区块链是目前一个比较热门的新概念，蕴含了技术与金融两层概念。本文以联盟链为例，简单描述了实践一个联盟链的基本过程。

首先要确定这个区块链的类型，是公证型区块链还是价值型？

公证型区块链是指仅限一些关键数据自证、披露、防篡改等功能的区块链，通常是在价值型区块链中附带的功能，也可以单独扩展，用于公示公开等。价值型区块链是指可以进行资产所有权转移的一种记账账本。

如果确定是价值型区块链，我们又需要确定目标区块链的总体定位：到底是一个普适的价值传输区块链，还是特定场景下的区块链？如果是特定场景下的区块链，我们通常推荐超级账本作为技术原型，如果是比较通用的价值区块链，我们推荐以太坊的思路。

业务场景的构建与初步分析

首先要明确的观点是，区块链不是万能的。很多场景其实是不需要区块链技术也能解决的。像跨境支付领域，区块链能很好的发挥是因为存在很多点对点的跨境机构有大量的支付清算需求，而又不希望中间机构参与，区块链是很好的选择。但是在一些集团内部，大型公司内部，区块链解决方案基本上远远不如传统的企业资源解决方案。

A、需求痛点分析

一般需求痛点在满足以下条件的时候，可以考虑使用区块链：

存在一个不相互信任的P2P网络环境；

节点之间是对等的，不存在一个绝对仲裁者；

节点之间是博弈行为。

P2P网络可能包含输入和输出，当包含输入和输出时，区块链不再封闭。

对于某个节点一般有以下几种行为（包括但不限于）：

不信任其他节点；

保证自己的收益最大化；

自私获取但不贡献资源。

针对以上情景的业务建模，需要针对具体的业务逻辑结合博弈论推导出满足自己需求的方案。

B、非区块链技术能否解决

案例1：

通常我们有不同的机构A、B、C，存在不对称的信息交换需求，即A、B、C分别具有部分数据，但三者组合到一起具有市场的全量数据。但是作为A，想知道B、C是否拥有自己数据集合中的某个点数据，根据这个结果来调整自己的购买策略。

案例2：

有不同的机构X、Y、Z，存在信息反馈的需求，当Z收到Y的服务时，会给Y一个信息反馈，这种反馈可能是信用评价，也可能只是响应反馈。总之这种反馈需要记录在案，X会根据Z的信息反馈结果调整自己的购买策略。当X购买服务时，同样会给Y一个反馈，Z也会收到反馈。

以上两个案例首先考虑使用非区块链是否可以解决：

针对案例1，敏感数据和私有数据是不会公开的，即使加密也不会被允许上传到区块链。所以产生了一个数据输入输出区块链的过程，该过程是区块链不可控制的。

那么使用传统的技术是否可以控制呢？貌似也不行，能够满足不暴露敏感数据的方案也只有HASH计算和同态加密。但是这两者都要求数据传输到指定位置。

通常我们会考虑使用零知识证明作为解决方案，然而具体的算法可能需要根据具体业务逻辑进行构建，结合简单智能合约，根据查询结果产生不同输出。

针对案例2，反馈信息容易被篡改，可刷单等问题是最大的，如何保证这种信息反馈是客观中立不可篡改的，可以结合区块链代币的币龄使交易具有方向性来防止作弊行为。

业务场景建模

针对第二节中的两个案例，我们接下来要进行建模，除去核心痛点，我们必然还有记账的需求，本质上任何案例中每个节点都既是服务方，也是客户方，那么怎么衡

量自己贡献和索取了多少呢？

所以任何区块链平台上，必须是要有代币系统的，否则记账将非常困难。在业务场景建模过程中，我们主要关注如何使节点之间达成帕累托改进，而不是因为每个节点是自私行为，让区块链服务名存实亡。

开发路径

A、区块链原型选取

根据本文开头的叙述，如果是特定场景的区块链解决方案，建议Hyperledger fabric，当然搭建以太坊私有链也是可以的。下面是一些以太坊和Fabric的比较：

以太坊与HyperLedger相同点：

都是提供区块链业务实现的平台，业务实现都是通过智能合约来完成，以达到最大的灵活性和对底层的不修改。

以太坊是：EVM虚拟机，Solidity合约语言；

HyperLedger是: Shim链码容器，用GO编写合约。

官方版本都使用GO语言实现。

因为都是提供第三方可编程能力，由于难度大，内部难免存在漏洞。对外则存在恶意程序攻击的威胁。尤其是在做为公有链时，威胁将会更大。上个月以太坊已有报合约solidity语言漏洞。

以太坊与HyperLedger不同点：

以太坊只提供智能合约能力。也恰好吻合它的定位：智能合约和去中心化应用平台。对系统安全性或准入机制无底层无核心上的支持。而HyperLedger在吸收以太坊智能合约特点的同时，提供MemberShip及身份验证角色管理等模块，更贴近商业应用场景。

共识机制不同。由于共识的不一样，所以每秒可处理的交易量也不一样，以太坊是每秒千级别的处理量，而HyperLedger可以达到十万级别。

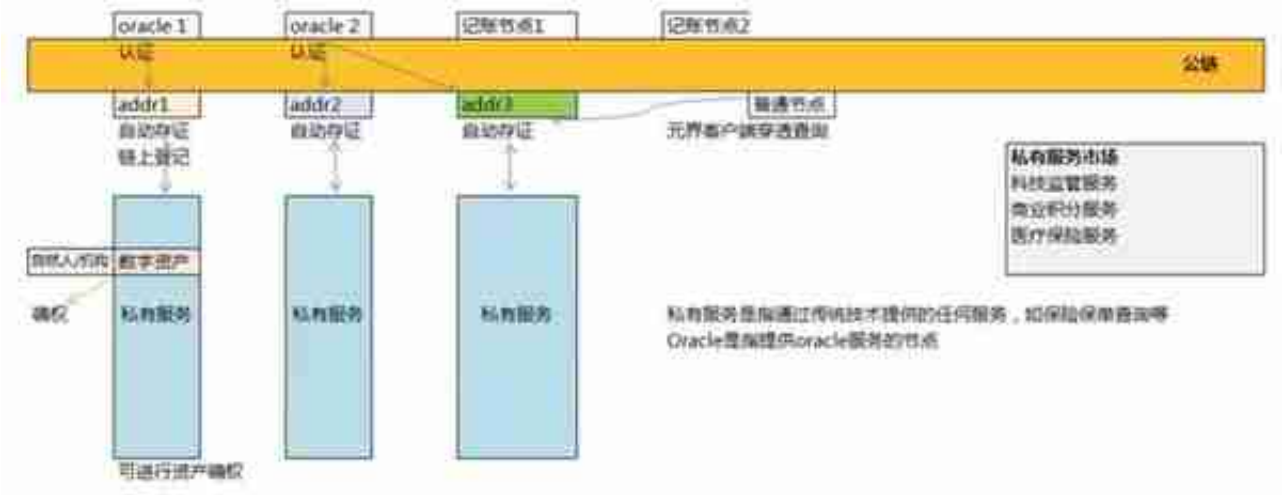
采用的技术实现思路上不一样。以太坊更多的是靠自己实现，自己造轮子，有点开

发人员炫技的感觉，如自己提供合约语言solidity，自己实现EVM（这个可能是实际需要）。

当然，如果考虑自行开发，建议搭建基础比特币网络，做加法，更改共识算法，网络传送协议以及附加合约（可选）。

其实智能合约在一些场景中不是必选项，对用户来说，可靠方便实时是第一需求，如果针对特定的应用场景，将“合约”固化在区块链里面，也是一种可行的思路。

针对以上两种联盟链实现，笔者还想强调，并不是所有服务一定得是区块链的，笔者构想了一个通用的保护伞型结构，如比特币的侧链技术，主链提供基础账本服务，侧链提供特定场景服务，侧链上的应用可以是非区块链实现的，只需接口注册即可。



B、交互接口设计

在交互接口设计上，推荐使用目前业界通用的Json-RPC接口，扩展性和友好性兼备。

一般我们将接口分为两类：开放接口和账户接口。开放接口是指区块链本身的描述信息，是不需要认证的，而账户接口是需要账户认证的。

C、基础账本设计

基础账本设计包含以下两个问题：

首先是原型区块链是否已经满足需求？如果针对以太坊，基本上不需要改动基础账

本，只需构建智能合约即可。如果以比特币体系为基础，则可能有较大的改动。

不满足需求时如何改动基础账本？这个其实要视账户模型而定，如果使用UTXO模式时，改动重点在如何嵌入模板交易体。如果使用Balance模式，那么则没有这个问题。

D、业务扩展层设计

业务扩展设计方面的内容比较复杂，篇幅问题这里也只是抛砖引玉提出两个问题：

1. 扩展层是外接区块链还是内置到区块链？

2. 如果包含数据输入，是否需要脱敏？脱敏后如何上链？

先想清楚这两个问题或许能帮你更好地规划业务扩展层的内容。

开发转变和难点

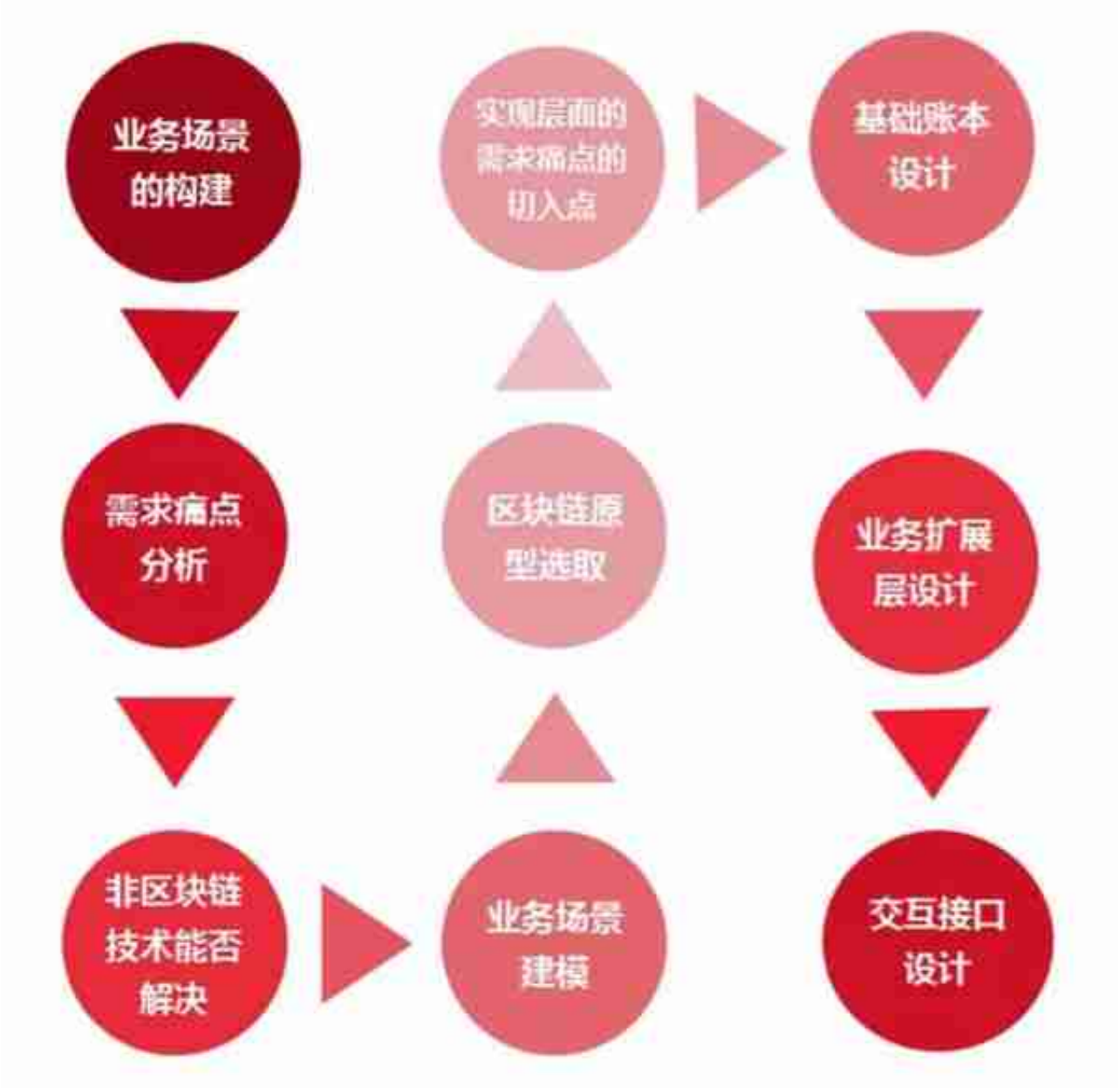
A、开发思维的转变

与传统网络服务不同的是，区块链开发不再以面向服务为主要关注点，而是面向账本和交易。

开发者面对的不再是以高可用高并发的应用程序为主要指标，而是切换到了面向用户，关注用户友好性和开发扩展性的终端程序开发。

所以高并发高性能不再是区块链终端的核心指标，安全性、可扩展性、友好性成了主要指标。

图2是一个适用于联盟链/私有链项目的工作流程。



B、开发难点

目前来讲，区块链项目开发的难点有三个：

1. 开发人力资源储备不足

目前比较成熟的技术体系有比特币及衍生技术体系、以太坊、超级账本HyperLedger fabric、比特股Bitshares、瑞波Ripple和未来币NXT。其中前三个是最有影响力的区块链项目。比特币以及衍生技术多以C++语言进行开发；以太坊支持大部分主流语言，官方以Go为主，也有其他分支的项目如Rust语言的Parity钱包；超级账本目前以Go为主。

从目前上海地区的区块链从业人员来看，保守估计在400~500左右。按一半为开发人员计算，也才200多个，面对巨大的市场需求，人才是极度稀缺的。

由于C++目前仅在金融和游戏领域有部分需求，所以C++工程师不多，尤其是高水平的C++工程师就更少了。Go作为新兴语言，发展势头很猛，但是Go的生态也不如Java大。

如果从Java的角度看，如何把其生态利用起来，目前区块链还没有做到那个地步。

综合来看，区块链在技术方面与其他技术的结合还有待探索。

2. 区块链是交叉学科，需要各方面工程实践的经验

在实践方面，我们希望区块链从业人员同时了解技术和金融业务，这个对人员的素质要求比较高，相应的符合标准的人就更少了。

3. 关于对各个区块链技术体系理解的偏差。区块链技术和概念日新月异，闭门开发可能会走到死胡同，如何保持一部分精力更新知识体系，同时保证开发进度对开发人员是有较大挑战的。

区块链作为一门新兴的技术，涵盖了去中心化、去信任、共享经济、分布式计算、分布式存储等多方面的内容，考验着技术人员的学习和思考能力。在未来，区块链将同人工智能一起，会影响到普通人生活的方方面面。

作者简介：陈浩，维优区块链CTO，曾任埃森哲高级软件工程师，擅长C++/Python语言，多年清算支付系统开发经验。万向区块链实验室2016上海区块链黑客马拉松比赛第三名，开源项目Libbitcoin开发组成员。