保险IT系统建设

BAOXIAN IT XITONG JIANSHE

主编 ◎ 谢用辉

首都经济贸易大学出版社
Capital University of Economics and Business Press

保险IT系统建设

BAOXIAN IT XITONG JIANSHE

主编 ◎ 谢用辉

首都经济贸易大学出版社
Capital University of Economics and Business Press
·北 京·

图书在版编目(CIP)数据

保险IT系统建设/谢用辉主编. —北京:首都经济贸易大学出版社,2016.7
ISBN 978-7-5638-2511-0

Ⅰ.①保… Ⅱ.①谢… Ⅲ.⑭信息技术—应用—保险业 Ⅳ.①F840-39

中国版本图书馆CIP数据核字(2016)第119464号

保险IT系统建设
谢用辉 主编

责任编辑 陈 侃
封面设计 砚祥志远·激光照排
TEL: 010-65976003
出版发行 首都经济贸易大学出版社
地 址 北京市朝阳区红庙(邮编100026)
电 话 (010)65976483 65065761 65071505(传真)
网 址 http://www.sjmcb.com
E-mail publish@cueb.edu.cn
经 销 全国新华书店
照 排 首都经济贸易大学出版社激光照排服务部
印 刷 北京九州迅驰传媒文化有限公司
开 本 787毫米×1092毫米 1/16
字 数 480千字
印 张 18.75
版 次 2016年7月第1版 2016年7月第1次印刷
书 号 ISBN 978-7-5638-2511-0/F·1416
定 价 42.00元

前言

本书分析了当前保险业IT系统的现状和发展趋势，描绘了保险IT应用架构和核心业务系统的应用架构、数据架构，重点选择保险核心系统的主要子系统：产品定义、保单管理、客户管理、理赔管理等进行分析设计，然后围绕核心系统，分析设计它们与周边系统之间的关系，包括与BPM、单证、打印、规则引擎、缓存服务、ODS等系统。同时也介绍了系统支撑性的组件，包括安全管理、技术架构、性能方案、部署架构和界面设计等内容。系统设计是一门平衡的艺术，在现实和理想之间、实用和完美之间选择合适的平衡点。本书尝试寻求一个比较好的平衡点。

本书第一章介绍保险IT现状和发展趋势，然后介绍保险应用架构。第二章介绍了核心系统的范围和保单管理的需求分析，这一章为保单管理的服务设计提供了需求基础。第三章和第四章分别介绍保单管理的承保和批改服务设计，介绍投保单、保单和批单三者之间的关系，并介绍了每一个服务实现的细节。第五章介绍了保单管理的数据库设计，以及如何设计可扩展的数据库结构。第六章介绍了核心系统对多产品线支持的复杂性，解释了为什么当前很多系统按产品线设计成多套独立系统。第七章介绍了和保单管理密切相关的当事人管理，重点介绍了由于保单管理和当事人之间的耦合关系，当事人管理和ECIF的关系。第八章介绍为支持多产品线的灵活性提供产品定义基础模块，首先介绍传统的参数配置，然后介绍规则引擎的应用，最后介绍灵活的保单元数据模型的基本思路。第九章介绍流程管理，着重介绍保单管理和BPM之间的关系，以及将流程从业务中剥离之后，可能导致问题以及相应的解决方法。第十章介绍了系统集成和ESB应用，讨论保单管理和单证系统之间的关系，重点介绍两个松耦合的独立系统集成时由于事务不一致性可能带来的问题以及解决办法。第十一章介绍了核心系统相关多个维度的安全控制。第十二章介绍了核心系统的系统架构，重点分析了模块间的耦合性和层之间的耦合性、系统性能方案和系统部署。第十三章介绍了界面设计相关问题，重点介绍了可配置的界面设计思路。第十四章介绍了系统的技术可管理性，如何实现的业务服务可管理性以及相应的应用。第十五章介绍理赔管理，综合前面章节的知识点，介绍了理赔相关的业务服务分析设计、赔案模型以及它们的功能实现。第十六章介绍保险数据架构，重点介绍ODS、EDW和数据集市之间关系和模型设计思路。

本书对近期的一些设计思路都有所涉及。例如，小核心的概念，认为核心系统应该仅限于保单管理，原来属于核心一部分的客户管理、理赔管理、收付费、再保险模块都应该是独立

系统，系统之间通过服务交互。即使对于保单管理系统，也要求前后台拆分，将后台作为保单数据服务的平台，允许不同渠道的接入。随着保险渠道，包括代理中介、经纪公司、电销、网销、电子商务、移动、互联网金融的发展，各种不同渠道有不同的录入系统，共享相同的保单管理。因此，将保单管理作为一个保单服务平台，是核心系统发展的方向。但是渠道衍生系统的开发工作量极大，都在重复开发录单功能，并且还要支持不同产品线的录入，每个录单系统都要解决多产品线的问题。将保单管理作为独立服务平台，并没有减少渠道接入的个数，也没有减少外围系统的开发工作量。无论是否采用小核心的概念，核心系统的本质在于如何支持所有产品线的管理，提供产品定义、保单数据存储模型、保单管理服务、功能界面、渠道衍生系统的一体化解决方案。多产品线的支持关键技术是如何分离和包装变化，将产品相关的个性化的数据、功能、流程剥离出来，而单独将保单管理从核心系统中剥离出来，并不能解决当前核心系统遇到的问题，这只是一个技术问题。

在应用架构方面，业内基本认可应该构建服务和数据两个基础平台，用于松耦合系统之间集成。监控管理平台将是未来的一个必备平台，对所有系统实现从前台到后台以及硬件的端到端的集中式的监控管理，提高效率，快速故障定位和故障预防。

支持所有产品线是核心系统设计的难点。常有人提产品引擎的概念，希望有这么一个平台，能够解决保险产品的个性化问题，尤其是保费计算。但是对产品引擎到底能做什么，认识非常模糊，有人认为是计算引擎，是在公司范围内的一个公共模块，负责所有保费和理赔相关的计算，也有人认为产品引擎就是规则引擎，能够支持业务规则定义和执行。但是当前规则引擎应用的效果并不理想，只是使用规则引擎的动态脚本技术，代替静态代码的技术，缺少在业务模型直接支持的规则引擎，属于粗粒度的业务灵活性控制，无法达到精细化控制，产品引擎应该提供两个方面的灵活性，即模型和算法。在模型方面，可以通过保单结构和赔案模型的元数据模型实现，在算法方面，应打造为保险业务模型量身定制的规则引擎。不存在独立的产品引擎，产品引擎与保单管理密不可分，不能单独存在。为了基础产品数据共享，应该将产品定义做为主数据管理。

伴随小核心的概念，松耦合是当前另外一个不断强化的概念。软件设计始终围绕“高内聚、松耦合”展开。所以，松耦合是所有软件设计技术上的要求，并不是核心系统的特殊要求。实际上，业务总是希望在一个操作中能完成所有应该完成的工作，是紧耦合的业务操作模式，而技术上要求系统应该是松耦合的，系统拆分比较彻底，耦合度尽量地低，相互之间甚至没有关联，这两者本来就是矛盾体。很多人认为的核心系统是一个紧耦合系统，业务很难拆分，恰恰体现了紧耦合的保险业务特点。本书许多地方都会涉及松耦合的讨论，在松耦合带来好处的同时，也要认识到松耦合可能带来业务上的损害，如操作不便利、数据冗余、不一致性、性能瓶颈等问题，有可能得不偿失。即使遵循松耦合思想设计的保单管理，在经过与其他系统集成之后，其松耦合特性也许不复存在，其结果可能还是一个紧耦合的系统，这和原本按照一个有机体整体设计的核心系统，没有本质上区别。本书中介绍保单管理和当事人关系，保单管理和单证系统的关系，都涉及两个独立系统之间集成时可能存在的一些问题以及相应的解决方案。

多年前业界就认识到流程和业务交织在一起，导致流程和业务相互影响，不能独立快速变化，认为核心系统应该将流程剥离，通过流程引擎或者 BPM 来解决流程，让业务只关注业务自身的处理。核心业务系统对外提供 SOA 服务，由 BPM 组织业务服务。本书关注流程的剥离，分析流程剥离所带来的问题，造成系统和流程集成的困难。没有良好流程剥离的思路，简单引入流程引擎，并不能达到流程灵活变化的目标。

大家都认识到，界面开发工作量庞大，似乎找不到好办法，只能按照需求个性化开发。本书专门就界面涉及做了深入探讨，希望能对界面开发效率有所提高，功能界面将是支持多产品线的核心系统的统一解决方案的一部分。

在数据架构方面，大部分公司已经做了一轮数据仓库和数据集市，部分公司还有 ODS，但是对于 ODS、数据仓库的定位，依然非常模糊。很多人认为 ODS 应该是同源结构，甚至就是核心系统的同构库。很多所谓的数据仓库其实就是数据集市，只是取了数据仓库名字。

当前各种流行的思想，包括小核心概念、松耦合、产品引擎、流程引擎应用、规则引擎的应用、SOA 思想等，偏重于技术。本书从始至终在强调模型的重要性，包括对保单模型的分析和赔案模型分析，这需要更多人静下心来思考业务模型。以客户为中心的设计思想，是每一个核心系统都要追求的目标，认为将客户数据集中管理，是其最重要的体现。本书专门介绍客户集中管理，深入讨论了客户集中管理之后带来的业务应用上的挑战。

本书不是技术文档，不要指望根据本书给出的服务接口、业务对象、序列图、界面就能实现 IT 系统，即使本书部分地方包含许多技术化的表达方式。图书和技术文档区别在于，前者告诉读者为什么要这么设计，后者告诉读者系统做成什么样子。本书的目的告诉和解释了为什么要这么做，例如根据序列图分析后台的服务清单，以及每一个服务应该考虑的要点，而不是描述一个真实系统的设计方案。书中对于投保界面、投保服务、投保单结构描述和数据库结构设计，只适合旅行意外险，这是虚构的设计，用于说明一般系统设计思路，现实中不存在只支持旅行险的核心系统，除非真的要设计一个只支持旅行意外险的系统，可以借用本书提供的例子，否则只能作为分析问题的思路，而不是最后结果。

本书观点主要来源于多年的核心系统的研发工作和客户的现场实施，借此机会对老同事和客户们的支持和帮助表示感谢。感谢史小六先生的长期教诲。感谢沈习武、王川、刘文华、叶寿生、贾晓谦、陈建芳等领导在工作上的支持和帮助。感谢 Lifepro 项目组成员：梁如见、吴万春、熊刚、李聪、郝秀峰、张道明、杨临芝、蒙剑、韩忠朝、张弘、李智、钟敬、杨洁、谢玉龙、蔡培茂、高卫、陈燕闽、王敏等全体同事们。感谢 Pharos 项目的国外合作者 Guillermo 先生和 Ciro 先生。感谢 Pharos 项目组成员：邵伟建、宋蔚、丘祺中、殷正栋、王凤燕、曹立刚、李芹、李洪斌、祁晓勇、李晓强、肖金华、姜义东、卫建军、高毅、高欣、王建波、王永霞、李贞、孙伟光、高静、茅诚等全体同事们。感谢温晓东、尹建斌、郭伟、杨晓园、卢嵘、王鹏宇、王娟、徐兵、黄莹、秦琼、高红山、刘冰等客户们。感谢甲骨文（中国）软件系统有限公司的领导许向东博士，让我有机会从 IT 产品角度梳理应用和产品之间关系和边界定位。

感谢编辑陈侃老师，接受这本小众类图书，同时涉及到保险和 IT，非常辛苦，付出太多努力，最终让本书面世，在此非常感谢！

保险应用是各种技术的综合应用，一般都没有什么难度，只要懂技术的人，都能做得好。业内朋友只要做过一段时间，都有很多关于系统建设方面的很深体会和见解，并有非常好的方案，只是少有落到文字上。本书总结了保险系统设计的一些经验，谈不上太多创新，只是起到抛砖引玉的作用，里面肯定有很多错误甚至谬误，欢迎大家批评指正，让我们共同进步！

谢用辉

2016 年 6 月 18 日

目录 CONTENTS

1

保险IT概论

1.1 保险业务和IT趋势

保险行业正发生碎片化、场景化和高频化的互联网金融变革。越来越多保险企业将大数据和云计算作为保险创新和发展的驱动力，实现以客户体验为中心的保险营销，产品创新、场景化的产业链整合和风险控制。保险业务快速沿着如下四个方面发展：

(1)多渠道建设和整合。新渠道不断出现和发展，包括电销、网销、社交媒体、微信、移动和互联网金融等各种金融创新业务快速发展，都要求对现有后台支撑系统进行全方面的整合，实现客户共享、产品共享、服务共享和数据共享，并进一步发挥垂直产业优势，进行上下游产业链整合，增加客户接触点，提高客户粘度。

(2)以客户为中心的销售和服务，包括提高客户体验，洞察客户行为，分析客户偏好，提供个性化服务，客户交叉销售，综合企业内外部数据进行分析和应用等。整合全企业范围内的客户信息，在企业范围内实现数据共享和应用，支持客户营销管理和客户分析，实现“以客户为中心”的服务。

(3)支持快速保险产品创新，根据细分市场和面向特定群体推出保险产品，实现客户增值服务，避免同质化低附加值的竞争。各种新产品不断涌现，包括面向政策性的新农险、医疗责任险，还有互联网金融创新型产品，如理财险、运费险等，实现产品差异变化竞争。这对IT系统灵活性提出了更高要求。

(4)强化风险控制，遵循合规性的要求，降低企业风险，在符合监管要求下的产品创新和服务。包括监管服从、风险分析和控制、理赔反欺诈。对保险数据分析应用和高可靠运营提出更高的要求，支持7×24的不间断持续运营。

随着保险业务的发展，IT技术有了新变化，反过来又促进了业务发展，尤其是云计算和大数据的出现，对保险IT产生了深刻的影响，作为后台支持的IT技术走向了前台，实现从保险信息化到信息化保险的转变。在IT技术方面沿着如下方向发展：

(1)采用面向SOA的IT架构，强调服务重用共享，系统之间通过服务交互。只要保持服务接口稳定，对一个系统升级改造不会影响到其他系统，可以灵活应对变化。随着多渠道系统接入，对服务重用性要求很高，如何设计可重用的服务，能够满足多个渠道要求，是系统设计非常重要的课题。

(2)技术平台化，应用小型化，将纯技术从业务中剥离出来做成公共基础平台，例如，剥离缓存服务、流程引擎、规则引擎、统一身份管理、服务集成平台等，它们都是具有很强

技术性的、通用的平台，可以作为企业内的共享平台，进而为业务系统瘦身，使得业务系统具有高密度的业务价值，更专注于业务本身，减少干扰，使得业务系统更加专业灵巧，适应变化。同时，从业务的角度，对现有紧耦合的系统拆分，实现小系统之间的关系达到松耦合的状态。

（3）云计算的应用。云计算是 IT 技术发展一个趋势，保险公司日益关注该技术在保险公司的应用。当前很多保险公司已经在数据中心和测试中心建设中，采用云技术，实现硬件资源集中管理，动态调配，提高设备利用率，取得了很好的成效，这是云计算应用的第一步，未来还有更广泛的发展空间。

（4）大数据的应用。随着互联网和社交网络的发展，每天都要产生大量非结构化数据，如何从这些大数据中提炼出有价值的信息，是目前各行业都关心的课题。保险行业非常关注，获取更有价值的客户信息，快速响应社会舆情，提高客户体验。

（5）移动技术的应用。手机和平板电脑是发展极其迅速的互联网接入设备，几乎所有公司正在提供移动设备的接入应用，包括保险展业需要的保单录入应用，也包括现场实时查勘定损理算应用，BI 报表查询工具等。

（6）互联网可扩展部署架构的应用。互联网架构通常采用高可用、横向动态线性扩展的特性技术，逐渐被保险公司 IT 人员所推崇，很多公司决定采用廉价 PC Sever 作为应用服务器，可以动态扩展，为企业提供高可用的基础架构。

综合上述，保险业务要求以提高客户体验为方向进行信息整合、信息共享和数据分析应用，而保险技术要求松耦合架构、共享数据和大数据应用。业务和技术在大部分程度都是面向共享和数据应用发展，但是又存在一定矛盾：业务上整合和技术松耦合，是一对矛盾。松耦合技术要求系统拆分集中和共享，导致系统之间交互性能变低，可靠性降低，业务整合要求支持以客户交互为中心的紧耦合业务形态。

随着云计算、移动技术、互联网金融、大数据等各种新技术发展，推动保险业务创新，成为保险业务创新源动力，实现“以客户中心”的服务转型。近期保险 IT 应用建设正沿着如下几个方向发展：

（1）整合全企业范围内的客户信息，在企业范围内实现数据共享和应用，支持客户营销管理和客户分析，实现“以客户为中心”的服务。

（2）开发低成本的新渠道，并与现有渠道进行整合，为客户提供全方位的、统一的客户体验，实现多渠道的大统一，提高客户满意度和续保率。

（3）开发创新的保险产品，对客户进行细分，面向特定群体有针对性的推出个性化的产品，实现客户增值服务，避免同质化低附加值的竞争。

（4）交互式的客户体验为中心的电商系统建设，以渠道整合、客户信息整合和产品整合为基础，实现客户行为实时分析和产品智能推荐。

（5）加强风险控制，遵循合规性的要求和控制，降低企业风险。

（6）整合现有的 IT 应用，支持系统间的融合协作，最大程度的实现业务流程自动化，减少没有必要的人工环节，缩短业务处理周期，提高处理效率。

（7）统一的数据平台建设，实现全企业范围内数据标准化，提供全局的数据服务平台，支持实时准确有效地决策智能分析和数据洞察。

（8）大数据平台建设，以支持大容量非结构化数据存储，以及数据分析，并能有效的融合到企业现有数据架构中。

(9)高性能、高可用的低成本云数据中心和异地灾备系统建设,实现资源共享,提高设备利用率,简化运维,并实现异地灾备,提高数据安全。

(10)端到端的、集中式的系统监控管理建设,实时监控系统可靠运营,提前故障预警,故障发生时快速定位和排查。

1.2 保险IT现状分析

保险IT引起了业务高管们的重视,并寄与厚望。但与此同时,业内普遍对当前IT现状不太满意,从最基本的核心系统到各种渠道系统,或多或少不尽如意,形成了强烈的反差。归纳起来,目前有如下几个方面问题:

(1)IT需求响应速度缓慢,一个小需求往往需要大工作量投入,实现周期长。

(2)流程变化很难,对某个保险产品的流程微整都不简单,需要开发工作,时效性差。

(3)新产品上线能力不够,经常需要多处产品定义,先在核心系统配置,然后再在各个渠道系统重复配置,并且经常需要修改代码,实施周期很长。创新性的产品上线周期更长或者无法支持。

(4)系统运行不稳定,性能差,经常有bug,需要中止业务等待修改,甚至出现宕机的现象。

(5)数据质量差,尤其是客户信息不完整、不真实等,报表数据经常不一致,数据再利用价值不大。

(6)新渠道接入周期长,工作量大,代价高。

(7)对新技术如云计算、大数据等,没有找到可观察的业务价值,还是停留在概念上。

上述现象归根到底还是IT响应慢,以及IT系统质量差、不稳定。从IT系统建设方面,技术上还是存在很多问题,从应用架构、技术架构、数据架构和部署架构方面,都存在问题。

1.2.1 应用架构现状

公司对应用缺少合理规划。保险公司历史大多不长,很多是最近十年刚成立,当时由于资金有限,急于开业,很多都是直接拿来现成系统就开始投入运营,只要系统覆盖面广,能够满足开业要求和最基本的日常运营,其他方面不太在意,根本就没有规划。尤其面对当前技术和业务快速变化的环境,随着多渠道快速发展,对系统重用性提出了很高的要求,希望能够重用现有后台应用,快速支持新渠道接入,才发现这些需求很难满足。对于一些老公司,由于有历史包袱,在引入新系统的同时,还要兼容老系统和老业务,很难能彻底另起炉灶。在当前环境下,能够提供第三方可选择的好系统比较少,很多公司选择自己设计开发,限于自身投入和人力限制,真正能够从规划重新做起的很少,大都是针对某个重要系统,改造升级,缺少顶层应用设计。

很多系统定位不清楚,边界不清晰。公司一般都有比较完整IT系统,从核心系统、渠道系统、销售管理、单证、打印,财务和人力资源等,应有尽有。但是这些系统之间边界不是特别清晰,尤其与核心系统紧密相关的系统,边界相当模糊,有功能重叠冗余的,有缺失功能的。例如,网销系统有本地费率表,可以做保费计算,而对于部分产品,还是调用核心系统计算保费,原因是前者保费计算简单,为性能考虑,在本地实现保费计算,而后者保费

计算复杂，不得不调用核心系统来计算。这个问题说明，系统边界范围的原则性不强，比较随意。再如，核心系统中一般有各种查询功能，而报表系统也有查询功能，有时候为了减少核心系统压力，将查询功能剥离出来，转移到外围系统。那么什么查询功能要剥离，什么查询功能保留在核心系统，要依赖查询的性能。要剥离出性能差的功能到报表系统，而没有性能问题的功能将继续保留在核心系统。也有公司对系统中报表功能是否在剥离到数据仓库中实现，是根据人员能力来决定，有能力剥离就移到外面、否则留在核心。

耦合度很高，系统重用度不好，重复建设多。一个核心连接着多个渠道系统，但是服务接口不能重用，为每一个渠道系统开放一套特殊接口，每个接口都要重新实现开发，不能重用相同服务业务逻辑，原因是不能解决不同渠道处理逻辑的变化。这样，系统对渠道接入的需求响应缓慢不可避免。就是连最简单的打印平台，很多公司可能还有很多套，一个用来纸质打印，为不同产品制作不同的打印模板，另一个用来生成电子保单，两个平台的模板不能共享。电子保单最早来自于网销的需求，因此，对电子保单平台调用，都是从网销开始调用，生成电子保单保存在网销系统，客户通过网销系统或者门户网站直接访问。随着后来业务发展，银保渠道或者传统渠道出单也需要出电子保单，于是在核心系统中也增加调用电子保单平台的功能。而很多核心系统有自带影像件存储模块，因此，公司可能有多个地方存放各种影像件和文档。很多公司核心不只是一套，为不同产品线分别有多套系统，在每一套核心都包含承保、保全或批改、核保、理赔、收付费、客户、再保险，基本功能基本相同，重复建设和维护，进一步恶化了外部系统，同一个外围系统，如销售管理、单证系统、打印等，都要和这些核心分别对接一次。类似地，由于每种佣金费用管理办法不同，为不同渠道分别建设销售管理系统，分为个人渠道，团险渠道，银保渠道，电销渠道等，它们都要与公司的多套核心分别对接，建设工作量巨大。

缺少主数据平台规划，如组织结构、业务员信息、员工信息、代码字典、产品数据、客户基本信息等共享的基本数据分散在多个不同系统中，多处维护，工作量大，存在数据不一致的风险。例如，销售管理系统一般包含业务员管理，在核心系统和其他系统依赖销售管理系统获取业务员信息，导致销售管理系统被依赖太多，对其可用性要求很高，而实际上销售管理系统负责对业务员的佣金计算和分配，关键业务都是批作业处理，不需要很高可用性，进而导致要么提高销售管理系统的可用性，要么降低核心系统或者其他系统可用性。而且，其他系统仅仅使用业务员信息，而依赖于销售管理系统，使得后者成为关键系统，对其稳定性要求很高，避免每次销售管理升级引起其他系统故障。由于在多个系统中进行产品销售，受限于核心系统中产品定义没有按照主数据思路来设计，无法共享，在各个渠道系统中分别有各自产品定义。同一个产品在多个系统重复定义，存在一定时间差，同一个产品还不能在所有渠道中同时上线销售。很多公司没有对客户集中管理，而将客户数据直接作为保单一部分保留核心系统中，客户信息在多处冗余保存，无法实现全公司共享，强迫公司重新开发一个事后的 ECIF 系统，用于客户信息集中整合和管理，需要长期从多个系统同步客户数据并经过清洗保留，过程有可能还需要人工介入。类似的，在多个系统中保留组织机构，并相互之间同步组织机构信息。多个系统都独立管理用户和权限，一个用户有多个账户，大部分公司还没有提供统一用户管理和单点登录的功能。互联网发展，外网和内网用户统一，更进一步加剧用户管理的难度。

1.2.2 技术架构现状

大部分应用采用JavaEE平台的B/S架构。由于Java开源框架很多,可以共享组件也很多,因此,很多公司都有多套框架在使用,不同项目采用不同框架,可能受开发商所采用的技术标准影响,也可能由于人员技能熟练程度影响,各种技术都有,会使用各种技术的人才也都有。但是真正高水平技术人才比较少,偶尔遇到开源代码无法解决的时候,开始怀疑技术架构的合理性,转而在下一个项目,采用一个全新技术框架,重复之前的工作。根据JavaEE标准技术,包括Spring框架,一般系统都会分为展现层、服务层、数据存储层。但是JavaEE在技术上对分层约束性并不强制,不像EJB有明确技术规范,强制要求使用EJB服务,每个层都可以自由采用各种技术,不做约束。很多在开发实践中,本来应该属于服务层逻辑,被放在展现层,本来应该属于展现层的逻辑,被放在服务逻辑层。同样的,在数据存储层,也没有严格遵循,很多对数据库访问直接放在展现层,也有业务逻辑直接放在数据存储层,具有一定随意性,一般根据实现难度和方便性来决定。总之,本来通过分层来提炼共享的服务,由于各层划分没有完全遵循,在系统中就找不到明确的可以发布为SOA的服务层。相反的,如果采用有明确技术规范框架,如EJB、Tuxedo、CICS等,服务层有明确技术实现约束,反而更加容易形成共享的服务。随着多渠道发展,各个渠道应用接入时,往往要另起锅灶,创建新服务,后者真正可以共享的内容很少,每个渠道可能都有自己的服务,每个服务都是独立的一套逻辑,导致当渠道不同或者产品不同稍微要调整一下流程时,变得非常困难,经常出错。

在同一个系统内部,模块边界也不太清晰。为了方便数据库操作,涉及两个模块的表查询时,直接做多表连接操作。模块之间通常直接通过JDBC直接访问到对方库表,导致系统模块之间无法拆分,很难保证对单个模块进行升级而不影响到其他模块。当系统难以支撑业务时,就采取全部重做的方式,甚至有些公司已经做了好几轮的核心系统。为了简化对象到数据库的映射,一般会采用Hibernate之类的框架,可以实现多个表的连接操作,甚至使用树型结构递归操作,导致系统性能极低。为了方便开发,很多对库表的更新操作,通过先删除再插入方式实现,IO量很大,浪费数据库资源,甚至导致后续的数据抽取或者数据同步到数据仓库时,数据增量变多,同时,数据仓库由于源系统无谓的删除和插入,导致历史轨迹数据量增多。有时为了简化代码,减少函数或者服务接口拥有过多参数,不保留已取到的有用数据,在同一个交易中,多次从数据库表中读取相同的数据,或者多次写相同数据,也导致IO量增加,性能变差。

事务管理存在一定问题。保险系统一般对数据一致性要求高,无论事务大小,无论是否关键数据,一般通过事务控制数据一致性,一个交易处理逻辑路径比较长,事务跨度长,导致数据库资源占用时间过长,性能低,并发差。服务之间调用,或者外部调用服务,或者内部逻辑处理,基本上是在一个事务中同步调用,等待时间比较长,系统响应比较慢。由于事务控制因素,很多服务间耦合度很高,不能灵活组合,重用度降低。当渠道接入时,经常需要重新开发服务,重写逻辑,剔除掉没用的逻辑,增加个性化操作,多套相似代码并存,代码维护困难。

并发性不高。很多地方还有临界资源限制,例如各种单证代码的生成,如保单号、批单号等,限制了系统并发能力,不能横向扩展。即使对团险,包含大量被保险人的保单,业务逻辑经常被设计成单线程处理,缺少并发处理,为了避免长时间处理,限制每次导入被保险人

的数量。很多批作业，如核心系统批作业和 BI 批作业，设计时也缺少多并发考虑，一般单线程执行，偶尔有并发机制，也只考虑单台服务器的并发。批作业一般只能逐个串行，导致持续运行时间很长，超过了规定的时间窗口。

缺少 Cache、规则引擎、BPM 应用经验。系统性能瓶颈大部分情况下在数据库 IO，Cache 服务是解决 IO 瓶颈最有效的手段。在互联网应用中，已经广泛使用到 Cache，但是在保险 IT 系统中，很少使用到 cache 来提高性能。曾经有公司发现系统比较慢，原因在于一个完整逻辑操作过程中，多次从费率表中读取数据，多次读写投保单，非常适合 Cache 来解决。规则引擎能够有效解决系统灵活性的有效手段，但是缺少合适数据模型和处理模型相匹配，导致规则引擎的应用并没有想象中那么有效果。BPM 被认为流程灵活变化业务重组有效平台型工具，但是应用本身没有做好 SOA 服务化，并没有真正带来流程可以灵活调整的预期。

1.2.3 状数据架构现状

数据类应用开始缺少事先规划，一般都是在核心系统开始投入运营之后，才从报表应用开始规划。很多支持开业的核心系统中，也包含开业所必须的保监会所要求报表应用，还有各种销售管理（佣金计算和分配）应用。后来由于有更多数据类应用，逐渐从核心系统中将报表应用拆分开来，转移到独立应用中，但是基础数据源还是核心系统的数据库。为了减少核心系统的数据库压力，数据库也从核心系统复制出来，做成备份库，报表应用基于备份库来运行。为了进一步降低核心压力，将查询应用也逐渐迁移到备份库之上。备份库和核心库之间通过实时工具进行同步。然后进一步开始构建数据仓库，从备份库抽取数据。很多公司的数据仓库也不是真正意义上经过整合之后的企业级数据仓库，而是根据各个部门提供的需求而设计的数据集市，其下可能有一层可以为多个数据集市共享的轻度汇总层，有一部分共享维度。许多小公司还没有数据仓库，但是客观上也有为保监会要求的相关报表，常涉及到财务数据。负责监管数据的部门通常属于财务部的职责，因此，很多源数据自然汇集到财务系统，使之成为是事实上的数据平台，为很多应用提供数据。

很多公司有 ODS，支持实时查询和部分报表应用，为数据仓库提供数据，后者从 ODS 抽取数据，而 ODS 实时从备份库或者核心库抽取数据。通常 ODS 是采用近源结构，即与源系统非常接近的数据结构，也有直接采用源系统的同构库。大部分 ODS 主要扮演着实时备份库的角色，没有对数据进行整合。如果公司只有一个数据源核心库时，ODS 可以采用同构库，但是当公司有多个系统时，ODS 就变成了多个源数据库结构的总和，结构没有整合，而将整合工作留给了数据仓库，没有发挥 ODS 应该发挥的整合作用。ODS 采用同源结构不是 ODS 必有的特性，是否采用同源结构，关键在于该结构能否整合和兼容来自源系统的数据。为实现源库和 ODS 之间数据增量抽取或者实时同步，要求在源系统的每个表中增加时间戳标示是否更新，并且要求应用维护该时间戳，导致已经投入运营的应用到处修改，经常修改不彻底，尤其存在直接从数据库删除记录的情况，记录时间戳更加复杂，加大 ODS 实施难度，甚至出现在源系统已经被删除的记录，而 ODS 继续保留的现象，导致数据不一致。由于缺少元数据管理，系统之间指标口径不同，也由于没有详细数据历史轨迹，导致业财数据不一致的情况经常出现。数据流转路径过长和数据抽取同步存在时间差，进一步加剧数据不一致的现象。

很多系统的数据库边界模糊清晰，系统之间经常通过数据库连接访问。由于很多系统从原有核心系统拆分出来，看似一个独立系统，但其数据库不能独立，依附于核心系统，因为