务系统化，就成为实现业务活动的计算机程序是在什么样的场景（事件）下开始执行，程序需要读取

哪些数据（实体），依据什么样的顺序（活动）、规则（任务）由谁（组织、角色）执行，执行之后产生哪些数据（实体）。任务会直接处理数据，而这种处理通常分为增加、修改、删除、查询四类。

一个业务领域是由一组活动构成的，而这些活动分布在价值链的不同环节，如果粗糙地划分业务组件，则将每一个价值链环节设为一个业务组件也未尝不可，不过这样未免太“偷懒”，对于业务复杂的大型企业而言，组件的内聚性会很差，所以我们需要更为精细的划分。数据模型都有主题域这个层级，就是将关系较近的数据实体聚合成一个分类，这种关系我们可以给出一个主题名称，比如，当按照产品划分主题时，FSDM 中产品分类下就可以建立一个“存款”主题域，将存款业务相关的数据实体放入其中，并使用 ER 图的方式表达。

在软件设计上，是可以考虑将关系较近的数据实体聚在一起，按照行为接近数据的原则，再将相应的功能聚合成一个组件。结合业务模型，就可以将与主题域中与实体相关的任务聚在一起构成业务组件。聚类过程中要注意：

- 数据主题域中的数据实体可能存在引用其他主题域数据实体的情况，这种情况下，在进行任务聚类时不会考虑此类数据实体，因为它们应当由其所在的数据主题域相关的组件创建，以保证在企业级业务系统中，数据生成职责的唯一性，这是应用企业级数据模型时非常重要的一点。
- 与数据实体相关的任务主要指对数据实体进行新增、修改、删除的任务，对同一数据实体进行新增、修改、删除操作的任务应当归属同一组件。只有这些任务具有数据的写权限，其他任务只具有读权限，这也是保证企业级数据一致性的重要措施。

上述原则会在一定程度上影响任务边界的划分，是否需要因为在任务中要表达对不同主题域数据实体的写操作，就需要将任务切分开，或者直接复用其他组件中已有的对该数据实体进行写操作的任务？表达上，当然切分开或复用任务最好，甚至可能复用活动，但是实际建模过程中则要具体问题具体分析了，这一方面是建模的问题，但另一方面其实也是应用模型过程中很重要的一个环节——解读模型的问题，如果两者比较统一，那模型具体长成什么样子就不必太纠结了。所以，该如何处理，同时取决于这两方面的情况，考验的是“统一语言”是否真的建立了。

熟悉 DDD 的朋友可能会问，任务与实体的关联主要基于对实体的增删改，这不是有点儿“贫血模型”的意味吗？其实不然，流程对数据的更丰

富的处理规则其实可以包含在任务的描述中，从而摆脱“贫血模型”的问题。

企业级数据模型与企业级流程模型相互支持，而其中最重要的概念都是企业级，企业级在操作层面上，说到底是个标准化问题，我们将在下一讲阐述这个问题。

业务架构和中台的难点，都是需要反复锤炼出标准模型

企业级业务模型的建设离不开标准化操作，因为做企业级模型要横向对比分析企业所有业务领域，以期望在设计上实现“以更少支持更多”，这是很多企业搞企业级系统建设或者企业级转型的目标，希望能够同时实现系统实现的快速灵活和减少重复开发以降低成本这两个目标。这个愿望是非常美好的，也是很多企业级架构设计追求的目标，关于对这一点的体会，留待以后再讲，这节我们还是一起讨论下为实现这一目标所需要进行的标准化工作。

基本的标准化方法

标准化最重要的是数据标准化，数据建模中已经提到了，企业级数据模型要保证数据实体和属性的唯一性，这样就不会由重复的概念产生，重复的概念会造成数据的“同义不同名”。影响数据的使用和统计结果，数据模型的唯一性从工具角度比较容易控制，通过对定义、取值的比较，能够

筛查出多数概念问题，但是依然有些定义问题不易发现，这就需要通过与流程模型的配合检查了。

数据模型是标准化的基础，我们先假定多数数据概念重复问题已经通过工具筛查解决，数据实体和属性基本保持唯一，这时我们可以将数据与流程对应起来，对应的主要方式就是识别任务需要使用的数据实体，包括读和写两类，上文说过，对组件的识别就是先通过数据聚类，进而到任务聚类实现的，任务聚类主要就是对写操作的任务进行聚类，因为读操作不会改变数据，可以由负责写操作的组件提供读取服务实现，所以，写操作是聚类任务的基本原则，这样是为了保持数据的一致性。

在对应过程中，经常会遇到多个不同的任务都可能要对同一个数据实体在不同时间进行写操作的情况，比如，个人客户初次到一个银行存钱，申请银行账户时，银行要建立客户的信息，会包括姓名、证件类型、证件号码等基本信息，也会包括电话等联系信息，或者邮寄地址等地址信息，这时的整体业务场景是存款。而客户过了一段时间再次来办理业务时，联系信息可能会有变化，这需要更新客户信息，但是此时的场景有可能发生变化，不再是来存款，可能是来购买实物黄金，从产品的角度，这就是两个不同的业务领域了。

那么，在进行企业级标准化以前，对客户信息的建立和修改完全有可能在存款和实物黄金的领域各有一套流程，可能是任务级别的重复，也能是在不同的任务中各自的内容有重复，实际上，以前做竖井式开发的时候，这是很常见的现象，每个业务系统都是独立的、完整的，都各自有一套客户信息，不仅重复，最重要的是经常会不一致。

当我们通过企业级数据模型去除重复的数据概念之后，通过任务与实体之间的写操作对应关系，会清晰地发现重复的操作。这时我们就需要做出建模的决策，是分开建模还是将所有对客户信息进行写操作的部分集中到一起建模。在 FSDM 提供的数据模型上，参与人这个分类中可以容纳与客户信息相关的所有数据，建模上可以把此类实体聚集在一个主题域下，比如叫做客户主题域，那么从企业级的角度也就可以将各业务领域中与之相关的任务或者涉及到该操作的任务中的客户信息部分全部抽离出来，集中到起来组成一个组件，而其他领域的任务经过调整后，不再包含此类内容，这就完成了一个标准化过程。

可能会遇到的问题

上述操作是相对较为简单、清晰的标准化过程，还有些标准化过程要

更难以判断，这种情况通常出现在流程看似相近的业务领域，以及一个领域内部的多个产品之间，后者其实更难判断，因为一个业务领域内部的流程本就相近，会很容易让人产生“整合”的冲动，尤其是对于业务建模来讲，因为业务建模毕竟是一种“纸上操作”，分、合都是很容易的，调整下结构而已，而整合对建模者来讲又有很大吸引力。为了避免这种错误，需要从业务和数据两方面下手，业务上自然是要重新审视、理清业务流程，搞清楚具体差异；而数据上要重新检视数据实体划分的颗粒度是否正确，是否包含的属性太多而导致内聚性不够。数据实体的颗粒度太小，会放大业务差异，而颗粒度太大，则会抹杀业务差异，二者都会导致不合理的标准化结果。

因此，流程模型与数据模型之间的配合检查是一个反复锤炼的过程。尽管标准化问题很重要、很困难，不幸的是，并没有什么很好的方法能够帮助大家快速解决问题，这就又回到了之前说的，模型质量严重依赖建模者的经验，除了经验之外，还要依靠高质量的建模输入，既包括完善的业务资料，更需要有丰富经验的业务人员，看资料是学不会业务的，尤其是业务中经常会有“活情况”。

业务人员与技术人员融合得越好，就越能产生高质量的模型和系统，这也难怪高盛、大摩这些金融机构中数字化转型的坚定执行者，会引入占员工比例 15%、甚至 20% 还多的技术人员，并直接派驻到业务部门与之共同工作。一般国外金融机构技术人员占比不足 8%，国内通常为 4% 左右，近年才刚刚有所增加。

尽管标准化过程很痛苦、自身又似乎很不“标准”，但是因其对企业级系统的构建意义非凡，因此，所有做企业级转型、希望建设企业级系统的企业和开发者，都必须认真对待这一过程，尽管这一过程未免有点“纸上谈兵”，但它的优势也在这里，这一阶段的任何调整都是代价极低的，而不合理的设计一旦传导到开发上，就将产生较大的纠正成本。本人原来所在单位规模很大，业务庞杂，因此这一过程耗时两年，其后仍在持续优化、完善。对于大型复杂系统而言，因其面对的问题域异常庞大，所以需要一套清晰的业务与 IT 的架构映射关系指导企业的持续建设，这就如同我们对地图的需要一样，只有践行标准化才能提供一张准确的地图。

这种标准化也是识别中台能力的基础，其实阿里的中台也是在不断的标准化和去重过程中沉降下来的。

如何为一个商业银行设计业务架构？

从实际操作的角度讲，企业级业务架构设计及其建模过程是一个充满可能性和争议的过程，并没有一个直观的量化标准能够用于判断一个架构方案的好坏，我们可以通过一个虚拟的例子体会一下。

假定我们为 A 商业银行设计企业级业务架构，为了集中感受组件和标准化的过程，我们跳过战略分析，不导入更多目标，比较单纯地从简化的现状入手，推导可能的目标架构。

价值链设计

假定只分析存款和贷款这两个大家耳熟能详、无论做没做过银行系统都能基本了解的业务，并且假定产品只面向对公客户。首先派出我们的架构设计团队，设计团队的组成人员最好是具有丰富项目设计、实施经验的人员（能拉上开发人员更好），了解业务分析、数据分析、架构设计，由他们与业务人员（或者叫需求方）共同组成工作团队。团队搭好后，按照

套路，既然做企业级，就先设计那一“横”——价值链，我们先暂定 A 行价值链由五个环节组成：产品设计、客户营销、运营管理、风险控制、统计分析五大环节，产品设计主要指金融产品上市前的设计过程，包括分析客户需求、开发系统、配置参数等；客户营销则包括客户信息管理、细分客户、销售产品、签订合约等；运营管理一般指需要后台集中处理的业务或者配送服务；风险控制是银行业务的重点领域，通常考虑各类风控模型的设计、风险视图的构建等；统计分析是指各类报表，包括业务报表、分析、监管报表等。这五个环节基本可以构成金融产品从设计到销售再到售后管理的完整过程。从这五部分的定义可以看出，价值链侧重于划定业务环节并分析环节包括的业务能力。由于是虚拟案例，我们只考虑前两个环节的简化分析，不对整个价值链做展开。

存款领域的模型设计

设计了价值链，我们就先开始分析存款领域。先来看下存款领域的“产品设计”环节，我们可以将这个过程起名为“设计上架产品”，定义为一个活动，其大致流程可以如下图：

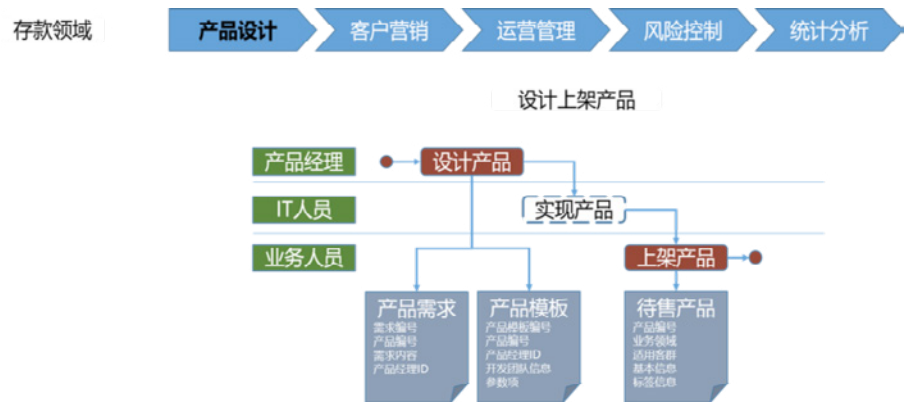


图 1 存款领域产品设计流程图

在这个活动中由三个角色：产品经理、IT 人员、业务人员；三个角色分别有三个任务：设计产品、实现产品、上架产品。产品经理负责分析产品需求，设计并运用产品模板为业务部门整理业务需求，并提交给 IT 人员去开发。这个岗位在不少银行的开发团队中是需求分析岗，但某宇宙行确实具有此类岗位。产品经理设计好产品模板之后交给开发团队，由于实现产品的过程是个复杂的开发过程，因此，在业务模型中可以用一个虚拟的任务代表。开发完成后，业务人员添加关于产品的基本信息、标签信息等，做上架前的最后配置，配置完成后就成为了一个待售产品，可以随

时出售。这个活动中，我们主要关注产品需求、产品模板、待售产品这三个实体，前两个由任务“设计产品”创建，最后一个由任务“上架产品”创建。

之后，进入“客户营销”环节。营销中我们通常一定会遇到“获取新客户”、“维护老客户”、“存款”这三个活动，第一个活动当然是面向银行刚刚挖门子盗洞抢来的新客户，第二个活动则是已有的存量客户信息发生了变动，第三就是营销的目的了——拉存款。这三个活动简要情况如下：

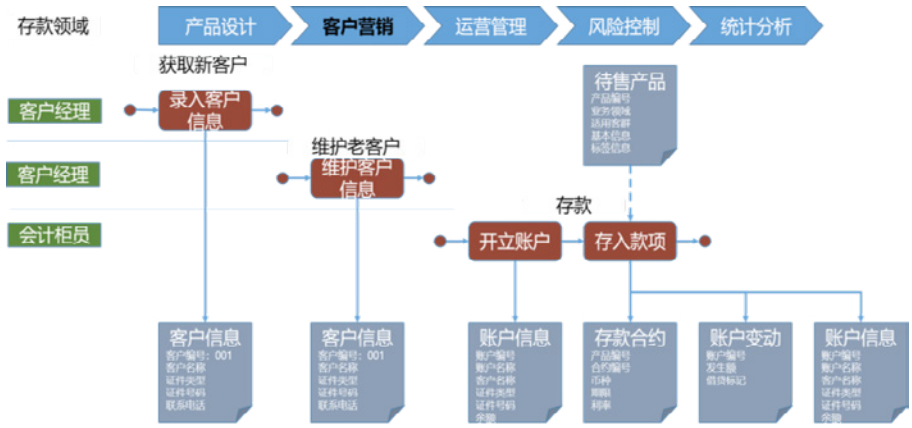


图 2 获取新客户、维护老客户、存款的简要流程

对公客户信息在银行，尤其是规模较大的银行中通常是由管户的客户经理负责录入，一般也无需审核，自己搞定。客户信息发生了变动自然要维护，比如联系信息。这两个活动都可以只包含一个任务，至于是否是分成两个活动，其实取决于建模习惯，当然，合并成一个活动就需要更改活动的定义和范围了，毕竟这两个活动中的任务都是在围绕同一个实体做文章，“录入客户信息”是创建“客户信息”实体，“维护客户信息”是变更“客户信息”实体。客户信息建好以后，就进入业务办理过程。客户到会计柜台去开立对公存款账户，开户是个麻烦的过程，要审一堆证件，不过这里我们略过这些内容，仅关注“开立账户”任务对“账户信息”实体的创建。完成账户开立后，就是存入存款了，其实客户无论是存活期还是存定期，都是跟银行建立了一个“存款合约”，代表了一种债权债务关系，而合约主要记录的要素其实来自我们在上一个环节中创建的“待售产品”。因此，“存入款项”这个任务读取了“待售产品”实体，将其实例化建立了“存款合约”、“账户变动”这两个实体，由于余额的变化，该任务还变更了“账户信息”实体。

以上这两个价值链环节的分析虽然简单，但也包括银行业务的基本过程：设计金融产品、营销客户、销售产品。由于是企业级设计，我们可以

先不急着分析组件结构，可以再分析下贷款领域再做决定。

贷款领域的模型设计

先来看贷款领域的产品设计，其实，从产品设计的抽象流程来看，两者过程上并没有太大差别，从产品模型的角度，是产品结构和参数项的差别；而从开发视角，则是功能上的差异。因此，从业务模型的角度，二者在设计阶段可以共用一套模型：

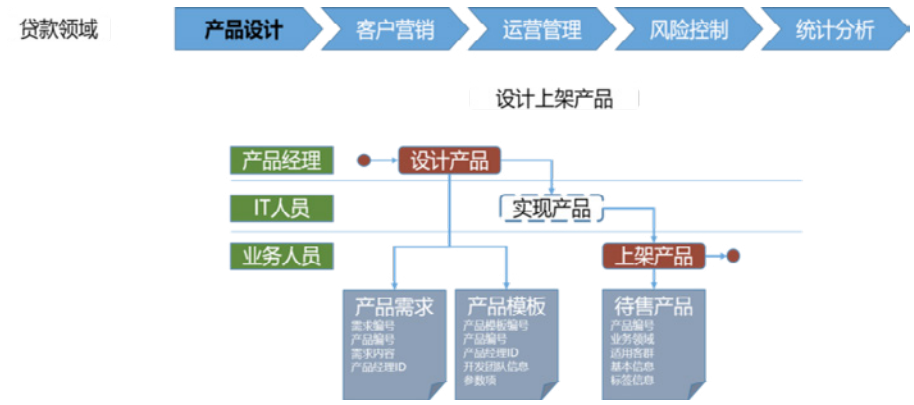


图 3 贷款产品设计流程

这实际上表示，当把开发过程剥离出去时，负责产品设计的组件可以是企业级的。

接下来进入“客户营销”环节，可以理解，“获取新客户”、“维护老客户”也是一样的过程，区别在于销售产品：

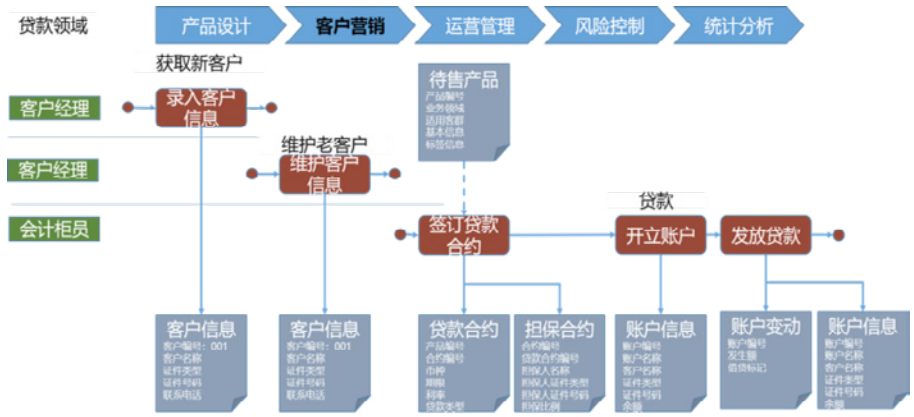


图 4 获取新客户、维护老客户、贷款的简要流程

让客户签订贷款合约是我们的“终极目标”，但是贷款不同关于存款，客户要提供一定的保证，保证通常有抵押、质押、担保等形式，对公客户

中，非常优质的客户也可以采用信用贷款的形式，不提供任何保证。本例我们假定是由其他客户为该客户提供了担保，在贷款合约之外签订了附属合约——担保合约，合约中记录了担保人信息、担保比例等。对公贷款通常是签订贷款合约之后再开立贷款账户，然后才是发放贷款。数据实体就不再一一介绍。

跨领域的标准化

如果不搞企业级，就是竖井式开发，那这样两套业务模型就可以分别使用了，各自构建一个业务系统；但是搞企业级，就需要一起分析了。

对于“产品设计”环节，我们之前已经分析了，可以放在一起，这样就可以有一个“产品管理”主题域，包含“产品需求”、“产品模板”、“待售产品”三个实体；处理这三个实体的是“设计产品”、“上架产品”这两个任务，后者可以聚合成“产品管理”组件。这样我们就根据数据关系的紧密程度将与之相连的任务设计成了组件，这个组件的定义和范围就是对这些任务和实体的概括性描述。按此类推，“客户营销”环节中“客户信息”实体可以构成“客户”主题域，而“录入客户信息”、“维护客户信息”则可以聚合成“客户信息管理”组件。再往下就到了相对复杂的业务部分，这里我们有两个问题要考虑，一是，担保合约中的担保人信息与客户信息非常类似，而且维护需求也类似，而且这种维护可能会不必要地造成担保合约的变化，因此可以考虑将其从担保合约中剥离，但是直接交给客户信息实体处理在概念上又不合适，因此，可以增加一个“角色信息”实体，专门记录客户在银行中不同业务领域可能承担的不同角色，但是这样原来的“客户信息”实体再叫客户信息也不太合适，所以我们可以在抽象程度上上升一格，将其改为“参与人信息”，这个实体应当是一个“客户”在银行有且仅有一个，并且是与业务无关的，其在各种业务中承担的角色由“角色信息”实体记录，这样也有利为“客户”构建更完整的全景视图。一个比较容易理解的比喻就相当于一个初创企业的领袖——董事长兼 CEO，这种情况下，人都是这一个人，但是有两个不同的身份、不同的职责，所以，把他本人定义为“参与人”，而把他担任的两个不同职务定义为“角色”，没准儿哪天他又兼任了 CTO，我们都可以很方便从他本人的信息出发，看到“角色”实体记录了哪些角色，而不用把董事长、CEO、CTO 的个人信息拿过来比较看是不是一个人。当然，这种抽象不是一层不变的，取决于实际需要和系统建设目标。优化后如图 5 所示。

第二个问题是工作流的顺序，存款开户在前，签约在后，贷款则相反。

从实际业务中考虑，首次开户时，开户和存入款项的顺序一定是开户在前，贷款实际上也是开户在前，涉及记账的放款在后；除去首次开户外，都是利用已有账户存钱或放款，并不需要考虑开户问题，因此，不考虑合约时，两个领域间的流程也是可以整合的。

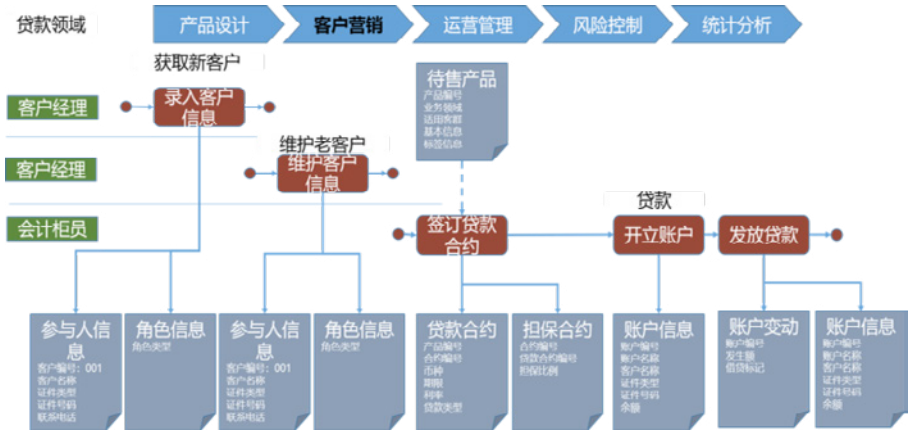


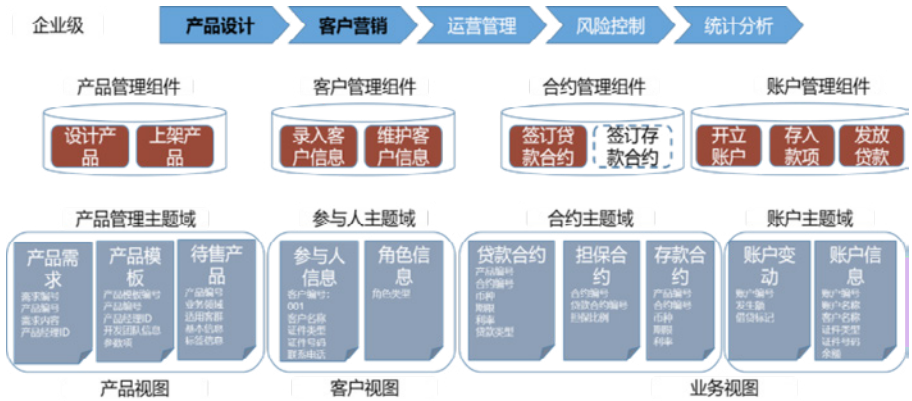
图 5 优化后的获取新客户、维护老客户、贷款流程图

那么引入合约之后呢？其实这就遇到了企业级设计很常见的一类问题，涉及跨领域整合时，流程可以调整吗？这不是指建模能不能改，图上当然很容易，但是实际业务能不能改？愿不愿意改？有的时候大家会觉得，既然都决定做企业级了，那不是有了尚方宝剑了，实际工作还真未必，想想之前举的综合积分的例子，如果搞不定利益分配，企业级也不是那么容易做到。回到当前的例子，我们可以设想，把存款合约部分从“存入款项”任务中分离出来，考虑建立一个签订存款合约的任务，实际执行过程中其实客户还是无感的，毕竟无论开户在前还是签订存款合约在前，客户都是提交了申请之后就在柜台前边等着，是柜员在里边操作；那么对柜员而言呢？如果是账户开立在前，那就意味着不管客户存活期还是存定期，都是先审核开户资料，再选择存款产品。如果签订存款合约在前，那就是客户先选择存款产品，建立合约信息，如果系统发现客户没开户，那就弹出开立账户界面，或者存款签约界面中直接嵌入开立账户功能，如果客户没有账户，展开这项功能；如果有账户，这部分就不展开。也就是说，其实也可以调整流程，那么存款流程就跟贷款流程很类似了，具体的流程图我们就不再画了。

组件设计

根据上述过程，我们在可以把数据实体“贷款合约”、“存款合约”、“担保合约”都放在“合约”主题域下，而与之相关的“签订存款合约”、“签订贷

款合约”任务聚合成“合约管理”组件；数据实体“账户信息”、“账户变动”放在“账户”主题域下，而与之相关的“开立账户”、“存入款项”、“发放贷款”任务可以聚合成“账户管理”组件。业务架构设计如下：



这只是一种设计方式，也可以根据客户实际需要等其他因素变更设计。比如，将“存入款项”和“发放贷款”中的记账动作分离出来，增加一个“记录账务”的任务，这样“存入款项”和“发放贷款”将更加关注流程，也就是“交易”，而“记录账务”会更加关注记账，于是账户管理组件中就会变成“记录账务”、“开立账户”两个任务，而在合约管理组件中填入“存入款项”和“发放贷款”，延长了合约管理的范围；进一步，如果并不关注企业级合约管理，更关注的是产品级的合约管理，则可以将合约管理组件拆分成存款和贷款两个组件，存款组件下放入“签订存款合约”、“存入款项”，贷款组件下放入“签订贷款合约”、“发放贷款”两个任务。可见，根据关注点、设计思路的不同，架构设计也会有变化，并没有绝对的对错之分。此外，上述划分产生的组件，是不是也有“中台”的意思呢？我们可以清楚地看到一个用于今后中台沉降过程的起点。

总结

本节的例子是为了说明问题而虚拟的例子，实际业务场景比例子中复杂的多，但是通过这个简单的模拟，我们可以意识到：

1. 业务架构设计并没有简单的衡量标准；
2. 设计思路和关注点对架构方案有很直接的影响；
3. 架构设计需要迭代和反复，虽然我们不情愿，但是实际操作中是难免的；
4. 基于上一点，架构师经验很重要，可以减少反复，尤其是关键设计的反复。

架构设计是一个不断精炼和确认的过程，上文提到的过程对于业务人员而言并不难理解，因此，需要架构人员、技术人员在设计过程中努力“培养”客户，这是一个深度融合的过程，而且上述设计思路对于业务人员日常分析自己的工作环境、设计工作方案、改进工作流程都有帮助，是一个可以跨出 IT 边界的工作方法，对于这一个过程的投入，是双赢的。

不神秘但很麻烦的业务架构落地过程

将模型转化成方案

业务架构设计不是替代需求分析的

经过之前的努力，我们终于建立了通过业务模型设计的企业级业务架构，建模过程中，已经分析了企业战略、企业价值、组织结构、价值链、业务领域、岗位角色、业务流程、数据等一系列架构元素，通过模型方法对上述元素进行了分析，并形成了组件、主题域的划分。业务架构设计并不是为了停留在纸上，而是为了实践，为了推动开发，尤其是面向复杂系统的企业级开发。但是，模型能够表达的信息其实是有限的，所以，将模型应用于开发之前，我们还是要明确好模型的使用价值和定位。

按照之前的方式做的模型是用来进行高阶设计的，从战略出发，分析企业的目标和为实现目标需要的业务能力。然后，按照业务领域，将能力需求落实到实际的业务流程中，并根据架构设计方法，划分出能力组件，

形成企业的能力视图。这样，就产生了高阶架构。但是这个架构虽然分析的有条有理，却并没有细到可以直接出 IT 设计方案的程度，也就是说，它只能明确企业级架构，不足以替代具体的需求分析。它更关注的是企业视角的整体结构，而非每一个细节。企业级业务架构是一个规划，而非详细的实施图纸，它要求的是 IT 设计继承这套结构，继续向下分解为 IT 的设计元素。

业务架构设计到当前这个层级是可以与实施之间保持一定自由度的，即，不完全受实施方式的约束。这种自由度是好事儿，可以保持架构的稳定和灵活；但也有不利的一面，它与实施的结合离不开架构设计人员与实施人员的密切沟通，它不是一个无需解释就可以直接继承的设计。

业务模型向业务架构方案的转化

明确了这个定位，我们再继续探讨业务架构落地过程的第一步，也就是将模型转化成业务架构设计方案。模型涵盖了很多东西，但是直接看模型并不是一个好选择，特别是做企业级项目，多个项目组同时开工，如果业务架构设计以模型的方式直接扔下去，必然会解读的千奇百怪，这是很自然的事情，“一千个人眼中有一千个哈姆雷特”。所以将模型转化为方案的过程其实是个导读和解释的过程，我们一起理理方案应包含的内容。

IT 人员做系统分析时会关注三个比较重要的东西：边界、结构、关系。首先是边界，边界就是系统的范围，也是我们工作成果的范围，没有边界的项目永远做不成，因为永远也做不完。所以，业务架构方案要阐明业务的边界，这是最高阶的上下文。对于整体价值链和全领域视角的整体介绍可以单独形成文档，作为整体概念向整个企业传导。而领域级阐述的方式则应首先解释业务领域的定义、范围和利益干系人视图，利益干系人视图可以解释清楚所有业务参与方及其诉求，也就是大家对功能的期待。在阐述清楚上述内容之后，要明确业务目标，就是方案要达到的业务效果。如果目标比较宏大，则可以在总目标下分解出子目标或者分阶段的目标，以使 IT 人员能够理解具体的目标点或者实施路径。

解释了项目边界与目标之后，进入结构、内容的编写，也就是阐述活动、任务、实体及组件的设计，以及它们之间关系。对于新建领域，可以从价值链的简介开始，明确高阶的结构。之后，可以选择活动的介绍顺序，按照活动的内在顺序介绍还是按照价值链的顺序介绍其实都可以，并没有严格规则，但是一个企业内部最好采用同一种顺序，这样便于跨领域阅读架构文档。我们上一讲介绍的案例比较简单，如果我们采用它继续做示例，则可以按照价值链的顺序依次介绍各个活动，比如存款领域，依次介绍“设

计上架产品”、“获取新客户”、“维护老客户”、“存款”四个活动，以活动为单位介绍活动的范围，要达到的目的以及活动之间可能的衔接关系；详细介绍参与到活动中的角色，包括其归属的部门；简述每个任务的执行过程，任务间的衔接关系、角色在任务中的权限，每个任务中如何创建、修改数据实体。对活动和任务介绍完后，则对本业务领域涉及的数据实体按照主题域进行 ER 图展示。流程模型和数据模型介绍完之后，介绍本领域涉及的组件，对每个组件的边界、包含的任务和数据实体进行说明。业务架构方案大体包含这些内容。

需要说明的是，从业务领域入手形成业务架构方案，实际上是从应用视角着手，对于较为复杂的业务领域，可能包含多个应用，比如金融市场领域中，外汇、贵金属、利率等业务都可以形成不同的应用，那么一个领域下就可以再细分为多个业务架构方案，而不必都混在一起，把方案搞得太过复杂，部头儿太大，难以阅读。另外，业务领域或者说应用与组件之间是多对多的关系，比如存款领域中的客户管理、账户管理组件跟贷款领域就是共用的。习惯上 IT 通常会按照应用视角组织项目，这样比较容易管理项目目标和处理协作关系，但是项目内部又会考虑按照组件来划分实施团队，这样便于功能开发的分工。

所以，业务架构方案必须具备两种视角的构造能力，既能从领域视角形成组件的协作视图，又能从组件视角形成一类业务能力如何被整个企业运用的分工视图，这是两种不同的文档，做为架构方案来讲都应该支持，否则，负责组件开发的实施团队就难以把握企业级的实施要求，对于粉“中台”的同学来讲，组件文档也是规划中台需要的，但是文档制作是件非常麻烦的事情，所以，最好采用工具支持文档生成。

写方案的过程要下定义、讲范围，好多时候看起来是枯燥的文字工作，甚至有些时候为了区分一些相近的概念，还会玩起“文字游戏”，但是，整理业务架构方案的过程其实是对业务架构设计的再次确认，而非单纯的图纸翻文案、搞 PPT 汇报，一定要把这个过程当作是一次全面的模型质量检验来做。

落地的关键：对模型的解释

人是社会性生物，群体力量远胜过个体，而群体力量的发挥依靠的是明确的分工和有效的沟通。沟通顺畅，不同族群的人也能一起把巴别塔修到天上；而沟通不畅，再伟大的工程也只能半途而废。企业级项目就是一类典型的巴别塔项目，巴别塔要成功就要所有人形成统一语言，而用于描

述企业级业务架构的业务模型，其主要作用之一就是承担统一语言的职能，通过模型传播业务知识。

为什么还需要解释？

经过建模过程，我们将企业目标落实到业务过程中，并将其整理成业务架构设计方案，做为“语言”的模型和做为“书籍”的方案都齐备了，但是知识的传播并不会就这样顺理成章地进行下去，如同练功一样，需要一个不间断的实践过程。

大家可能会觉得，建模过程不就解决了各方对模型的理解了吗？如果项目范围小、参加人数少，这自然不会成为大问题，但是如果在大型企业，数万人、十几万人、甚至几十万人的企业，按照管理学定律，沟通的复杂度是呈几何级数上升的。大型企业中，直接参加项目的人也是非常多的，以笔者所在单位曾经做过的企业级项目而言，集中建模阶段，参加业务建模的人数约 500-600 人，而实际参加开发的人员则达到上万人次。可见，真正做过建模的与参加项目的人数比例大约在 1：20 以上。

集中建模阶段过后，常规负责建模工作的人大约不到 100 人，也就是后续需求产生时，即便只按照项目平均参加人数计算，在同一时间内，建模人员与项目人员的比例也在 1：50 左右，可以说还是非常少的。在这种情况下，如果建模完成后，没有建模人员到项目中去以模型沟通需求、以模型宣讲企业级理念，指望模型自己说话是不大可能的。

现实情况中更可能的是，做为需求提出方的业务部门和做为实施方的项目团队坐到一起，在双方对模型理解不一致或者对模型中有些地方解释不清时，如果得不到业务架构团队或者建模团队的及时支持，自然会倾向于双方直接协商需求，而这时的沟通结果很可能与最初的建模会有差异，如果这种差异在多个项目组中形成、累积，势必会造成业务模型的崩坏，最终结果可能使模型和企业级业务架构失效，而失效之后更麻烦的是，将没有一个统一的视图能够对模型失效产生的“真空”进行有效填补，时间一长，就会形成架构的紊乱，“统一语言”可能会变成“统一误解”。

跟上去，业务架构师

解决上述问题的办法并不难，必须要求做建模的业务架构人员跟进各个项目组的概设过程或者敏捷过程，最好能实地跟进。项目启动时，首先由业务架构人员就业务模型向参加项目的业务和技术人员统一解释项目目标、架构方案、业务需求，由参加项目的业务和技术人员共同确认，因为这时参加项目的人员很可能并没有参加过建模，对模型中的细节、建模中

对原有业务基于标准化进行过的处理并不了解，需要业务架构人员再次统一思想。之后，再由业务人员和技术人员在模型框架下进行需求沟通和功能分解，遵守基于模型产生的项目边界和分工。

如果要形成“统一语言”，业务模型必须对项目具有较强的指导性和约束力，但这也对负责建模的业务架构人员提出了较高要求，要对业务、模型、开发都具备足够的了解，这样的业务架构师必须长期培养才能产生，他们不同于产品经理和项目上的架构师，其对企业需求的宏观把握能力只有通过长期的积累才能达到。要求模型具有约束力并不是模型无论对错都要遵守到底的意思，没人会这么“死心眼”地去做架构，因此，业务架构人员跟进项目的另一个目的就是判断模型的适合性，在建模期间掌握的信息很可能是不足的，而到了实施阶段则会对细节有更多的讨论，随着信息的增加，包括对客观因素的考量，很可能会对原有模型进行适度调整，这时，业务架构人员必须切实履行其职能，承担起架构职责，而非一味要求对模型的遵守。架构是有原则的，但架构也不能失去灵活。建立“统一语言”是为了提升效率，安排业务架构师，是为了让“语言”被恰当使用，而不是造就新的“技术官僚”。因此，凡是要建立企业级业务架构工作团队的，务必考虑清楚对其职责的定位和应当给予的信任与权力，同时，也应当坚持选用具有足够能力的人员。如果不能确定企业会长期坚持这种策略，就不要建立这样的固定组织，否则对从事该项工作的个人而言，会造成极大的机会成本。

此外，还有一点就是应当在整个企业内部不断利用业务培训的机会，用业务模型去解释业务，使各个条线的员工都能对整体的企业架构有所了解，对模型化、结构化的思维方式有所了解。其实每个条线或者领域的业务本身都会经常制作流程图，很多业务培训也是用模型的方式在培训，那么在企业级项目建设的背景下，使用同一种建模方式对业务培训课件和宣讲方式进行统一，会对“统一语言”的建设有很大帮助。

业务架构师也要经常反思自己做过的设计，要“多吃自己的狗粮”；在有业务架构师团队的情况下，经常开会讨论设计得失，集体吃“狗粮”，对于提升业务架构整体和设计思维的一致性也非常有意义。

大家也可能对如此大费周章地建立业务模型、业务架构感到不解，我们不妨回想下业务架构最重要的作用：跨越“数字鸿沟”，尤其是帮助业务人员跨越。现在，科技与业务深度融合已经成为共识，但是融合首先是人与人之间的融合，这种融合非常依赖沟通方式和工具，在科技唱主角的时代背景下，如果业务人员与业务人员、业务人员与技术人员、技术人员与

技术人员这三个层次之间的沟通都能具备更高的效率，那会为企业减少多少沟通成本、带来多大竞争优势！所以，在“统一语言”上投注多大的力量都不为过，而基于模型的沟通，就是打造“统一语言”一种有效方法。

现在很多企业都想仿效阿里的中台战略，通过这种中台方式支持业务的灵活变化，但是，大家是否深入思考过中台的积累过程？是像童年的小猪储蓄罐那样，你丢个硬币、我丢个硬币这样“攒”起来的吗？中台的沉降一定是个不断标准化、不断去伪存真的过程，反复也是一定会有的，而基于能够提供标准化判断依据的工具进行持续的沟通，是设计过程汇中必不可少的。阿里多以 DDD 方法为基础，而要想做更高一级的企业级规划，则需要采用能够从企业级视角着手分析的方法。并不是因为阿里的成功，那些看起来有点儿“老气”的方法就落后了，工具总是“尺有所短寸有所长”，工具的威力，更多还是在使用者自身的运用能力和决心毅力。

企业级业务架构的实现需要不断沟通和调整

基于模型的设计与协调

前面讲过了模型转化为方案和基于模型的沟通，自然就到了基于模型和业务架构进行的企业级 IT 系统开发，相信到这一讲很多人都会有所期待，毕竟从业务到模型只是一个复杂的“预备过程”，开发才是大家关注的重头戏，然而，我不得不以自己六年的企业级项目经验告诉大家，这里既没有什么神秘，也不该有什么神秘。可能让你失望了，但这是事实。为什么很多企业，甚至包括科技企业在内，业务转型或企业级方面做得不理想，其实是因为一个很简单的问题——脱节。一开始把企业级或业务转型搞得“高大上”，请咨询顾问、搞战略项目，但是往往停留在企业高层，缺少向执行层面的贯穿，当然也就更谈不到传导到开发，甚至搞战略时技术人员都很少参与。

搞企业级或者业务转型，实际上是个“全民工程”，不能只做顶层设计，

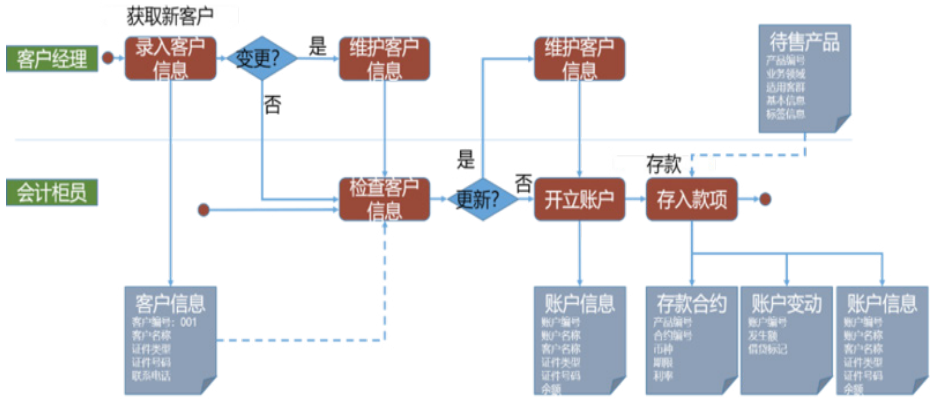
必须逐级分解；不能只是业务自己想转型，而是必须拉上技术人员一起研究。转型不是搞一套新需求，而是要实实在在落地的。而基于模型的企业级业务架构设计方法正是关注战略到落地的“一气呵成”。

基于模型的设计

模型驱动开发（MDD）并不是新鲜事物，搞开发的人，尤其是科班出身的开发人员，上学时可能就知道 MDD 了，特别是 UML 这类耳熟能详的需求分析方法；企业架构也有 30 多年历史了，就算只以 TOGAF 为起点，业务架构也有 20 多年历史。虽然业务架构这几年提的人越来越多，并且大家着手的落地实践也日益增多，不再觉得业务架构，特别是企业级业务架构只是“画大饼”，但是，它只是在以往的工程方法之上累积发展起来的，所以，它更多的是对使用者决心和意志有较高要求，而非有什么点石成金的神奇方法。所有的企业级业务架构都是在痛苦的摸索、怀疑甚至反复中，靠着“革命乐观主义”精神和持之以恒的努力，迭代演化出来的，无论最初的设计看起来多出色、多惊艳，终归也要随着时代发展更迭，何况大部分的企业级业务架构设计最多只能算是给出了一个可以被多数人接受的演化起点而已。

基于业务模型进行项目设计在具体方法上可以根据自身企业的特点和习惯去决定实施工艺，但是其过程本质上是对业务模型的细化。业务模型为了设计企业级业务架构，必然对活动、任务进行适当的抽象，甚至减少一些工作细节，这样的模型可以让项目实施团队了解其标准化的核心，以及功能复用方面的设计思路，但是在具体开发中，实施团队是要了解到细节需求的，这与建模过程有一些矛盾之处。对建模而言，需要了解一定的细节，细节有助于判断抽象的正确性和合理性，但是，建好的模型中却不能表达太多的细节，过多的细节会让抽象的表达变得混乱，而且，以后我们还会讲到，过多的细节也会让模型的应用过程笨拙，使维护变得不可能。所以，到了设计阶段就必须对模型表达的工作流再细化，这也是为什么说业务架构的建模不能替代需求分析的原因。设计阶段的细化允许对原有的工作流进行拆分重构，但是有一个前提，就是新产生的元素必须基于旧元素，并且标明继承关系，这样才能保证设计的连续性。比如，之前虚拟案例中“获取新客户”、“维护老客户”、“存款”这三个活动，在实际设计中，如果存款部分和客户管理部分按照组件边界划分了项目组，存款项目组负责合约管理组件和账户管理组件，客户项目组负责客户管理组件，这种情况下，需求分析或者开发人员可能更愿意用一个较长流程来表示应用视角，这样可以更好地描述两个项目组提供的功能如何在具体的存款业务场景中

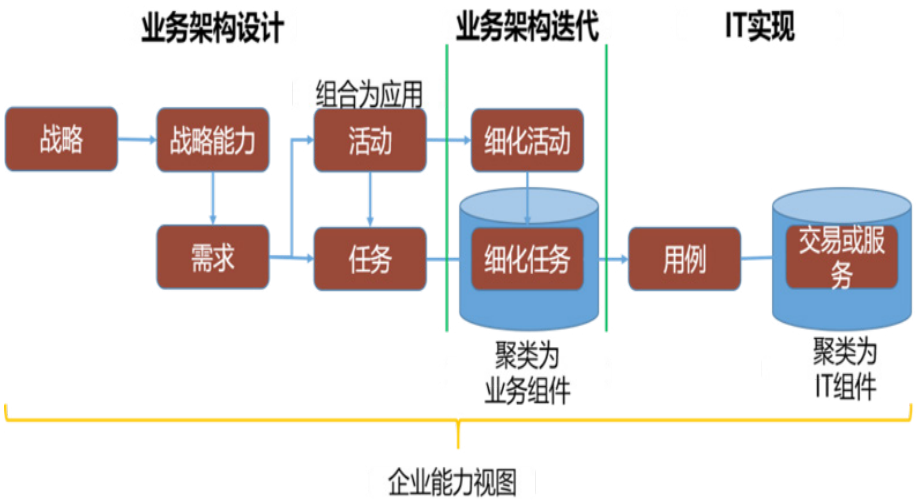
衔接，而重构后的流程很可能如下图：



现实场景中，对公客户经理录入客户信息时很可能对方还只是潜在客户或者客户并没有马上进行开户操作，而是隔了一段时间之后才开户，这时客户信息可能需要更新。如果是客户经理首先了解到需要变更客户信息，则会走上边泳道中流程；如果客户直接去了柜台，柜员则会首先检查客户信息，这时发现需要更新，则可以走下边泳道这个过程，更新之后再开立账户。图中也可以发现，原本没有“检查客户信息”这个任务，在最初的建模过程中，它是可以写在开立账户这个任务中的，但是在细化流程阶段，则可以考虑是不是将其独立出来。这个取决于整体的架构判断，如果其他流程中也涉及这种拆分，确实可以调整企业级业务架构模型，如果不涉及或者很少领域涉及，也可以将其标记为“开立账户”任务的衍生。这种细化可能产生新的数据需求，设计新的数据实体，或者在原有的数据实体下产生根据某种维度细分的子实体。

之后就是根据细化了的、已经达到需求分析标准的业务模型，进行大家已经比较习惯的项目开发，设计用例、设计功能或者服务、分组开发、测试上线。但是企业级项目很重要的一点是要建立各阶段工作之间明确的衔接关系，尤其是设计元素之间的联系，最好能建立“战略 - 战略能力 - 需求 - 活动与任务 - 细化活动与任务 - 用例 - 交易或服务”这样明确的关联关系，形成一个完整的、分层级的企业能力地图，当然，这需要工具的支持。其参考逻辑关系如下图。

企业能力视图不能仅局限于业务侧或者到组件层级，而是应该延伸到最终的实现，业务与技术的深度融合是今后企业发展的大趋势，业务就是技术，技术也是业务，企业未来的作战地图必须是基于业务技术一体化的视图，而非分裂的视角。



基于业务架构的协调

“检查客户信息”这个任务的增加，还引出了企业级项目中很常见的另一项工作——跨项目协调。业务模型最初设计时将任务归类给了各个组件，每个组件都包含了一定数量的任务和数据，从而构成了自己的边界，这个边界可以演化成各个子项目的边界。假定根据业务模型，企业内部已经完成了第一次“争吵”，所有项目的边界在下达项目计划给项目组时，已经暂时接受了，那么，在进行需求分析产生的这个“检查客户信息”任务应该归谁呢？从流程图中，似乎挺明显，它读取客户信息数据做判断，生成判断结果，按照数据聚类的话，应该归负责客户管理组件的项目组实施，尤其是在多个领域都涉及这种拆分时，它的“企业级”属性看起来比较明显。但如果涉及的领域很少呢？比如只有存款领域用它，虽然图中做了简化，数据实体没有增加，但在实际需求中，如果要求它不但生成屏显的判断结果，还记录判断结果并生成推送信息给客户经理呢？如果这个推送信息又被视为运营和销售两个部门间的沟通呢？数据会增加，而这些数据的归属似乎也是模棱两可的。不但任务的“企业级”属性模糊了，连数据归属都有点儿模糊。如果我们再引入一些常见的影响因素，比如，项目组的预算管理比较严格、项目组都面临工期问题、新加任务识别的较晚等等，在这些因素的作用下，项目组对于这些调整是很不愿意接受的，毕竟，企业级项目的一大特点就是，功能谁都能做，谁做了也都可以实现企业级复用，为什么就非得拍给我？这就是对基于模型的企业级架构协调工作的考验，一方面考验模型本身的质量，而更重要的是考验人和制度。人指的当然是业务架构师自身的设计和协调能力，架构师提出的调整方案是否具有足够的

说服力；制度指的就是整个企业给企业级项目或者企业级转型提供的配套管理措施是否到位，项目组对企业级管控的执行是否给力。文中的案例是虚拟的，但在实际工作中，大家难免会遇到类似情况，如果这类问题解决的不好，组件间的不规则“边缘”会越来越多，而企业级项目协调难度之大，甚至会令一些项目组望而却步，为了赶工，宁愿多干活儿、少开会，产生一些连架构层都不清楚的“违章建筑”，最终形成一个到处都有“兔子洞”的企业级。

基于模型的企业级业务架构对解决上述问题有多大帮助呢？个人的经验认为，最大的帮助在于大家可以用同一种“语言”进行“吵架”，在进行项目协调的过程中，业务模型会很自然地吸引“火力”，它是项目设计的源头，向上追溯一定会追溯到模型这里，使大家不再只是“公说公有理，婆说婆有理”，为最终达成一致结论提供了有益的标靶。而模型也是一种很好的、结构化的结论记录形式，比起“一纸文书”的会议纪要或者发个内部电邮更容易让项目组理解和执行。

可能也有人会觉得，形成一个至高无上的强力架构不是更简单吗？但是，从实际工作角度来看，一是模型很难一开始就做到十分完善，很难达到可以照做不问的程度；二是，做企业级不是为了造就新的技术官僚，是为了打破部门边界、提升企业能力，所以让架构自己过于中心化了，反倒不是一个很好的选择。从这里引申开去，做企业级项目的成功标志，我个人认为不是项目上线，而是业务能力被全面地固定在机构中并且能够可视化，让大家都能在一个框架下朝着一个目标努力，不再过分依赖个人，无论是业务还是技术人员，这才是企业级的真谛。

从设计的角度再说说中台，中台其实也不过是个规划和迭代的结果，如果抛开规模导致的技术复杂度来看，只要坚持企业级的方向或者合理的领域规划，经过业务设计和沉降过程，产生中台规划只是个必然结果，每个企业都能找到属于自己特点的中台，而这个“寻找”的过程对企业而言胜过“中台”这个结果本身。从企业级的视角出发，你也许能够找到比中台模式更适合你的组件化结构，而不必沉迷于中台这个概念。

实施中产生的架构调整

前面已经多次提到了业务架构设计是个迭代的过程，会有不断的反复和调整，站在旁观者或者事后回头看的角度，这是挺容易理解的，但是在执行过程中，却是一个需要慎重考虑且非常“烦人”的事情。如果把你摆在业务架构师的位置上，花了好大精力，动用了不少的人力、物力，几经周

折汇报，过五关斩六将地完成了业务架构设计，任务发到了项目组，然后没过多久，以各类合理不合理的理由被提出种种调整，这当然是件很让人掉头发的事情，作为企业级项目，这些调整往往涉及的不是一个项目组，而是一条绳上拴着好几只“蚂蚱”，企业级架构管控经常在做些“按下葫芦浮起瓢”的操作。每每出现这种情况，上上下下都会想为什么架构不能直接把事情拍了（当然很多时候还是站在自己立场上想，为什么不按我说的拍）？业务架构师自己也会觉得，我说的很在理，我是最站在企业级立场的，为什么不干脆让我直接说了算？其实业务架构师，如果具有较大权力确实有助于贯彻整体架构，中心化毕竟是一种高效的执行结构，但是中心化的决策方式其实不符合建设企业级项目的目标，上一讲最后我提到，理想的企业级项目，实现之后应该达到不对个人的依赖，而过于强势的架构师自然会导致这种依赖的产生，所以，无论是从职业的角度还是企业级项目目标的角度，架构师都必须接受、鼓励、坦然应对这种调整要求，当然，这不是要求架构师允许无限制的浪费精力和项目时间，而是首先坚持“以理服人”，再要求对架构设计的贯彻，讨论是让所有项目人员深入理解企业级的过程，这个过程必须，也只能允许反复和挑战。基于实施情况进行调整，对业务架构和模型都是有益的，没有此类反馈的模型，十之八九是没有被真正使用的模型。

应该做的调整

既然允许调整，又不能无限浪费时间，那我们就来梳理下应当调整的情况：

1. 原有架构设计中的疏漏。这一点是架构师们不愿意看到，出现疏漏证明了原有工作的缺失，不过，“知漏就改还是好同志”，别给自己找什么借口，坦然承认，补充设计，调整架构方案，越是虚怀若谷，越是赢得尊重。项目都是有周期的，每个环节都必须有一定的时限，除了首次做企业级转型之外，业务架构设计是非常重要却又不能被分配太多时间的环节，想想对项目的“敏捷”要求，如果让业务架构师自己慢条斯理地搞起细致的业务架构设计，相信所有人都会疯掉。那么，业务架构师就必须在尽可能短的时间内给出覆盖度尽可能完整的架构方案，这是对业务架构师的考验。但是，平心而论，时间越短、信息越少，就越可能出现疏漏，在细化阶段发现问题就很正常，自然调整就好，各方都不必苛求。
2. 出现了更好的设计。上一讲的虚拟案例中就提供了改良的可能性，“检查客户信息”这个任务独立出来，可能会给整个企业级设计带

来一定的改善，方便其他业务领域实现同类需求，这样的改良就可以回补到模型中，由此带来的架构调整是有益的。

3. 对现实妥协的等价方案。架构设计经常不是只有一个可行方案，人们常说，架构师就是在手里永远准备两套方案，并随时准备抛弃其中一套。我个人也经历过这样的事情，原本设计时有两个方案可以选择，在为 C 领域做业务架构设计时，A 领域和 B 领域都有可以采用的会计引擎实现能力，A 领域的业务性质与 C 领域更为接近，于是做架构设计时选择了 A 领域，但是实际推进项目时，负责 A 领域的项目组由于客观条件限制，无法按时实现，只好再转到 B 领域，两个方案基本等价，但是架构和模型上必须要做一定的调整，这是现实条件制约的。对了，不要问我为什么两个领域都会有类似的会计引擎实现能力，这是另一种现实。
4. 架构设计错误。与第一点不同，这是实实在在的错误，是架构师们一直竭力避免的情况，这对架构设计和架构师自身能力的可靠性都是直接的挑战，对此，架构师应当做的就不仅是调整了，更重要的是深入了解错误原因，总结经验，反省和提升自我，由此，也看出经验在架构师综合素质中的重要性，好的架构师都是时间和项目铸就的。但是，如果一个架构师经常出现此类问题，就必须考虑对其进行调整了，或是到项目组重新锻炼，或是不再让其担任架构职责。如果是整个架构师团队经常出现此类问题，那就很可能是工作机制的原因，是不是架构师没有机会深入参与到项目过程中去了解项目实际情况？是不是上一讲提到的“违章建筑”太多，导致架构已经失灵？总之，集体问题与个体问题不同，需要区别对待。

不应该做的调整

以上是几种应当做的调整，还有些情况则不应当调整，大致如下。

1. 明显违反既有规则的调整。比如客户统一视图这类需求，这是典型的跨领域需求，但是又具有一定的领域特性，因为金融行业中客户可能同时在多个领域发生业务，但是每个业务条线从自己领域出发，应用客户视图时不一定要看所有内容，这就相当于在一个统一的数据基础上分领域定制，这样的需求其实由客户管理组件实现，或者由专门负责数据仓库、数据主题的项目组实现都可以，因为客户管理组件掌握客户基本信息但未必掌握业务数据，大型企业中，通常会考虑以数据仓库的方式归集各组件形成的数

据，因此，无论是哪个项目组实现，本质上都是通过数据仓库加工。但是，分工一旦形成，就不要再随意调整，如果既定是由客户管理组件做，特别是客户管理组件已经实现一部分时，就不能允许客户管理组件再以各种理由推脱后续需求，把其他需求交给做数据加工的项目组去做，这样会导致架构的混乱，导致决策原则的不一致，不要轻易推翻已经成为事实的判断原则，要通过事实建立共识。

2. 不必要的重复造轮子。这样例子相信在大家都常见到，一般的重复造轮子我们不谈了，反对的理由大家也都清楚。还有一类重复造轮子则可能是“企业级”给“逼”出来的，这么说有点奇怪，有点给企业级“泼脏水”的嫌疑，但是实际工作中确实会遇到，特别是在企业级转型过程中。企业级项目最大的难度其实是实施过程中的跨项目协调，这时各种利益冲突都会在某种诱导条件下爆发出来，背后的原因即可能是错综复杂的，涉及到遗留系统难以拆解、缺乏关键业务人员、第三方不配合、技术不过关等非常客观甚至是由来已久的因素；也可能是简单到令人无奈的，就是给不想干找一堆理由，这些令人心力交瘁的协调工作，也可能会“逼”得大家宁可重复造轮子，也不愿通过协调解决问题。这种情况，架构管控上也要坚决制止，因为转型之路虽艰难，但是开倒车会导致更大的艰难，会让之前的努力都付之流水。

经过多年的企业级实践，有时我也会问自己，企业级这么费劲儿的事情，真的物有所值吗？坦白地告诉你，无法计算。所谓降低开发成本，这个是常挂在嘴边，但不一定是企业级要去解决的核心问题，技术五到八年差不多就换代了，节省的成本就算能算出来可以挺几年？你还得扣除转型成本。所谓系统互联互通、数据共享，其实这个是通过对关键问题的企业级处理，也就是点上的企业级就可以解决，比如数据标准化加数据仓库。所谓灵活响应、快速上线，这个更多是开发方式、Devops 等关注的内容，毕竟领域之所以为领域，还是因为领域之间有差别，可以因共享而提升的反应速度可能是有限的。所以，企业级价值在哪里？我个人认为，花费这么大力气搞企业级最重要的是转变企业文化，打破部门边界，让企业融为一体，让业务与技术融为一体，这种一体化带来的企业内部的变化，带来的高效协作才是最有价值的，是企业未来长期竞争力的基础。衡量企业级项目成功的标志并非单指一个系统的实现，文化没有转变、思维没有转变，是不会真的诞生这样一个系统的，即便交给企业一个这样的系统，也可能

会被改回去。做企业级，难点不在技术，企业级真正解决的是业务问题、组织问题、思想问题，是超越技术之上的建立一个什么样的企业的问题。企业级业务系统是给具备此类文化的企业使用的配套工具，也是给不具备此类文化的企业提供的一个转型过程，至于结果，要由时间、感受和市场去检验，而非标准。

不同的系统也会反映不同的企业文化，这点如同之前对阿里中台背后的分析，希望朋友们能够多去思考和观察你所能接触到的系统和使用这些系统的企业。

业务架构设计“笨重”，它能跟敏捷沾边吗？

传说中和现实中的双模开发

“天下武功唯快不破”。电影《功夫》中火云邪神这句台词可谓深得互联网时代竞争的要旨，也不乏业内人士常常感叹，一个产品的成功可能只是领先对手一周甚至两三天的上市时间，产品创新速度、市场响应速度越来越被企业重视，但这两个指标似乎都是大型企业，特别是传统行业中大型企业的弱项。所以，不少人都致力于教大象跳舞，不断有关于软件过程、项目管理的概念应运而生。比如，Gartner 在 2014 年提出了“双模开发”，敏态加稳态，可预见性的业务使用传统瀑布式开发，也就是稳态；探索性业务使用敏捷开发，也就是敏态。

2015 年，Gartner 对全球 2800 多位 CIO 进行了一次调研，结果显示：38% 的企业已经实施了双模 IT，26% 的企业将在未来 3 年内实施双模 IT，只有 13% 的企业不会实施双模 IT，另外还有 23% 的企业则不确定。

虽然没有找到近期的数字，但是按照 Gartner 的说法“双模开发”少说也占据了半壁江山。我无意在这里去质疑或者争论这种统计，但是，显然这些 CIO 们口中的敏捷开发未必是“敏捷宣言”所提到的敏捷开发，更多的还是应对特殊需求时，打破常规的“快速”开发，省略了通常的管理环节，组个小分队，快速上线。这种剧情相信大家都经常遇到，市场着急、领导着急，大笔一挥，桌子一拍，下月上线。当然，对互联网科技公司而言，可能就是下周上线。所以，大家都经常在稳态和敏态，也就是按部就班和大干快上中做来回切换。

无论是符合敏捷开发定义的敏捷，还是特事特办的敏捷，都意味着忽略既有流程中的一些环节，压缩周期，快速实现目标，那么，相比之下，经过业务架构设计、应用业务模型驱动的开发是不是显得有些笨重，是否与敏态不兼容？是否在敏捷过程中应该被省去？要回答这些问题，我们还是从符合敏捷开发定义的敏捷和特事特办的敏捷这两个角度分别看看吧。

跟正宗的敏捷比比

说到正宗的敏捷开发，自然要说到“敏捷宣言”中提出的四个核心价值：个体和互动优于流程和工具、工作的软件优于详尽的文档、客户合作优于合同谈判、响应变化优于遵循计划。最直接的冲突莫过于第二项，敏捷开发素来以“不重视”文档闻名，但其实敏捷开发并非不要文档，而是不要那么详尽的文档，过于详尽的文档会消耗大量不必要的精力，而用途又非常有限。对于开发人员而言，在软件维护方面，高质量的需求文档远不如整洁的代码加上详尽的注释会更让人清楚软件的结构，所以敏捷开发在这方面做了大胆的改变。但是，敏捷开发也还是要求有一份恰当的概括性文档作为项目的总体目标，而不是什么都没有就甩开膀子干。如此，业务模型这套方法要想与敏捷开发融合起来，自身也必须具备一定的简洁性。

从之前的介绍来看，首次企业级转型肯定不适合这么干，而且企业级转型需要深思熟虑，也不是应该去“敏捷”完成的事情。一旦转型结束，具备了企业级架构之后，就是如何快速应用架构工具的问题了。我之前提到过，业务模型不应该承载太多的业务细节，要保持适当的抽象度，否则不利于对企业级的描述，至于细节要描述到什么程度，不同的企业可能会有不同的要求，大的原则是，可以基本解释清楚任务对数据实体的创建和变更，以便划分任务以及组件的边界。因此，模型本身具备快速应用的潜质。而且，模型本就是一张“作战地图”，通过模型也可以更快摸清项目范围、涉及的组件和团队以及潜在影响。敏捷并不是不顾一切的敏捷、更不是牺

性企业级架构一致性的敏捷，否则，敏捷项目就成了为短期利益而有意忽视长期影响的盲目行为。无论多着急，不看看地图就冲向战场，都是危险行为。不过话说回来，模型分析和调整需要一定的过程，企业级这类庞大的模型体系也需要工具来支持，否则真的谈不上快。

这似乎跟第一项核心价值也有冲突。其实不然，敏捷开发的速度来自于节省不必要的步骤和提高协作的效率，所以“个体和互动”才“优于流程和工具”，注重面对面直接交流，减少分歧。这就要求业务架构师必须参加敏捷项目，在项目中快速完成架构分析，把控项目引起的架构调整，而不能在项目之外等着按“流程”操作。敏捷开发团队在要在业务架构师的直接参与下，在项目一开始就根据需求描述快速使用模型工具澄清项目范围，列出对架构的改变事项，这其实也是对企业“统一语言”构建效果的检验。在项目期间，业务架构师则可以根据项目的具体情况完成对模型的详细调整，实现并行作业。所以，应用业务架构和业务模型驱动的开发过程，可以转变为敏捷过程，并为敏捷过程提供更好的分析依据。

跟“特事特办”论论

再说到特事特办的敏捷。这种敏捷其本质就是“临时事项”，因事而立，事过则废，这种方式其实对企业整体的架构管理带有一定的破坏性，往往会直接要求一些违反“架构”整体安排的改动。而事后也经常无人对此负责，做了就做了，然后交给运维团队去维护。有些真有长期价值的系统可能会持续使用，还好些，但是做了之后就再无人问津，半死不活地躺在哪里的系统也屡见不鲜。

与上文提到的“真”敏捷不同，“真”敏捷是软件过程的差异，并不意味着要去违反企业级，可以与企业级很好地融合；但是有些特事特办的敏捷确实有不管不顾的意思，上线是唯一目标，手段可以不问。这种情况就很棘手了，因为它超出了业务架构师的控制能力之外，是对企业文化的考验，是对企业维护其企业级架构决心的考验。

不过我们还是坚持下具体问题具体分析，别一概而论地扣大帽子。

首先，有些项目确实形势逼人、速度第一，不上线毋宁死，这种要举全局之力搏一隅的情况不支持也不行，但是业务架构师要切实参与到项目中，不但要给出当时的架构设计建议，也要尽快给出影响分析和事后的重构方案，如果成本允许，尽可能要在企业“搏命”之后，回归正途，避免架构遭到破坏，如果成本不允许，那这块架构“飞地”也要标识清楚，尽可能让它成为以后架构设计可以利用的“轮子”而不是会踩的“坑”。

其次，对于不具备上述价值的“特事特办”，架构师应该从业务架构的角度申明其立场，给出认真的分析意见，这是架构师自身的职业操守，而能否容忍架构师的意见则是企业文化的真实体现了。如同会议表决中的“保留意见”一样，如果架构师真的反对项目的设计和实现方式，那企业就必须在项目文档中保留其分析意见，这倒不是非要“立贴为证”，而是在未来真的需要调整时，作为参考意见使用，同时也供所有架构师进行检验和学习。这是对企业级的一种警醒，但并不是业务架构师工作中的常态，也不应是其工作追求的目标，架构师不是以参倒了宰相为荣的“言官”，要有原则但也不要让自己孤立，更不能变成言必称企业级、处处拿大帽子压人的“反对派”，架构师的宗旨是解决问题，而不是让自己变成问题。虽然跑了点儿题，聊了下架构师的素养，但这是“特事特办”牵出来的，足见这种事的麻烦，它超越了正常工作的范畴，带有点儿其他色彩，需要架构师谨慎对待。

综上，无论是否企业有意识地推行过“双模开发”，大家也总是忙碌在一般流程和“特事特办”之中；无论是否建立了敏捷开发体制，也总有项目必须要快上。大型企业中，企业级体制建立不易，打破却很容易，不要为了“快”而牺牲企业级。管理大企业开发就像指挥大兵团作战，既有担当主力、要稳扎稳打的方阵部队，也有要处理特殊任务的游骑兵，多兵种之间的有序协作是克敌制胜的关键，这就离不开有序的架构管理，通过业务架构方法、应用业务模型驱动开发本身与敏捷并不矛盾，敏捷可以是、也应该是一个有序架构体系下的敏捷，而不是“叫嚣乎东西、隳突乎南北”的乱闯，好像有句格言，“捷径是迷路的最快方法”。所以，一定要有效利用业务模型这个“作战地图”，培养好业务架构师这个“领航员”。

说到这里，朋友们也不妨想一下，仅凭中台模式就能很好地解决敏捷问题吗？其实，这应该还是一个更大范围的企业管理或者企业文化要去解决的问题，它并非一个单纯的开发模式或者技术架构问题，之前我也提到过，企业级业务架构设计是希望实现企业整体的联动、文化的转型，而不是仅仅建立一套系统，实际上，阿里的中台背后支撑其运作的也是阿里的企业文化。没有敏捷起来的往往不单纯是工具，而是人和人所在的环境。

企业级业务架构设计的“五难”

我们简单回顾一下，以业务架构的发展过程和对业务模型基本介绍作为开始，结合笔者的工作经验和自身一些不成熟的理解，在业务架构设计方面陆续讲到了企业战略解读、企业组织结构的影响、如何划分业务领域和流程、与流程建模配套的数据建模、企业级的模型标准化，并设计了一个虚拟的案例；在业务架构驱动开发方面，讲到了如何将业务架构设计转化为业务架构方案、业务架构师如何基于模型与项目开发团队沟通、项目开发团队如何基于模型开展设计、项目团队之间的协调、模型基于实施的调整和企业级项目完成后如何继续建立持久的企业级工作机制，之后还分析了与敏捷开发的关系。这已经算得上是一个完整的历程了，包括了企业级转型的规划、设计、实施及建成后的应用机制。企业级建设是个很艰难的过程，经历前面的介绍之后，我们不妨聊一聊企业级的实施之难，也给各位已经投入或者即将投入企业级转型的同仁们提供一点儿思路上的参考，或者就算帮大家发发牢骚、吐吐槽吧。

企业级是一个美好而艰难的愿景，了解“领域驱动设计（DDD）”的朋友可能会知道，DDD 是不对企业级抱太大希望的，认为企业级的建设路径只能是一个领域一个领域的不断尝试融合，换句话说，DDD 不认为企业级真的可以通过自顶向下的规划产生，只能是自底向上的生长；科技公司中如果说企业级的代表，可能莫过于阿里的“大中台”模式，但这个模式是“演化”出来的，有兴趣的朋友可以读读相关书籍，但阿里毕竟有个好处，其业务范围总体而言是垂直的电商领域，当然，这并不是说电商业务很简单、很单一，而是领域中还是有一定的公共部分可以抽离的，这个观点也得到一些阿里同学的认同。但说到金融，学过或者做过金融的同学可能有体会，这是一个很不“专业”的专业，里边东西五花八门，看似大家都在同一个领域，实际上却是“各怀鬼胎”，传统的存贷款跟票据业务其实没啥直接关系，票据跟金融市场沾边，但是关系也不深，代收代付不过是个约定转账，现金管理是个大杂烩，托管是另外一个领域，后来还多出个养老金，这几年新兴的资管、理财完全可以自成一体，不然支付宝、余额宝也不会发展那么迅速。说到底，大家的共性无非是客户都是同一群客户，围绕客户共建了一个账户体系，业务虽然差别很大，但是多数都得记账。也就是说，如果自顶向下看，客户和账务是应该企业级的，而其他部分，严谨地说，真就像 DDD 主张的那样，得一个领域一个领域去研究，这也是建模和标准化的难点。所以，企业级建设的难度跟企业所在行业的特点有直接关系，没有一个通用的企业级业务模型可以随便套，甚至一个行业内，企业跟企业之间内部特点的差别，也会决定企业级建设路径和结果的不同。

这算得上是企业级的第一难吧，也即，很难通过简单复制的方式快速切换到企业级。别人的经验，无论成败，对你而言都是个借鉴，自己的路还要自己走，但是实践中找个“老司机”带带路，找个做过企业级开发的科技公司帮助做转型还是比较稳的。

第二难，企业级多数情况下不是个技术问题。这是非常让技术人员为难的，因为这根本不在他们的能力范围之内。前面提到过综合积分的事情，这只是众多要协调的事例中的一个，如果是一个业务种类繁多、部门庞杂、等级森严的传统企业，建企业级不次于一场“内战“，一场对部门边界、协同关系的重新界定。你可能会觉得，真有那么可怕吗？如果没有那么可怕，我倒宁愿相信是以下两种情况中的一种：一是企业之前分工非常合理，无可挑剔；二是大家都没去触动真正要解决的问题，一团和气的结束了。前者基本是不可能的，而后者是非常可能的。如果真的是下了决心要做，对于一个传统企业而言，要改的东西实在太多了，而引入新方法、新思维产

生的冲击也需要大量的时间去消化，是一个彻头彻尾的大转身。这其中，需要业务上做的调整不亚于技术上的调整，而对企业文化的调整尤为重要，现代管理学之父彼得·德鲁克曾说过这样一句名言："文化能把战略当午餐吃掉（Culture eats strategy for lunch）"，这的确是个难题。

第三难，应对理想与现实的落差。做项目很重要的一项工作是管理好用户的预期，企业级建设也是如此。因为要耗费大量人力物力，所以，企业级项目启动之前，往往会将蓝图描绘的太过美好，但是建设周期的漫长、建设过程的曲折，以及中间不断对现实做的一些妥协和折衷，会让很多泡沫被挤掉，这会让实现的结果看起来很"骨感"，之前文章中我也提到过，有些目标其实不是企业级要去解决的问题，有些成果也不是非得记在企业级的功劳簿上，甚至做企业级的成本和收益都难以直接计算。这有点儿像从单体应用到SOA、微服务的演变，看起来零件化了，灵活性上升了，但通信、维护也变复杂了，企业级效果的积极方面可能也要随着时间才能逐渐显现。这会产生对企业级的怀疑，尤其是在项目刚结束的一段时间内，大家都期盼着出现跟以往迥然不同的"大转变"，但是，往往需要"让子弹飞一会儿"。所以，要管理好企业的预期，不需要给企业级项目戴上太多的"高帽"，而忽视了真正该戴的"高帽"——完成一次企业文化的建设，实现整体转型，如果这个目标没实现，那才是真正该失望的，不要只用系统去检验企业级。

第四难，架构的权责定位。在组织中，一件事情能做好，其前提就是做事的人权责匹配，无论是临时事项还是长期事项，否则，成功就是侥幸而不可复制的。企业级转型期间，作为临时性项目组织，架构可以有较大权力去保证项目落地，但是转型期结束，转入常态开发时，架构如何定位呢？我之前给出的机制是一种解决办法，毕竟架构就是架构，不是企业的管理者。但是，架构定位的困难在于，权力太小，不足以维护企业级，甚至让企业级随着时间的流逝而"名存实亡"；权力过大，又会发展成新的部门化组织，一旦开始以架构"卫道士"自居，就会导致对架构创新的阻碍。这种说法可能科技公司不太容易理解，但是对传统大型企业而言，是很正常的，因为这些企业中本就有强烈的"官本位"思想。企业级建设实际上是要让这些习惯了业务管理的企业去正视技术，定位好自身的科技基因，如何对科技中很重要的一股力量——架构师（既包括业务架构师也包括其他架构师）做出合理定位，就成了对企业的一个大考。

第五难，志贵有恒。企业级的长期坚持是件难事儿，大家可能会觉得，业务架构有了、模型有了、地图有了、机制有了，还会很难吗？当然会的，爱美之心，人皆有之，都知道体型好又漂亮又健康，花钱、花时间减肥的

大有人在，但是真正坚持到底、不反弹的有多少？企业和个人都是一样的道理，水会自然流向阻力最小的地方，所以，企业级的放弃和崩坏，未必是把架构组织撤销、机制停掉这么激烈的动作，而是各种“畏难情绪”、“客观原因”导致的缓慢的无序，跟减肥、忌烟失败差不多。

说了一堆难处，读者也能体会到，传统企业，尤其是大型企业谈企业级，跟那些互联网科技公司是不大相同的。对于后者，虽然也有管理方面的因素，但更多还是技术规划、技术栈建设的问题；而对于前者，自始至终，非技术因素的作用与技术因素相比，至少是等量齐观的。但是时代已经进入了数字化时代，正如某次交流会上，一位嘉宾豪言，“未来已来，你爱来不来”。随着国家开放程度的不断提高，民营领域创新能力的不断提升，大型传统企业已经进入了被动的数字化转型之中，是否会迎面走上、顺利走通企业级转型这条举步维艰之路，我们拭目以待吧。

小结：对业务架构的再次思考

虽然做了六年的企业级业务架构，但是总觉得业务架构不是个好讲的东西，业务架构离不开业务模型，所以讲它就会搬出一堆枯燥的模型来，甚至会让人觉得业务架构就是建模。但建模只是个手段，建模的目的是把现象总结成模式，再从模式中找到结构，将业务上看到的结构传递给技术，如果二者能够基于同一结构思考，沟通上将产生最大的便利，这就是通用语言的基础，其实说通用语言，还不如说通用结构，因为说语言，经常会把人带到语法层面，纠结于规则、概念、标准之类似是而非的东西。所以，我总结建模的原则无非是把握整体、穿透现象、保证落地，建模即不能死守规则、冥顽不化，也不能脑洞大开、信马由缰，必须从一开始就关注如何落地。建模不是建个自圆其说的乌托邦，而是传给后续过程的设计图纸。业务建模可以有前瞻性，但是所谓的前瞻性是能够看清分阶段实施路径的前瞻性。

业务架构是不断演进和迭代的，它有生命力，可以成长，如果架构管理工具本身支持历史记录和模式比对，你也可以看到企业架构的演进历史，而不是只看得到现在，只能听别人讲讲过去，过去是可以看见的。这种可视化的历史是一种宝贵的学习资源，人是从历史中学习未来的，毕竟有很多行业还是需要积淀的。

但是，业务架构的形成过程的确是在一种看起来科学的方法论下，不完全科学地操作的，这点我曾经也很纠结，后来软件架构的书看多了，再加上到项目中的观察，也逐渐释然了。软件架构其实很羡慕建筑架构，觉

得建筑架构有力学基础做支持，有很多可以计算的东西，但是软件架构却没多少能算出来的。在开源思想时兴之前，行业内部交流分享较差，都比较愿意看别人的架构，而不想亮出自己的，很多研究者都抱怨这个非常需要标准的行业反倒是很隔离的。开源为架构和软件带来新的成长方式，共享让思维发展更快、普及更快，但是，软件架构本身却只是增加了大量的案例，依旧难以标准化，哪怕是同一个行业的企业，给这家做的软件也不一定能直接搬到另一家去，很多商用化了的系统软件也还是离不开个性的本地化改造过程。云计算带来的 SaaS 虽然让软件应用省去了许多部署过程，但是，依然难以改变这个行业个性化程度严重的局面。软件架构尚且如此，业务架构也就不需要纠结了。

业务架构设计可以很快，也可能很慢。快无非是两种情况，一是架构师自身炉火纯青、天生慧眼，设计能力超强；二是原有业务模型已经很清晰，可以快速分析业务变化，形成架构设计，我们可以追求的是第二种，这也意味这首次建模，尤其是首次建设企业级模型，不要过快，对模型设计方法、业务流程分析、标准化过程，都要细致点儿，基本功扎实了，才有后边的“敏捷”。企业级转型没有轻松的，不少企业是把转型仅当成一个项目，而忽视了对自身的调整。一个普通士兵变成一个特种战士，不是因为给了他一身价值 10 万的装备，而是经过了地狱般的训练。上至最高管理者，下至普通员工，人的思维不转变，哪来的企业转变呢？

为了推动企业真正的数字化转型，业务架构设计人员永远不要忘记，业务架构最重要的职责不是传递需求，而是藉由自身的努力，推动业务和技术的深度融合，桥梁作用才是业务架构最重要的职责，如果不能实现这一目标，也就不能真正实现一个快速响应内外部变化的企业级业务系统。

其实中台并非万能，客观地讲，一个优秀的架构设计人员是不会“迷信”于任何一种架构设计方式的，也不会执着甚至偏执于方法间的争论，没有哪种设计方式是完美无缺的，软件行业没有“银弹”，任何一种方法都需要坚持与灵活的结合，都需要通过长期的实践不断总结和改良，如果一个方法没有被坚持数年以上，可能连入门都谈不上吧。我对中台认识更多还只能算个一般观察者，论述中难免有失，感谢读者朋友们能够宽容地看我一路“叨叨”下来。

版权声明

InfoQ 中文站出品

出版 : 北京极客邦科技有限公司
北京市朝阳区来广营叶青大厦北园 5 层

欢迎共同参与 InfoQ 中文站的内容建设工作，包括原创投稿和翻译，请联系 editors@geekbang.com。
网址 : www.infoq.cn

扫码关注InfoQ公众号

Geekbang. 极客邦科技 | InfoQ