# 决胜低代码

## 作者：老 K

作者介绍：老 K，“技术领导力”公众号作者，文出过畅销书、武做过 CTO，被生活逼得一身才华。

支持媒体：“技术领导力”、“BAT 架构”公众号：

技术领导力

BAT 架构

目录



# 1、终于有人把“低代码”说清楚了！

今年 2 月 23 日，外国低代码平台提供商 Creatio 宣布获得 6800 万美元融资；2 月 22 日，国内 SaaS 软件厂商黑湖智造宣布完成 C 轮近 5 亿元人民币融资。国内外的低代码开发平台备受投资方青睐，被各大软件和互联网媒体捧上“C 位”。老 K 也写过几篇评论低代码开发平台的文章，想必你都见闻不少。

那么，你真的懂低代码了吗？抛开冗长晦涩的专业术语，我想写一篇最接地气的文章，给大家说清楚“低代码”是怎么一回事儿。

## *01*

### “低代码”的起源和走过的路

低代码的故事要从上世纪 80 年代说起，当时计算机科学理论已逐步发展成熟，不少高级程序设计语言都逐渐开发完善。这时，编程界推出了“结构化语言”，即以功能指令为单位，把相应的代码封装好。当程序员要系统运行某个功能时，只需发出指令，计算机就知道要运行对应的代码。

到了 2000 年，“VPL”（可视化编程语言）出现了。顾名思义，就是在第四代编程语言的基础上，把系统运行的过程以更视觉化方式呈现，例如图标、表格、图表等形态。

随着高级编程语言不断发展成熟，以及国内外计算机人才的培养规模逐渐扩大，2010-2015 年称得上是传统软件和 SaaS 软件兴起的时代，市场规模稳步增长。就是在这一时期，编程人员承接了许多软件开发项目。他们发现：软件的功能大同小异，重复度很高，导致很大部分的软件开发成本都浪费在重复的功能编程上。

而 Forrester，一家国际知名的技术和市场调研公司，敏锐地发现了这一问题，并在 2014 年首次提出低代码和零代码的概念：只需用很少甚至几乎不需要代码就可以快速开发出系统，并可以将其快速配置和部署的一种技术和工具。随后在 2018 年，Gartner 提出 aPaaS（应用平台即服务）和 iPaaS（集成平台即服务）的概念。


图自：CSDN

在这两个概念出现并逐渐传播的时间里，国外软件厂商就陆续发布出低代码或零代码开发平台，探索并证明了这类产品成功的可能性。基于外国的成功初探，中国市场也掀起了“低代码/零代码”的热潮，并在近两年逐步形成完整的产品生态体系。


图自：海比研究院

## 02

### “低代码”为何而生？

低代码开发平台至今已发展得较为成熟，现在我们站在较高的“上帝视角”，回顾“低代码”诞生的合理性。其实，低代码平台除了击破重复编程的高成本痛点之外，还解决了两大难点：沟通隔阂和效率问题。

1、需求方与技术方之间的认知和沟通隔阂

传统的软件定制开发环节中，需求方往往会提一大堆业务流程、数据收录、界面设计等要求。经验丰富的技术员能理解甲方的业务流程，用正确的逻辑完成开发。而欠缺业务经验的技术员则照着“单子”来开发，这种粗暴

的方式往往也埋下了不少系统逻辑不自洽、出 bug、流程不通等隐患。技术方不懂业务怎么运转，需求方不懂系统语言和逻辑，双方存在认知和沟通隔阂。

低代码开发平台凭着自身可视化、易理解的配置功能，让业务人员更清楚如何用上面的功能来开发应用；开发人员也能借助平台的界面、功能使用指南，更轻松地让业务人员理解应用实施逻辑。现在市面上绝大多数的低代码平台也在主张由业务人员自行实施应用，背后也是这个道理。

2、友好的操作界面提高应用实施、漏洞排查和修复效率

也是因为可视化、交互化、简洁的平台界面，应用开发者能更高效地实施开发，不用对着满满一屏幕的黑底白码埋头苦干。同样地，排查及修复 bug 的效率也因信息简化了而更容易提高效率。

以轻流产品为例，展示数据表字段配置界面

## 03

### “低代码”的技术特点

谈完低代码是为降低软件开发的成本、沟通和实施效率而生，我们来看看它有哪些技术特点。

1、两种模式：基于表单或引擎驱动 以及 基于 aPaaS 平台

目前大部分低代码开发平台都属于下述模式的其中一类。

| | 基于表单或引擎驱动 | 基于 aPaaS 平台 |
|---|---|---|
| 定义 | 通过建立多张表单，使用流程串联，定义报表输出方式，构建表单类轻应用 | 以应用开发平台为核心，承载各种开发工具和复杂技术手段，并将其可视化、低代码化来使用 |
| 优点 | • 功能简单易用易学<br>• 具备基础的自动化流程运转能力<br>• 采购成本较低 | • 功能更多元<br>• 应用细节的颗粒度更高<br>• 应用开发的灵活度更高<br>• 开发技术壁垒高<br>• 场景局限性弱，满足大中小客户的需求<br>• 基本可实现复杂的系统开发和对接 |
| 缺点 | • 开发技术壁垒低，缺乏技术竞争力<br>• 难以实现复杂的系统对接和功能配置<br>• 场景局限性强，主要服务中小客户 | • 对应用开发者有技术能力要求<br>• 采购和实施的各项成本较高 |
| 更 适 合 | 表单类应用，如：人事行政、资料归档、 | 复杂场景应用，如：ERP、生产全流程管理、 |

| | | |
|---|---|---|
| 的应用场景 | OA 审批、客户管理等 | CRM、物联网等 |
| 典型代表产品 | 魔方网表、云表、活字格 | ClickPaaS、氚云、宜搭 |

大家可能还感觉不到有啥区别，让我来举个对比例子：

表单驱动模式的低代码平台主要以表单的形式运转业务流程；而 aPaaS 模式能借助应用平台打造一个立体空间，让不同部门的不同业务线彼此交叉贯通，还可以对接外部的系统。

2、颠覆传统：“低代码”和传统企业管理系统架构的差异

低代码开发平台除了自身模式不同，和传统企业系统管理相比，在系统结构和管理理念上也有颠覆式差异。

“低代码”将多个“系统烟囱”归整为一个集大成者，更灵活敏捷地创建中台架构。

传统的企业系统中，每个部门有不同的系统需求，于是各自采购自己的系统。但这些系统彼此孤立，独立运作，导致企业采购的软件系统冗杂。低代码平台则让绝大部分部门的业务系统都能在一个平台里搭建，彼此联系，打破信息系统孤岛，同时降本增效，提升内部生产力。

“低代码”重塑业务部和技术部的分工定位，为业务部赋予系统定制化的能力和自由。

重塑业务和技术的分工定位，主要在于宏观到微观的企业系统管理运维上。技术部负责统筹企业在低代码开发平台上的整体架构分布，维护系统运维的稳定性和安全性，修复漏洞。而业务部则有更多自由，利用“低代码”自主开发出业务所需的管理系统，并实现跨部门应用交互。另外，当重新定义了二者的分工后，企业技术部的价值才能从修电脑、装 wifi、买服务器这些琐事中进阶，为公司数字化管理做建设性实事。

## 04

### “低代码”能否继续干得漂亮？

1、势头：稳定增长

2021 年初，海外研究机构 Infolob 表示，低代码应用平台保持着 40%的年复合增长率，预计到 2022 年，低代码应用程序市场总规模达 212 亿美元。Gartner 预测 2024 年应用软件开发活动中，65%将通过低代码方式完成；75%的大型企业将用至少四种低代码开发工具开发应用。

“低代码”在国外发展势头强劲，无论是市场培育还是商业模式都渐趋成熟。在国内，它的表现也毫不逊色。2020 年企业数字化浪潮让低代码市场规模迅速扩展，也因此鼓动了不少软件厂商转型做“低代码”。海比研究院预测，2021 年至 2025 年，中国低代码市场将保持规模扩张的良好态势。


图自：海比研究院

2、机遇：物联网和大数据也需要“低代码”

物联网和大数据都是时代的技术主旋律，而它们的发展也需要“低代码”助力。像物联网平台需要调度“云、管、边、端”各方资源，还要兼顾传感、语音等交互，并适应环境变化的状况——可想而知它的开发难度之大。“低代码”凭着灵活敏捷的开发功能，恰好能帮助降低物联网项目的开发门槛，缓解成本、人才等痛点。

据我所知，像优锘科技、畅图科技等物联网和地理信息大数据系统厂商，都在与低/零代码厂商明道云合作，并取得良效。个人认为，低代码开发平台能抓住物联网和大数据的风口，挖掘自身产品在高精领域的协作可能性，是很聪明的差异化拓业策略。

3、挑战：客户观念尚未扭转

APICloud 创始人刘鑫曾在一次访问中提到：“很多企业都说需要一个大数据、人工智能工具，但很少会说我需要一个开发工具。客户的需求并不是一个低代码平台，而是低代码能够产生的价值。”客户依然习惯性寻求贴身服务，观念尚未扭转，自身也难以培养低代码开发能力。“低代码”要真正普及，还需要继续教育市场和客户。

“低代码”的市场在时刻变化着，头部厂商在主动普及低代码教育，也有小众厂商探索该市场下的细分赛道，还有传统 ISV 躬身入局，加入战场。在机遇与挑战激荡的成长期里，我们尚且一起见证“低代码”的变迁。

**尾语**

日本富豪前泽友作买下马斯克火箭公司的第一张绕月飞行船票，并赞助 10 位艺术家与自己同行。他说：“过去的人类宇宙史里，只有科学家能上太空。我希望让艺术家也能去看看太空，看看另一个曼妙的世界。”

低代码仿佛也有这样的力量，让不会代码的人也能通过可视化操作，感受开发一套应用软件的成就感、获得感。不少程序员觉得低代码、零代码开发平台就是个玩具，但我认为，低代码、零代码不是所谓的“低智盛行”，而是“人人平等”，人人都可以是开发者，去探索技术的宇宙。

# 2、特斯拉放弃 SAP，仅 25 个人 4 个月就开发了 ERP！

特斯拉是老 K 非常喜欢的高端电动车品牌，除了贵以外，找不出它有什么明显的缺点。从特斯拉发布第一款车开始，老 K 就一直在存钱，按目前的存钱速度来看，可能还需要 99 年。

梦想还是要有的，万一见鬼了呢？

特斯拉老板埃隆·马斯克，也是老 K 喜爱的企业家，作为现实版的钢铁侠，多金、有梦想、大嘴巴，连找个女朋友都这么朋克。

图片自网络@版权归原作者

刚开始看到这则消息的时候，也是很惊讶的，25 个人 4 个月开发了整套 ERP，这不是闹着玩吗？

老 K 早年做程序员的时候，玩得一手好 ERP，因为活好、手快，获得“中关村 ERP 小王子”的美誉。所以，没有人比我更懂 ERP。

汽车制造行业的 ERP，是所有行业当中最为复杂的，包括了供应链、产品规划、库存管理、订单管理、资产管理、财务模块、客户管理等等。

这绝对不是 25 个人 4 个月能完成的。然而人家特斯拉 CTO Vijayan，牛逼啊。反正是干了 4 个月就上线了，业务用起来还觉得挺好。

翻阅了一些资料后，发现这个事情还挺有意思的。

当时，特斯拉刚推出 Model S，CTO Vijayan 正在跟团队评估，是继续升级 SAP，还是找其它的 ERP 套件来替换。

这时，伊隆•马斯克找来 Vijayan，对他说：“不如我们自己建一套 ERP 吧。”

Vijayan 看了看手表，现在是早上 9 点呀，心想老板没这么快昏头吧。

又看了看老板的气色，也不像是喝大的样子。

Vijayan 正准备说出一万个理由来拒绝的时候，想想老板的暴脾气、再想想手里还有大把期权没兑现，家里还有房子要供，咬咬牙说了句，“喏。”

说完带着 25 个人关了小黑屋，4 个月后，手捧着几百万行代码出来，一套叫做“Warp ”的 ERP 系统上线了。至此，特斯拉有了自己的 ERP。

我很好奇，这帮老外是怎么在四个月内捣鼓出一套能用的 ERP 的。

因为类似的项目我也干过，当年我作为“中关村 ERP 小王子”，带了 4 个开发，入驻某钢铁制造企业实施 ERP 项目。当然了，是在标准产品基础上做二次开发。但是，我们那个标准产品啊，跟没有差不多。

老板说，给你 6 个月，不成功就 TM 死在那里。老 K 年轻的时候多机灵啊，我心想：项目成不成功谁说了算？钢铁厂老板啊。于是老 K 前两个月啥也没干，就陪老板唱歌、跳舞、搓背，钢铁厂老板都好这口。开始的时候我是拒绝的，一切为了项目，我就豁出去了。

两个月下来，跟钢铁厂老板那感情深啊，就差点没把女儿许配给我了，老 K 默默看着那位身高 160，体重 160 的千金，咬咬牙还是婉言谢绝了。

做人要脚踏实地，别老想着天上掉馅饼，还是看看地上有没有钱捡比较靠谱呀。

但是，我搞清楚了一件事情：最让老板头疼的系统问题，就是每月不能及时拿到他要看的几张报表。

于是，跟团队说，所有工作围绕那几张报表来做，后来只用了 4 个月，钢铁厂老板在每个月末都能够及时看到他要的报表，他很高兴的打电话给我老板，一顿狠夸，项目款也结得很爽快。

扯得有点远啊，拉回来。特斯拉的 ERP 建设，跟老 K 说的案例可不一样啊，人家马斯克虽然也喜欢唱歌、喜欢美女，但人家懂 IT。早年创立了 paypal，技术还是很牛逼的，这就没法用老 K 实施项目的方法去忽悠他了。

那么，CTO Vijayan 究竟是怎么做的呢？只用了 25 个人，短短的 4 个月就做出一套 ERP。

原来，Vijayan 也不傻，从 0 开始做，做到猴年马月啊。于是，他使用了当时最好的低代码平台 Mendix。梳理完业务流程，就直接在 Mendix 上搭建了特斯拉第一套 ERP。

低代码这玩意，这两年在中国突然火了起来，其实早在 2002 年，国外就有类似的解决方案了，可见国内 2B 领域的软件水平，跟国外差距不是一点点啊。

Vijayan 在接受采访的时候透露，特斯拉自主研发的 ERP 系统“Warp”，最早是搭建在 Mendix 之上，经过一段时间的使用，性能、架构方面遇到了挑战。

于是他们使用了 C＃编程语言，在微软的.NET Framework 平台进行重构，并且运行在特斯拉的私有云上。

根据资料显示，目前“Warp”系统的开发和维护人员在 200 人左右。这套系统每天被全球 3 万多名员工使用，网站和 App 每日用户访问量达到千万级别。

仔细思考 CTO Vijayan 的整个决策过程，你应该会有所收获：为什么不继续用 SAP？为什么一开始用低代码平台来构建？之后又推翻，重新用 C＃重构？

# 3、零代码时代已来！程序员真的要去送外卖了？

“零代码”和“低代码”的概念是同时提出的，二者经历的背景都一致，所以就不做赘述了。

软件业经过几十年的发展，对于可视化编程语言和高级编程语言的开发逐渐成熟，同时市场对于 SaaS 软件的需求逐渐旺盛。

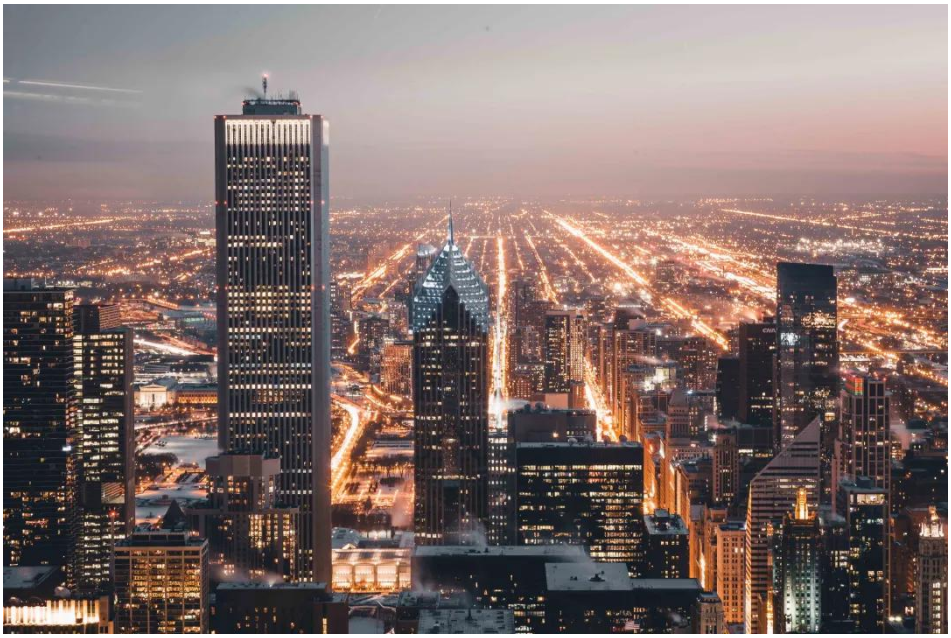
程序员都希望把软件项目的开发成本降低、开发效率提高，于是著名的咨询公司 Forrester 提出了“零代码”和“低代码”的概念，并带动小众的 ISV 开始研究“低代码”和“零代码”应用开发平台，开创出一条软件行业新赛道。

回顾一下“低代码”和“零代码”的概念：只需用很少甚至几乎不需要代码就可以快速开发出系统，并可以将其快速配置和部署的一种技术和工具。

由于“零代码”对编程工具可视化、简洁性的要求非常高，很少 ISV 能开发出完全“零代码”的产品，也因此“零代码”的发展速度比“低代码”稍慢一些。

“低代码”应用平台在 2014 年左右进入中国市场，目前处于成长期；而“零代码”应用平台则晚了两年，还在市场导入期。

在今天来看，国内相对知名的低代码软件厂商有 34 家，但零代码软件厂商只有 17 家。（数据来自中软网海比研究院《2021 年中国低代码&无代码市场研究报告》）



**“零代码”的技术特点**

**如果你试着百度搜索“零代码”就会发现，现在市面上的零代码开发平台除了色彩不一，内部的功能设计、列举的产品特色都大同小异：**

- **功能灵活，随改随用。**
- **实时生成应用搭建效果，所建即所见，所见即所得。**
- **交互设计友好化，直观化：拖拽小组件来完成表单配置；以垂直流程图的视觉来呈现自动化工作流或智能机器人的流程动作。**
- **灵活对接外部系统或插件：钉钉、企业微信对接是标配，Zapier、Processon 对接是极客玩法，各种电商、SAP、供应链系统对接则是高水平动作。**
- **支持跨平台使用：PC、Web、APP、H5、小程序。**

**以上“零代码”的技术特点自然能延伸出它的优点：**

- **高效开发应用，节省时间和人力成本。**
- **操作界面和开发逻辑都视觉化、交互化，对电脑小白而言更容易理解和使用。**
- **开发和维护费用低，因为底层的功能都做好了，开发和维护都只是功能调用的工作，复杂度降低，所以比传统软件定制开发的项目报价便宜不少。**
- **和外部系统的兼容性、互补性强，只要外部系统提供数据接口，双方就可以测试接通，在数据传输和响应上灵活互动。**

**作者本人也是“零代码”赛道的从业者，要说好话是十分容易。而谈到缺点，我更希望以我在工作中的所见所历，帮助大家代入场景去理解。**

**高度灵活化、去代码化会牺牲平台固定设计的自定义自由度。**

我曾经碰过一个客户，她对某个零代码平台的界面设计不满，坚持要求数据要横向排列而非纵向排列；应该提供一个调色板，让界面所有色彩部分都能自定义搭配。对于产品要求如此“拧巴”的客户，产品经理听到了恐怕只能扶额摆手，感慨不易。

处于市场进入阶段的零代码产品，会更注重功能完善、平台稳定、用户理解和使用门槛。过度强求产品固定设计的编辑自由度，就有点舍本逐末了，倒不如直接走全定制开发来得痛快。我相信，对于比较常见、确实影响使用的功能需求，产品经理一定会认真考虑，纳入规划路线图的。

超出现有功能的需求仍然需要写代码来实现。
将用户的新历生日自动换算成农历生日，从用户的身份证号码中提取出生日期和性别，去掉手机号字段中的“+86”……这些需求看似还挺日常和普遍的，但其实现在大部分零代码应用平台都无法做到这么细致的功能点。不过，平台开发者留出了“代码块”功能，让 IT 极客们满足特殊需求。

CSDN 上有部分程序员提出：一些简单但少用的数据处理动作，在 Excel 上用函数也能轻易完成，用代码块反而会变得更绕了。的确，这一点也是当前“零代码”的劣势。

外部系统集成需求无法完全满足。
目前虽然绝大部分“零代码”厂商都宣称能做到外部系统对接，但真正成功的例子却较少，因为它除了考验平台扩展能力外，还考验厂商技术团队的服务能力和甲方技术团队的配合度。

一家大型国有传统制造集团曾经找过一家零代码 ISV，希望做一个 ERP 系统，并把他们老 ERP 系统的部分功能对接过来，以便集团员工逐步从老系统过渡到新系统。理想很丰满，现实却泡汤了。

失败的原因是综合的：甲方光提需求，没有配合乙方提供老 ERP 系统的接口数据；甲方和老 ERP 系统开发商的沟通也是难题，一般老开发商不太情愿配合，毕竟是要把自己的生意让出去；乙方的实施顾问“手艺”未精，也很容易把项目弄垮。

很多技术人员都会从产品功能角度说用“零代码”集成外部系统的无能；但作者反而认为，是现实项目中多方解决实施困难的决心和行动力不一致导致了这个问题。

看完以上的缺点，或许有读者会觉得：那“低代码”还是比“零代码”好一点呀，毕竟还能更灵活地实现比较复杂的需求。对此，我找到一张“低代码”和“零代码”的对比图，帮助大家做多维度对比，对号入座，选择更适合自身情况的一种。

| | 低代码平台 | 零代码平台 |
| --- | --- | --- |
| 使用者 | 技术人员 | 非技术人员 |
| 定位 | 面向技术开发人员的下一代快速应用程序开发工具 | 普通用户自主开发应用程序的平台 |
| 比传统开发速度 | 2倍 | 8倍 |
| 易用性 | ⭐ | ⭐⭐⭐ |
| 技术要求 | 懂编码 | 无需编码能力 |
| 个性化程度 | 高 | 低 |
| 多平台运行 | ✅ | ✅ |
| 适合企业 | 有资金、有研发团队的中大型企业 | 无研发团队、软件需求紧急、变化快的中小型企业（或中大型企业的某个项目/部门） |

图自：明道云博客《零代码与低代码快速开发平台的区别》

**关于“零代码”，我的三个观点**

**上一篇科普“低代码”的文章里，我写了不少关于“低代码”行业前景的观点和材料。这次，我给大家分享三点感触，希望帮助大家对“零代码”树立新的体会。**

**“零代码”是中年编程小白的末班车**

**互联网行业爆发以来，人人都见证了程序员在社会地位上的跃进。招聘网站上铺满了程序员急招信息，高校、中小学、甚至连幼教都开设编程课，朋友圈和公众号上充斥着“30 天快速上手 Python”的网课。时代在告诉每一个人：不学点编程，你就落后了。**

**可是对 80 年代以前的编程小白而言，学编程谈何容易，能把房子、工作、一日三餐处理好就不错了。“零代码”对他们而言，算得上是“学编程”的替代品，能借助“零代码”做出和代码处理效果类似的成果，也能多添一个本领。（当然，替代程度大概仅为 50%，也不错了）恰好，这个群体在职场上也到了管理者的位置，熟悉业务运转和行业特点，十分适合挑起大旗，搭建业务管理应用。**

**用什么来证明“零/低代码”有好未来：4.5 亿个应用**

**我读过一篇很有意思的文章，里面提到：微软曾简单计算过，未来 5 年将有 4.5 亿款新应用程序将被开发出来，这比过去 40 年里开发的所有应用程序都要多。微软公司公民应用平台副总裁 Charles Lamanna 说：“如果这是真的，那么这 4.5 亿款软件必须使用低代码或零代码工具。而专业的开发人员应该专注于比费用提交表单或审批表单更困难的挑战。”**

**截至 2019 年 4 月，AppSheet 已经在谷歌的零代码平台上创建了 180 万个应用——4.5 亿的进度条已启动了 0.4%。**

**别让人人都自信地成为“零代码”应用开发者**

**“零代码”虽然降低了业务管理应用的开发门槛，但也很容易让对业务架构理解不当的员工盲目自信，开发出设计不当的应用，引发更大的管理问题。**

**曾有一家美国大型保险公司继承了 1.6 万个基于 Quick Base（一款低代码开发平台）的应用程序，但业务员把它们运行在 Quick Base 的退役版本上，导致系统运作混乱。我遇到过某家公司的业务经理，他梳理的业务流程图就像缝衣服一样，线路交错，只找到流程开头和结尾，叫人哭笑不得。至于后续他把零代码应用做得如何，就不得而知了。**

**个人认为，真正对客户负责的“零代码”ISV，必然是对客户教育负责的。他会建立适当的流程和基础设施，作为使用引导；并配备丰富的功能学习资源，如标准的业务应用模版、学习资料、定期用户交流活动等。先给用户赋能了，再让他们开始大干一场。**

# 4、低代码革了谁的命？

**网上写“低代码”起源历程的文章多不胜数，但大多是基于宏观历史的回顾和复述，不乏陈词滥调。而这次，我选择走近亲身经历过低代码本土化的第一批群体——软件厂商、客户、产品团队，在他们身上深挖“why”背后的答案。**

**为了让大家更有兴趣看完下文，这次我们结论先行：**

**1、万物起源皆混沌，第一批做“低代码”的国内软件厂商，一开始都不在认真做“低代码”；**
**2、第一批国内“低代码”产品团队，都有很“稚嫩”的产品思维；**
**3、“低代码”只是一个被套用到软件上的概念、形容词，实际功能才是产品本质；**
**4、第一批“低代码”国内客户是初代产品快速成长的最强辅助，极度包容、认可和支持“低代码”的发展。**

## *01*

### 大荒野：锁定中小企业市场

时间定位在 2015 年左右，当时国内 SaaS 软件正处于风口期，不少软件厂商都在寻找机会与灵感，做个小 SaaS 软件，说不定就成了“爆品”。而同时，BI、报表类比较复杂的 toB 数据软件已包揽国内绝大部分中大型企业客户。

也由于各家软件都不太一样，数据迁移也比较麻烦，因此企业一旦选定了某个软件后，基本不会弃用和另购新软件。中大型企业的市场基本已成定势，要在软件市场里更上一层楼，就要锁定新潜在客户——中小企业群体了。

当年，大数据的概念已经被提出和讨论，不过相关的技术工具还处于研发和不断迭代中，只有大企业、政府机构会涉足这个领域。

中小企业管理者也有“大数据”概念，想要系统地整合全业务数据，通过清洗、整理、分析数据，得出更准确科学的商业决策依据。然而，适合中小企业管理者使用的方法论和工具仍未出现或完全成形，他们被卡在很尴尬的境地。

传统软件厂商想攻占中小企业这片市场，同时想参与 saas 软件的赛道，于是奠定了入局基础。中小企业管理者对数据管理意识的觉醒，对于系统化业务管理和企业内部数据分析的无力感，为软件厂商们释放了一点机会信号。

## 02

### 造物：殊途同归——在线表单工具软件

循着这个信号，软件厂商们找到了少许产品设计的灵感和方向，不过这离最终做出低/零代码 APaaS 还远着呢。人们很容易会以为，软件厂商就是直接借鉴外国早期的低代码产品，便造出了第一批低代码本土化软件。但其实第一批“低代码”国内软件厂商，一开始都不是在认真做“低代码”软件。而且，它们的产品团队，最初都有很“稚嫩”的产品思维。

伙伴云的创始人之一袁兆江曾在一篇访问中谈到，伙伴云从最初的伙伴网发展到现在，一共经历了三次产品变革的试错。起初团队不知道要做一款怎样的企业协作应用，于是借鉴外国的 Yammer，做了企业内部协作通讯软件。

新品虽然一下子得到不少客户的青睐，但袁兆江团队反思到中小团队因为人少，几乎没有社交需求，所以这个产品没有真正切准中小企业的需求，不会走得长远。

后来，他推出数据协作领域的团队版 Evernote。然而，团队不甘于做个小应用，于是花了一年时间推出“伙伴 3.0”，号称一站式企业协作解决方案。产品做“大”了，但大到袁兆江也很难形容这是什么产品，陷入对产品的怀疑。痛定思痛后，团队给产品做减法，砍掉 2/3 的功能，最终推出了伙伴云表格——也就是今天的伙伴云。

比起伙伴云的故事，简道云的起步显得更“低开高走”。

2015 年，帆软成立了一个产品小团队，计划推出新软件，进驻中小企业软件市场，但是一开始还没想好做要什么软件。团队思索良久，决定先做一个简单的企业内部数据收集和管理工具，解决用 Excel 管理数据的麻烦问题，让企业能把数据处理标准化。2015 年，简道云 1.0 上线；2017 年，简道云正式商业化。

我很诧异，擅长做精细报表和 BI 软件的帆软是如何把报表工具做得轻量化的。简道云的业务负责人单兰杰告诉我："当时简道云团队的成员大多是行业新兵，这样反而没有定势思维，所以能放开了做，回归商业的基础数据结构需求，做最简的业务数据管理软件。"

简单的工具获得不少中小企业的喜爱，简道云商业化的第一年就有了上千家客户（当时简道云团队甚至没有一个销售员），随后几年里基本以每年 100%的客户增长速度，迅速开拓市场。因为有了更多用户，简道云团队获得越来越多用户需求，更快更深地了解用户所需，迭代出越来越完整的业务数据管理软件。

我向单总问了一个问题："当时你们知道有'低代码'吗？"

单总笑说："不知道，我们也不是很关注概念，更关注对用户有什么价值、解决什么问题。大家都是摸着石头过河的。我们一边做着，一边就觉得要把工具做得好用，小企业也能用。到后来，我们发现简道云的产品理念和'低代码'是契合的，于是慢慢地用它来概括产品特色。"

市面上很多"低代码"产品也是这样走来的，"低代码"只是一个被套用到软件上的概念、形容词，实际功能才是产品本质。

## *03*

### 开荒：包容、认可、热情的第一批"低代码"种子客户

大家猜第一批用上简道云的用户是什么人——IT 极客？互联网公司？
有，但仅为部分，更多的是自家楼下的眼镜店、水果店等小商贩。没想到吧！

简道云团队研发 1.0 产品的时候，亲自陌拜大街上的水果店、眼镜店、杂货店等小店老板，邀请他们成为简道云的种子用户。老板们大都是基于对产品功能的兴趣、解决业务流程管理困扰的好处，而受邀成为第一批用户。

单总谈到，种子用户们虽然没有"低代码"意识，仅有 CRM、ERP、进销存的系统概念，但是十分相信简道云的产品价值，会热情地提使用反馈和产品需求，也会乐意推荐身边人使用简道云。

随着用户规模扩大、产品雏形渐显，高校老师、科技博客博主、科技工作者，甚至西门子、百威等大公司的小部门，也成为较早一批使用简道云的用户。

伙伴云的第一批用户来得更直接和亲密一些。他们都是伙伴云产品团队在创造 DiscuZ!时的老铁粉，知道团队转战 toB 软件了，便跟着来了解、探讨、体验，甚至极力支持伙伴云产品。

魔方网表是国内“低代码”软件的老元祖了。我翻查其最早的客户案例报道，发现在 2012 年，魔方网表就成功向满福珠宝交付出“低代码”的门店数据管理系统，堪称最早启蒙客户使用“低代码”产品的国内软件厂商之一。

要总结第一批“低代码”软件厂商是怎么获得第一批客户的，只能说：出其不易，出其不意。

“低代码”软件的概念太新了，普通人最初理解起来比较困难，于是简道云选择面对面介绍、推荐，同时用诚意来获得用户信任。伙伴云团队有粉丝基础，凭着信任背书来吸引第一波用户。用真诚来换信任，得到的用户不会差到哪儿。

*05*

**尾语**

**“低代码”能在国内做起来的原因。我认为有几点：软件市场形势的推动、早期“低代码”软件厂商的探索和试错、中小企业对自身管理需求逐渐清晰并主动寻找解决办法。三方共同作用、磨合、商业化成熟，最终让“低代码”软件在国内市场站稳了。**

**这篇文章是我写作以来收集素材耗时最长的一次，我翻遍了全网有关第一批国内低代码 ISV 的报道，和简道云的业务负责人做了一次深度访谈，和同事聊了不少产品经历。我离历史越来越近，看到越来越多前人留下的脚印。**

**IT 界内不乏有人在批判“低代码”，各有各理。但我完成了这篇文章后，想说：**
**如果你不走近一点、不亲眼看过国内“低代码”的发展足迹，就不会真正懂中国“低代码”的开荒路；不会明白一个新生概念从被提出到呈现为一个个真实事物的过程中，有多少人曾涉足尝试，探寻出路，拨云见晓。**

**我喜欢简道云团队的无畏敢为、放手尝试；佩服伙伴云团队的自我批判，不断追问内心深处最想做什么产品；也忍不住赞叹魔方网表在最早期就先行探索，而且现在发展出多元精专的产品线。**

**我相信每个读者都希望用更全面的眼光看待一切事物。而看懂了中国最早期的“低代码”产品是如何开荒辟野的，我们就不会再那么理所当然地站在上帝视角，缺乏辩证性、历史性地评价每一个“低代码”产品的发展选择、市场地位、长短板等。这，是这篇文章想带给大家的另一层价值。**

# 5、低代码扑街了？

**近日，某头部科技媒体发表了一篇文章《“行业毒瘤”低代码》，文中采访了某知名咨询公司中国区 CTO，他表达了一个观点：“以降低程序员门槛为目的的低代码是行业毒瘤”，文章标题就是取自这个观点。**

**后来明道云的任总站出来发声，写了一篇《低代码不是行业毒瘤，你才是！》来表达观点，一些自媒体也转发了这两篇文章，留言区的讨论很热烈，有支持的，也有反对的。**

**说下对这件事的看法：**

**首先捍卫表达的权利，这点是肯定的。但是用上了“毒瘤”这样的字眼，老 K 认为是不妥的。**

虽然老 K 为了阅读量，也经常搞标题党。但是理解归理解，作为头部媒体、百万粉丝的大号，用词还是严谨些好，毕竟这么大的影响力，要顾及的方面更多一些。

“低代码”作为一个“新”事物，一开始有争议是难免的，而且自身方法论体系也不完善，仍然需要一个演进的过程。鞭策它、给它挑毛病，一点问题没有。

即便这位 CTO 表达的观点过于直接、口无遮拦，小编也应该拿出点职业精神，稍作处理一下。但是如果小编为了完成 KPI，而利用了 CTO 的原话、搞标题党，那就有点 low 了，有失百万大号的风范。

以上纯属个人观点，点到为止吧，不展开。这并不是本文要探讨的主题。

言归正传，在上一篇低代码的文章中，老 K 讲了一个观点：从 ERP 到中台，再到低代码，演进的逻辑是企业数字化转型要解决的主要矛盾发生了变化。

ERP 解决的是，企业大规模生产管理问题；中台解决的是，企业快速创新的问题；“低代码”满足了企业“敏捷能力”的诉求。

由于篇幅有限，上篇文章没有展开讲，本文接着探讨：是什么驱动了主要矛盾的变化，其背后的核心要素是什么？它又是如何演进的？

首先思考一个问题，推动社会进步、企业变革、商业发展的最根本动力是什么？

说到底，就是技术进步。

## 社会化大生产时代：<br>解决大规模生产效率问题

上个世纪初，主要的技术进步是：蒸汽机的改良、电力的广泛应用。它极大释放了生产力，使得人类进入到了社会化大生产时代。

为了应对大规模生产的效率问题，管理学正式成为一门学科，大家熟悉的科层制组织、绩效考核、平衡记分卡等等管理方法论和工具，先后被发明出来。

随着计算机技术的普及，在 60 年代，MRP 作为一种企业管理软件，也开始应用到企业管理当中，企业的数字化水平有了进一步的提升。

直到 90 年代，Gartner 正式提出了 ERP 的概念，至此，ERP 开启了它辉煌的一生。可以说，ERP 是许多读者的“同龄人”，一晃快 30 年了，扎心啊。

## 互联网时代：
## 解决企业创新问题

随着互联网技术的日益成熟，“上网”成为了许多企业的标配，也成为了许多人娱乐和生活的方式。

老 K 就是在 90 年代末开始接触互联网的，第一次进聊天室、混迹 BBS；逛“榕树下”看网络文学；也看了许多岛国老师们，不可描述的图片和视频 ……哎，谁还没有青春呢。

90 年代最重要的技术变革就是互联网，互联网深刻改变了人们的生产和生活。

人们个性化的需求被释放，要求产品创新、服务创新，于是“大规模定制”、“柔性生产”的要求被提出来。

企业为了应对市场的需求，首先在组织上进行变革：“扁平化”、“去科层制”。日本的稻盛和夫提出了“阿米巴”经营，很好地解决了企业创新的问题，让企业更快速地应对市场的变化。

2015 年，国内的阿里巴巴提出了“中台”模式，采用“大中台、小前台”的方式，合并相似组织，将企业的核心能力进行沉淀和输出，支持企业快速创新和试错。许多企业纷纷效仿，推出了自家的“中台”系统，不同程度地解决企业自身的发展与模式创新的问题。

## 智能时代：
## 赋予中小企业“敏捷”能力

近几年，技术的变革主要体现在：移动化、大数据、云计算、人工智能等方面。

尤其我国在移动化应用方面的发展，已经领先全球。移动支付、出行、购物、导航、社交等等极其丰富地应用场景，产生了海量的数据，这些数据给人工智能提供了丰富的“养料”，再加上云计算技术的发展，使得大规模运算成为可能，为人工智能又提供了硬件支撑。

于是，我们大步迈入智能时代。

**智能时代下的生产，对组织的灵活性、对个体能力的要求进一步提高。出现了“去中心化组织”、“创客化”、“自由职业者”，也就是“个体崛起”的时代。**

**微信生态养活了数千万的“自媒体”、“微商”。阿里生态之上的电商公司、中小企业也有数百万之多。**

**这些数目庞大的个体、中小企业快速成长起来，他们对于数字化赋能有着更敏捷的诉求。比如疫情期间，几个星期就完成从线下到线上的服务和管理的转型。这在以前是完全不可想象的。**

**而且这种敏捷能力，对企业来说，是一种融入 DNA 的能力，没有就得死。**

**“低代码”，以成本低、部署快、云原生、模板化、可少量定制的优势，天然满足了这些中小型企业对敏捷能力的诉求。**

**那么，为什么大型企业也需要“低代码”呢？前文提到的，大企业正在“去中心化”、“创客化”，比如海尔、美的这样的传统制造企业，也被“拆分”成数百个独立的“小微”企业，它们本质上是跟微信生态里的“自媒体”、“微商”是一样的小微企业。**

**所以，企业的需求极其旺盛、资本不断涌入，“低代码”行业出现了“井喷”的发展态势，成为了近年来少有的“风口”。**

## 结语

**总结一下本文观点：**

**1、从 ERP 到中台，再到低代码，演进的逻辑是企业数字化转型要解决的主要矛盾发生了变化。**

**2、社会化大生产时代，主要的技术进步是：蒸汽机的改良、电力有广泛应用。企业需要解决大规模生产效率问题，于是出现了 MRP、ERP 管理软件。**

**3、互联网时代，主要的技术变革是互联网的普及应用。企业要解决创新的问题，于是提出了“阿米巴”组织、“中台”模式。**

**4、智能时代，主要的技术进步是移动化、大数据、云计算、人工智能。VUCA 时代下，企业要具备的是敏捷能力，“低代码”，以成本低、部署快、云原生、模板化、可少量定制的优势，天然满足了企业的诉求。**

最后，没有哪种技术是完美的，也没有哪种技术生来就是“行业毒瘤”。IT 从业者有他的价值主张，媒体也有它的责任和使命，用尼采的一句话与君共勉：“你凝视着深渊时，深渊也在凝视着你”。

# 6、杀死低代码

最近，“低代码”被媒体推到风口“刀”尖。因为一篇文章《“行业毒瘤”低代码》，“低代码”被扣上了行业毒瘤的大帽子。

尽管徐 CTO 对低代码的部分看法不无道理，但是毒瘤一词显然说得太“过”了。我和老 K 对此交流不少意见，我们认为：要给新事物足够的生长空间，让其在批判和进步中补其所短，增其所长。

## *01*

### 低代码，怎么了？

现在，我们的确看到了低代码给企业带来的不少好处：

1、高效开发应用，节省时间和人力成本。

2、操作界面和开发逻辑都视觉化、交互化，易于理解和使用。

3、开发和维护的复杂度和费用都比传统定制开发的低。

4、和外部系统的兼容性、互补性强。

同时，技术专家也看出了它的弊端，比如：

1、宜创科技 CEO 宜博称，低代码的技术壁垒在于懂技术的看不上，懂业务的学不会；

2、大数据治理专家彭文华指出，低代码一时爽，数据治理火葬场。

低代码确实是一个被讨论了很多年的话题，自身也经历了多种形态、产品模式的探索式发展。小到给小学生摆弄的可视化编程软件，大到低代码企业应用开发平台，既有其成功的闪光点，又有瑕疵短处。批评是有必要的，但也给它们一点时间去改正、进步吧，总能成为“三好学生”的。

话说回来，为什么低代码会遭受开头所述的抨击呢？其实，向低代码“捅刀子”的绝不仅仅是个别自媒体。那么，还有谁？

## *02*

# 谁在杀死低代码？

## 一、自媒体：他们的话足以影响读者的判断

以某知名科技媒体的《“行业毒瘤”低代码》为例，徐 CTO 站在软件发展的潮流角度评价低代码只是一个重复流行的概念，从 IT 从业者的维度批判低代码的实用性；还过分夸大低代码的弊端，已经到了明显与实际情况不符的程度。这些观点的确为文章赚得不少热度，但也强行歪曲了部分事实。

之前陈果 George 也发布过“低代码烂大街”的观点，但立场相对中肯得多：
1、“低代码”本身很有用，但是其应用广泛性还有待实践验证，作用不能被夸大了，否则，就像“中台”一样，本来是个好东西，却被媒体、厂商和外行们炒作得烂了大街。

2、企业内大面积的公民化开发的应用创新是个伪命题，开发软件是一回事，能用起来是另一回事。

3、低代码不适合开发复杂逻辑的核心业务，不适合管理企业主数据；使用低代码开发，会对企业的数据治理、信息安全产生一些隐患。

个人认为，人人都有表达的权利，自媒体亦然。但自媒体是有公众影响力和号召力的，切勿为了流量和热度而发表过于绝对的观点，扼杀新事物。保持专业，保持慎独，真实表达。

## 二、开发者、程序员：一些程序员害怕低代码革了自己的命

他们是软件开发项目里最基层的一员。当低代码挥着解放劳动力、提高开发效率的大旗出现，他们很难不焦虑。在很多低代码、零代码的文章里，总能看到这类群体的留言，比如：

“低代码骗小白的东西还有人吹，996 可不仅仅是需求多，而是需求改动的多，还想搭积木开发产品，可能中间还没搭建完，发现底座需求都变了。”

“低代码，对于刚入行人员，一点帮助都没有，全世界低代码千千万，都是不可复用性，所以学了一点用处都没有。”

“低代码最终会生成，运行效率低又难以维护的垃圾，最后不得不重构。低代码上面投资越多，技术债越重，推倒重来的时间点就越近。”

我不是码农，挺难站在程序员的角度感同身受；但我接触过许多已经习惯用低代码、零代码完成项目开发的程序员。他们依然用传统编程开发的严谨思维设计着系统架构，但同时熟练迅速地操纵着低代码工具，快速完成项目交付。

他们也会吐槽低代码开发工具的缺陷，有时是找 bug，有时是反复质疑某个老功能的问题。他们的话很专业严厉，一针见血，让低代码工具的产品经理赶紧提需求。

看，一部分程序员已经在和低代码开发工具建立新的相处模式，有协作也有批评，没有好坏之分。因此，不妨把低代码当作自己新的生产工具，思考着学习者如何用它来提升自己的生产效率。能早点收工回家吃饭，不香吗？

三、传统企业软件服务商：害怕低代码动了自己的奶酪

本文不好指名道姓，但一些 CTO 和老 K 私下交流时曾表达过：甲方客户对低代码感兴趣，但他们（传统企业软件服务商）又没有推出这方面产品，所以就会跟客户说低代码的种种弊端。

为了获客，打击对手是常见招数了。不过，希望这类软件服务商也能静下心来好好了解低代码、零代码，对比自己和它们的区别，去理解为什么客户为它们着迷，自己又拥有怎样的竞争力。

我相信传统企业软件依然有它们的价值，特别是在一些技术壁垒较高的软件产品领域，低代码很难一时攻克。积极拥抱变化，顺应趋势，取长补短，加快转型，为之正道。

四、企业中低代码工具使用方：低代码影响了企业的“内部政治”，这是他们在软件功能之上更看重的地方

这是我在甲子光年的《低代码局中局：是 IT 革命还是高级外包？》里读到的故事：
2017 年宜创科技服务一家大型集团企业，却遭到了 CTO 的“嫌弃”。CTO 对着宜创科技的人员大吼：“你们不就是一键建站的吗？凭什么要这么多钱？”宜博告诉「甲子光年」：“CTO 的抵触心理也可以理解。我原来管 300 个人，你现在告诉我用了你们的工具，30 个人就能搞定。我 CTO 以后怎么混？”

更高效的生产力工具，意味着更少的人能干更多事情，低代码软件改变了企业组织管理模式。目前国内不少大型企业依然有着传统的办公室政治风气，部门各立山头，以团队人数彰显势力，进行着办公室版“权力的游戏”。

理想地思考这个问题的解决办法：企业管理者牵头，梳理出使用低代码工具的部门生产模式，探寻创新高效的部门管理模式和架构。从部门试点调整架构，到逐步普及，最终构建出一个个术业专攻、灵活高效的 IT 或业务部门。

## *03*

低代码，势不可挡！

低代码被“追杀”，高手则能沉住气分析形势。对于低代码的未来趋势，当前 IT 界里有几个突出观点。

低代码的未来潜力可以被量化

微软（中国）CTO 韦青说：“未来 5 年，将产生 5 亿个新的应用，而且是基本属于逻辑应用，等于是过去 40 年的总和。而当前的软件专业开发人员或者专业软件公司数量是不可能去应对这种需求。现在，恰恰正是低代码/无代码开发的时机。”

Gartner 预测，到 2025 年，70%新应用都由低代码开发工具所开发。

不会出现通用“低代码”平台

“低代码”只有根植于业务场景，才有生命力。不存在通用低代码平台，通用意味着不够深入业务，必将失去价值。

伙伴云官方平台称：未来 5-10 年，低/无代码平台的发展趋势有三个特点，其中一点是低代码平台行业化。目前有一部分低代码平台是基于 PaaS 服务构建的，这些低代码平台本身有较强的行业背景，也有大量成熟的开发方案，所以可用性和稳定性都比较强。

大数据治理专家彭文华表示：低代码的核心就是对各种现有场景通用功能的抽象。在我看来，就只有一个场景，就是信息流转的流程设计——所有场景，全部适用。

“低代码”正处于技术成熟度曲线中的“期望膨胀期”

2021 被称为“低代码”元年，不少技术团队扎入低代码赛道，低代码正在经历“市场泡沫”。不过这就是行业发展的必经阶段，有质疑和追捧都很正常。

甲子光年曾从多位采访对象里得到评价：中国低代码领域还没能成功验证商业模式。

“国内的低代码市场，无论从产品能力，整个理念，技术深度和广度上都没超过国外厂商。”阿里云 SaaS 加速器负责人黄省江表示。

盈动资本合伙人蒋舜坦承：“我觉得并没有纯粹的、标准的低代码创业公司。只是今年投资本身没有什么题材，大家都会去看一看。”

伙伴云表示：低/无代码早就不是新概念了，更谈不上是软件业革命。毕竟第一代应用平台产品诞生在上个世纪末，距离现在已经 20 多年了。

正如明道云创始人任向晖所说，“是革命，也早就革命完了。APaaS 品类早已过了那个过山车的顶峰，今天正在走进‘寻常企业家。”

*04*

**结语**

这一轮低代码的争论风波逐渐平复，总有下一波，只是迟与早的问题。作者希望大家能理性看待网络上的议论，不仅是低代码，还有中台、新能源造车等等社会议题，让理性慎独成为自己的思维习惯。

否定一个事物很容易，但有理有据地说出来却很难；蹭流量、带节奏很容易，但最终能为读者、为某个行业的发展献计献策却很难。无论是媒体还是从业者，都应该选择难走的那条路，而不是轻易扼杀新事物。愿我们，都敢于踏上难走的那条路，大步往前。

# 7、揭露一个伪“低代码”平台！

今年“低代码”突然就火了，老 K 和流水姐也写过几篇介绍低代码的文章，阅读量都不错，除了因为流水姐的文笔犀利之外，更重要的是大家对低代码的关注度很高，随便一写都会火。

在中国，只要一个概念迅速火起来，立刻会吸引许多人纷纷入场，这就导致了从业者的技术水平和经营能力差别巨大，形成了鱼龙混杂的局面。

就像前几年的中台赛道一样，只要是个做软件的，都说自己是中台，你批评他，他还不高兴。老 K 就是因为写了几篇评论中台乱象的文章，被一些同行记恨到现在。

正所谓，以前称兄道弟，以后不再联系。也好也好，道不同不相为谋。

## *01*

### 低代码为什么突然就“火”了？

老 K 作为一名理工男，是不相信“时机成熟”、“时候到了”这种笼统说法的。我认为商业社会背后遵循的是商业规则，只有透过表象，才能洞察本质。

低代码之所以火起来，背后一定是有原因的，我稍作了些分析。

从外因来讲，疫情导致中小企业数字化转型的进程被提前。以传统餐饮行业为例，他们需要迅速建立起：在线订餐、客户管理、营销管理、员工办公管理等系统，但是传统的 IT 开发成本太高、交付周期长，不适合中小企业的敏捷特性。

另一方面，中大型企业的数字化服务市场，经过 10 几年的发展，进入增长平台期，不能够满足软件服务企业的业绩增长需求，需要开辟一个崭新而广阔的市场，于是中小企业数字化转型市场被挖掘出来。

从内因来讲，中小企业数字化转型迫在眉睫。在全世界疫情常态化的新局势下，中小企业只有拥抱数字化，实现经营、管理、服务的转型升级，才能够降本增效、服务创新，实现新的业绩增长。

在内因、外因的共同作用下，低代码成为被风口选中的行业，加上资本的涌入，整个行业突然就火爆起来。

低代码火爆的同时，随之而来的是行业乱象，随便一个软件公司都说自己是低代码。为了更好的研究低代码，我们首先要探讨，如何定义低代码平台？

## *02*

### 如何定义低代码平台？

著名咨询机构 Gartner，于 2020 年 9 月发布的《企业级低代码开发平台的关键能力报告》(Critical Capabilities for Enterprise Low-Code Application Platforms）中，定义了低代码的 11 项关键能力。也就是说，这 11 项关键能力是衡量一个平台是否能够称之为低代码平台的关键因素。


图片@Gartner

**先简单了解一下各项要素的含义：**

**1、Intuitive, No-Code App Development：易用性，不写代码时的开发能力。**

**在不写代码的情况下，能够完成多复杂的系统搭建。这是标识低代码开发平台生产力的关键指标。**

**2、Application User Experience：所开发出来的应用的用户体验。**

**它指的不是低代码开发平台本身的用户体验，而是通过低代码平台开发出来的应用，给到用户的使用体验。**

**3、Data Model and Management：数据模型和管理。**

**这个指标就是通常所讲的“模型驱动”，相比于表单驱动，模型驱动能够提供满足数据库设计范式的数据模型设计和管理能力。开发的应用复杂度越高，系统集成的要求越高，这个能力就越关键。**

**4、Process and Business Logic：工作流与业务处理逻辑。**

**流程应用与业务逻辑开发能力和效率。这个能力有两方面：**

**第一，是指使用该低代码开发平台，是否可以开发出复杂的工作流和业务处理逻辑；**

**第二，是开发这些功能时的便利性和易用性程度有多高。**

**5、Platform Ecosystem：开发平台的生态系统。**

**低代码开发平台的本质是开发工具，当内置的开箱即用的功能无法覆盖更多应用场景时，就需要基于该平台的完整生态系统，来提供更深程度、更全面的开发赋能，比如开放的插件机制。**

**6、API and Integration：编程接口与系统集成能力。**

**为了避免"数据孤岛"，低代码开发平台需要提供系统集成能力、编程接口，跟其它系统进行数据的互联互通。**

**7、Architecture：系统架构。**

**系统需要支持服务化、分层的架构方式，支持高可用、集群的应用部署方式。**

**8、Quality of Service：服务的质量。**

**系统的健壮性、无故障使用时长、故障恢复时长，对 CPU 资源、硬盘资源占用情况，对云是否友好。**

**9、Persona and SDLC：用户模型与软件开发生命周期支持。**

**软件开发的生命周期中，包含设计、开发、反馈、测试、运维等多个环节，低代码平台要支持单元/集成测试、联调、发布、回滚、持续迭代等。**

**10、Governance：治理及运维。**

**现代软件开发中的敏捷开发、代码库管理，版本权限，发布管理等，在低代码平台中，都要能够支持。**

**11、Security and Compliance：安全与合规。**

**低代码开发平台需要提供：灵活的部署方式、安全机制和权限控制、SSL 数据传输、密码强度策略、跨域访问控制、高粒度的用户权限控制等等。**

**以上就是 Gartner 对低代码平台定义的 11 个要素，相信许多技术人员已经对低代码有个大致认识了。但是，想要识别和判定一个低代码平台的话，直接使用以上 11 个要素，就有点太繁琐了。**

## 03

**如何识别伪“低代码”？**

**老 K 结合软件工程全生命周期实践，以及 Gartner 定义的 11 个要素，给“低代码”平台拟定了 6 个维度的判定标准：**

**1、拖拽式开发；**

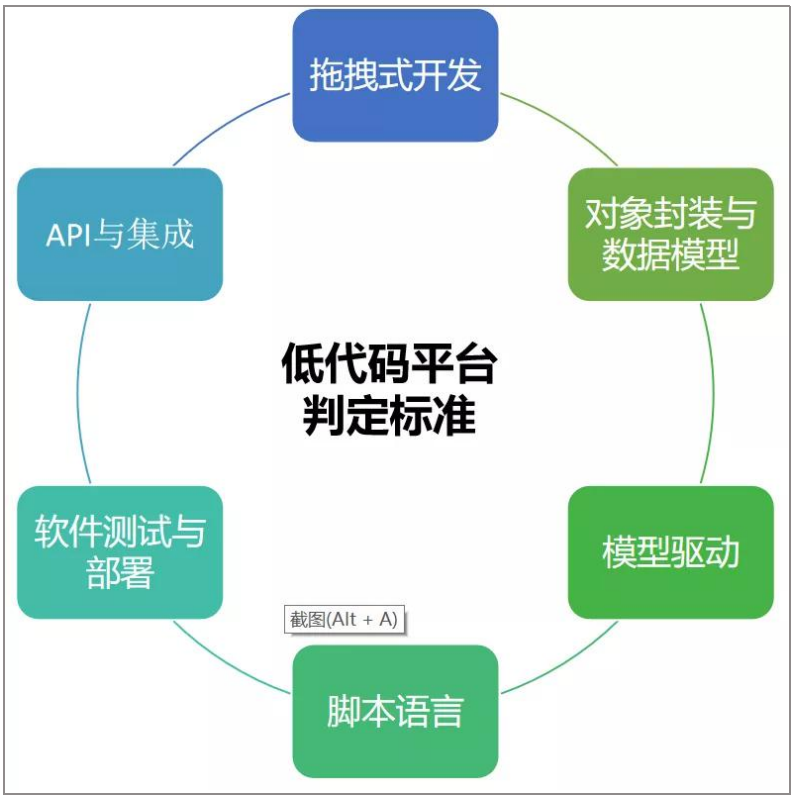
**2、对象封装与数据模型；**

**3、模型驱动；**

**4、脚本语言；**

**5、软件测试与部署；**

**6、API 与集成。**




图片@技术领导力 Mr.K

**一、拖拽式开发**

**拖拖拽拽做开发，就是“低代码”开发平台给大家最直观的印象，所以也是“低代码”开发平台最基本的特征。**

**这一点，世面上的许多“低代码”平台都能够做得到，可以说是个送分题，如果你家的软件产品连这点都做不到，就不要吃“低代码”这碗饭了。**

**二、对象封装与数据模型**

**这部分指的是低代码平台要操作的对象、数据模型、表达式等等，它可以是高度抽象和封装的对象，可以省略掉“类”、“接口”、“函数”这些编程语言的高级特性，以更简化的方式提供出来，供程序调用。**

**三、模型驱动**

**“模型驱动”是相对于“表单驱动”的，指的是对于数据进行建模和处理，比如国外的低代码平台 OutSystems、Mendix，就有很强大的模型驱动的能力，包括了定义实体、实体关联、主键、索引、数据查询等等。**

**四、脚本语言**

**脚本语言实际上就是编程语言了，是低代码平台实现复杂业务逻辑的扩展，可以使用 JavaScripts、Python、Java 等语言进行编程。**

**但是，低代码平台会把语言的编译过程做好封装，做到一键发布，即时运行，方便代码调试。**

**五、软件测试与部署**

**低代码开发平台，本质上是软件开发工具。所以整体开发过程也要遵守软件工程的流程规范。只是把许多环节都做了简化、内部封装，降低了学习成本、开发成本、测试成本、部署成本。**

**六、API 与集成**

**主要是解决低代码平台开发出来的系统，跟其它外部系统的数据互联互通，否则又是造了一堆大烟囱，一些数据孤岛。**

**以上 6 个判定标准怎么使用呢？**

**每个维度 1~10 分，总分 36 分是及格线。分数越高，表示这个低代码开发平台越完善、成熟度越高。**

**通过这个方法，可以简单判断一个低代码平台，是真低代码，还是伪低代码。**

## 结语

**以上，总结了低代码平台 6 个判定标准：**

**1、拖拽式开发；**

**2、对象封装与数据模型；**

**3、模型驱动；**

**4、脚本语言；**

5、软件测试与部署；

6、API 与集成。

最后，留个小作业给大家思考：

1、以前非常流行的 Delphi、Power builder，是不是低代码开发平台？为什么？

2、国内和国外，有哪些是真低代码平台，哪些是伪低代码平台？为什么？

留言告诉我。

# 8、低代码将干掉 70%的软件开发工作

根据 Gartner 预测：到 2025 年，70% 的新应用将由低代码/无代码技术完成开发。

也就是说，“低代码”将干掉 70%的传统软件开发工作。

什么是传统软件开发工作？

说白了，就是目前绝大多数程序员每天在做的 CRUD 工作。

你说程序员的命苦不苦？AI 想要取代他，“低代码”要革他的命，甚至干到 35 岁自己就被淘汰了。

那么，程序员应该如何应对呢？本文就来聊聊这个话题。

时代在进步，人也要与时俱进。之前，国内某咨询公司的 CTO 说：“低代码是行业毒瘤”，那篇文章在业内广为流传，我看了其中的观点，觉得挺有意思的。

这让我想起一个故事，在工业革命时代早期，有工厂主发现纺织机经常在夜里遭到破坏，就派人躲在角落里观察，看看到底是谁在使坏。因为当时没有监控摄像，只能用人肉监控。

结果他们发现，是一些小作坊的纺织工人在搞破坏，原因是他们认为先进的纺织机抢走了他们的工作，而织布是他们赖以生存的手艺，不甘心这几十年修炼出来的一身本领，一夜之间被一台机器所取代，所以破坏机器泄愤。

你也觉得很可笑吧，历史总是惊人的相似，一百多年后的今天，这一幕居然还在重演。

## 01

### “低代码”是更先进的软件开发方法吗？

先看看资本怎么说，别人拿真金白银投票的，总不是开玩笑的吧。

OutSystems 宣布获得 3.6 亿美元投资、估值过 10 亿美元。Mendix 被西门子 7 亿美元收购。

国内的简道云、明道云、氚云、钉钉宜搭、轻流、易鲸云等，也先后获得了数额不菲的融资。

再看看，科技巨头在“低代码”领域的布局。研究一个新技术的应用情况，就看看以阿里、腾讯等公司为代表的互联网巨头究竟是什么态度。

一、阿里

阿里年初推出了“云钉一体”战略，把钉钉、宜搭、阿里云等平台进行整合。为企业提供全生命周期的 IT 解决文案：

基础设施层，由阿里云提供网络、主机服务，为企业打造云端 IT 基础设施。

中台体系，由阿里数据中台、技术中台、钉钉中台，组成的标准化服务能力。

前台应用，通过宜搭、以及用户自建的业务应用软件构成。

可见，以宜搭为代表的低代码平台，在阿里 toB 解决文案生态中，占据重要地位，是“云钉一体”战略中很重要的一块拼图。

二、腾讯

低代码在腾讯内部有着非常广泛的应用，比如“星图低代码平台”是为游戏营销活动开发而设计的。微信支付、腾讯广告相关的部门也有相关的低代码产品，也都是为了提升各自业务场景下的研发效能而建设的。

腾讯将各个事业部的低代码平台进行整合，推出了 OTeam 平台。它是真正意义上的低代码平台，包含了：UI 可视化、逻辑可视化、DSL 代码语言、生产和运行模块、质量保障模块，以及配套的 IDE 开发工具。


来源：腾讯大讲堂

以上，分析了资本对低代码的热捧，以及腾讯、阿里等科技巨头对低代码的布局。

可以看出，低代码绝不只是停留在概念炒作的阶段，从资本到科技巨头都对这个领域极其重视，并且都做了许多积极的探索和实践。

## 02

### 低代码究竟提供了什么价值？

简单来说，低代码为企业提供了“降本、增效、提质”的价值。

降本、增效、提质，就是为企业降低研发成本、人力成本，提升研发效率，缩短产品交付周期，加快企业试错的速度，降低试错成本。使得企业的产品和服务以更快的速度进行迭代和优化，在激烈的市场竞争中胜出。

在接受 Creatio 调研的 1000 位开发高管中，95% 的人认为低代码开发速度相对于传统方式有提高，其中 61% 的高管认为提高速度在 40% 以上。

低代码为什么能够降本、增效、提质？低代码平台所具备的能力有哪些？

1、开发过程可视化。可视化交互是低代码平台所具备的一种必备能力，不再面对冷冰冰的传统文本 IDE 编辑器，转而和可视化的编辑器进行交互，不管是 UI 界面，交互事件、后端接口、数据库/Redis 调用，都能通过优雅而简单的可视化交互完成配置和编辑。

2、代码开发组件化。这个能力和中台化、SDK 的概念有相似之处，就是将重复的公共的能力沉淀出来，封装起来，让开发人员可以在低代码平台上，直接拿出来作为工具嵌到产品中，这样开发者就不用再关心这个功能/组件的内部实现。

3、一次开发，多端发布。对于前端研发人员来说，经常需要多端发布同一个项目/页面，H5/小程序/IOS/Android 的开发工作，经常需要不同技术栈的研发人员。而对于低代码，就屏蔽了具体的代码选型，内部编辑都用一种低代码语言，最后发布上线，可以发布到小程序/安卓/IOS 等多个端，而且能尽量保证 UI、交互、功能的一致性。

## *03*

### 低代码的使用者是谁?

Creatio 调查结果显示，低代码平台的使用者中，约 67% 的人是 IT 开发者，剩下的则是业务人员。也就说是，低代码的使用者以研发人员为主。

还记得特斯拉技术团队开发 ERP 的故事吗？20 几个人在 4 个月里，通过使用低代码平台 Mendix，搭建了第一套 ERP/MRP。开发主力仍然是那 20 几个程序员，而不是特斯拉的业务人员。

即使低代码平台做到很高级的可视化，轻松实现拖拉拽来生成代码，它仍然属于软件研发的范畴，仍然具备很强的专业性。

需要掌握需求分析、业务建模、代码调试、模块测试、发布和运维等工作。这些并非一个普通业务人员能够胜任的。

也就是说，低代码仍然需要大量的专业程序员，只是低代码平台把程序员从低效的、没有技术含量的 CRUD 当中解放出来，做更有价值、更高效的软件开发工作。比如：业务建模、领域建模、数据结构设计、业务流程设计、业务系统调试和部署等等。

举个例子来讲，你可以用记事本写 JAVA 代码，然后通过 JAVAC 命令指定类路径来编译，K 哥 10 几年前做程序员的时候就是这么干的，也许你会觉得这才是高手，这很牛逼。

但是你不要忘了，这很低效，所以我们现在都使用 idea 等高级开发工具来辅助编程。

低代码并不是要干掉程序员，低代码是新一代的软件开发方法和理念，将程序员从没有技术含量的 CRUD 工作中解放出来，做更有技术含量、更有价值的事情。

**结语**

**以上，我们讲了低代码是一种更领先的开发方法，深受资本和科技巨头的青睐。**

**低代码能够给企业带来降本、增效、提质的价值，成熟的低代码平台具有：开发过程高度可视化、组件化、一次开发多端发布等特性。**

**低代码的主要使用者仍然是程序员，通过低代码平台完成：需求分析、业务建模、代码调试、模块测试、发布和运维等环节，实现软件研发全流程的提效。**

**一套真正意义上的低代码平台，能够覆盖软件研发全生命周期，带来工业级的效率提升，这将是一次不可逆转的、具有颠覆性的软件研发效率革命。**

# 9、上“低代码”半年，30 名程序员被裁，CTO 离职！

**一位读者小 M 给我讲述了发生在他们公司的真实故事，为了避免不必要的麻烦，隐去一些敏感信息，我将整个事件的经过整理出来：**

**小 M 是广州某制造企业的技术负责人，下面带了 50 个技术人员，负责该公司 OA、CRM、人事等多个日常运营类系统。小 M 也是我的知识星球“老 K 星际不迷航”的会员，私下咨询过我不少关于个人职业规划、技术团队管理、低代码方面的问题。**

**感觉上，小 M 是个挺有想法的人，也很勤奋，在知识星球里经常输出、帮着回答其它会员的问题。**

**在小 M 看来，目前他们公司的问题是：整体研发效率不高、业务方想法太多、老板对 IT 不是很重视。**

**在这种环境下，技术经理的工作压力非常大：一方面，业务方觉得系统交付太慢，提任何需求都需要排期；另一方面，老板觉得 IT 的成本太高，一直不愿意增加 IT 投入；此外，技术人员觉得业务的需求经常变，没上线就已经推翻原来的需求了，经常导致返工。**

**小 M 曾经问过我，这种情况应该怎么破局，我就问他：你是为自己，还是为公司？因为不同的目标，就有不同的打法。如果为自己，那就走上层路线，唯一要做的事情就是伺候好老板，只要他在老板那里说得上话，业务方的压力、员工的压力，那都不是事。**

小 M 跟我说，为公司。这让我挺诧异的，职场当中不把自己的利益放在第一位，确实难得，这让我对小 M 高看了两眼。我给他的建议是，抓主要矛盾。小 M 梳理了一下，现在最大的矛盾就是：系统交付慢、业务方满意度低，经常到老板那里投诉。

我建议他研究一下“低代码”，用两条腿走路：一、把有限的研发资源，投入到核心业务系统的研发当中，跟业务方一起做需求治理；二、用“低代码”承接非核心业务系统的开发需求、或创新类需求。

于是，小 M 经过一番调查，选择了国内一家“低代码”供应商，最先把 OA、CRM 系统用低代码进行替换。在供应商的帮助下，顺利完成了系统搭建、用户权限打通、还做了数据迁移，基本上算是用起来了。

渐渐地，业务方觉得“低代码”平台挺好，调整流程、加个字段，不再需要排期，上午提的需求，下午就能上线。这个消息很快传到了老板耳朵里，老板两眼放光，让小 M 赶紧给他汇报一下，这个低代码究竟能够做什么。

经过几次汇报之后，老板总算明白了低代码到底是干啥的，于是他非常支持小 M 的想法，希望他把尽可能多的系统用低代码平台来实现。小 M 听了之后，很高兴，连忙跟技术人员一起梳理，在后来的 3 个月里加班加点，又把工单系统、供应商管理系统陆续迁到低代码平台中。

也就是说，除了核心业务系统之外，能迁移的系统都完成了迁移。原本小 M 的计划是，之前负责这些系统的技术人员，把更多的精力投入到核心业务系统的开发当中，底层架构做个升级，顺便还技术债，再丰富一下接口，让核心业务系统的健壮性、代码可维护性、可扩展性得到进一步提升。

但是奇葩的事情发生了，上周被老板叫到办公室，HR 也在场，让他把非核心系统的开发人员列一个表，小 M 也不知道老板的葫芦里卖什么药，就把名单整理了出来，总共 30 人。

第二天，名单上的人员就都接到了 HR 的裁员通知，给了 N+1 赔偿，要求当天办理离职和交接。

小 M 完全蒙在鼓里，非常气愤地找老板理论，老板笑脸相迎，说是董事会的决定，自己也是刚刚得到通知。小 M 不傻，一听就知道被老板当猴耍了，老板的骚操作搞得他非常的被动。

但是，眼下也顾不得生气，赶紧联系行业的朋友，挨个给被裁撤的员工找下家，我也帮小 M 内推了好几个岗位。

一周后，小 M 在微信上跟我说，本来一心为公司降本增效，才引入低代码，通过梳理核心系统和非核心系统，优化 IT 投入产出比。哪里知道，老板在背后给他来这么一手，搞得他的处境非常被动，现在团队的气氛也变得非常微妙，没被裁撤的员工，也纷纷在找下家。

我说，求仁而得仁，又何怨？你当初的目的不就是为公司降本增效吗？

小 M 沉默了，其实道理他都明白，这是他的职责所在，但是他过不了自己这一关，30 几个下属都是他一个一个面试进行的，现在却要一个一个地送走，换了谁不难受？

摆在小 M 面前更严峻的问题是，今后在公司将如何自处？老板明摆着是对他不信任的，连裁员这样的事情，他也是最后一个知道的。下属们也没有了安全感、归属感，研发效率上肯定会受到影响。业务方已经尝到了“低代码”的甜头，反而对 IT 的容忍度变低，应该会变本加厉的在老板面前捅刀子。

小 M 的选择其实并不多，其实经历了这个事情还是利大于弊的，虽然感情上接受不了，但是至少让他看清楚了老板的真实嘴脸，否则等到公司临上市前被扫地出门，不是更惨？而且积累了低代码的实施经验，简历上也加分了，塞翁失马，焉知非福。

小 M 的故事就聊到这里，接下来聊聊关于低代码方面，大家关注的几个问：低代码究竟适合用来做什么？不适合做什么？低代码会不会彻底干掉程序员？

## *01*

### 低代码适合做什么？

从小 M 实施低代码的故事当中我们了解到，除了业务核心系统之外，他们几乎把非核心系统都用低代码实现了。

这种用法其实是低代码非常典型的应用方式，一般来说企业不会用低代码从零来开发整套核心业务系统，比如 ERP，因为如果你需要一套 ERP，直接购买成熟的解决方案就可以了。除非你的情况跟特斯拉一样，公司的业务模式跟同行业有很大区别（传统汽车销售是经销商模式，特斯拉是直营模式），那就另当别论了。

就目前而言，低代码不适合用来从 0 到 1 构建厚重的企业核心数字化系统，低代码更适合基于核心数字化系统之上，构建创新类应用、敏捷运营类应用。

有个更形象的说法，低代码更适合做企业数字化建设当中“最后一公里”的事情。

从小 M 的案例当中可以看到，这样的应用方式极大提升了技术研发和系统交付的整体效率。

低代码当前正在处于“技术成熟度曲线”的创新萌芽期，随着低代码平台的日益成熟，模板越来越丰富、生态越来越完善，未来低代码的适用范围也会得到扩展。

当前比较适合采用低代码来开发的 5 类应用有：

1、企业门户。包括 App、小程序、PC 门户等等，数据都来自中台、后台，企业门户只是做展示，以及简单的互动。

2、数据操作及展示应用。通过连接企业的数据库，把生产经营的数据进行编辑删除查询等操作。

3、基于表单的应用。基于数据库的表单收集、处理、统计类应用。

4、业务流程应用。定义复杂的工作流，跨部门协作流程，复杂审批流程，比如：OA、人力、财务等系统。

5、移动端应用。基于已有核心生产经营系统，进行移动化的应用场景。

讲完低代码适合做什么，我们再来聊聊：低代码不适合做什么。

## 02

### 低代码不适合做什么?

如上文提到的，就目前而言低代码不适合用来从 0 开始构建厚重的企业核心数字化系统，因为受限于模板、生态、可扩展性。

低代码毕竟不是高级开发语言，因此对于界面效果要求特别高、复杂的算法和数据挖掘、高性能和复杂系统架构、要求较高的底层开发等等，都不适合使用低代码。

总结一下，不适合用低代码开发的应用场景：

1、构建厚重的企业核心数字化系统。比如构建一套完整的 ERP、CRM 等等，有这类需求的企业，更适合购买专业的企业核心数字化系统。

2、对界面效果要求较高的应用。比如短视频应用、交互酷炫的游戏。

3、复杂的算法和数据挖掘。虽然低代码可以处理复杂的业务逻辑，但是不适合用来处理复杂算法和数据挖掘，这类应用应该采用更专业的 BI 开发工具、机器学习平台等工具。

4、高性能和复杂架构。许多互联网巨头的并发量动辄千万、上亿，为了优化性能需要做很多措施，如服务化、中台化、集群化、云化等等。低代码是相对标准的：界面层、逻辑层、数据层的架构模式，无法应对高性能和架构灵活性。

5、要求较高的底层开发。如设备、硬件接口、驱动程序等等，这类开发工作可能 C、汇编语言更适合。

## 03

### 低代码会彻底干掉程序员吗？

从小 M 的故事中可以看出，低代码能够把程序员从技术含量较低的 CRUD 工作中释放出来，去做一些更有价值的架构优化、底层服务升级、复杂算法方面的工作。

从低代码的发展趋势来看，越来越多的程序开发工作，将会通过低代码来完成，低代码干掉程序员的事情已经在发生了。虽然，小 M 的故事有一定的特殊性，是在特定的公司，特定的领导，特定的业务场景下才会发生。

但是，危机给了我们足够的警示，需要我们冷静面对、仔细思考，K 哥结合自己的思考以及同行的建议，总结了 5 项应对策略，希望每一位从事软件开发的朋友引起足够的重视：

1、警惕重复性编码工作。如前文提到的，低代码已经能够完成复杂业务流程类开发工作，能够基于表单驱动、模型驱动的方式进行软件开发工作。最先被替代的就是那些技术含量低的重复性编码工作，如果你是一个 CURD BOY，请立刻停止低水平的重复劳动，否则离被淘汰的日子不远了。

2、在低代码不擅长的领域深耕。低代码因为受限于模板、生态、可扩展性，因此对于界面效果要求特别高、复杂的算法和数据挖掘、高性能和复杂系统架构、要求较高的底层开发等方面工作还不能胜任。所以，这类技术含量较高的工作，就目前来讲是相对安全的，建议程序员们在这些领域进行深耕，提升职场核心竞争力。

3、远离短视的老板。如果你的老板对技术不重视，没有把企业数字化能力看做公司的核心能力，建议你尽早远离。因为，随着低代码的不断成熟，越来越多的业务系统研发工作将由低代码平台来完成，短视的老板会更看中投入产出比，像小 M 公司里发生的事，在行业里会越来越多。

4、永远不要停止学习。技术在不断演进，工程师的工作方式也在不断发生改变，只有不断学习新的技术才能跟上时代的发展。一旦停止学习，就不再适合从事一线技术开发工作，所以，如果你热爱技术，那就成为一名终身学习者吧，这是唯一的出路。

5、启动职场 B 计划。以前在许多文章里都写过“ABZ 计划”，这里就不多介绍了，“ABZ 计划”不仅给了你职场上的保障，还能够帮助你找到一生热爱的职业，比如《三体》的作者刘慈欣，脱口秀演员呼兰、庞博，都是通过 B 计划转 A 计划，找到了一生热爱的职业，实现了人生跃迁。

### 结语

最后，本文的目的并不是想引起焦虑，相反，当我们察觉到了危机的来临，提前做好了应对的准备，就没有什么好担心的了。拿破仑说：“伟大的人之所以取得成功，都是因为他们懂得顺应法则，懂得努力适应一切障碍。”多说无益，干就完了。

## 尾语

有人瞌睡，就会有人送枕头。有人看到了软件开发过程复杂、漫长的痛点，于是做出低代码应用平台。有人看到了低代码应用平台在使用门槛和灵活度上的不完美，于是做出“极致”的零代码应用平台，让“Less is more”的“Less”做到“Least”。

软件行业作为一个反脆弱的系统，必然会生生不息，不断迭代。“零代码”不会是软件开发的句号，总有一个新名词会取代“Least”。我们一起拭目以待，距离下一个替代零代码的“枕头”出现，还会有多远。

更多技术干货资料下载，请关注“技术领导力”、“BAT 架构”公众号

技术领导力

BAT 架构

知识星球：老 K 星际不迷航