文章
Qiao Peng · 九月 22, 2022 阅读大约需 25 分钟

精华文章--漫谈应用集成的现在与未来

关注FHIR的大侠们估计都注意到了，FHIR更新了它支持的互操作范式，除了消息、文档、服务、API这4种，增加了2个：资源仓库、订阅。前面4个好理解，为什么资源仓库和订阅会成为FHIR的新的互操作范式？互操作与应用集成是什么关系？

这里借FHIR的新互操作范式，聊聊应用集成，看看集成平台是什么？有什么样的集成方案？以及怎么评价不同的方案。

一. 什么是应用集成？

集成的概念应用广泛，往往被滥用和误用。例如数据集成，传统意义上是指通过ETL（数据抽取、转换和加载）实现的数据集中；界面集成是指通过单点登录解决人工用户跨应用访问的问题。它们都和我们讲的应用集成是两码事。

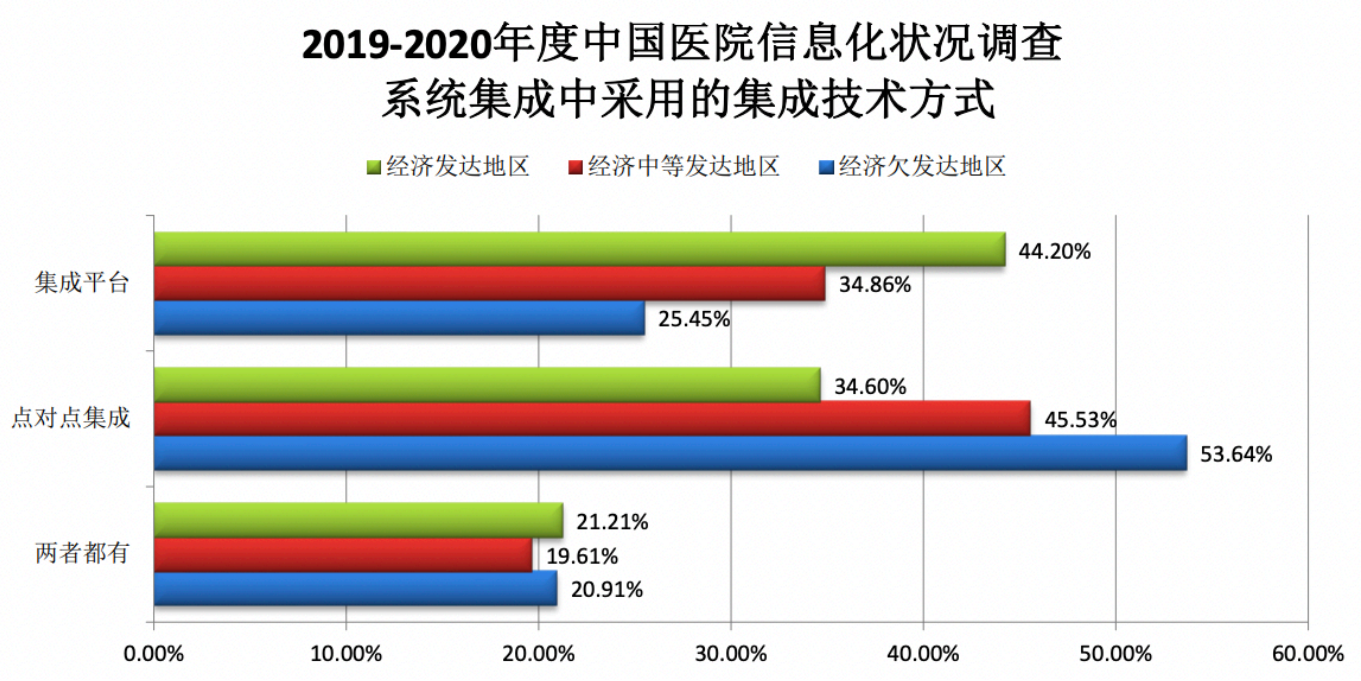
那什么是应用集成？

应用是一系列业务功能的组合。传统应用基本都是单体架构应用，无论是哪种软件架构开发的。单体架构应用的特点是独立运行、业务功能边界清晰。正因为不依赖于其它应用就可以完成业务边界内的功能，通常单体架构应用并不具有完善的应用交互能力，例如完善的接口。当单体应用间在特定业务流程上需要协同工作时，例如医生站下达的检验医嘱需要立刻通知到检验科信息系统，就需要把这2个或多个应用集成在一起自动实现业务流程协同工作，这就是应用集成。

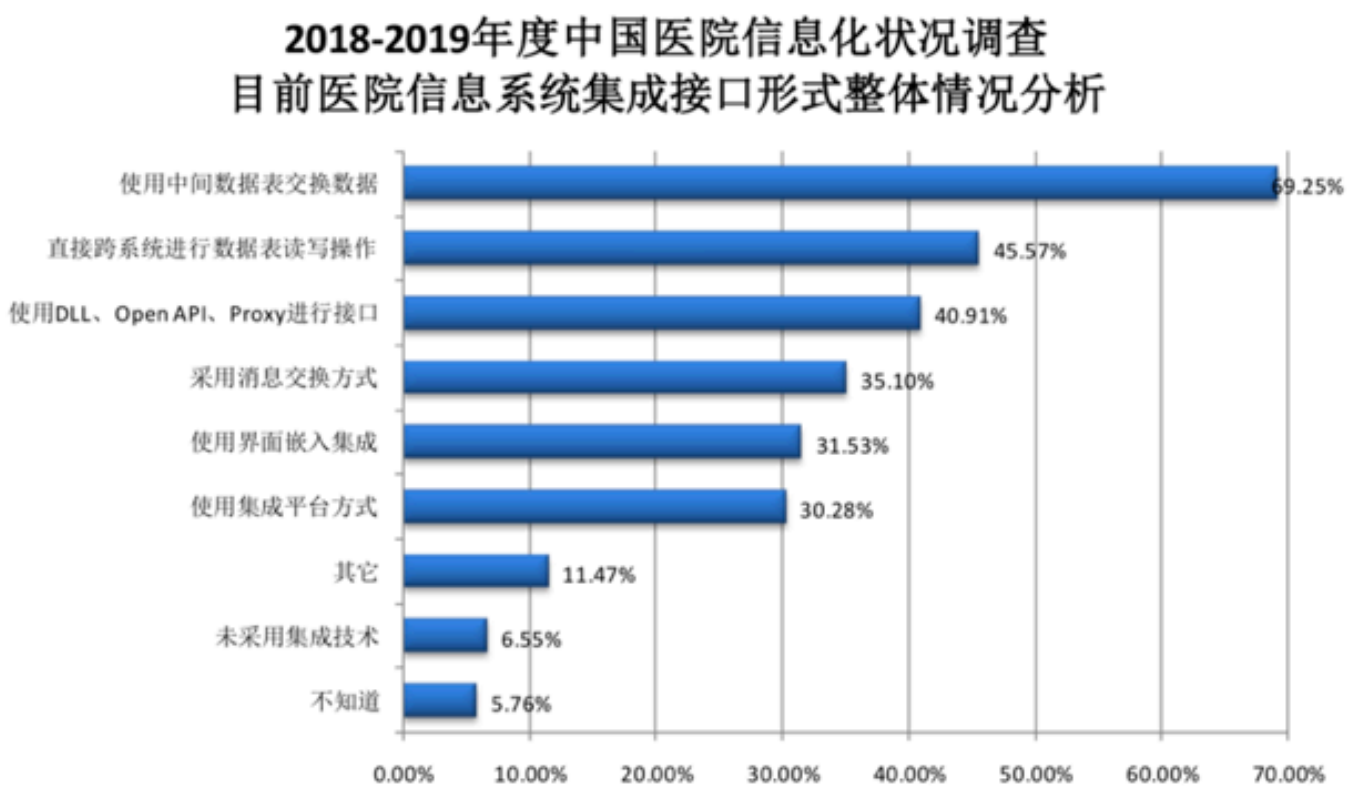
既然集成的是应用，应用集成架构是随开发应用的软件架构发展而发展的 -
这就是上一篇《从软件架构发展谈业务集成技术演进与展望》的主题。

要实现应用集成，双方应用必须解决接口问题、采用同步还是异步通讯问题、数据格式互相理解问题、术语统一问题、出现数据与业务异常如何表达异常问题、互相调用接口出现网络等错误时是否重试以及如何重试问题...
所以完善的应用集成并非易事。

早期解决这些问题是通过点对点的方式，从《2019-2020年度中国医院信息化状况调查报告》来看，即便在互联互通成熟度标准化评测已经开展了8年的今天，在经济发达地区，点对点集成的方式仍占比高达55.81%（34.6%+21.21%）。



《2018-2019年度中国医院信息化状况调查报告》揭示了更详细的应用集成方式统计：



由于没有中间层，点对点集成的双方必须事先同意接口方式、同步/异步通讯方式、数据格式、术语、响应异常代码等，同时要处理包括通讯异常在内的各种连接故障和异常。而且对于复杂的业务流程，一个应用可能需要同时与多个应用对接以交换数据、调用接口，开发变得非常复杂。在应用数量不多的情况下还能勉强支撑，对于动辄数十甚至上百个业务系统的行业环境，点对点的开发成本、运行风险、排故难度都非常高，而灵活性很低。这也是集成平台诞生的原因。

二. 如何做应用集成？

如何做应用集成？

无论什么应用集成方式，重要的都是梳理和实现应用**集成的三个要素**：**事件、上下文、流程**。

2.1 集成三要素

•事件

事件是数字化表达的业务行为记录，是应用间业务流程的触发点。它在业务节点上，触发应用之间集成的流程。通常事件是由上游业务系统的操作触发的，例如医生在医生站下达药嘱，即药嘱事件。

仅由这些基础业务系统的事件驱动业务，往往是不够的。例如如果这个药嘱有用药风险，是不是要提示，这个提示其实是一个临床决策事件。灵活的应用集成方案应具有事件生成能力：基于基础业务事件产生决策事件的能力。

基础业务事件需要梳理，要具有足够的业务覆盖度，且应该与被集成的应用无关。也就是业务决定了业务事件，更换任何一家厂商的应用，都不会影响医院有这个业务和相应的业务事件。行业集成标准通常都梳理了基础业务事件，例

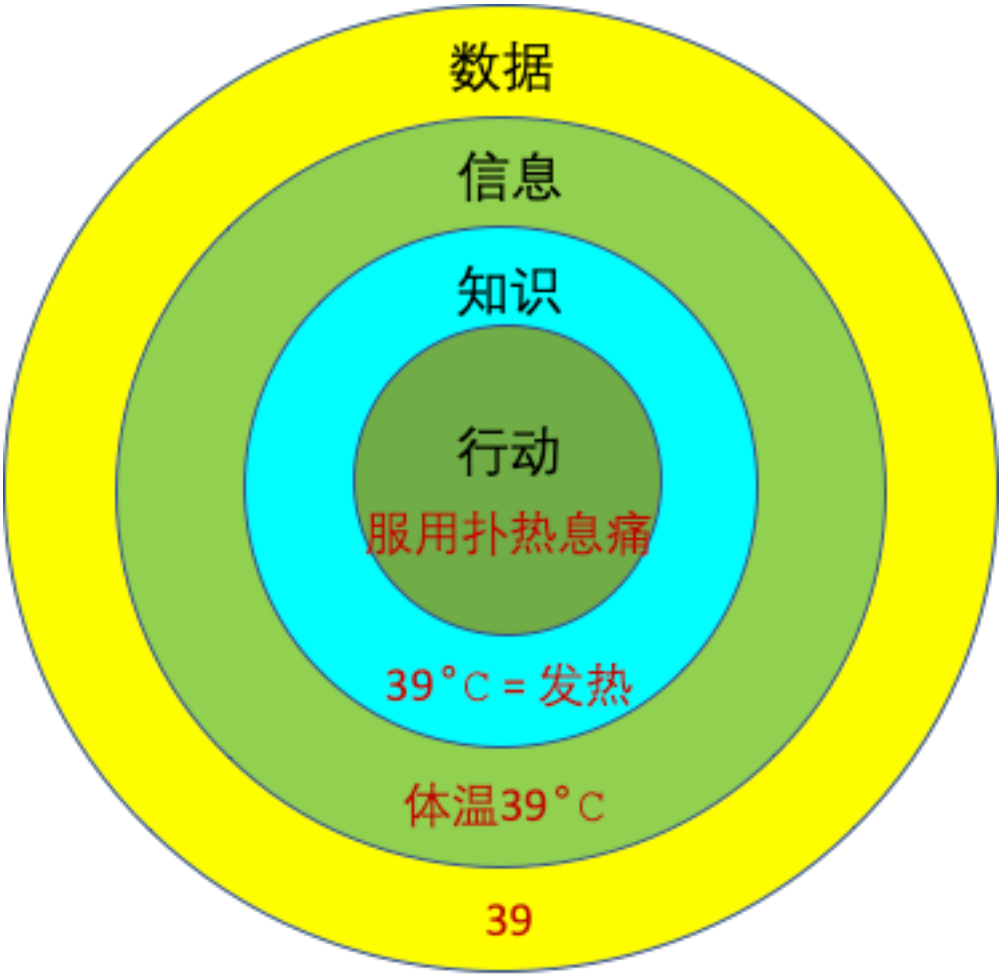
如HL7 V2、 HL7 V3消息里都定义了业务事件。

在项目方案中，集成平台如何实时捕获这些业务事件？这些事件通常是业务变化时在应用中可以捕获的信息变化，集成平台通常可以通过这些方式得到这些信息变化：

1. 应用主动通知，例如应用调用集成平台发布的SOAP服务。这种方式说明被集成的应用具有基于标准的接口能力，或需要在项目现场改造。
2. 被动由集成平台发现，例如平台通过SQL适配器发现应用后台数据库的数据变更。

•上下文

被集成应用间，需要传递的
不仅仅是数据，而是能表达业务行为的完整信息，我们
称之为**上下文**，或者**语义**
。就像只给一个值为39的数据是没有意义的，需要完整表达患者的额温是39摄氏度这样的信息。这仅仅是数据项，还需要通过数据模型表达数据项之前的关系和完整度，例如表达体征记录事件的数据模型，不仅要包含体征的信息，还要表达是哪个患者的、是发生在哪次就诊过程的...



能完整表达这些信息的数据模型才是我们说的上下文。

这个数据模型应该具有业务场景的普适性，即这个模型应该能普遍表达业务场景里业务实体的数据，而不是特定应用要求的那些数据。例如在医疗行业，共享和交换患者的基本信息，它的模型应该能满足大部分业务场景对患者的数据需求，而且应该尽可能只通过一个患者模型就可以满足多数场景的需求。原则是不能仅根据项目需求去谈数据需求，否则模型大概率会随业务需求的轻微变化而发生重大变化，从而影响整个集成环境的稳定。

•流程

流程是被集成的应用边界之间的信息流程。梳理流程是做应用集成的基础。例如，医生站下达的药嘱，要先给药品知识库系统判断风险，在没有用药风险后再发给发药系统，这就是典型的应用间集成的流程。随着越来越多决策支持系统的应用、流程闭环需求的增加，跨应用流程建模与管理越来越重要。

流程可以通过改造各个应用，由各个应用实现，但这样流程被割裂在各个应用中，难于理解和再造。因此流程通常会由集成平台实现，从而实现：

- 流程的统一建模与管理
- 图形化、直观
- 灵活的流程调整和再造

流程如何建模？可能完全通过代码实现、可能通过设置消息路由规则、也可能通过图形化的业务流程建模。建立流程，还要考虑调用是同步还是异步？使用什么流程架构，例如通过事件驱动架构（EDA），来实现更灵活、松耦合的应用集成流程？

另外，在应用系统边界之间，特定的流程节点可能需要人工干预，例如新冠大流行期间，门诊分诊时对没有24小时核酸的患者，护士需对其进行抗原检测，这个流程并不在HIS系统中存在，是当前应用之外的人工工作节点。
这些人工工作，也应该被流程建模实现。

2.2 应用集成的级别

业务集成，可以做的非常简单、也可以做的更完善，取决于集成的目标 - 要做到什么级别的应用集成。

HIMSS 将**应用集成**定义为4个不同的级别：

基础级别，仅仅打通了系统间进行数据通讯的通道，也就是接口。

结构级别
，在基础级别上，定义了数据交换的格式和语法；在这个级别上，通常这些交换的格式和语法是集成的双方坐一起讨论出来的，换一家应用集成，大概率格式语法就不同了。

语义级别
，建立在行业通用的基础模型和数据编码上，使用标准化的行业语义来定义数据元素、使用标准的值集。语义级别的信息集成是全
行业可以理解，并有确定行业意义的信息集成级别。或者说语义级别的信息集成才是基于**标准**的信息集成。

在应
用集成领
域还经常说“ 互操作
”这个词，语义级别的应用集成就是互
操作。这也是互操作和应用集成的差异！后面我们会更多使用互操作这个词，表达语义级的集成。

组织级别，
通常都是由国家、国际行业协会和行业标准开发组织开发的。
它加入了政策、社会、法律等方面的考虑，分析了通用的业务流程和工作流，在此基础上设定了参与信息集成各方的角色、权限和知情同意策略等。我们的互联互通成熟度标准，就是组织级别的互操作。

2.3 行业互操作标准在集成中的价值和作用

在医疗行业，通常需要的**信息集成**要达到语义级别，才能保障医疗信息准确和医疗行为安全。

要达到语义级和以上级别的信息集成，需要基于或参考标准。这些标准应该是行业级的、五位一体的标准：

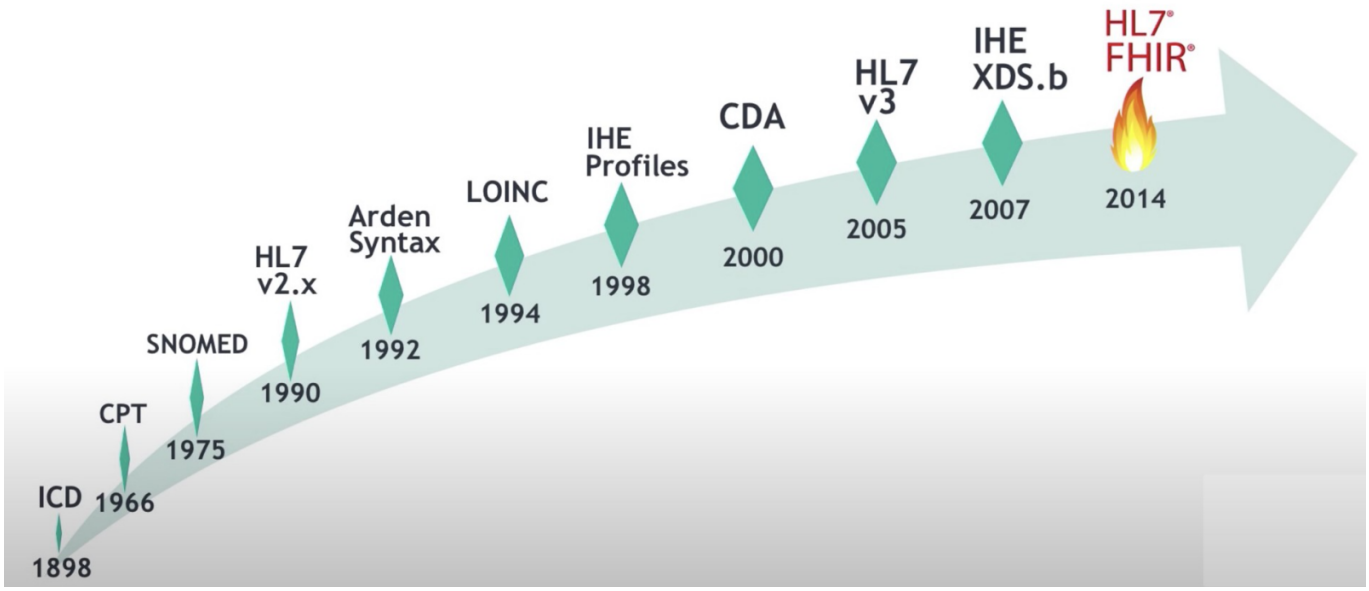
- **词汇/术语标准：**相互通信的健康信息系统依靠

结构化的词汇、术语、代码集和分类系统来表示健康概念。**词汇/术语标准**就是表达健康概念的标准，例如世界卫生组织国际疾病分类标准：ICD-10, ICD-11。

- **内容标准**，描述信息交换中，数据内容的结构和组织。它还包括通用数据集的定义。CDA、HL7 V2消息都是内容标准。
- **传输标准**
  定义了计算机系统、文档架构、临床模板、用户界面和患者数据链接之间交换的信息格式和传输方式。传输方式确定了健康信息交换的“推”和“拉”方式。DICOM、IHE等规定了传输标准。
- **隐私和安全标准**
  是确定谁、何时、出于何种目的、使用哪种个人健康信息的权利，以及如何护健康信息的机密性、可用性和完整性的标准。美国的HIPAA和欧洲的GDPR都是关于隐私和安全的标准。
- **标识符标准**是用来唯一标识患者、机构、医护技、设备等实体的方法。例如咱们互联互通里用到的OID。

行业互操作标准由行业标准化组织发布，是基于对行业业务的梳理与分析，构建用于行业集成的标准。因此这些标准在表达业务上下文的完整程度、业务事件的归纳、业务流程的梳理与适配、可扩展能力通常都比较好，也就是对集成3要素的抽象更加完整和适用。由于行业采纳度上较高，通过这些标准的推行可以达到更一致的建设效果和通用性。

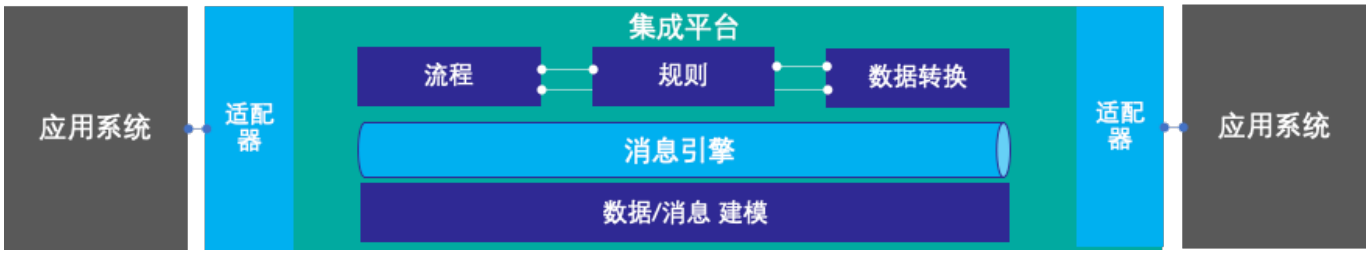
医疗行业的互操作标准还不少，例如DICOM的影像学互操作标准、HL7 V2、V3的临床消息标准、HL7 CDA的共享文档标准、HL7最新的FHIR、IHE的服务标准，还有我们自己的互联互通成熟度标准。



图：医疗行业常见标准出现的历史

3. 什么是集成平台？

集成平台的基础功能就是在不打破现有应用的边界（不改造、少改造应用）的前提下，提供跨应用的业务流程整合和闭环。它当然要解决点对点集成的问题，因此适配各种接口方式、管理同步/异步通讯、数据格式转换、术语注册与转换、处理通讯故障等，都是需要的能力。集成平台，作为一个技术平台，为应用集成工程师提供生产力工具。下面就展开说说集成平台需要什么功能。



图：典型的集成平台功能架构

集成平台要解决以下基础能力：

1. 互联互通（打通应用的连接性）
2. 数据建模和数据转换
3. 流程建模和执行：协同、甚至再造应用间的业务流程，例如业务闭环

另外，集成平台在这些领域往往也被给予众望：

1. 将旧有架构的应用现代化！-- SOA、EDA、DDD（主题域驱动设计）、MSA...
2. 基于已有应用与数据资产的复合应用开发

3.1 集成平台需要什么功能？

集成平台并非是一种技术或一个方法论，而是整合后的一系列与应用集成相关的技术和特性的工具箱。

上面提到了集成平台要解决应用的互联互通、数据建模和数据转换、流程建模和执行这些基础需求。集成平台通常包含这些基础技术/特性以应对基础需求：

1. **连接适配器**
   ：连接适配器是可复用的，高度可配置的，对各种技术、协议、应用的连接组件。真实业务场景中，各个业务应用都可能提供了不同的技术接口、不同的数据协议， 因此一个丰富的适配器库是保证对接口普适性连接能力的重要因素。另外，适配器通常有确保连接、发送的能力，例如连接应用网络异常时需要的重连、重发能力，而不是修改调用方应用使其具有这样的能力。 专业的集成平台产品通常内嵌了行业互操作标准适配器，提供对于这些互操作标准协议的开箱即用的连接、校验、转换、路由的能力，从而极大降低开发和实施成本，提高方案的敏捷性。连接适配器也是集成平台提供广泛连接性的主要方式。
2. **元数据管理**
   ：对上下文、消息进行建模和约束管理等能力。这是集成平台对数据建模和数据转换的能力支撑之一。因为上下文和消息的复杂性，多模型建模的能力尤为重要。
3. **数据转换引擎**
   ：用于对数据/消息模型进行转换，这也是集成平台对数据建模和数据转换的能力支撑之一。数据转换涉及多种数据模型的相互转换，包括XML、分隔符分隔的字符串、JSON、对象模型等，图形化的转换能力尤为重要。
4. **消息引擎/消息总线**： 用于建立消息路由规则，通过消息引擎在应用之间可靠地传递消息。路由规则是集成平台提供的流程建模能力之一，用于简单的流程场景。路由规则通常可以基于消息类型和消息内容进行设置，也就是需要它有拆包分析消息数据的能力。而消息引擎需要具有同步发送、异步发送、重发、编辑后重发等能力。要确保消息发送，还需要消息持久化的能力。
5. **业务流程引擎**
   ：对业务流程进行建模，是集成平台提供的重要的流程建模能力。业务流程引擎通常提供图形化的流程建模工具、使用流程建模语言，从而提供比路由规则更灵活的流程模型，例如流程分支控制、延迟控制、流程异

常捕获与处理，提高流程自动化程度、更适合对业务流程的再造，例如将人工智能加入业务流程。

6. **消息可视化追踪和消息仓库**
：将集成平台上的跨应用传递的消息按业务组织、以可视化的方式展现每笔业务流程历史（包括流程尚未结束的业务）的全貌，通常包括时间、请求消息内容、响应消息内容、状态、错误与跟踪信息等，方便集成工程师快速发现、定位和解决流程问题。实现消息追踪，需要持久化消息，并且可以查询检索，例如通过SQL。而消息仓库可以存储消息，并提供更丰富的消息分析能力。

3.2 与集成平台混淆的概念

对集成平台特性的不同理解也造成了市面上对于集成平台以偏概全或不准确的误读，例如集成平台就是消息引擎、集成平台就是ESB、集成平台就是中间件.....
这里试着讨论集成平台功能的同时，将这些不同技术、概念与集成平台区分开。

3.2.1. 中间件：

中间件是很多产品的统称，是处于源和目标中间的技术构件，因此称为中间件。例如消息中间件（消息引擎、消息总线）、API中间件（API网关）、事务中间件（TPMs)、企业应用集成中间件(ESB)、Web应用中间件（web应用服务器）、数据流中间件...

和应用集成相关的，通常是消息中间件、企业应用集成中间件、API中间件，它们都是广义集成平台的能力组成部分。因此中间件不是集成平台。

3.2.2. 消息引擎：

消息引擎，往往也被称之为消息中间件，是提供消息通道的中间件技术，可以应用在很多架构下，例如流式处理。因为其消息队列管理能力，尤其是消息先进先出、消息路由等能力，通常集成平台都会把消息引擎作为组件，用于消息处理顺序控制和发送控制。

消息引擎本身只处理消息机制，它不主动向目标系统发送消息，即消息引擎并不解决连接性问题。集成平台通常是通过适配器将消息主动地、按目标方要求的通讯协议，连接到并发送给目标方的。因此消息引擎单独做不了应用集成的事情！

另外，市面上很多消息引擎产品，其中一些不支持消息持久化，无法保证消息的发送成功、无法重发；一些不支持消息校验，无法检索分析消息；另一些不支持发布/订阅模式、或多个订阅用户。

3.2.3. 接口引擎：

虽然广义上，接口引擎具有更多能力，但狭义上，接口引擎是通过适配器解决接口连接性的技术，它的目标是使用应用/技术现有的接口能力，建立对其的连接，有可能是单向的、也可能是双向的。因此，接口引擎只解决应用/技术接入的问题，它通常是集成平台解决应用接入的技术组件。

3.2.4. 企业服务总线（ESB）：

这里的服务是面向服务架构（SOA）语境下的服务。什么是服务，和前面提到的接口有什么区别？

接口是指基于某种协议的交互规范的实现，例如基于SOAP的接口；而在面向服务架构里，服务相对于单体应用架构而言的，它是按功能封装的应用组件，可以被别的服务（应用组件）调用、因此可以复用，例如HIS的患者查询服务。接口是服务是现实方式，例如HIS的患者查询服务SOAP接口，服务是业务概念、接口是技术概念。

企业服务总线是随面向服务架构（SOA）发展起来的，用于封装、注册、协同调用和监控服务。SOA封装的服务，

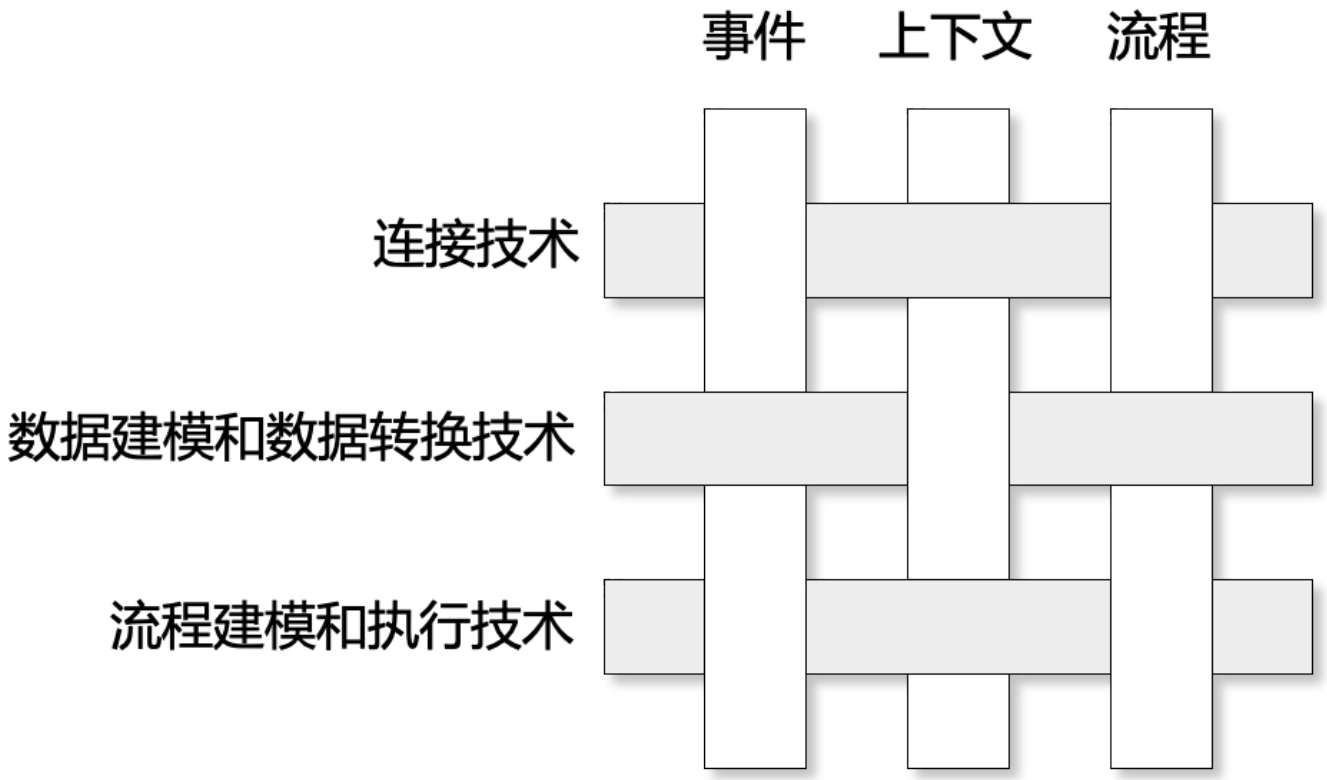
实践中最多的是Web服务：基于XML的数据模型、基于WSDL的服务定义、基于SOAP的服务接口。但理论上服务并不一定是SOAP服务，也可以是HTTP服务、TCP服务、文件服务...

ESB既然要协同调用服务，通常都具有集成平台的核心能力，例如流程建模、数据转换、适配器等。这是最容易和集成平台混淆的概念，甚至一些大厂在不同场景下把其产品同时称之为集成平台或ESB，差异在于集成平台不必要基于SOA架构、不必要封装服务。

4. 集成方案与评价

应用集成项目的效果和实施有很大关系，且和底层采用的集成平台技术相关。评价它的因素太多了，例如性能（消息吞吐量）、高可用、持续集成能力、建设成本.....

而我们这里关注集成3要素和集成平台基础能力，这些集成的核心需求来考察集成方案。



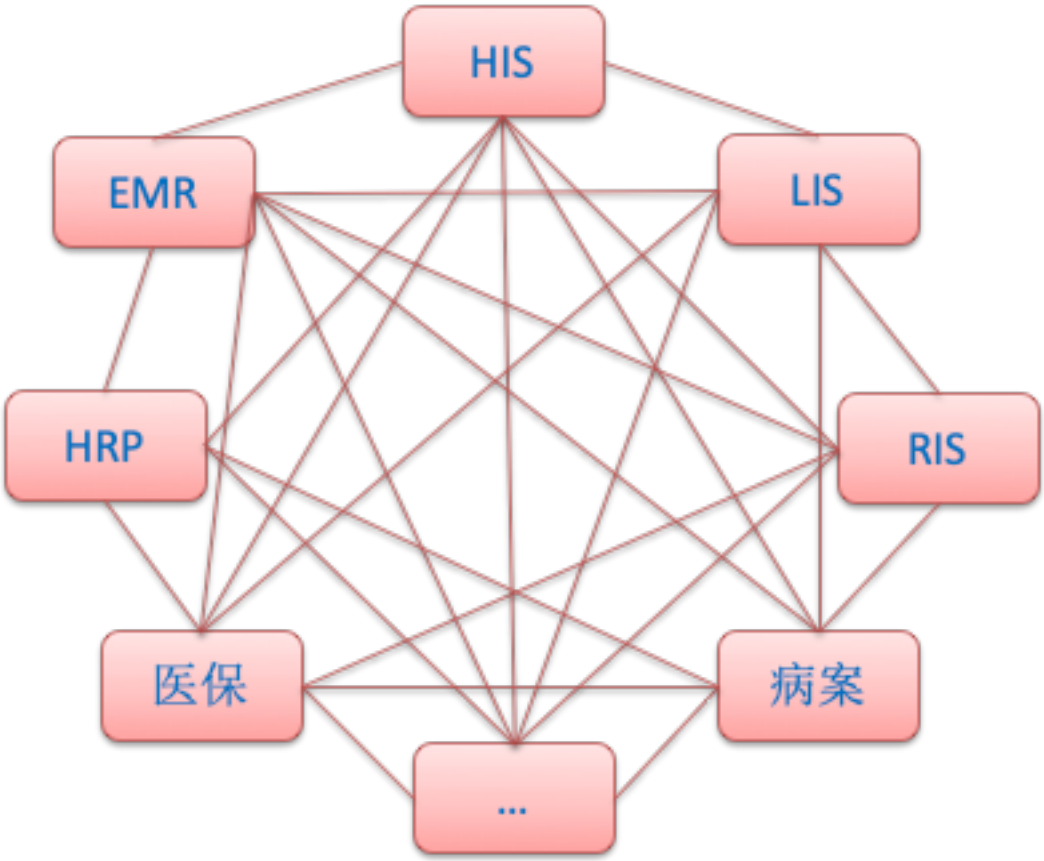
图：集成三要素与集成平台基础能力

即无论什么样的方案，都需要留意是否关注在集成3要素上，是否解决了3个核心问题。为分析市场上常见的集成方案，我们把不基于集成平台的常见集成方案也纳入进来。

4.1 基于点对点接口

不依赖于集成平台。应用需要大量的接口改造工作，复杂度随被集成应用的数量呈指数增长。因为通过各自应用的改

造，事件和完整的跨业务边界流程被割裂在各个应用中，没有地方可以一览应用间的流程，可以说缺少流程和上下文梳理。虽然它目前仍是应用集成的主流方式，但其固有缺点使其很难持续部署在日益复杂的多应用业务场景下。



4.2 基于中间表的数据交换

使用中间表的方案是上面点对点接口的一种特例，其思路是通过SQL这一种接口单向打通应用：将需要传递的信息放在中间表里，数据用户在业务流程节点上去中间表获取信息。 这种方式的流程自动化程度通常较低，适合的业务场景也比较有限。例如医生站下达了检验医嘱，医嘱并不会推送给检验系统；患者拿着检验申请单到检验科，检验系统通过扫申请单二维码获取申请单编号，并在这时去医生站开放到检验医嘱中间表获取完整的医嘱信息。
“ 事件 ”、“ 流程 ”是在患者人工参与下完成的：
人工传递医嘱，将前段业务流程（医生下达检验医嘱）和后段业务流程（检验科处理医嘱流程）关联起来；
“ 患者到达检验科窗口 ”是后段流程（检验科处理医嘱流程）触发事件。

4.3 基于消息交换

“ 基于消息交换 ”是主流的集成方案，通常是利用消息路由作为“ 流程 ”的承载机制，通过消息路由规则解决集成三要素中的“ 流程 ”问题。
而消息本身就要承载另外二个要素：事件和上下文了。因此对于消息标准的选择是影响方案的关键因素之一。

这类方案要关注:

- 是否解决了应用连接问题。如果仅是一个消息引擎，而要求被集成的业务系统都需要被改造以连接到消息引擎去发送消息、轮询消息引擎获得自己需要接收的消息，那么这个方案的成本就非常高了！
- 是否有平台统一的消息标准。每个业务系统对相同的业务对象 - 例如患者 - 要求不同的数据，如果在平台上没有做标准化，会在平台上传递太多的消息类型，而且都是残缺的信息，这些消息的管理成本太高、而消息的数据价值很低。作为上下文载体的消息，其对业务的抽象能力非常重要。

4.3.1 基于行业互操作消息标准

行业互操作消息标准对业务事件和上下文都有梳理。例如，在对事件的梳理和实现上：

HL7 V2的消息名称代表了业务场景和业务事件。例如ADTA01消息，ADT代表患者管理业务：入院（Admission）、出院(Discharge)、转院(Transfer)，A01是患者入院事件。

HL7 V3也类似，但其结构层层封装，远比HL7 V2复杂。例如PRPAIN101001UV01，PRPA是业务场景(PR：Practice)的患者管理域(PA：Patient Administration)，IN是交互（interaction）规范。而其内部定义了这个交互对应的事件：

```
<?xml version="1.0" encoding="UTF-8"?>
<PRPA_IN101001UV01 ITSVersion="XML_1.0" xmlns="urn:hl7-org:v3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
...
    <controlActProcess moodCode="EVN">
    <code code="PRPA_TE101001UV01" codeSystem="2.16.840.1.113883.1.18"/>
    ...
        <subject>
        <registrationEvent moodCode="EVN">
        <id extension="393736355" root="2.16.840.1.113883.19.200.51"/>
        ...
        <identifiedPerson>
            <id extension="000197245" root="2.16.840.1.113883.19.3"/>
            ...
            <identifiedPerson>
            <name use="L"> <given>Patricia</given> <family>Patient</family> </name>
            </identifiedPerson>
            ...
        </identifiedPerson>

        </registrationEvent>

        </subject>
    </controlActProcess>
</PRPA_IN101001UV01>
```

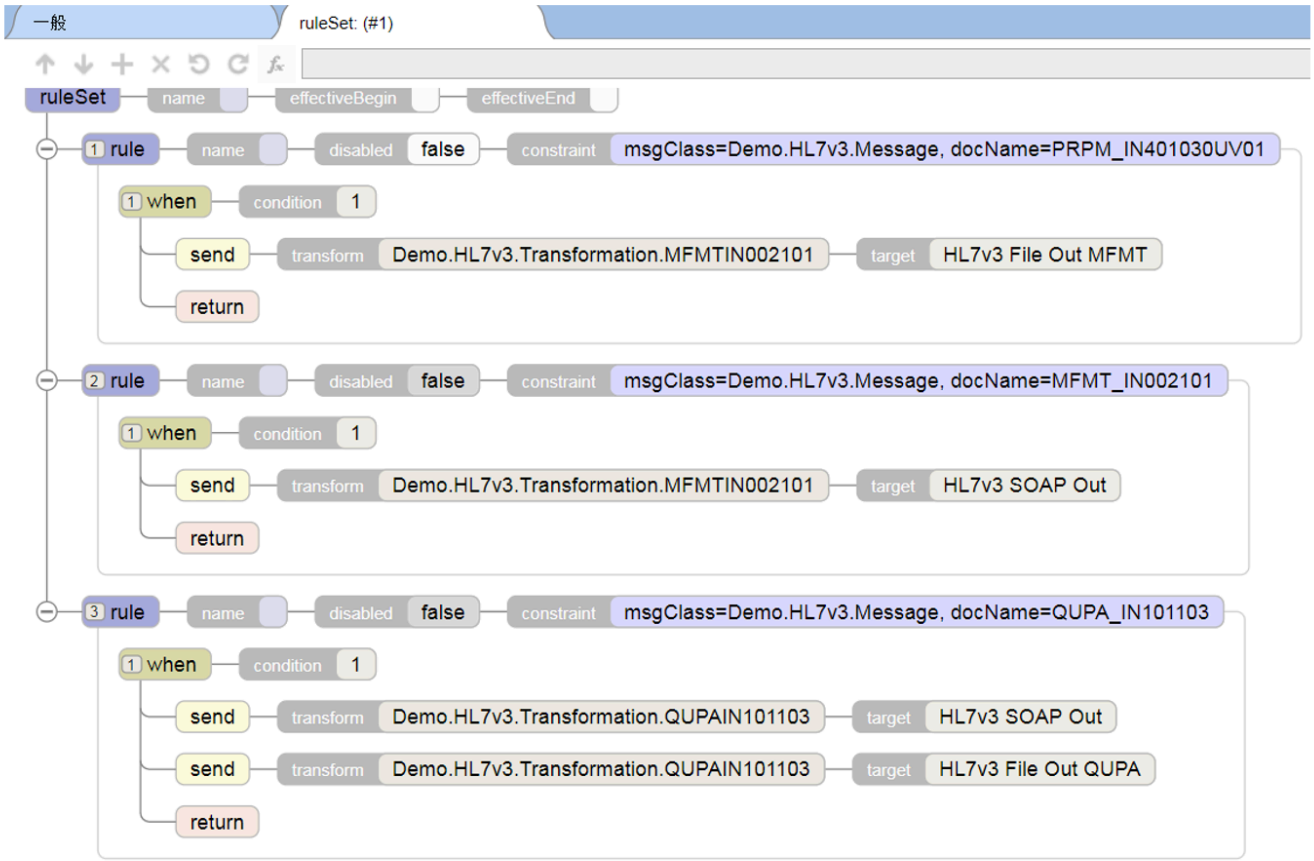
主题域内容

事件

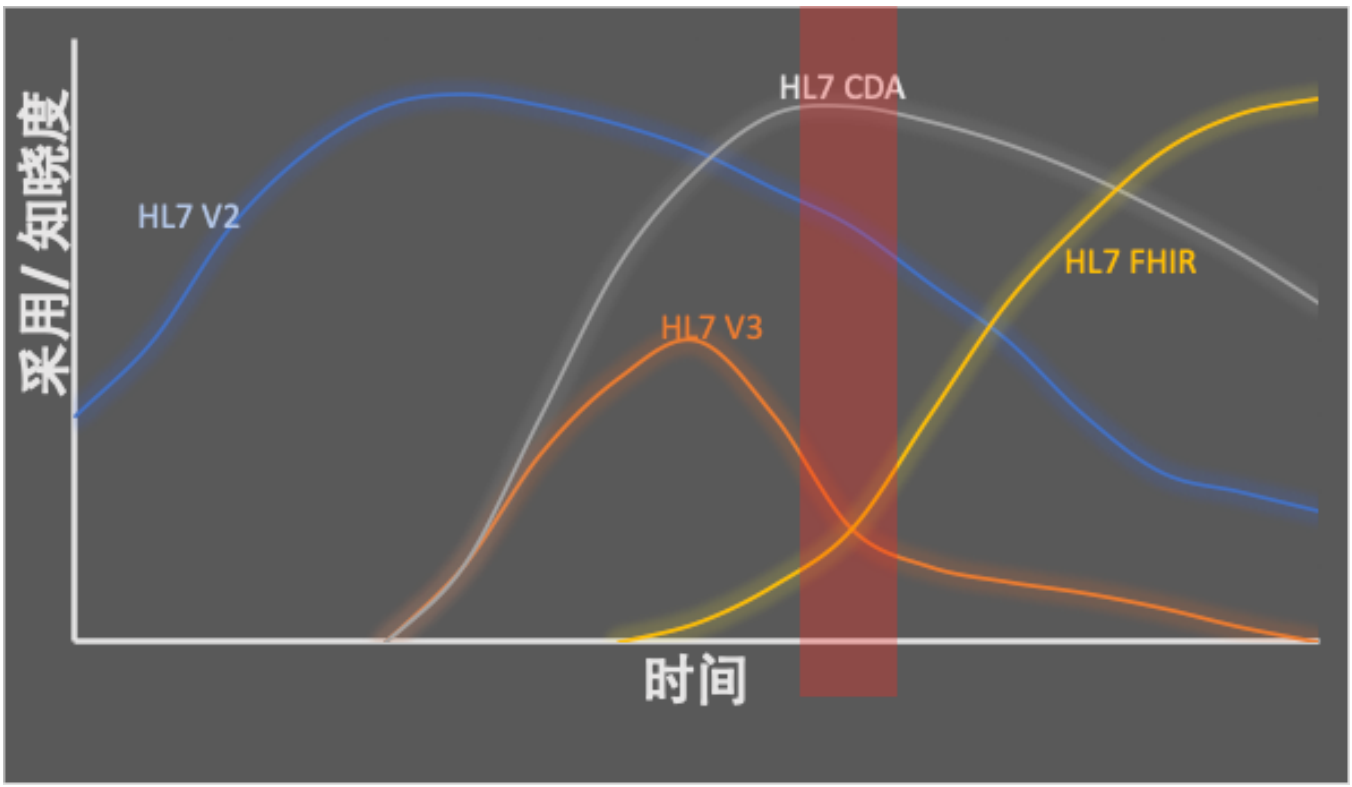
触发器事件封装

传输封装

而消息路由规则承载了“ 流程 ”：



目前在医疗互操作标准中，HL7组织发行的标准采纳范围最广、影响范围最大。HL7的历史很长、标准众多，如果我们要参考或采用其标准，必须要了解它发行的这些标准的用途和差异。国内市场上，所谓基于HL7 V3消息的方案不少，但大多数对HL7 V3标准的遵循质量并不高。



图： HL7组织不同标准的采纳度随时间的变化及趋势，枣红色柱状代表当前时间。

需要注意的是，行业的互操作标准对于应用集成来说非常重要，但其标准往往来自于对既往业务的总结。实际中总有超出现有标准覆盖的业务，因此基于标准消息的方案也要考虑可扩展性。 例如
HL7 V2具有非常简单和灵活的可扩展性，通常通过自定义的Z字段扩展；而HL7
V3由于其RIM方法论，相当难于扩展。这也是图中大家看到的HL7 V3全球采纳度远不及V2的原因之一。

对于标准通过扩展也不能满足的业务，就依靠底层的建模与转换能力和方案的丰满程度了。

4.3.2 自定义消息

当需求变化时，消息结构自主可控，这是用户自定义消息的核心优势。自定义消息很考验经验 - 需要对业务对象和流程抽象。如果不能覆盖主要的用例，意味着消息结构可能需要经常变更，从而影响方案和接口的稳定性。

这个方案有一个比较流行的分支：

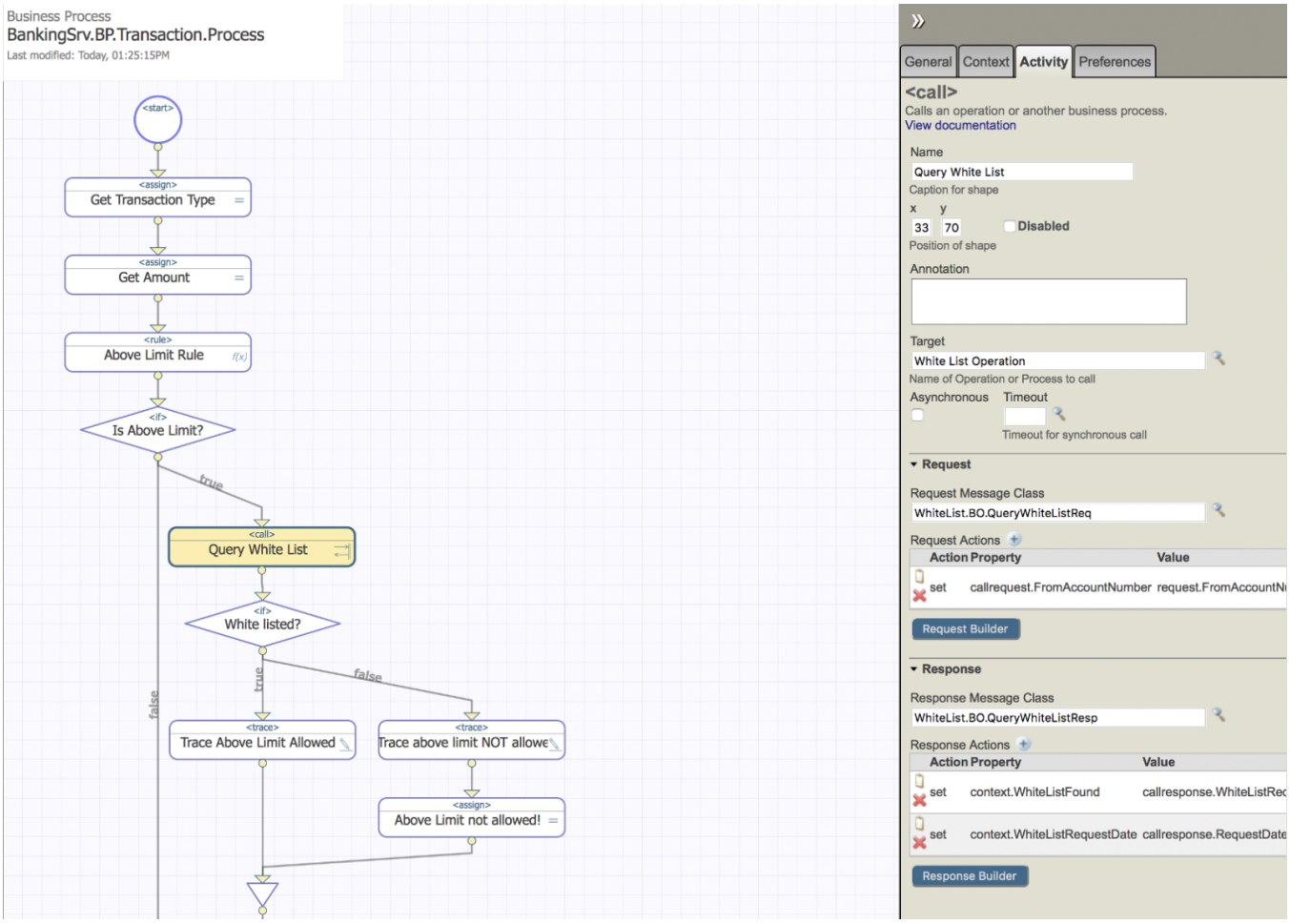
对消息体不建模、路由时不做拆包处理 - 任何的消息体内容都可以传，从而避免因为需求变化而更新消息模型（schema）的工作。它不做消息校验和基于消息内容的路由，仅根据消息类型、甚至定死的路由目标设置进行路由。方案最大好处是平台建设方“ 无事一身轻 ”：平台仅提供基于消息类型的路由机制；而各个应用需要改造以发送、接收、校验和处理消息，工作量较大。

4.4 基于文档交换

文档通常是小结性的信息，发生在业务阶段性节点上，例如“ 出院小结 ”发生在出院时，它不是细颗粒度“ 事件 ”。所以文档交换并不是主流的集成方案，通常作为基于消息或服务的方案的补充。
文档结构是对“ 上下文 ”的定义，通常采用的有互联互通文档、HL7 CDA文档或用户自定义文档。
基于文档交换的“ 流程 ”特别简单，通常是：注册、查询、获取。而应用需要具有使用文档流程服务的能力。

4.5 基于服务总线（ESB）

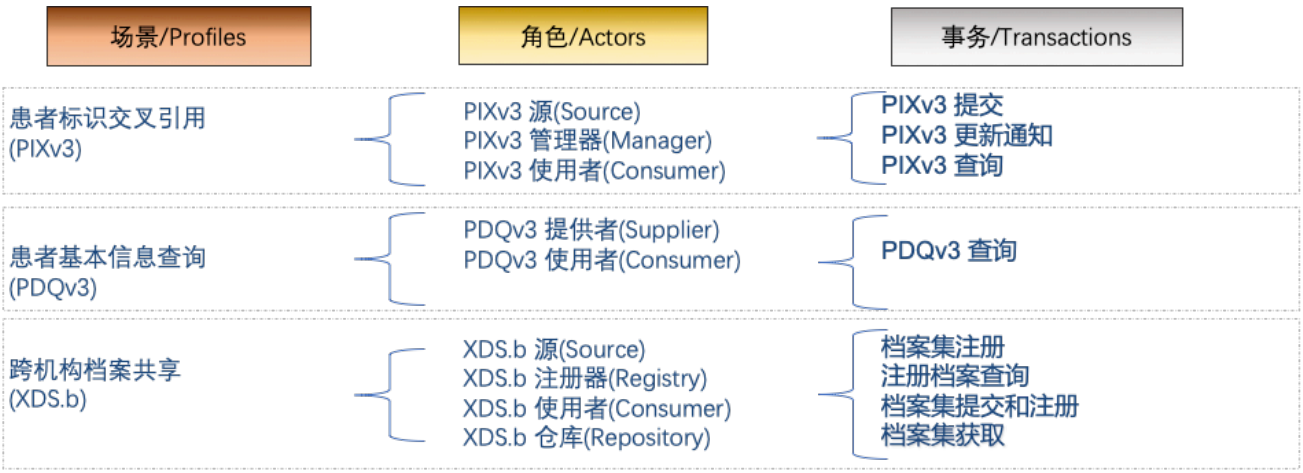
将应用的功能在ESB上封装为服务，并通过ESB来协同这些服务，就是基于服务总线的集成方案，它也是主流的集成方案。 服务本身封装了事件、上下文。ESB协调这些服务调用的流程，通常是通过业务流程建模能力实现的，它提供比消息路由规则更直观、更灵活、更丰富的流程建模和管理能力，甚至可以被业务专家直接使用。



需要注意的是，有服务不等于用服务总线。例如一些方案中，集成厂商扔出一份“ 服务 ”文档，要求各个应用厂商建设“ 服务接口 ”，但后台并没有ESB，不做服务注册、数据转换等，这其实是一种点对点方案。 还有一些解决方案仅有注册、发布固定服务的能力，例如自定义的SOAP服务。并不能注册别的服务-
例如TCP或HTTP的服务，或新的服务。这样的方案连接能力不足。

4.5.1 基于行业互操作服务标准

医疗行业，IHE、互联互通成熟度标准，都抽象和定义了服务标准，例如下面IHE常见的几个服务的场景、角色和事件（事务）：



可能大家注意到了，上面没提3要素之一的上下文。通常这些服务标准也规范了服务的请求和响应模型，例如IHE使用HL7 V3消息或FHIR消息；而互联互通服务使用互联互通消息标准。

基于服务总线的集成方案，通常需要具有持续的服务注册与发布的能力。这也是评价方案的因素之一。

4.5.2 用户自定义服务

由厂商基于经验或项目需求定义服务，并使用ESB注册、发布服务。对服务的规范能力和设计弹性是这类方案的关键。

另外，方案是否提供良好的连接能力，也是自定义服务方案的评价因素。

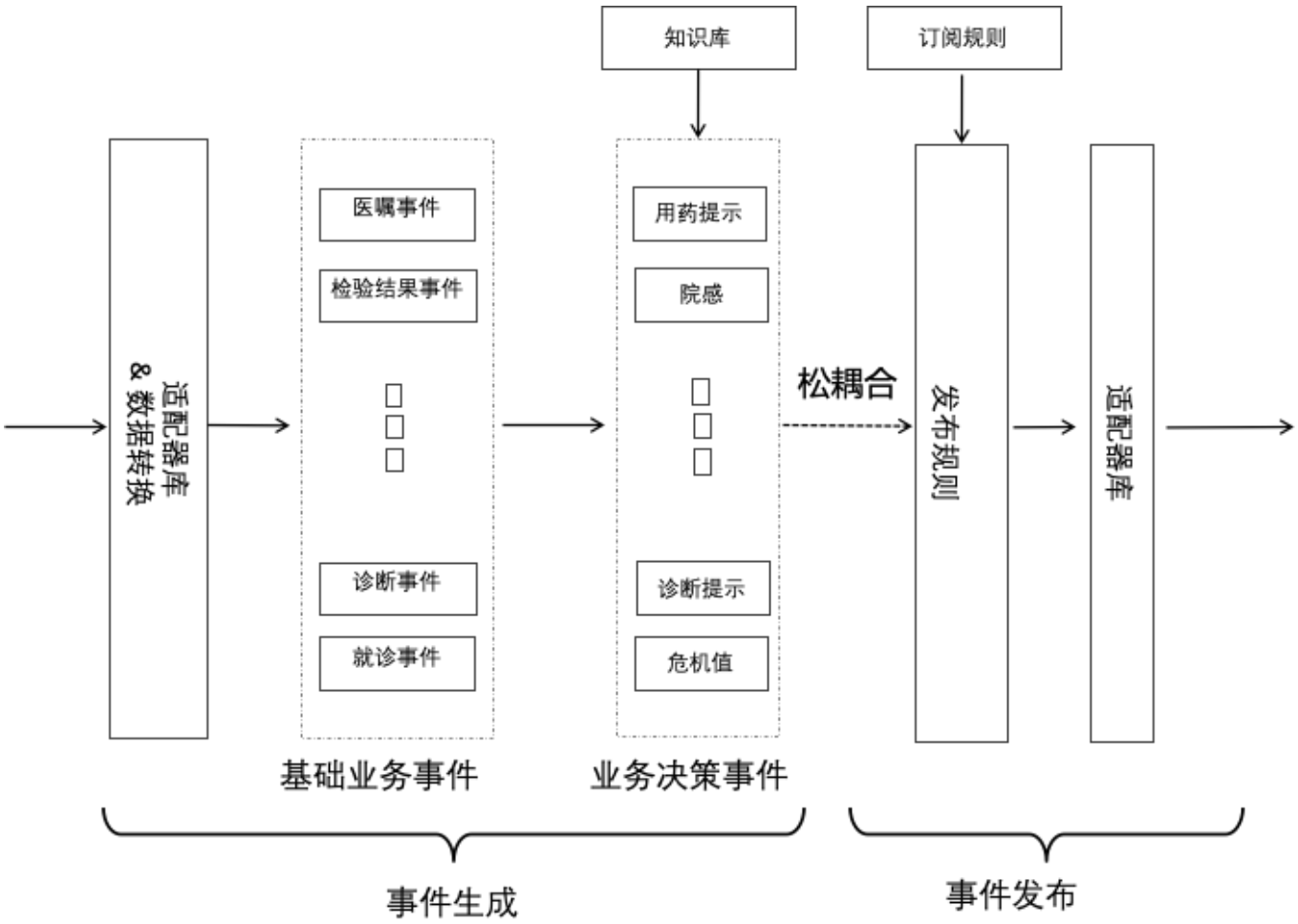
4.6 基于事件驱动（发布/订阅）

绕了这么久，终于提到了开篇所说的**FHIR订阅** 范式。

事件驱动（EDA）是SOA的继承者，其核心特征是通过事件生成能力和事件（及表达事件的上下文）的发布/订阅，打破紧耦合的服务调用，以松耦合架构实现更灵活的服务调用流程。通过对事件的灵活定义和发布/订阅机制，EDA应对业务变更只需要增加、调整订阅关系，而无需修改固定的业务流程模型或消息路由规则。

基于事件驱动的集成方案现在越来越广泛地被采用，为用户提供了可持续集成的灵活架构，对集成厂商而言也降低了业务变化时的开发实施成本！

事件生成能力和灵活性、发布/订阅便捷性是评价方案的关键。另外，事件驱动是以SOA为基础的，它仍要基于ESB封装服务和协同前道业务流程，而且对服务标准化程度和调用方式也提出了要求 -
例如应该尽可能使用异步服务调用方式，而不是同步调用。



FHIR订阅基于W3的WebSub， 提供FHIR的订阅资源Subscription
和订阅主题资源 SubscriptionTopic，本身就是一个API时代的事件驱动实现。

5. 应用集成的发展

应用集成随应用的软件开发架构的进化而进化。 FHIR的6个互操作范式涵盖了上面提到的消息、服务、文档，另外3个 - API、订阅和资源仓库正体现了近年软件架构上的发展。

5.1 基于API的应用集成

上面提到的应用集成方案基本都是中心化的，应对主流的单体架构应用。而软件开发架构发展的一个趋势是去中心化，例如微服务架构。

微服务架构是站在SOA肩膀之上的软件架构，去中心理念、开发的敏捷性、部署的弹性、快速迭代能力让其近年几乎和各行各业的数字化转型画上了等号。微服务架构通过按主题域驱动设计(Domain-Driven Design)
，将业务进行细颗粒度的划分，使用API打破了应用边界。

既然微服务架构下，应用边界都被突破了，基于应用边界的应用集成显然对微服务架构应用不再必要了，转而需要对这些微服务进行集成。这涉及到微服务的发现与注册、微服务的编排、微服务路由、微服务流控、微服务安全、微服务监控等诸多领域，架构复杂度相当高。

市面的API网关一定程度上可以应对 “ API ” 集成，通常有API发布、API协同调用、数据转换、API转换等能力，让其能涵盖API的调用 ” 流程 “ ，而对目前API的主要载体RESTful
API的连接能力当然不在话下。同时，API网关也不必中心化部署，适合微服务软件架构向外暴露服务。

在医疗行业，FHIR提供了一套支持资源CRUD的API，并且提供了用户自定义API的扩展能力。虽然FHIR API不等于微服务，但它赋与了基于FHIR的应用间使用API进行互操作（集成）的能力，再借助其行业语义级的资源模型和事件定义能力，让FHIR API成为行业中一种可行的互操作范式。

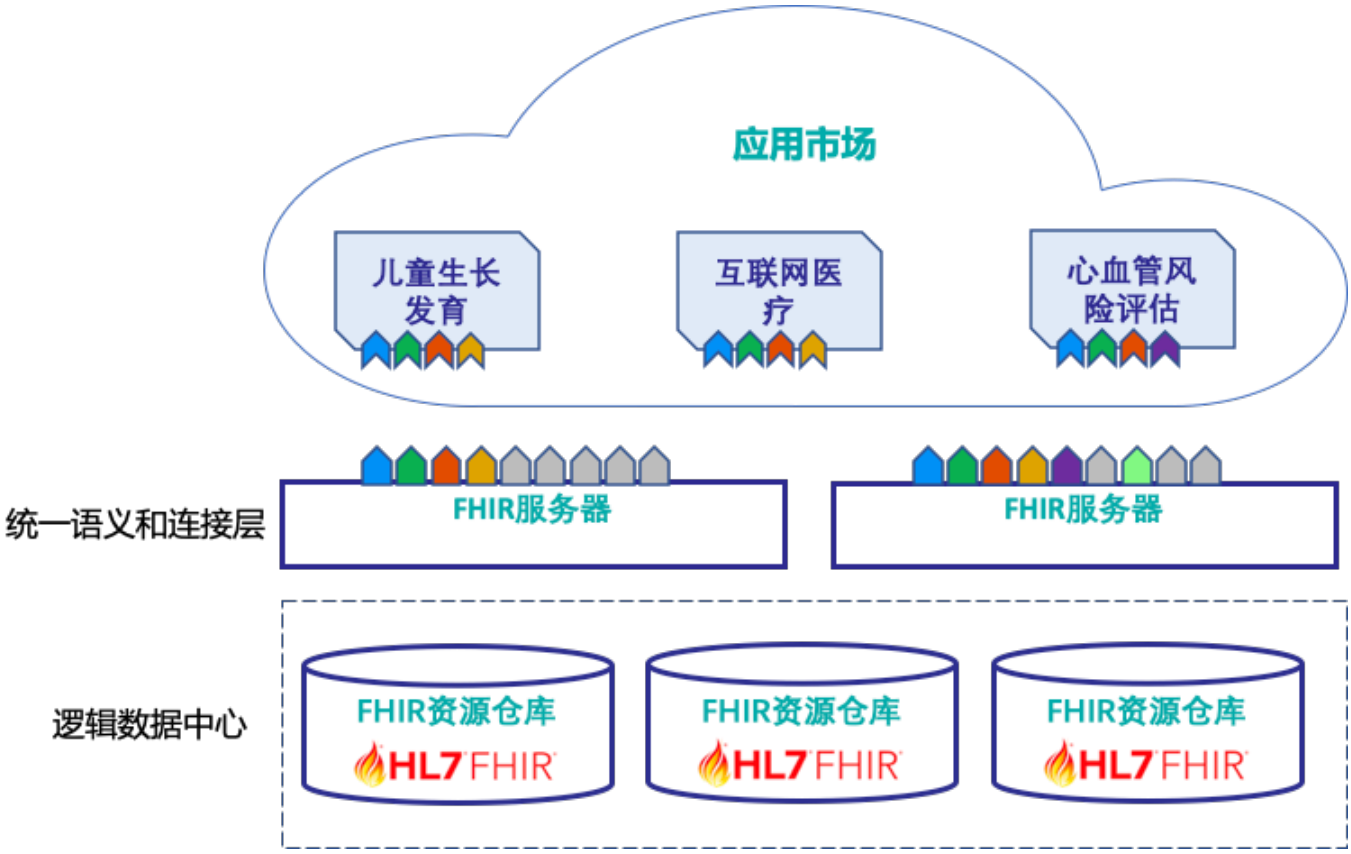
需要注意的是，
API网关不是集成平台，API集成严格上也不能叫做应用集成。只有在微服务架构的应用间，它才能充分发挥作用。而在多种架构应用并存的今天，仅靠API网关显然解决不了应用集成的需求。在微服务架构应用全面、大规模部署前，应用集成仍需要集成平台的支撑、API网关来补充。相信微服务架构未来会越来越重要，但并不会一统天下。

5.2 基于FHIR资源仓库的应用集成

微服务架构中，服务可以非常弹性的部署、快速的迭代，但数据并不容易 - 高效地同步或复制数据是个难题。能否在微服务架构上，让数据保持逻辑上的集中存储以解决这个难题？

FHIR的第5个互操作范式 - 资源仓库，基于这个思路，为基于FHIR API的应用集成带来了一个有趣的分支。FHIR的资源仓库为这种方案提供一个统一行业语义的、逻辑的数据中心，而且其中的数据是可以读写操作的。之所以说是逻辑数据中心，是因为不同FHIR资源可以存在多个FHIR仓库中，互相之间通过资源引用 - 例如URL - 关联在一起，而不必把所有资源数据物理上保存在一起。

FHIR资源模型提供上下文语义和事件定义，FHIR服务器提供统一的语义层和连接层，API网关以及FHIR生态中的CDS hooks、订阅等提供流程层面的支持。而“生长”在FHIR资源仓库上的应用，天生就是互联互通的！



这个方案不知道有谁已经真的实施了，但的确开拓了具备行业天生互联互通能力的应用开发架构的新思路。

当我们讨论未来的时候，其实未来已来。数据编织、数字孪生、组装式应用、超级自动化.....已经出现在我们身边，也深刻影响着应用架构和应用集成。对应用集成的发展，让我们拭目以待！

#InterSystems API管理器（IAM） #InterSystems 业务解决方案和架构 #业务流程 (BPL) #持续集成 #InterSystems IRIS for Health

源URL: https://cn.community.intersystems.com/post/%E7%B2%BE%E5%8D%8E%E6%96%87%E7%AB%A0-%E6%BC%AB%E8%B0%88%E5%BA%94%E7%94%A8%E9%9B%86%E6%88%90%E7%9A%84%E7%8E%B0%E5%9C%A8%E4%B8%8E%E6%9C%AA%E6%9D%A5