宇信易诚消费信贷管理系统

——架构设计说明书　**v0.1**

目　　录

1　概述 4

1.1　文档目的 4

1.2　背景与建设目标 4

1.3　设计规范与约束 4

1.4　参考资料 5

1.5　述语 5

2　架构需求分析 6

2.1　消费贷关键业务场景分析 6

2.1.1　场景：申请 6

2.1.2　场景：电核 6

2.1.3　场景：审批 7

2.1.4　场景：面签 8

2.1.5　场景：还款计划与费率计算 9

2.2　消费贷业务特征 9

2.3　设计目标与原则 9

3　架构设计 11

3.1　系统业务架构 11

3.1.1　业务模式 11

3.1.2　业务流程 11

3.1.3　功能划分 12

3.2　系统逻辑架构 13

3.2.1　功能层次划分 13

3.2.2　功能层次关系 14

3.3　系统技术架构 15

3.3.1　子系统划分 15

3.3.2　技术选型 17

3.3.3　技术架构分层 17

3.3.4　关键技术点 19

4　功能设计 24

4.1 功能模块划分 24
4.2 功能结构设计 25
5 非功能设计 28
5.1 性能设计 28
5.2 安全设计 28
5.3 容错设计 29

1 概述

1.1 文档目的

《架构设计说明书》 用于确定消费信贷系统的整体架构， 明确业务功能结构、 技术方向、以及设计原则， 为后续阶段进行概要设计、详细设计、编码开发以及测试提供方向性、 原则性的指导。

消费信贷系统主要针对消费金融公司、 银行消费信贷部门的业务运营需求而设计， 本说明书将从消费贷业务特征分析为切入点， 从业务架构、逻辑架构、 技术架构等多个维度，逐步分析采用何种技术架构可以在最大程度地满足现有业务需求的同时， 也能兼顾将来一段时间内的业务发展变化。

1.2 背景与建设目标

基于国内整体消费金融业务的发展情况和银行关注消费金融的程度， 以及国家加速发放消费金融牌照的趋势， 为了能够抢占消费系统服务市场份额， 特别研发新一代消费信贷管理系统。消费系统建设整体目标如下：

1、建立先进、有效、多类型的进单渠道，并建立与渠道的沟通方式，以扩大与外部合作机构、消费者的联系和服务质量；扩大客户群体和异地服务的能力。

2、为了支持消费贷款业务短、平、快、业务量大等情况，建立适合的业务处理流程。实现业务的精细化管理、 统计分析、 监测、审批、 控制的电子化和自动化， 提供存储、 汇总、收集、反映，为各层次的经营管理者提供监控、决策、分析、预警等功能，为信贷业务的创新、经营决策提供充分的信息支持。

3、高效的影像审批流程：通过消费信贷管理系统和影像系统的整合，以及通过系统提供在线通知、在线打印等自动化功能，实现业务审批模式的突破，满足消费业务集中审批、无纸化办公的要求。

4、智能的数据决策分析平台：通过各种报表的展示和数据的统计分析查询，为不同层次的人员提供相应的数据，提高决策的效率和决策的效果。

5、标准外围系统接口和数据规则：建立统一的外围系统接口， 包括人行征信 /公安身份核查 /总帐接口等；建立导入 /导出数据和模板标准，便于与第三方进行数据文件交换。

1.3 设计规范与约束

Web 容器（ Servlet） JSR154规范

流程引擎遵循 WFMC 规范

规则引擎遵循 JSR199规范

内容服务器遵循 JSR170、 JSR283规范

基于宇信公司 EMP 平台进行构建，在遵循 EMP 的架构规范的同时，在技术实现方式上接受 EMP 约束.

流程引擎参照 WFMC 标准进行设计，与非 WFMC 标准的其它流程平台对接实现上存在一定的技术困难，需作为关注的技术实现约束条件。

1.4 参考资料

《消费金融公司试点管理办法》

1.5 述语

个人消费贷款 ：金融机构向个人客户发放的有指定消费用途的人民币贷款业务，用途主要有个人购物、住房、汽车、一般助学贷款等消费性个人贷款。

CC：电话客服中心系统。

电核 ：金融机构通过电话沟通方式核实借款人身份、贷款用途等基本情况。

面签 ：贷款审批通过之后，约客户到场签订借款合同、缴纳费用，向客户说明贷款权利义务。

合作方 ：指金融机构在营销贷款产品时的合作伙伴，如合作商户、大卖场、4S 店等等。

功能模块 ：系统技术实现时，封装一类业务功能的组织单元，一个功能模块下有一个或多个功能组件。

功能组件 ：系统技术实现时封装业务功能与处理逻辑的组织单元，每个单元在概念上与业务过程中的业务实体基本一致，如客户组件。

页面部件 ：系统技术实现，用于在系统界面中展现业务要素的基本单元，如文本输入框。

2 架构需求分析

2.1 消费贷关键业务场景分析

2.1.1 场景：申请

消费贷款申请，是信贷类系统的典型场景之一，申请主要目的是为收集客户贷款融资需求与客户的基本信息，同时申请还需借助各种渠道来充分发挥其市场营销功能。在消费贷的申请场景下，会有多种运作模式，例如由客户经理直接登录系统操作完成、由客户通过各种渠道操作完成、由合作商户通过各种渠道代理客户操作完成等等。消费贷申请用例图如下：

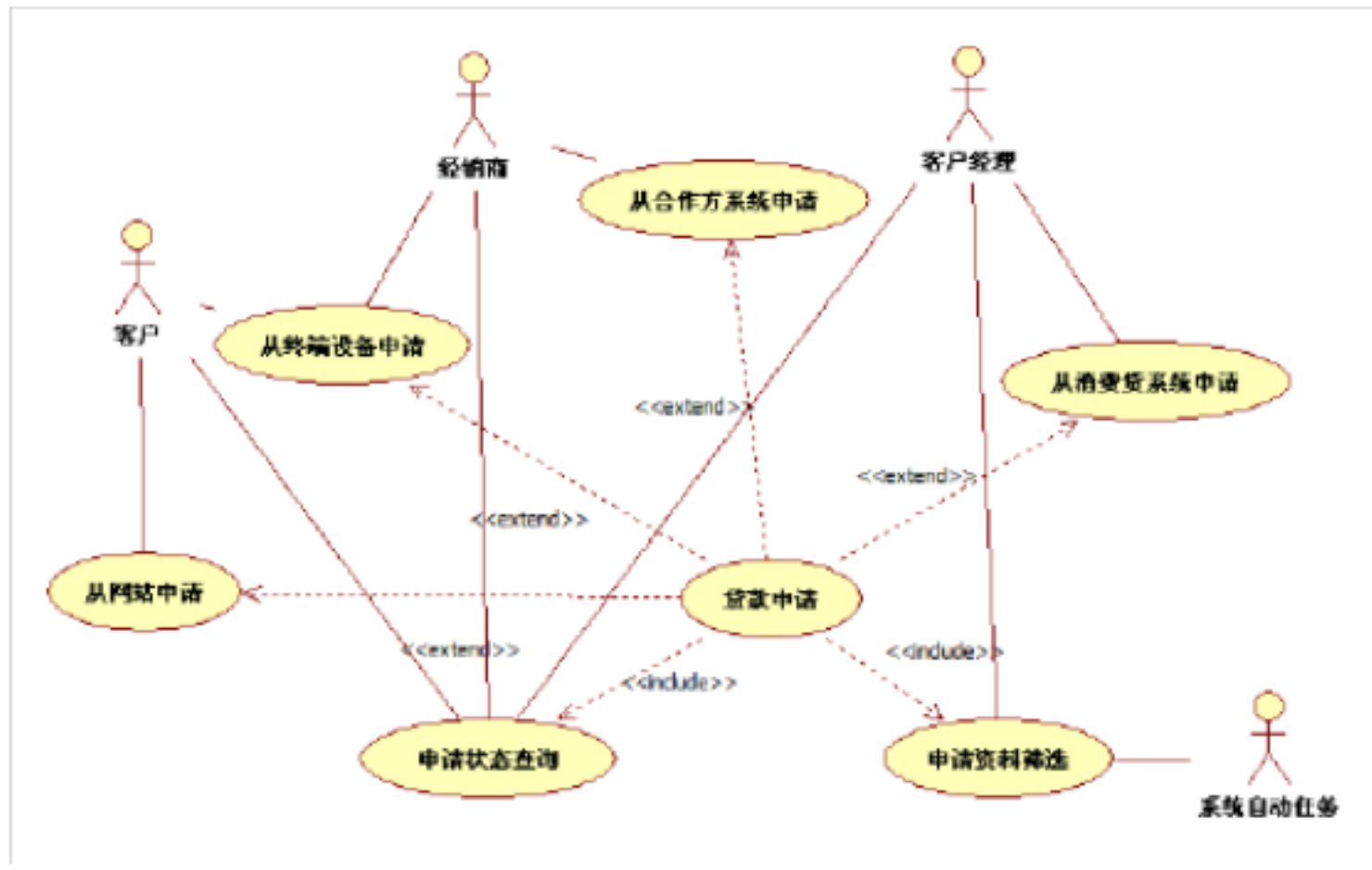

图 2-1

从上图中可知，消费贷申请过程中涉及到的角色有客户（申请人）、经销商（合作商户）以及客户经理。客户可以从网站（银行本身或第三方）、终端设备发起贷款申请；经销商可以从终端设备、合作商户子系统（消费贷子系统）发起申请；客户经理可以从消费贷系统发起申请。对客户、经销商与客户经理均可以通过系统查询申请的状态（其权限范围内）。客户经理与后台自动任务可以对申请资料的合规性进行筛选。

消费贷申请，是整个系统的业务操作入口，其涉及角色、渠道多，情况复杂变数多，同时在消费贷下的不同产品申请时所关注的业务要素可能有较明显差异，系统需要充分考虑金融机构消费贷产品种类井喷时系统的应对方式。由于涉及不少渠道是由客户主动填写并发起的，所以其数据量可能会很大，相应垃圾数据量也会很大，所以需要有人工与系统自动相配合的数据筛选功能，对大量申请数据进行甄别。由于业务申请量可能较大，在设计时需要考虑尽量将录入工作从客户经理的日常工作中剥离出来，减轻客户经理重复工作量。

2.1.2 场景：电核

消费贷款电核，用于对客户身份与申请信息真实的核查，一般情况下，电核为金融机构

内的独立部门使用独立的系统（如 CC）进行，消费贷款系统仅配合其完成电核操作即可，而不再去实现与 CC 系统相关的功能（如任务调度、座席管理等等）。消费贷用例图如下所示：

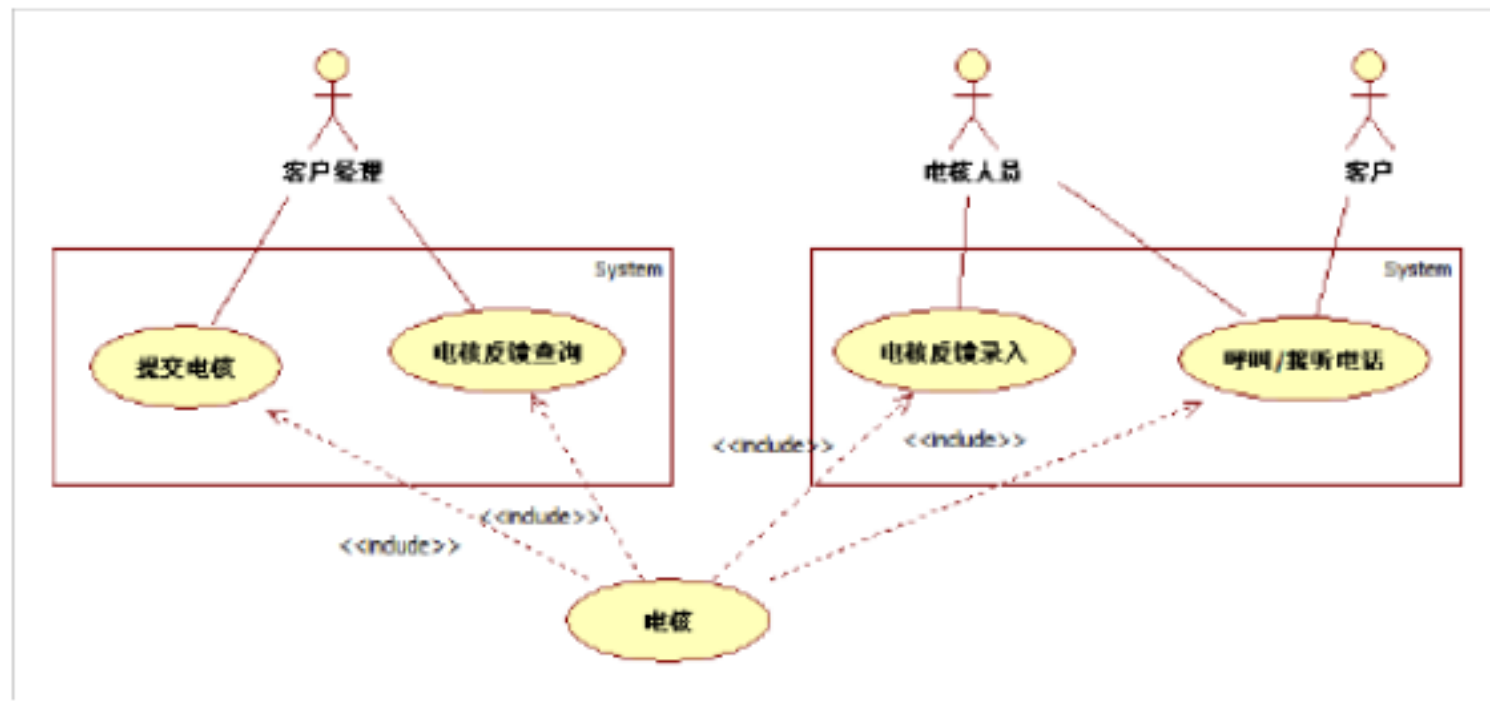

图 2-2

从上图中可知，消费贷电核过程中涉及到的角色有客户经理、电核人员与客户。客户经理从消费贷系统中将资料提交至电核系统，同时也可以查询电核反馈结果；电核人员在电核系统中与客户电话联系，并将电核情况记录至系统中。

电核过程，对于消费贷系统而言是一个可选部分，不是所有消费贷产品都需要这一步骤，而对于没有类似 CC系统的金融机构，但需要对客户作电核，则可以考虑直接作为系统业务流程的一部分来简化实现，即在消费贷中系统录入线下电核结果即可。

由于电核步骤涉及独立电核系统，所以在设计时需要充分考虑与电核系统间可能的交互模式，保证在接入不同的电核系统时，对消费贷本身业务流程的实现没有影响。

2.1.3 场景：审批

消费贷申请审批，也是信贷类系统的典型场景之一，审批主要目的是对客户的贷款申请，按金融机构内规章制度的进行逐级审批。不同金融机构、不同的产品其审批流程步骤、参与人员、以及审批规则将会有明显区别。消费贷申请审批的用例图下如：

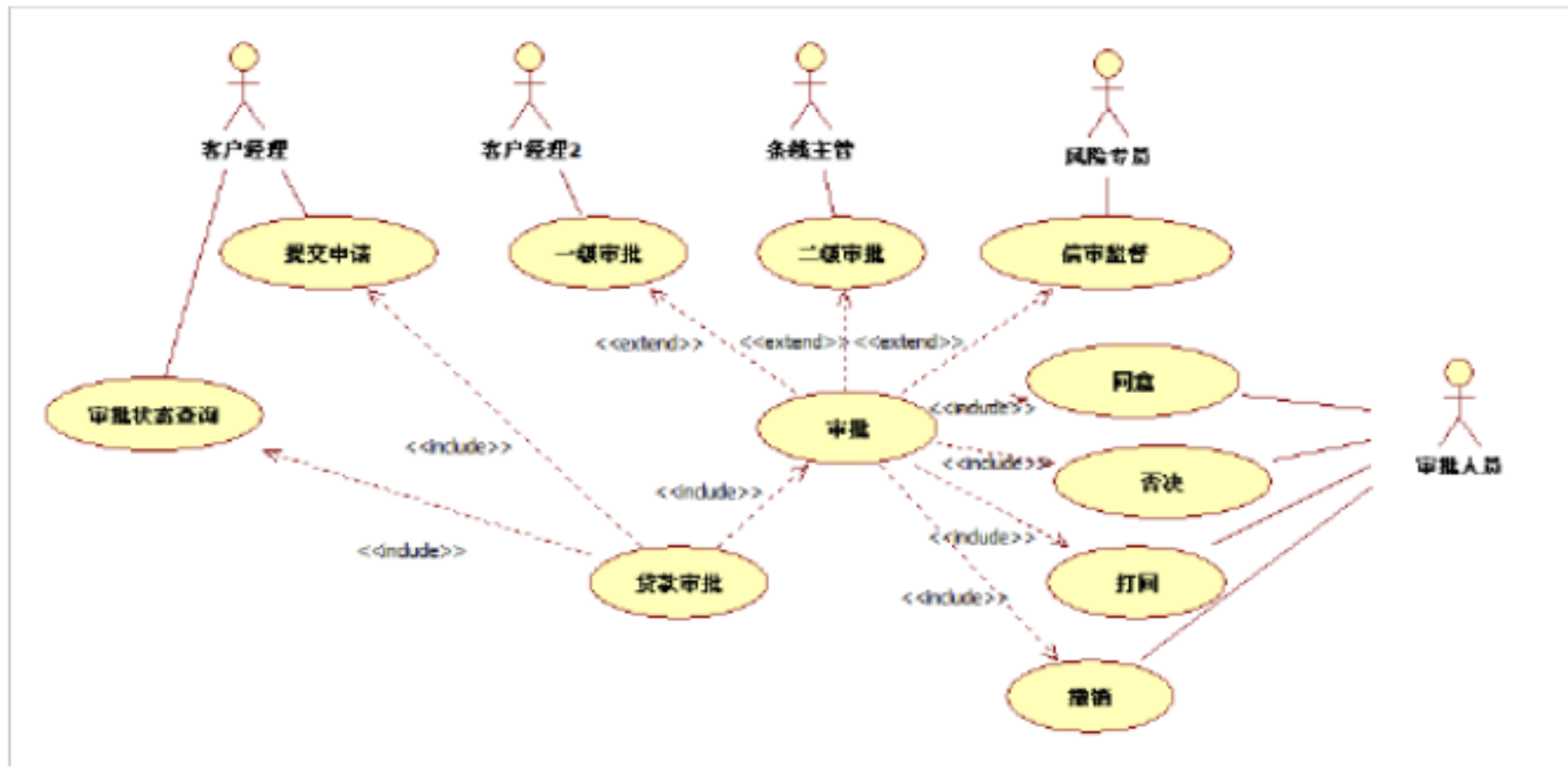

图 2-3

从上图可知，参与审批的角色主要有客户经理与审批人员（如，其他客户经理、主管、风险专员等等），担当审批人员的角色不同机构、不同产品、甚至同一产品内不同的申请（如，新件与复议件） 都会有不同角色人员来完成审批工作， 其完全取决与行内的规章制度。 但无论审批角色如何变化，其需要完成的动作基本一至，主要包括对申请的同意、否决、打回或撤销等等。

在设计时需考虑消费贷审批过程的不确性， 系统需要提供灵活配置功能来应对不确定的审批需求。由于每天申请单量可能较大，需要在设计层面充分考虑如何提高审批工作效率，例如批量审批操作、自动审批、关键消息提醒等等。同时，需要提供充足的审批统计数据，为行内的审批流程优化再造提供数据分析支撑。

2.1.4 场景：面签

面签是信贷类系统的典型场景之一， 面签主要目的是在审批通过之后约客户到场签订借款合同、 缴纳费用， 向客户说明贷款权利义务等等。 一般而言面签主要工作是在系统线下完成，线上主要合同打印、登记面签结果、确认合同已签即可。但对于消费贷而言，对时间要求高，面签过程可能会直接在场外完成，以提高贷款的效率。消费贷面签用例图如下：

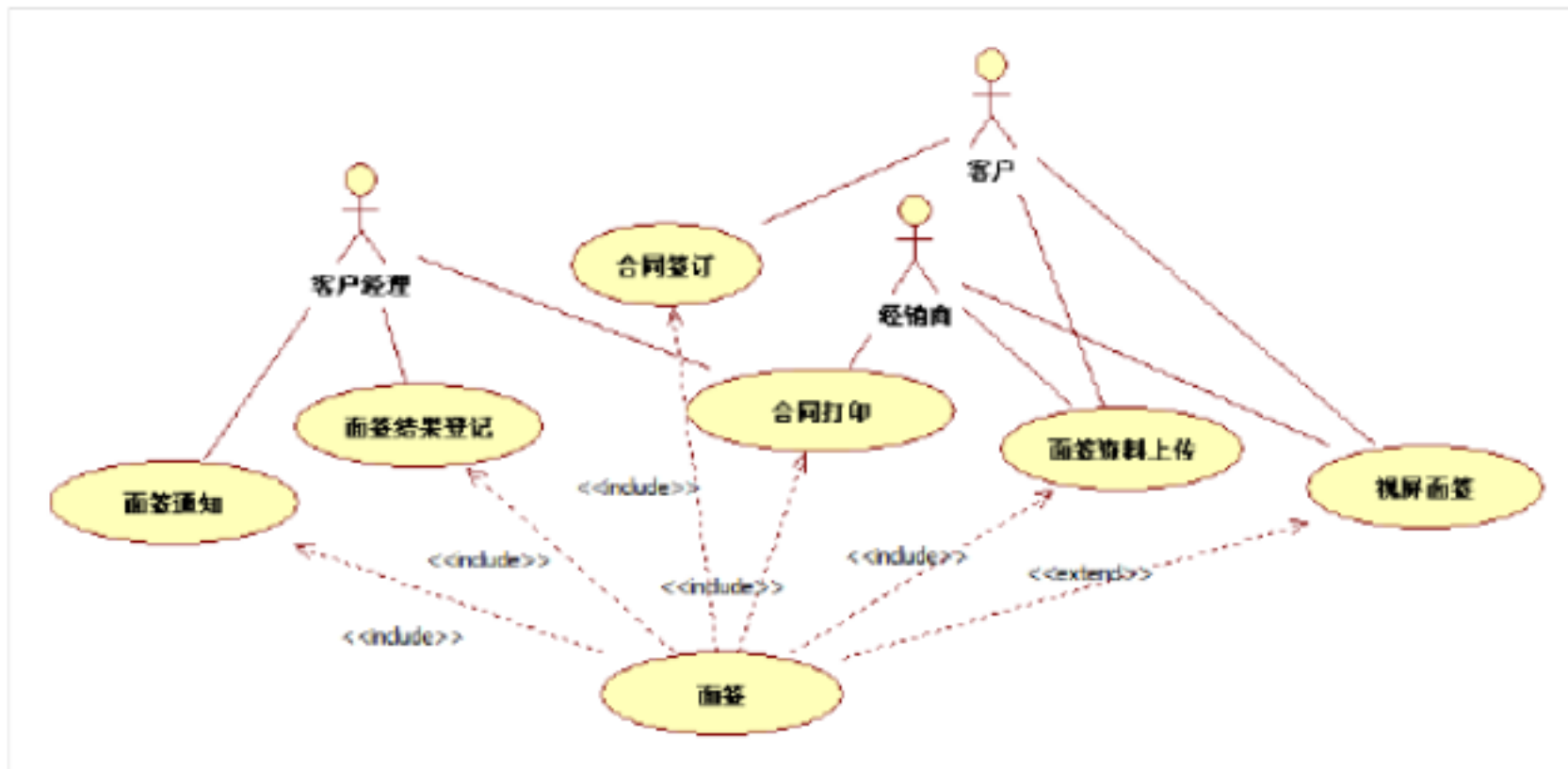

图 2-4

从上图可知，面签过程参与角色主要有客户、客户经理、合作商户（如经销商） ，如果在场内签订合同， 则客户与客户经理参与即可， 如果在场外签订合同， 则可能由客户、 合作商户、与客户经理共同参与，也有可能在场外通过专用设备进行远程面签（如视屏面签） ，此时就不需要客户经理必须到场，仅需合作商户通过配合即可。

由于消费贷面签过程大多都是在场内线下进行，但随着 IT 技术的不断发展，面签功能会逐步放在场外上线进行，系统需要充分考虑在新的面签模式下如何能很好的支持。

2.1.5 场景：还款计划与费率计算

由于消费贷大多数都是采用按揭方式还款，对灵活的还款计划计算的必不可少。系统需要能支持各种方式的还款计划计算，以及相应的费率计算。由于金融公司可能没有核算系统，所以系统中需要能提供部分账务核算功能；而对于有核算系统的银行，则需要实现与其核算系统的接口。为此系统需要充分考虑如何支持有核算系统与无核算系统两种情况。

2.2 消费贷业务特征

消费贷款产品与其他贷款产品相比，其基本特征是小额、快速与灵活。在运作模式上主要特征是与特约商户合作推广业务。在目标客户群体在主要特征是针对年轻人、年轻家庭以及由合作商户推荐的客户。服务模式上，主要特征是 IT 自助型服务模式。

消费贷‘快速’这个特征最为突出，其中最快的贷款产品要求可以直接在商户卖场中 1 小时之内完成，一般的消费贷款产品，要求在 1~3 个工作日内完成。而机构内审批时间要求最快能在 30 内分钟完成。可以说没有什么贷款产品比消费贷对工作效率要求更高，这对于承载消费贷业务运营的 IT 系统而言，如何满足对高效率的要求将是一个挑战。

消费贷‘灵活’主要体现在还款方式、利率与贷款品种上。由于消费针对的目标客户的用款需求十分明确，相对与信用卡（等其贷款产品）而言，还款方式与利率定价的针对性更强，对于客户而言享用金融服务的方式（如更适合自身条件的还款方式）也更为多样化，更灵活。由于消费领域十广泛，与商户合作方式很多，对应的消费贷会衍生出很多产品，如针对耐用消费品的贷款产品、针对电子消费品的贷款产品、针对教育培训的贷款产品、针对汽车衍生品的贷款产品等等。对于 IT 系统而言，需仔细考虑如何应对业务产品形态的多样化。

也许是由于年轻人更能接受超前消费的观念的原因，市面上已有消费贷产品面向的客户主要群体一般是年轻人，消费贷产品将来一定会考虑采用更有针对性的自动营销渠道，例如直接在网上商城内营销、手机 /PAD 等终端设备上营销、微博营销等等。对于 IT 系统而言则需要能支持多种的营销渠道来源，同时也能提供给客户更多享用金融服务的渠道，如通过手机/PAD 等 IT 自助服务。由于金融机构的消费贷产品未来发展，在很大程度上取决于对合作商户的不断发掘，对于 IT 系统而言，需要充分考虑如何应对不断增加的合作商与合作模式给系统功能带来的负面影响。

2.3 设计目标与原则

消费贷系统作为一个独立、新兴的贷款类业务操作系统主要围绕着以下三个主要目标进行设计与建设：

1、系统能提升消费贷业务运作的效率；

消费贷业务对时效性要求很高，在设计时需要优先考虑，系统能提供哪些模式，让工作

效率得到最大限度的提高。提高用户在系统中的工作效率，首先是要有良好的 UI 设计，在 UI 设计时考虑能达到不同业务场景下突出不同的重点的效果，尽量实现‘消息驱动’的 UI 操作模式。

要提高效率仅仅从 UI 层考虑是不够的，更应该从优化业务操作流程着手，进一步考虑 IT 技术如何帮助业务流转加速，例如，自动化数据筛选、自动化审批、任务智能分配、大任务拆分、基于数据分析的流程再造等等。

除此 UI 与业务流程优化之外，在系统设计时需考虑提供相应机制，保障在业务高峰期的系统性能，尽量保证不会出现因为系统响应速度原因而影响业务效率，例如，合理的子系统划分、流量控制、支持系统的横向扩展架构等等。

2、 系统能规范业务流程、屏蔽操作风险；

消费贷系统作为业务操作管理类系统而言，能够监控与约束对机构内人员的行为，屏蔽人为操作风险的发生，是最原始的初衷之一。系统设计时，不仅仅把贷款的审批过程用工作流模式来实现，而是考虑采用其来实现操作层面的主线功能，通过在系统中构建各业务场景间流转逻辑关系，来实现业务规范在系统中的落地。同时，工作流模式还需要与风险识别与拦截机制结合起来，才能真正发挥规范业务流程的作用。

3、 系统能支撑消费业务的不断发展；

消费金融公司在国内处于起步阶段，未来的变数还很大，其主要表现在新消费贷产品可能会大量出现，甚至会出现全新的业务模式。现在所设计的消费贷系统，必须要认真考虑如何能适应将来较长一段时间内（ 3 年）的业务需求变化，至少要保证不影响业务的发展，同时，尽量能缩短未来新产品的落地开发周期，不影响新产品的投放时间。

3 架构设计

3.1 系统业务架构

消费贷业务不同于传统的个人消费类贷款，其有着更灵活的业务运营方式，下面将从业务模式、业务流程、功能划分三个部分来分析消费贷业务的架构。

3.1.1 业务模式

消费贷的业务模式与传统的个人消费类贷款、以及个人信用卡相比，最大的不同就是营销渠道的不同，采用现场主动营销方式，即直接在客户消费场所营销，而不是被动地等着客户上门来贷款或者刷卡消费。为此，消费贷的运作需要合作方支持，业务模式见下图：

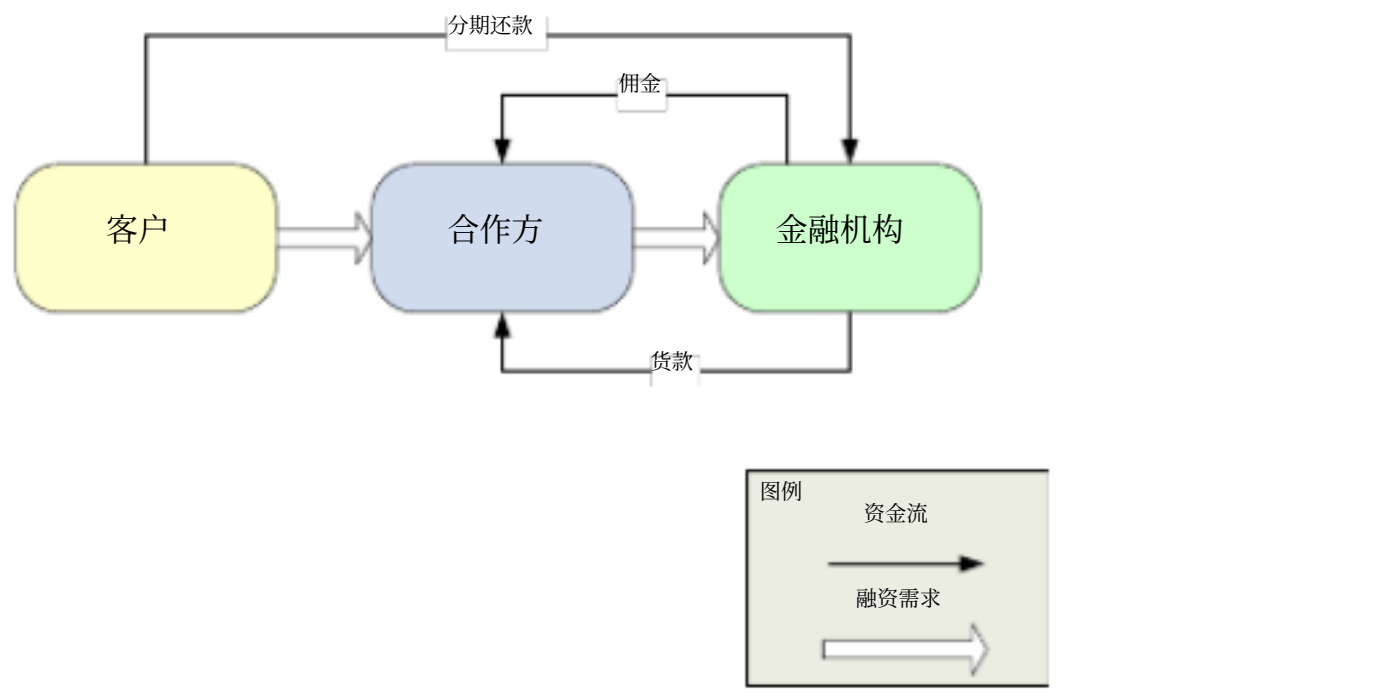

图 3-1

在传统贷款方式下，客户的贷款需求是直接面向金融机构的，而消费贷是通过在合作方（如卖场、 4S 店、培训机构等等）的内主动挖掘需求。在成交之后，金融机构直接将货款打给合作方；客户方分期向金融机构还款；金融机构在一定条件下，定期付给合作方一定的佣金。在这种模式下金融机构赚取了利息与手续费、合作方扩大自己的客户群并能赚取一定的佣金、客户则提前买到了合作方的产品或服务，实现了三方的共赢。

在这个模式下，对于金融机构而言，最重要的是如何找到合适合作方，且能尽可能多地扩展新的不同领域的合作方，同时提供合作方感兴趣的激励机制（如佣金等等）；对于客户则能提供更有针对性的还款方式。

3.1.2 业务流程

消费贷在业务流程上与普通贷款在本质上没有太多区别，其目地都是为了实现审贷分离。区别在于消费贷为提升效率，流程中各节点专业性更强、分工更明确，即采用类似信贷工厂的模式（ 注：消费贷并不是工厂模式 ）。业务流程图如下所示：

图 3-2

消费贷业务流程中，电核为可选节点，有的机构或产品不需电核这一步骤，其它步骤均会必选步骤，面签为线下节点，今后可能会放至线上。

申请节点，由客户经理或客户填写并发起；电核节点，由独立的电核人员完成对客户的核实；审批节点，由管理人员完成对申请的审批；面签节点，由客户经理与客户一起完成的同合的签订；放款节点，由财务人员完成打款操作；还款节点，由系统自动完成从客户还款账户上扣款，或客户主动发起提前还款。

随着业务量的增大，今后可能会出现无法满足对放款时限要求的情况，届时将考虑将上述流程中部分业务外包出去，例如申请与电核。同时，尽可能采用自动流程部分代替其中的人工操作。

3.1.3 功能划分

消费贷系统主要面向独立的金融机构，一般其核心职能会划分到四个子部门：产品管理部门、风险管理部门、营销部门、财务部门。对应图 3-2 中，申请、面签交由营销部门中的客户经理完成，审批由产品管理部门与风险管理部门负责，放款由财务部门完成。具体消费贷功能划分如下：

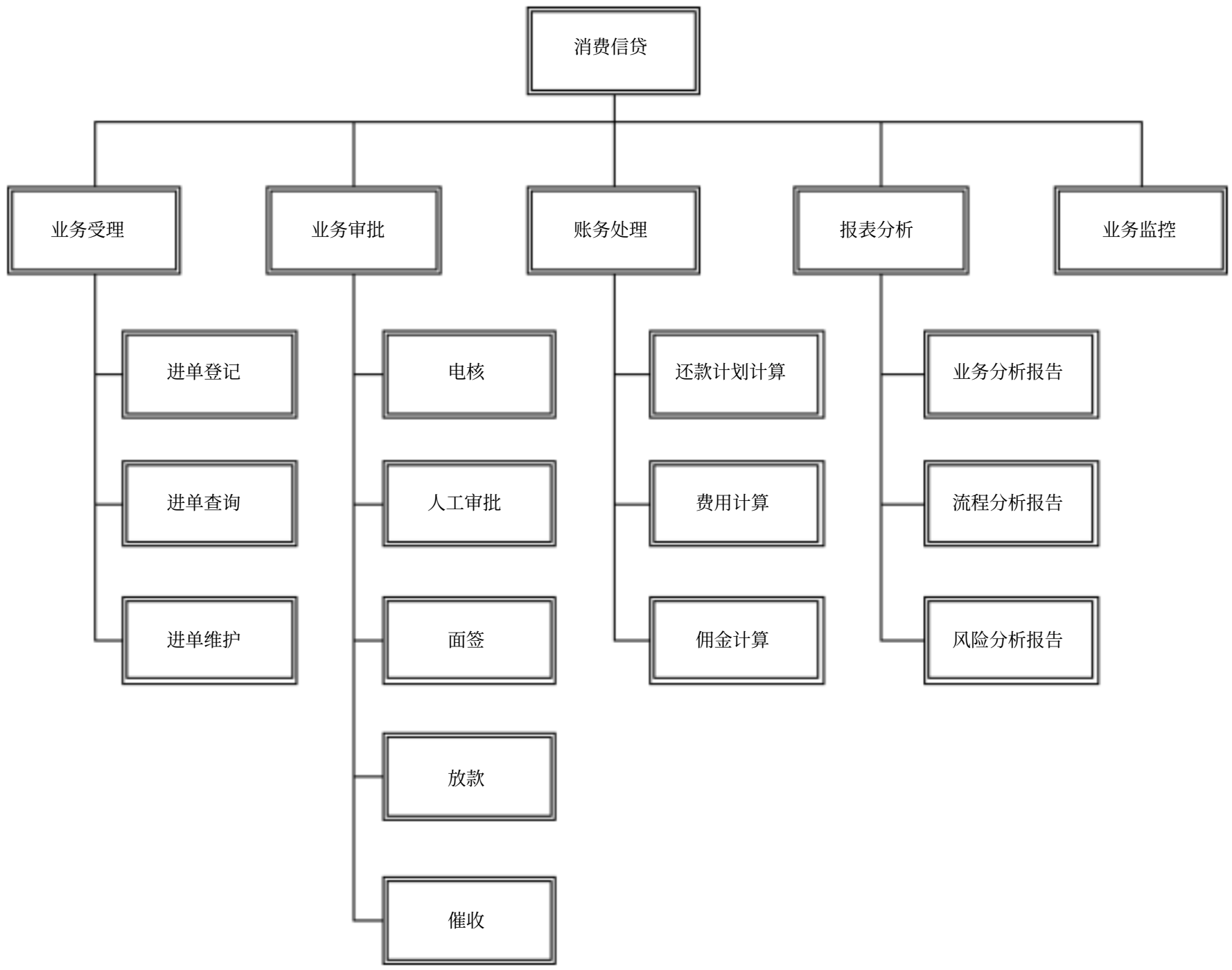

图 3-3

消费信贷业务功能由五个部分组成，包括业务受理子模块、贷款业务流程处理和管理子模块、帐务模块、报表子模块、业务监控子模块，其包含了完成消费主体业务所需的全部功

能。各子模块完成专项工作，　　如受理模块主要完成营销和渠道进单功能；　　业务流程处理和管理模块主要完成金融机构内部对业务的所有业务处理，包括电核、　　人工审批、　放款、　贷后管理、催收和保全；帐务模块主要完成贷款或卡业务的各种帐务计算，包括还款计划生成、利息/贴息计算、罚息计算、复利计算、费用计算、佣金计算等；报表模块主要完成对所有业务分析、流程分析、　客户分析、　风险分析等统计和展现；　业务监控模块主要用于对业务的实时业务量监控。

3.2 系统逻辑架构

消费贷款系统功能从逻辑上划分成六个层面——业务操作层、　　业务管理层、业务工具层、决策分析层、账务核算层以及技术支撑层。功能逻辑结构如下图所示：

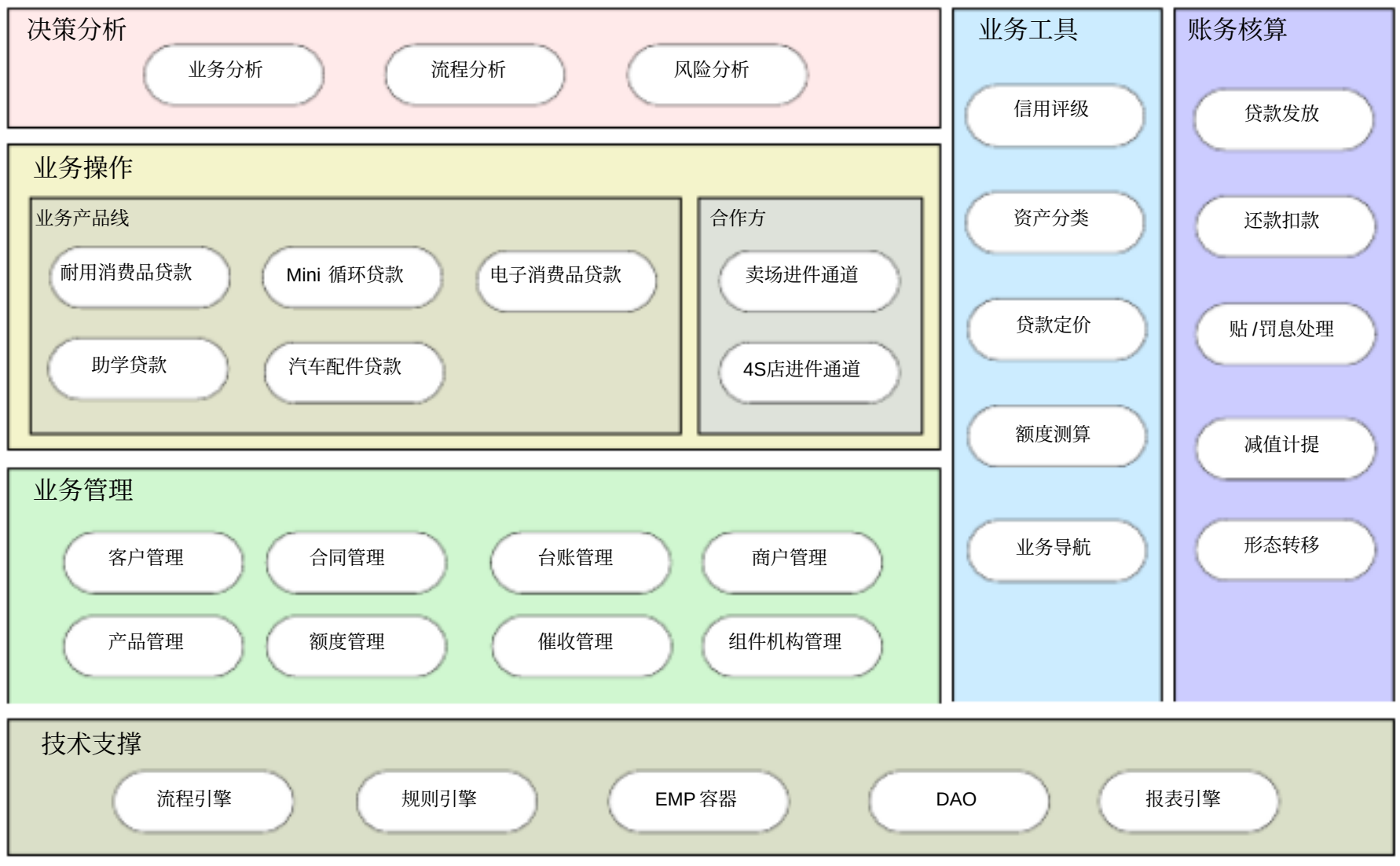

图 3-4

3.2.1 功能层次划分

业务操作层

在业务操作层中的功能用于直接面向客户的业务操作与呈现，　　可以简单地认为其相当于系统的面门，用作信贷基层人员对最终客户的业务门面与操作入口。其中包括业务产品线、合作方两大部分。　产品部分以业务产品为基本组织单位，　　其承担着一个产品在生命周期内所有对外的功能。　之所从在概念上与物理上将这一层分离出来，　　一方面能将系统功能与现实中的业务角色功能映射上，　减少现实业务与虚拟系统之间在基本概念与组织结构上的差异；　　另一方面也是因为在业务操作层面上，　　业务形态丰富、变数大，　期望通过架构上的分离，　来确保系统能适应这种特性，　不会因为新业务的加入而影响已有业务。　　可以说业务操作层就是整个信贷业务的外延。

业务管理层

在业务管理层中的功能用于对业务操作层的支撑与管控，可以简单地认为其相当于中后台，用于中、基层管理人员对业务运作的监控与管控。本层以业务生命周期过程中的业务实体为基本组织单位，其承载着各业务环节的管理、配置、监控、以及对应业务实体的基本功能（与现有划分基本对应），并为业务操作层提供具体功能服务。由于采用业务实体为功能单组织单位，其相对操作层而言是稳定的，这是因为业务实体作为业务生命周期中承载业务要素的介质，其种类个数是一定是可穷举的，其在不同贷款业务间是相同、相似甚至是可共用的，并且在不同银行间之也是相似的。而相对于业务操作层面，业务产品种类个数（尤其是对未来新产品）是不可预期的，不同业务产品间使用的业务实体也不会相同，更重要的每种产品的运作模式都会完全不同，不同银行间产品的运作模式也有明显差异。可以说业务管理层就是整个信贷业务内涵。

业务工具层

在业务工具层中的功能，用于为业务操作与业务管理的正常运作提供必要的专业的工具箱，但其并不对业务运作的形态、业务规则或流程产生直接影响。同时，业务工具是完全可以与的业务操作、管理相分离的，甚至可以替换成第三方独立系统。之所以划分出这一层，主要原因是为银行内部对专业化技能的要求越来越高，随着银行自身的发展，会形成越来越多专业职能岗位（或部门），或者干脆将部分非核心的业务操作外包给第三方。独立的工具层，可以与银行机构组织职能一一对应上；同时，也有利于在核心业务与非核心业务之间划分出一条明显界线，从架构层面去减轻两者间的耦合程度。

决策分析层

在决策分析层中的功能，用于对业务运营过程中产生的数据进行统计分析，例如，对业务办理效率分析、对贷款人群分布分析、逾期贷款占比分析等等。决策分析层中，主要通过日终批数据加工处理、报表工具来实现对数据的分析与展现。由于数据分析不是现阶段消费贷系统的关注的重点，因此，独立划分出决策分析层，有利于保持核心功能的稳定性。

账务核算层

在账务核算层中的功能，用于为没有核心业务系统的金融机构，提供一个 Mini 版的核心业务系统，其只含贷款功能，不含存款功能（注：与银监管会相关策略吻合）。其能完成贷款发放、还款扣款、罚息减免、贴息处理、减值计提、贷款形态转移等基础的账务核算功能。由于不是所有金融机构都没有核算系统，所以将本层独立出来，保持其与它层次间的独立性，有利用增强产品的适应能力。

技术支撑层

技术支撑层，用于为消费贷系统提供基础开发平台与运行容器，是所有业务功能落地的基石。其中包含的 EMP 容器、流程引擎、规则引擎、报表引擎与数据访问工具等等。本层将在后继章节进行详细介绍。

3.2.2 功能层次关系

消费贷款系统中六层功能之间是一个逐层依赖的关系，如下图所示：

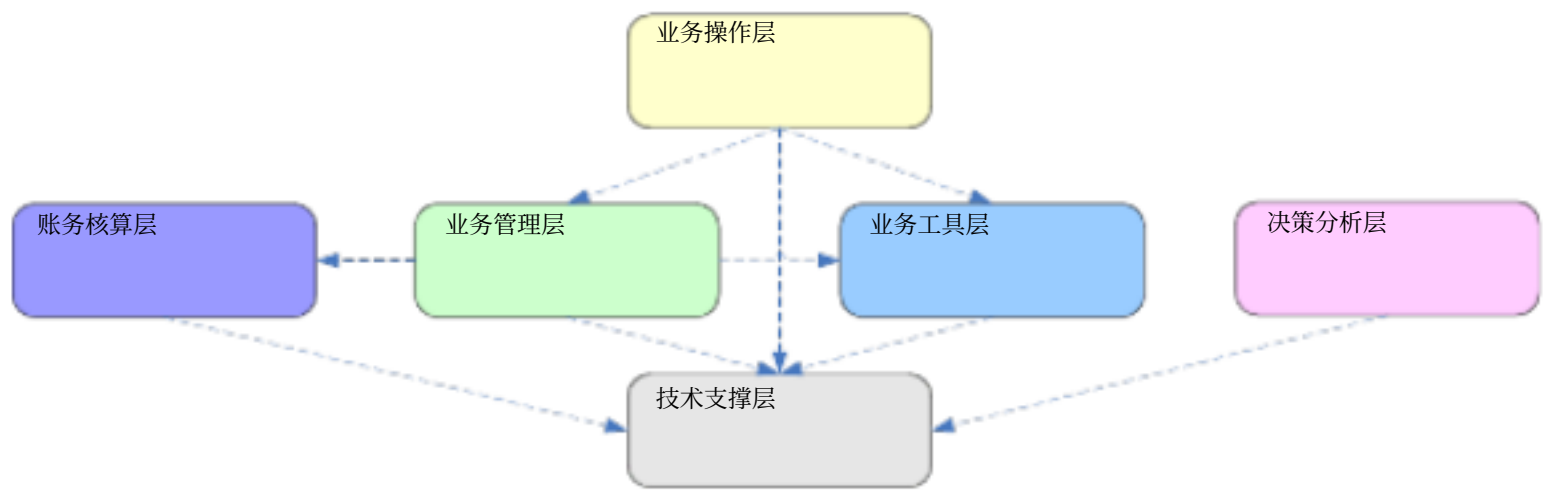

图 3-5

消费贷系统本质上是一个业务操作系统，业务操作层自然便成为了整个系统的主要门户，是所有业务的入口。业务的正常运营，离不开业务管理部门（风控部门）的管控，与专业部门（例如，财务部门、产品部门、客服部门、IT 部门等等）的大力支持。因此，业务操作层依赖与业务管理层、业务工具层、账务核算层以及技术支撑层。业务管理部门为了能更好的对业务进行管控，也需要专业部门（如产品部门、财务部门、IT 部门等等）的积极配合。因此，业务管理层依赖与业务工具层、账务核算层与技术支撑层。金融机构财部门是一个相对独立的部门，其完全依照财务规章制度运行，一般而言，其运营主要有 IT 部门的支持即可；同时，由于不是所有金融机构都需要消费贷系统提供核算功能，这就需要尽量保证其独立性，因此，账务核算层，只需要依赖技术支撑层。决策分析主要是针对数据的统计分析，因此，其仅仅依赖与技术支撑层，而对于其它功能层次，也均不直接依赖于决策分析层。

3.3 系统技术架构

3.3.1 子系统划分

消费贷系统按功能职责划分成四个子系统——消费贷款管理系统、消费贷款合作方系统、消费贷款核算系统、消费贷辅助系统，如下图所示：

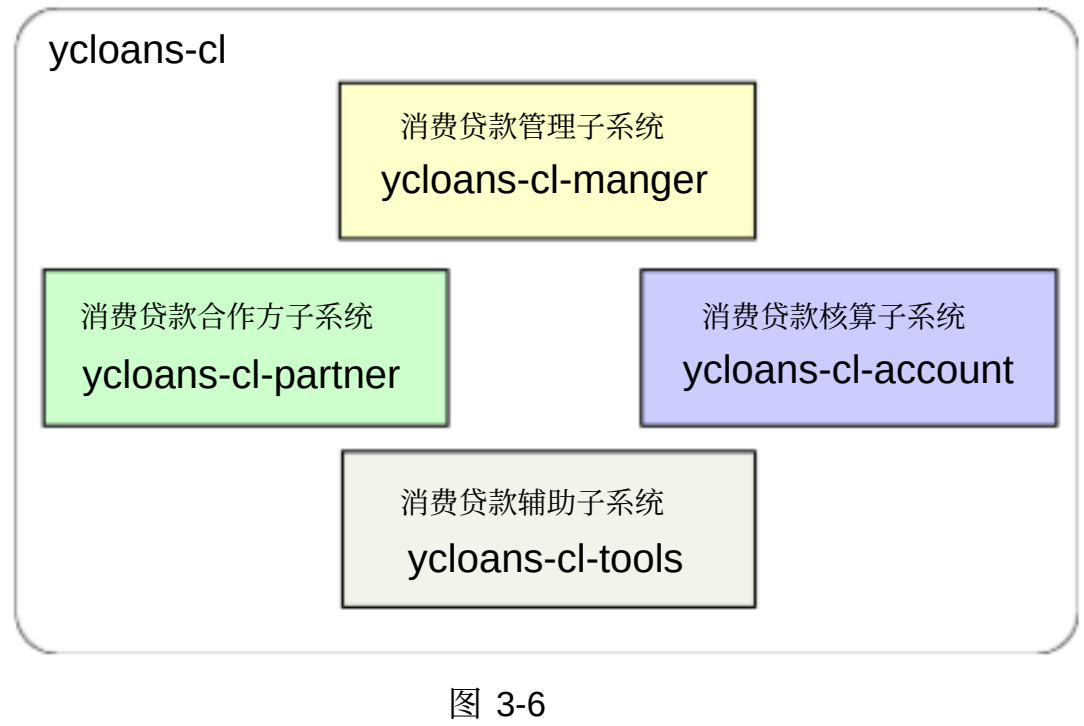

图 3-6

消费贷款管理子系统

消费贷款管理子系统，主要面向金融机构内部的客户经理以及相关管理人员，负责完成消费贷款进件流程与业务管理。其中包含了逻辑架构中的业务操作层、业务管理层、业务工具层中的绝大部分功能（注：不含操作层中的合作方功能），例如，贷款申请、人工审批、合同管理、放款等等。消费贷款管理子系统是整个系统业务运营的核心系统。

消费贷款合作方子系统

消费贷款合作方子系统，主要面向金融机构之外的合作机构，如卖场、4S 店、培训机构等等，其负责为合作方与金融机构之间建立渠道与门户。其主要功能是进件处理与审批状态查询。之所以将合作方独立成子系统，首先，是出于金融机构与合作方之间合作方式存在不确因素的考虑，例如，现在是对卖厂、对 4S 店，但今后可能会对网店、对 PAD、对外包公司等等，因此将合作方的功能独立出核心业务系统，作为系统对外扩展的网关，会有利于消费贷的不断发展；其次，是出于安全方面考虑，由于需要直接面对互联网，所以必须将对外的功能独立部署，屏蔽任务从互联网上直接访问至核心业务系统的渠道。

消费贷款核算子系统

消费贷款核算子系统，主要面向金融机内部的账务人员，负责完成所有与贷款相关的账务处理，如贷款发放、还款扣款、罚息减免等等，其与逻辑架构中的账务核算层相对应。独立的核算子系统，有利于增强系统的适应能力。

消费贷款辅助子系统

消费贷款辅助子系统，主要负责完成数据加工与分析、定时任务调度、报表生成与展示、历史数据查询等后台任务。之所以划分出辅助子系统，主要是出于对系统性能的考虑，通过独立部署的子系统来完成后台任务，避免后台任务对日间业务运行效率产生不良影响。

子系统间依赖关系

消费贷款四子系统之间的依赖关系如下图所示：

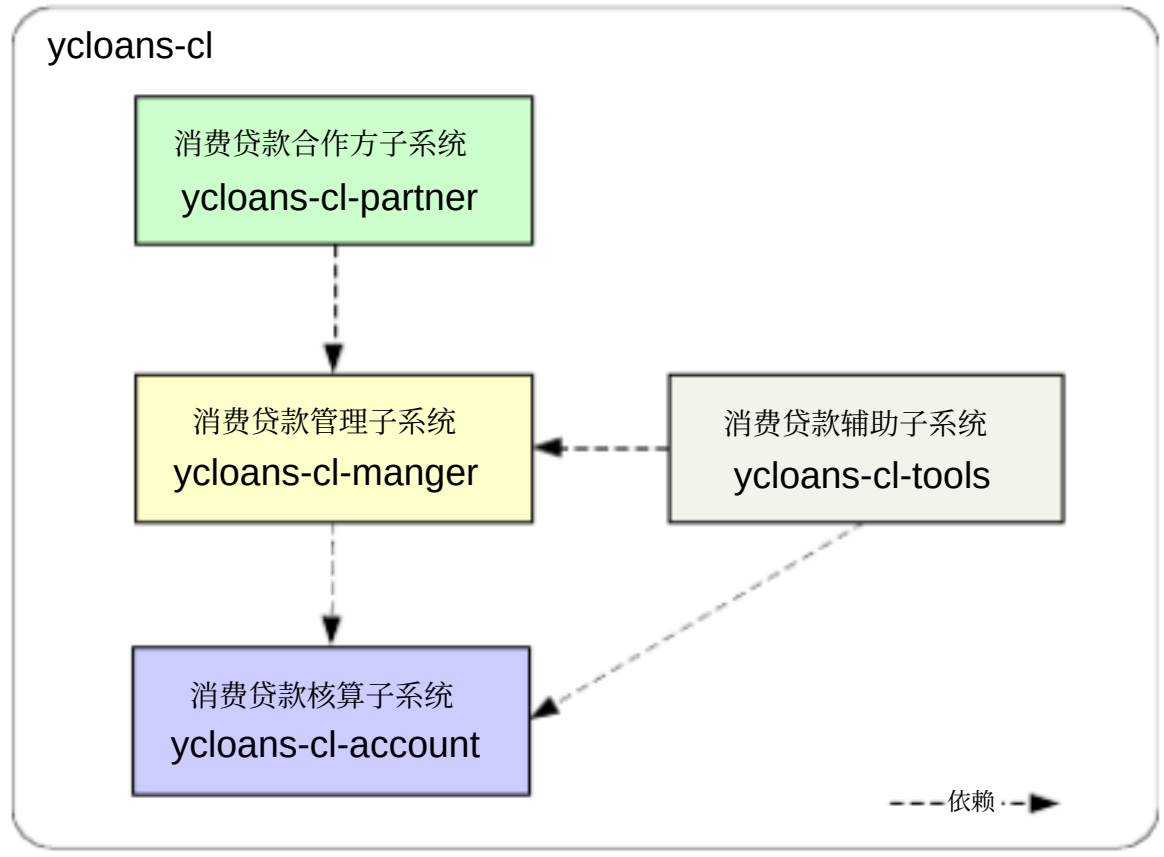

图 3-7

合作方子系统的运作必须依赖与管理子系统的支持，合作方子系统负责对外渠道，其独立运行没有任何意义，需要通过管理子系统来完成对每笔进单的业务处理。

管理子系统的运作，需要有核算系统的配合，核算系统可以是消费贷系统内嵌的核算子系统，也可以是机构内的核心业务系统。管理子系统通过核算子系统完成所有贷款账务相关的业务操作。

辅助子系统的运行，需要依赖管理子系统、核算子系统的正常运行，辅助子系统用作消费贷系统的后台处理单元，其独立运行没有任何意义，需求通过调用管理子系统与核算子系统中的业务功能来实现自身功能。

3.3.2 技术选型

消费贷系统是基于 JaveEE的 WEB 应用，其后台技术框架采用公司 EMP 实现，前台业务展现框架基于 Jquery+EasyUi 实现，具体如下图所示：

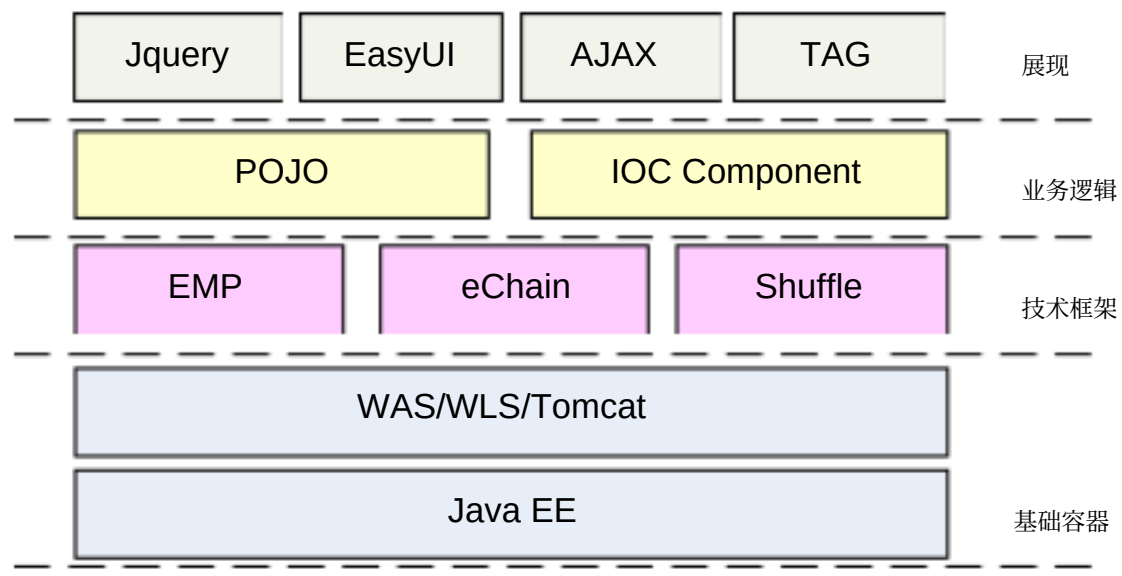

图 3-8

消费贷系统基础支撑环境为 JavaEE以及在此之上的 WEB 容器中间件（如 WAS、Weblogic 等等），对 Java 虚拟机版本要求为 1.5 或 1.6，对 WAS版本要求为 6.1 以上（含），对 Weblogic 版本要求为 10g 以上（含）。

技术基础框架采用公司产品 EMP2.2，流程引擎选用公司产品 eChain2.2，规则引擎 shuffle1.0 ；业务处理逻辑实现，采用组件注入机制与 POJO为业务数据结构方式；业务展现逻辑基于 Jquery 实现，页面部件基于 EasyUI实现，并用 Jsp Tag包装，前后台通讯有使用 Ajax 方式。

业务逻辑处理没有选用 EMP 提供的标准开发方式实现（逻辑流 +KCOL），而是选择了组件+Pojo 方式实现，由其是在数据结构上不再使用 EMP 的 KCOL，而是选择传统的 Pojo 方式。这是因为，希望通过功能组件 +Pojo 方式来实现面向对象设计思路，不再去走面向过程 +快捷开发的老路。这是因为消费贷系统是业务操作类系统，其中的业务运行模式、信息结构都是具有积累价值的，所以可以用面向对象的设计思路，将业务模式转换成技术框架，将业务功能包装成一个个组件，将业务要素信息映射成数据对象，最终，达到提升系统可复用性与产品形态的目地。

3.3.3 技术架构分层

消费贷系统技术架构上划分为四个层次——业务展现层、服务提供层、业务组件层、持久层。技术架构层次划分如下图所示：

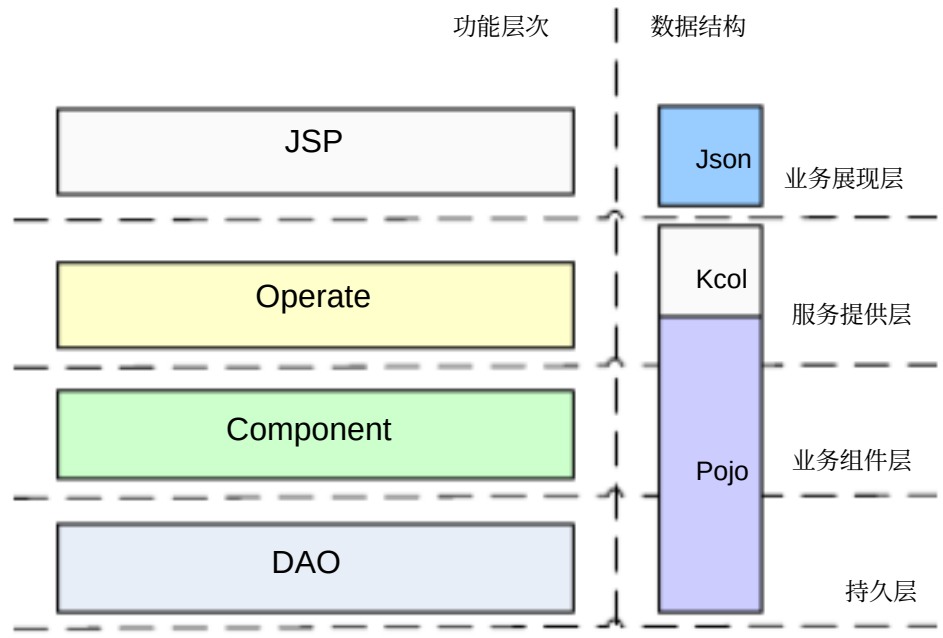

图 3-9

业务展现层

业务展现层负责消费贷系统与用户之间的交互接口，在这一层中的业务逻辑处理统一由JS负责实现，不会出现其它形式业务逻辑处理（如，JAVA代码）。无论在逻辑处理时，还是加载后台数据时，还是向后台发送请求时，其数据结构统一使用JSON格式。

服务提供层

服务提供层负责将系统内的功能组装成独立事务的原子服务，并以此为单位响应来至展现层（或来至其它渠道，如SOCKET、WEBSERVICE）的服务请求。这里提到的服务，并不是指通常概念中的服务，其主要是指系统内部暴露给其用户的功能，当然，完全可以将其中的一部分功能，通过各种渠道发布给其它系统使用，此时其才是真正意义上的服务，这里只是借用并扩展了服务这个概念，将系统提供给用户的功能也认为是一种服务，确切地说是细粒度的原子服务。在服务提供层中，通过对组合对业务组件层中组件的调用来实现服务功能，服务层中的每个服务对应着一个业务操作，其名称都应该是一个动词，表示一个业务过程的业务操作（如，发起申请、合同签订等等）。同时，在本层中还负责将从展现层（或渠道）过来的EMP KCOL格式的业务数据转换成POJO格式的数据。

业务组件层

业务组件层负责对业务功能进行划分、封装与实现，组件层为服务层提供功能实现层面的支撑。其中以组件为单位对业务功能进行组织，每个组件有着明确的业务含意，其在概念上与参与业务过程的业务实体基本一致，每个组件的名称都是一个业务名词，表示一个业务过程中的业务实体（如，客户、合同、耐用消费品贷款产品等等）。在本层中的各个业务组件内，一律使用POJO为业务逻辑处理时的数据结构。

持久层

持久层负责实现消费贷系统对数据库的访问与操作。在持久层中一律使用POJO为数据逻辑处理时的数据结构。

各层间关系

四个层次之间类的关系如下图所示：

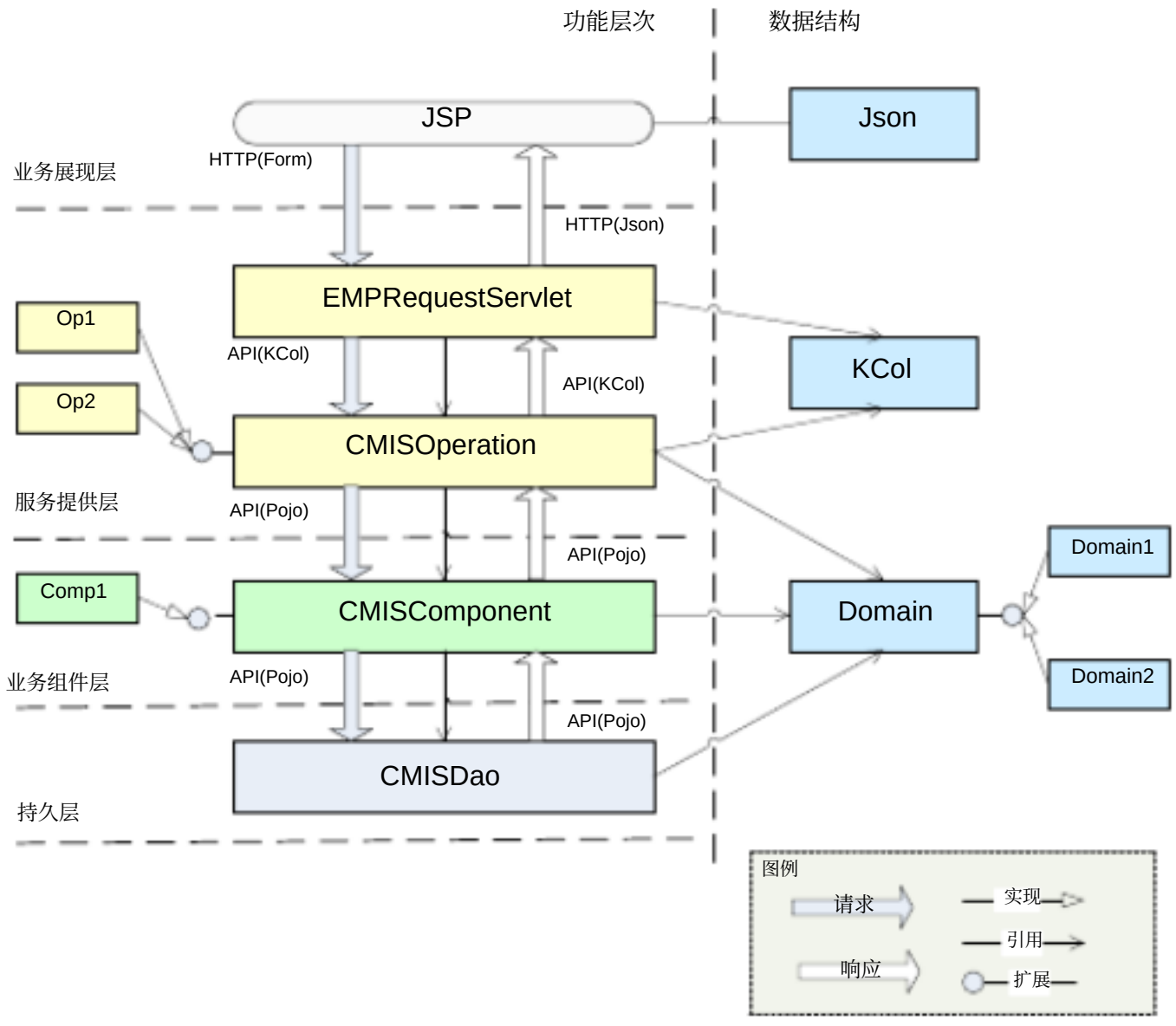

图 3-10

展示现层通过 HTTP请求方式调用后台服务层提供的服务，此时请求报文为标准的 HTTP 报文格式，且报文中的内容也是标准的 FORM 格式（示例：Key1=Val1&Key2=Val2&…），后台服务层返回至界面的数据内容则为直接为 JSON格式。后台服务通过 EMPRequestServlet 接收请求（此处省略了对 EMP 框架内部处理描述），接收后通过抽像类 CMISOperation 来调用对应的具体服务实现功能 OP；在 OP中将 KCOL通过特定方法转成 Domain 类（POJO格式），然后根据业务逻辑处理的需要来调用具体的组件类；在组件类中将不再使用 KCOL格式的数据来处理业务逻辑，在组件类中直接调用持久层 CMISDao 中公共类 SqlClient，完成系统内最基础的数据操作；在持久层中也将不再处理 KCOL格式的数据，而直接使用 Domain 格式数据来操作数据库。

3.3.4 关键技术点

3.3.4.1 数据库访问

在数据库访问层面，我们总结了在项目开发过程中使用基础框架遇到的问题，并考虑客户一些建议，对 EMP 中的数据访问层进行了设计重构。

首先，存在的问题我们归纳为下表：

序号	问题描述
1	不支持复杂的自定义 SQL的执行，二次开发人员只得另行编写数据访问层，最终导致系统数据访问接口不统一、混乱，不利于问题排查和日后维护。
2	不支持使用普通的业务实体对象进行数据传输： EMP 中的 TableModelDao 只支持 KeyedCollection 和 IndexedCollection 这样动态存储结构的数据传输，虽然灵活，但是数据

	存取不明确，一旦有问题，不利于排查。
3	不支持批量执行 SQL，导致项目开发人员在开发过程中编写了大量循环操作数据库的代码，对系统性能影响较大。
4	依赖配置文件实现 ORM 映射关系，这样导致系统配置文件过多，使系统文件的维护过于繁琐；另外，一个 ORM 配置文件对应一个表的僵化映射关系，对多表关联查询的 SQL支持乏力；另外通过配置文件进行映射必定要用到 JAVA 的反射机制，这也是性能消耗的地方。
5	开发人员在 JAVA代码中编写大量 SQL并根据前端传入参数拼写 WHERE子句的条件串，容易留下 SQL注入攻击漏洞，为排查和修复这些漏洞以及应付甲方的安全扫描需要花费不少工作量；另外，甲方希望 SQL和 JAVA代码严格分离。

基于以上问题，新的数据访问层提供了以下解决方案和 API 支持：

动态 ORM 映射方案

该方案是基于业务实体对象的一种动态 ORM 映射方案，该方案的关键在于业务实体对象的内部结构，我们在业务实体对象内置了一个 MAP 用来存储业务实体对象的属性值，而对外使用时，依然通过业务实体对象的 get 和 set 方法进行存取，这样既保证了上层调用时的明确性，同时内置的 MAP 结构为多表关联查询和动态映射提供了很好的支持，保留了 KeyedCollection 结构的灵活性。另外无需再考虑数据库表字段和业务实体对象属性名称的不一致性，这种映射关系动态灵活，方便开发人员维护，也避免了反射的使用。具体的业务实体对象的结构可参考下列代码：

User.java

不足的是以后的业务实体对象需要实现 CMISDomain 接口中的方法，不是纯粹的 POJO。

SQL的可配置方案

该方案强制要求把 SQL从 JAVA逻辑中剥离出来，类似 iBatis 在相应的 XML 文件中进行配置，然后程序根据配置的 ID 动态装载 SQL,并用“？”替换配置的 SQL串中的变量，最后交给 JDBC的 PreparedStatement 对象执行。具体的 SQL配置文件格式可参考如下文件：

demo.sql.xml

通过这种强制性方案，开发人员失去了编写带有 SQL注入攻击漏洞代码的机会，对这类问题进行了提前预防；同时 SQL的外置可配置，使业务实体对象在进行动态 ORM 映射时，不需要指明其对应的物理表名和主键字段名等信息，这样两个方案结合在一起，对复杂的多表关联查询和动态映射进行了很好的支持，同时也支持了在系统逻辑层中通过业务实体对象进行数据传输这一需求。

此外，对于 SQL配置过程遇到的一些特殊问题，我们也提供了较好的解决办法：一是动态 SQL的问题，一些 SQL条件是根据运行时期的值动态决定是否要拼入整体 SQL中，对于该问题的解决办法是，把相应的 SQL条件逐一配置在配置文件中，并用相应的条件 ID 标识区分，然后由开发人员在程序中根据客户端传入参数值决定要把哪些条件 ID 传到数据库访问层，数据访问层的核心类根据这些条件 ID 动态去追加这 SQL条件；问题二对于 IN 子句的支持，开发人员可以在 IN 子句的括号中指定对应的变量名，并指明该变量的类型，其类型可以是数组、LIST集合对象以及逗号间隔的 String，然后数据库访问层核心类会拆分和

计算出 IN 值的个数，并用对应个数的 ”? ”替换配置的属性变量名。

最后，没有直接使用 iBatis 原因是 iBatis 的 SQL配置方式和配置语句相对复杂，且相应的辅助 JAVA类过于冗长繁琐，和数据库访问层整体解决方案的契合度不高。

支持 SQL的批量执行

前述两个方案结合在一起，解决了问题 3 以为的四个问题，对于问题 3 中提到 SQL批量执行， 新的数据库访问层 CMISDAO中的核心类 com.yucheng.cmis.pub.dao.SqlClient 提供了相应的 API 支持，主要是采用 JDBC的 PreparedStatement 对象的批处理功能来实现

另外，考虑到兼容性，对于之前的数据类型 KeyedCollection 和 IndexedCollection，新的 CMISDAO依然提供部分支持；对于之前的 TableModelDAO 的优点，新的 CMISDAO依然继承和发扬，例如对单表的面向对象方式的存取访问接口依然提供， 只是内部实现方式有所改变。

CMISDAO本身在小数据量时（千条之内） ，对系统性能无明显影响，能将 Domain 至数据库之间的来回映射的性能消耗降至最低，另外，由于 JVM 堆容量的限制，建议在批量数据处理时要谨用。

3.3.4.2 数据缓存

CMISCache用于信贷系统运行时缓存常用的业务数据、 过程数据与配置数据， 以缓解系统在面对复杂业务需求时的对数据库的压力。缓存机制在信贷系统中已经比较常用，例如，数据字典、 组织机构信息等。 实际上缓存还可以广泛用于对数据一致性要求不高， 但对数据库性能消耗高， 或用 SQL难以实现的复杂的业务场景， 例如即时消息提醒、 数据即时统计、数据呈现的修饰与深加工等等。

CMISCache现阶段主要解决对业务数据缓存使用的规范统一问题， 集群成员之间缓存同步问题， 并提升缓存检索性能， 加强使用安全性。 将来考虑实现分布式 +分级式的缓存机制，平衡时间与空间之间的矛盾，进一步让缓存能提升系统性能，并考虑使用类似 XPATH语法模式对缓存检索，让缓存机制能替代部分 SQL实现功能，能为复杂业务的实现提供更多支持。

3.3.4.3 数据权限控制

数据记录级权限为菜单级权限的补充， 用于更细粒度地控制用户对每条记录各类操作权限，不一样的是在菜单级权限中的动作完全由用户自行定义， 而对于记录级权限则仅分为三类动作 ——查询、修改、删除，其分别对应查询权限、修改权限、删除权限。

查询权限 ，用于控制数据记录对当前操作者对的可见性，其作用于 展示列表之时 。当某数据无权被当前用户看见时， 在该用户访问其所在列表时， 系统将自动过滤掉无权限的数据。注：系统现只在访问列表时进行查询权限，对于单笔数据的查看权限， 则仅仅使用界面菜单权限进行控制。

修改权限 ，用于控制当前操作者对数据记录的修改权，其作用于用户 进入‘ 修改 ’界面之时 ，以及在 发起修改请求之时 。当某条数据记录无权被当前用户修改时， 系统将在进入

修改界面、发起修改请求两个时刻进行拦截。

删除权限 ，用于控制当前操作者对数据记录的删除权，其作用于用户 发起删除之时 。当某条数据记录无权被当前用户删除时，系统将在发起删除请求时进行拦截。

数据记录级权限所 保护的资源 范围是在表模型中所配置的表， 其控制的依据 取决于表中的权限归属描述字段（用于描述被约束的对象）中的值与当前登录者相关信息（如工号、机构码等等） 的匹配程度， 约束的对象 是信贷系统中的用户。 在信贷系统中有三类常见的约束对象 ——本人、本机构、本机构及下级机构，其与记录级权限一共有七种，如下图所示：

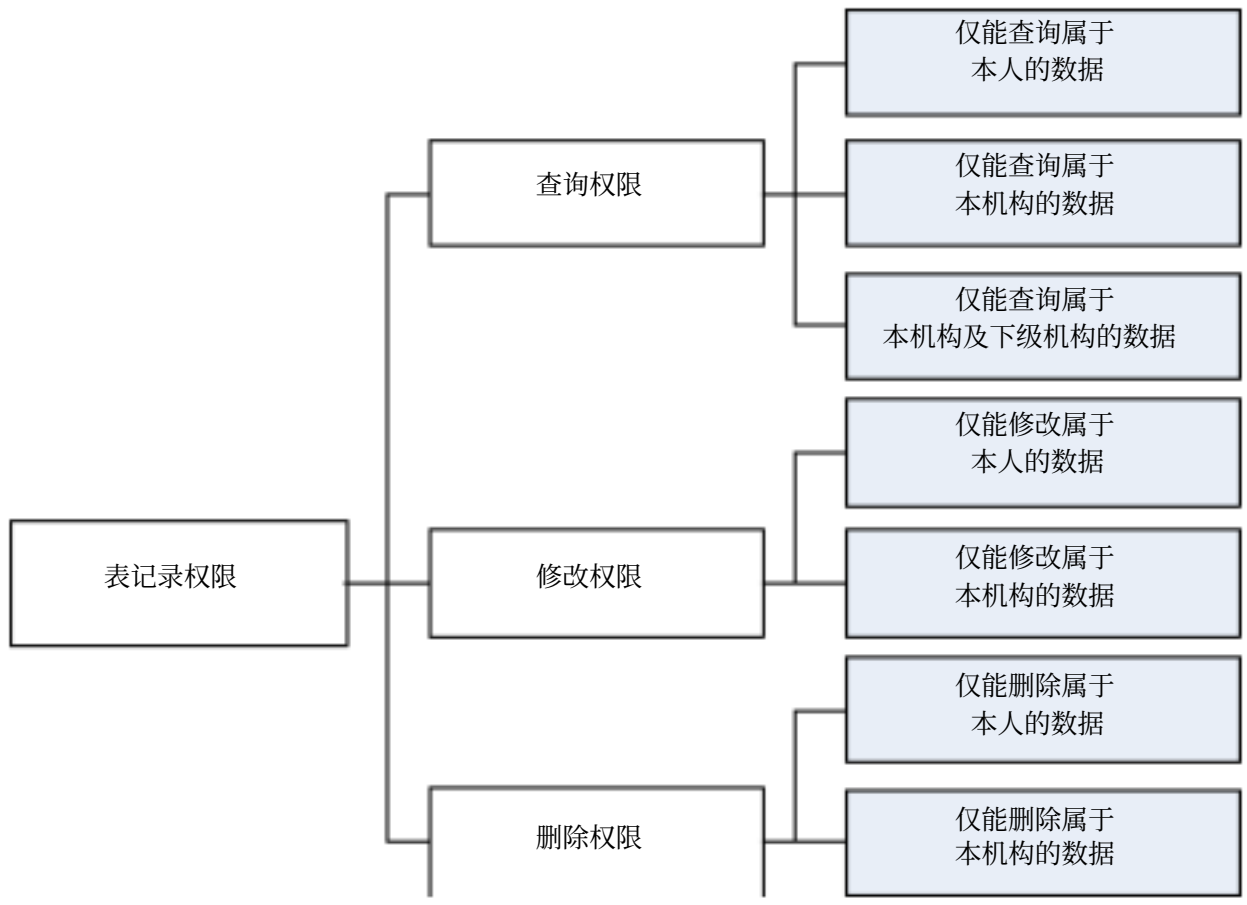

图 3-12

注：在流程审批过程中的对数据记录的权限控制移交给流程引擎负责 ,操作查看。

在上图中描述了记录级权限控制的层次划分， 及针对每一张表有三类权限 （查询、 修改与删除），针对每一类权限又按约束对象细分为若干个具体的 权限控制点 。

由于在信贷系统中， 被约束对象基本上是确定的， 而用于描述这些对象的权限归属字段的字段名基本一至， 因此可以考虑针将常用权限控制点制作成 标准记录级权限控制模板 ，用于覆盖绝大部份的权限需求， 对于少数特殊的需求， 可以参照现有的权限控制模板进行扩展即可。

现对每个权限类型下的最有可能出现的约束对象提供标准记录级权限控制模板， 一共提供以下七个：

（1） 由查询本人记录权限模板 ，控制仅能查询属于本人的数据

（2） 由查询本机构记录权限模板 ，控制仅能查询属于本机构的数据

（3） 由查询本机构及下级机构记录权限模板 ，控制仅能查询属于本机构及下级机构的数据

（4） 由修改本人记录权限模板 ，控制仅能修改属于本人的数据

（5） 由修改本机构记录权限模板 ，控制仅能修改属于本机构的数据

（6） 由删除本人记录权限模板 ，控制仅能删除属于本人的数据

（7） 由删除本机构记录权限模板 ，控制仅能删除属于本机构的数据

为保证模板是可替换的，　应该按 表模型 +操作类型 +岗位 来配置具体哪张表在作什么类型操作时应该如何控制当前用户的权限。模板使用的大体步骤如下：

在执行某操作之前，　根据当前操作的表模型、　操作类型（是属于查询、　修改还是删除），以及当前用户的岗位调用对应的模板　（注：岗位在配置中为可选项，　如果没有配模板对应岗位，则该模板适用于所有人）　，用以检查当前的操作者是否有权限操作当前数据，对于权限检查不通过的，则直接返回无权操作的提示界面，截断用户的操作请求。

3.3.4.4 业务导航

为提高用户在系统中的工作效率，　首先需要有良好的　UI 设计，通过良好的　UI 设计可以实现消息驱动　+流程驱动的操作模式。可以考虑将整个业务办理过程，按实际业务拆解成若干个步骤，　每个步骤是功能明确，　并且相对其他步骤独立的，　而让客户每笔发生的业务都按步骤一步步执行，　从而将整个业务办理步骤，　行成业务操作向导，　通过业务向导指示实现在操作层面，　将业务功能组织更条理且简单明确；　消费贷系统中，　将通过业务导航与流程引擎的配合，来完成业务操作，示意图如下所示：

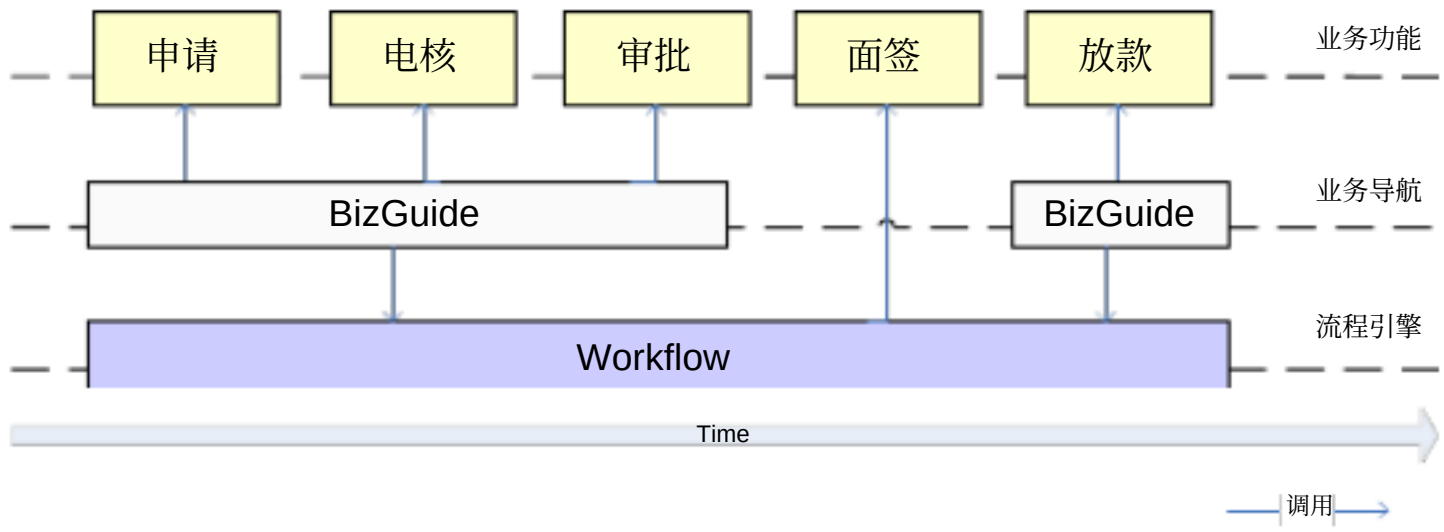

图 3-11

在图 3-11 中，通过导航将业务办理过程中的复杂业务操作划成若干个单简的场景，并用导航串联一起，　使其业务操作过程更明确与规范，　从而达到提升效率的目地；　通过流程引擎将整个业务各个环节联接起来，实现业务流程的流转。

3.3.4.5 JQuery 与 EasyUI

jQuery EasyUI 是一组基于　jQuery 的 UI 插件集合，而　jQuery EasyUI的目标就是帮助　web 开发者更轻松的打造出功能丰富并且美观的　UI 界面。开发者不需要编写复杂的　javascript，也不需要对　css 样式有深入的了解，开发者需要了解的只有一些简单的　html 标签。 jQuery EasyUI为我们提供了大多数　UI 控件的使用，　如：accordion，combobox，menu，dialog，tabs，validatebox，　datagrid，　window，　tree 等等。

jQuery EasyUI 是基于　JQuery 的一个前台　ui 界面的插件，　功能相对没　extjs 强大，但页面也是相当好看的。一些功能也足够开发者使用，相对于　extjs 更轻量。 jQuery EasyUI 有以下特点：

1、基于　jquery 用户界面插件的集合

2、为一些当前用于交互的　　js 应用提供必要的功能

3、使用　EasyUI 你不需要写很多的　javascript 代码，通常只需要写　　HTML 标记来定义用户界面即可

4、支持　HTML5

5、开发产品时可节省时间和资源

6、简单，但很强大

（注：本小节中对 jquery 与 easyui 的描述引用至‘百度百科’）

3.3.4.6 流程引擎

消费贷系统整个业务流转基于公司　　echain 实现。

3.3.4.7 规则引擎

消费贷系统整个业务过程的规则基于公司　　shuffle 实现。

4 功能设计

4.1 功能模块划分

基于以上从业务与技术两方面对消费贷系统的分析，　　可以初步明确系统功能模块如何划分，通过对模块的划分明确系统中各个功能职责的组织方式，如下图所示：

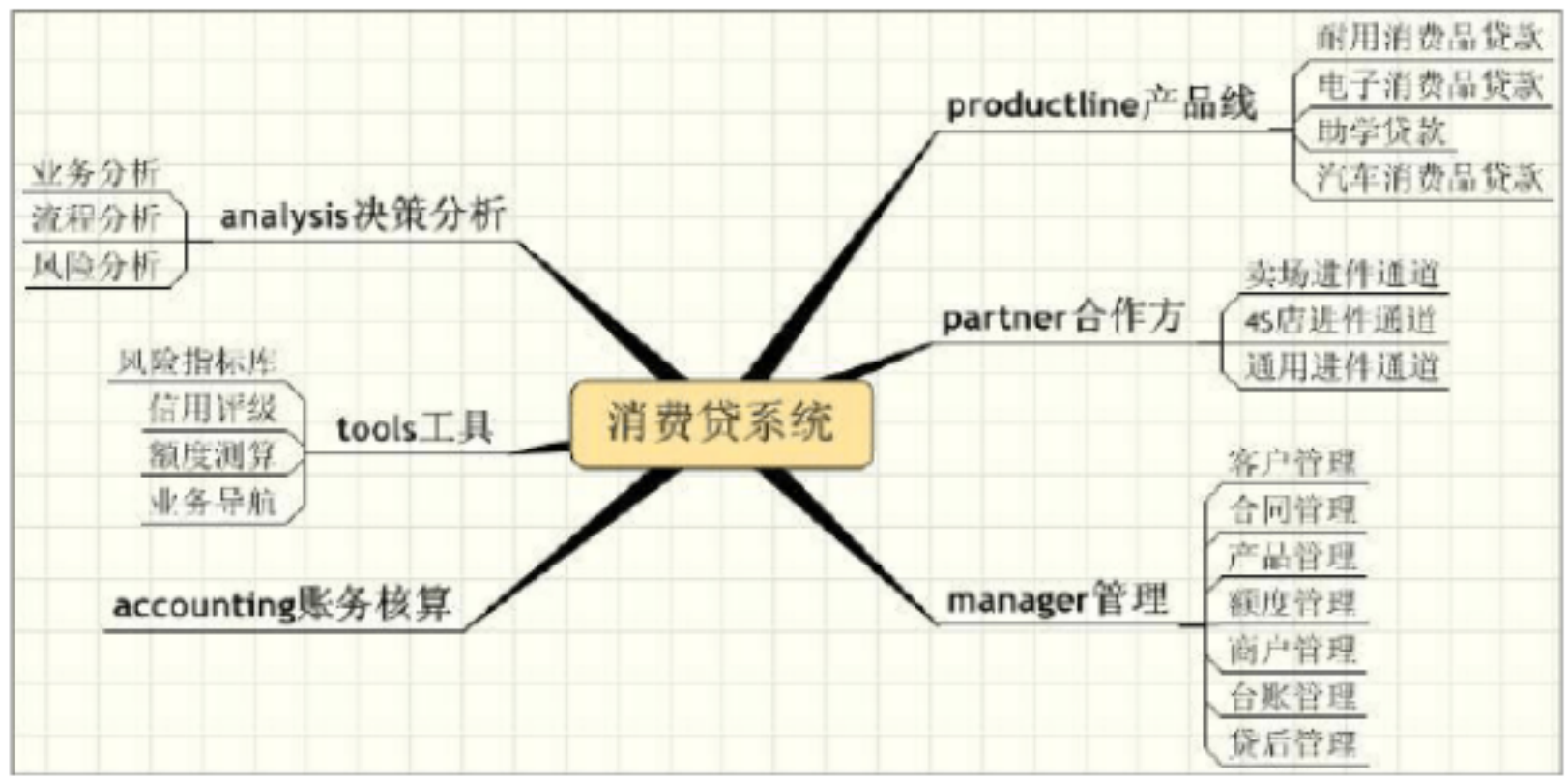

图 4-1

其中业务功能的一级包结构与逻辑结构一一对应，　　一共分为产品线、管理、　合作方、决策分

析、工具、账务核算六个包（　　注：这里不包含技术实现部分　）。各个包之间的依赖关系图如下：

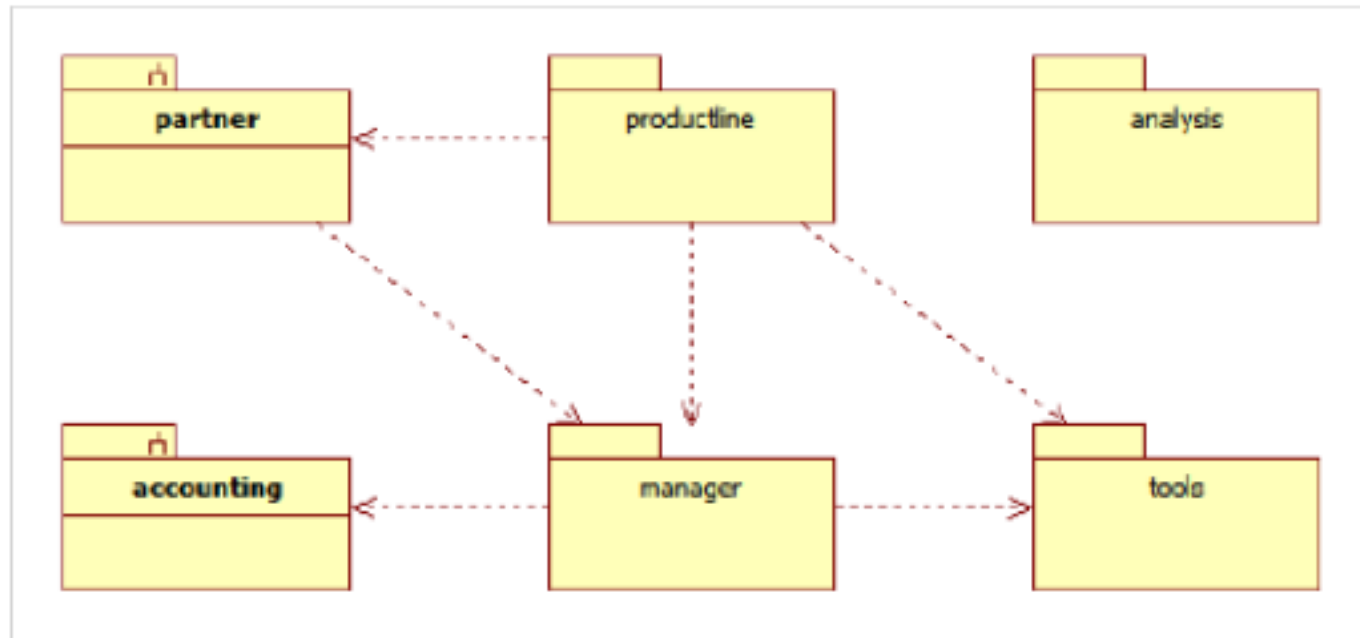

图 4-2

在产品线包（ productline ）中，用于放置业务操作与呈现相关功能，以产品别划分二级包，例如对耐用消费品贷款、电子消费品贷款等等。其中每个二级包内，以独立产品为单位形成组件类，　即一个产品一个类，　如果有两个十分相似的产品，　　则以继承的方式实现第二个类。产品线包为顶级包不为其它包提供支撑。

在管理包 （manager）中，用于放置业务管理相关功能，　　以业务实体二级包，　例如客户、合同、产品等等。 其中每个二级包内，　依据功能职责划分若干个组件类。　　管理包为产品线包、合作方包提供支撑。

在合作方包（　partner ）中，用于放置消费贷业务涉及的第三方进件渠道相关功能，例如，卖场经销商、　4S 店等等。其为产品线包提供支撑。合作方包将以子系统的形态与主应用工程分离部署，在主应用工程中仅保留其接口。

在决策分析包（　analysis）中，用于放置业务运作数据的统计分析相关功能。决策包为顶级包不为其它包提供支撑，同时其本身也不依赖与其它业务功能包。

在工具包 （tools）中，用于放置业务专业工具相关功能，　　例如，信用评级、 风险指标库、额度测算等等。其为管理包、产品线包提供支撑，其本身不依赖与其它业务功能包。

在账务核算包（　accounting）中，用于放置贷款核算相关功能，例如，贷款发放、还款扣款。 其为管理包提供支撑，　其本身不依赖与其它业务功能包。　　账务核算包将以子系统的形态与主应用工程分离部署，在主应用工程中仅保留其接口。

4.2 功能结构设计

通过对消费贷系统主要功能模块的划分，基本明确了系统中最粗粒度的功能组织单位，在本节中将采用类图的方式，进一步细化各功能之间的结构关系，如下图所示：

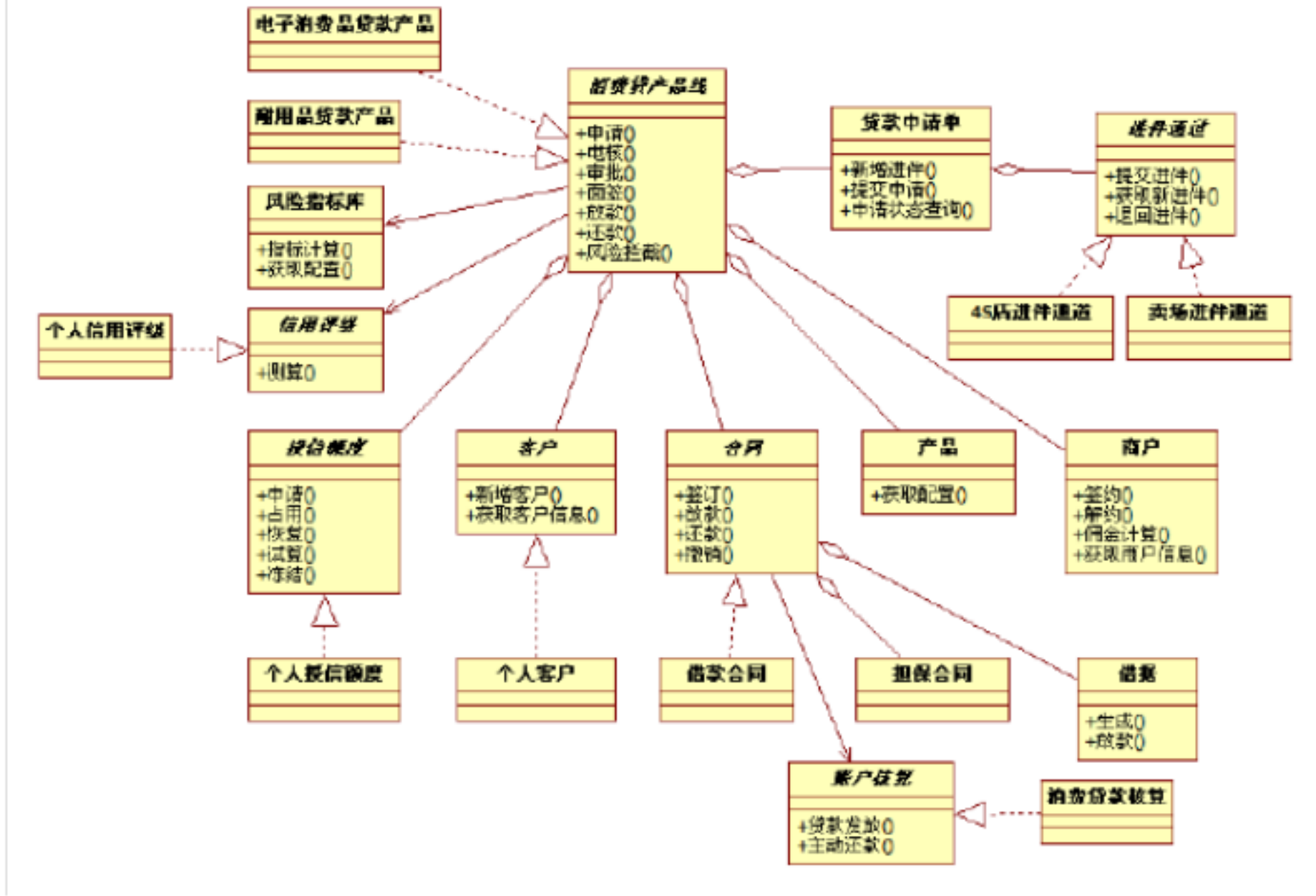

图 4-3

在图 4-3 中， 描绘了消费贷系统中大多数业务类以及各个类之间的关系。 从图中可以很容易地发现，处于整个系统中心的是 ‘ 消费贷产品线 ’ 这个抽象类。其负责实现业务申请至还款整个业务流程中所有与业务操作相关的功能， 如申请、电核、 审批等等。 围绕它的类的关系可以分三种：实现关系（如，耐用消费品贷款、电子消费品贷款等） 、聚合关系（如，申请、客户、合同等） 、引用关系（如，信用评级、风险指标库等） 。

实现关系：耐用消费品贷款类、 电子消费品贷款类均是消费贷产品线抽象类的具体实现，用于对消费贷产品的基本功能进行扩展。

聚合关系：申请、客户、合同、产品类均是消费贷款产品的构成关键要素，是其中的一部分，因此它们之间是聚合关系。

引用关系：信用评级、风险指标库类均属于在消费贷款产品运过程中需要使用的工具类，因此它们之间是引用关系。

消费贷产品线类负责所有业务操作层面功能的实现， 之所以要采用要采产品线作为中心来组织功能，这是因为不同的产品会有不同的运作方式（如， 有的产品需要走授信，有的不需要；有的进件渠道是来自合作方，有的来自机构内；有的需要进行电核，有的不需要；有的产品是自动人工放款， 有的是自动放款； ），同时，对于金融机构而言，产品是其组织开展业务的基本单元；对于客户而言，产品也是享受金融服务的基本单元；对于金融行业而言，产品更是彼此间竞争的基本单元。 因此， 以产品为中心来组织业务功能， 在概念上实现了业务与技术的统一。

在技术实现层面上， 通过产品线类来统一封装业务操作层的功能， 屏蔽后台的具体的技术实现细节与实现方式， 同时也屏蔽来自管理层的具体业务规则， 让其专注完成产品基本的公共的业务操作功能，如接收进件、发起申请、提交电核、合同签订等等。对于有新的消费

类贷款产品，可以通过直接新实现该抽象类来完成新功能，也可以继承已有的产品类来实现新功能（当产品间差异较小时）。产品线类相当于一个模板（或者说是一个业务操作框架），当有新的功能，复制一份模板，将调整其中有变化的地方即可。这样，即可以实现新功能，又不会影响已有功能，对于系统业务功能的积累十分有利。

在此之前的信贷系统的设计中，只有申请、合同、客户、额度等单个的业务组件，由于没有产品线这个概念，其结果是，申请有一堆不同版本的功能、合同有一堆不同版本功能，客户也一样…，随着时间的推移，现在没有谁能说明白哪些功能是应该配套在一起使用的，哪些功能是为什么业务实现的。现在有了产品线类，通过产品这个维度来统一为业务组件功能进行编目，就可以有效解决一直以来信贷业务组件功能组织混乱的问题。

从业务需求与技术实现两方面来看，以产品线为中心来组织、设计业务功能，可以较容易地满足各个层面对系统的需求，也能较好地应对将来的变化。

5 非功能设计

5.1 性能设计

由于消费贷系统对时效性要求较高，在系统设计时需考虑提供相应机制，保障在业务高峰期的系统性能，尽量保证不会出现因为系统响应速度原因而影响业务效率。系统提供以下五种机制来保障性能：

（1） 数据缓存机制

消费贷系统中基于 CMISCache实现对业务数据的缓存，其用于系统运行时缓存常用的业务数据、过程数据与配置数据，以缓解系统在面对复杂业务需求时的对数据库的压力。其还可以广泛用于对数据一致性要求不高，但对数据库性能消耗高，或用 SQL难以实现的复杂的业务场景，例如即时消息提醒、数据即时统计、数据呈现的修饰与深加工等等。

（2） 请求量控制机制

消费贷系统中，可以对于执行时间较长的交易，基于 EMP的访问控制器 accessManager，独立实行请求量控制。通过请求量控制，可以确保在同一时间内不会对同一个交易出现过多请求，以避免因个别复杂交易或问题交易影响整个系统的正常运行。

（3） 内存控制机制

消费贷系统中，将自动控制集合类中存放元素的个数（控制在 30000 个之内），以规避由于数据量过大或程序 BUG 导致的 JVM 内存溢出问题。

（4） 数据压缩机制

消费贷系统中，将自动压缩下传至客户端的返回数据，包括返回的 HTML、JSON、CSS、JS，其能有效节约 80%的网络流量。

（5） 合理拓扑结构

消费贷系统中，将分析查询、定时任务等耗时效的功能，从主应用中分离出来独立部署，以尽量避免后台操作影响日间交易运行（参见 3.3.1 子系统划分 ）。

5.2 安全设计

消费贷系统，由于是一个含账务核算的业务操作系统，且有部分功能面向互联网，所以系统安全必须有保障。系统除了提供在硬件环境内的安全保障（如防火墙、软硬加密等），在应用软件内部也提供以下五种机制来保障安全：

（1） 内外网隔离

消费贷系统中，将对外网的合作方使用的功能，从主应用中分离出来独立部署，两子系统间其于报文交易，以有效屏蔽从外网直接访问内网服务器的可能性，从而有效提升系统整体的安全性（参见 3.3.1 子系统划分 ）。

（2） 防跨站点脚本攻击机制

消费贷系统中，在后台通过特定的过滤器，统一过滤有嫌疑的字符（如 < {=:等），在前台针对所有 Ajax 异步请求结果，也作统一的过滤处理，从而有效避免前台页面恶意代码注入的发生。

（3） 防 SQL注入机制

消费贷系统中，通过在持久层统一对 SQL 的提出处理，将 SQL 与代码进行了分离，可有效屏蔽 SQL注入的可能性。

（4） 防止会话盗用机制

消费贷系统中，通过将后台业务会话 ID 与浏览器 COOKIES或 IP 进行绑定，有效规避了通过网络嗅探器盗用会话 ID 的风险。

（5） 防止越权访问机制

消费贷系统中，通过记录级权限机制，严格控制每个登录用户的数据访问、操作权限，有效屏蔽越权访问的风险（参见 3.3.4.4 数据权限控制 ）。

5.3 容错设计

由于消费贷系统对时效性要求较高，在系统设计时需考虑提供相应机制，能保障整个系统有较高的可用性。系统提供以下四种机制来实现容错：

（1） 双机互备机制

消费贷系统中在应用服务器层面，采用双机（或多机）集群方式，实现应用系统间的互备，在数据库层面，采用双机 HA 方式，实现数据库的互备。通过互备机制，能在一方出现严重故障时（如宕机、掉电、断网等） ，另外的服务器能继续对外提供服务。

（2） 交易存储 -重发机制

消费贷系统中，对于关键性交易（如贷款发放交易） ，一律再取先存储交易场景，再发送交易的机制，当交易失败时，可以视失败原因（例如，通讯失败）择机重发，或等待日终对账处理。

（3） 防重复提交机制

消费贷系统中，对于一些较复杂的操作场景（如贷款申请） ，可能会出现由于操作失误所导致请求重复提交，此时，系统在页面层面提供遮罩机制，避免对单一功能的重复提交；在后台，通过 EMP 的访问控制器 accessManager，有选择地控制短时间内重复报文的发送。

（4） 操作预防机制

消费贷系统中，通过对用户操作时严格控制（如，控制数据长度、精度、范围、以及业务处理规则等） ，配合友好的提示，尽量将系统的错误处理前移，实现整个系统的预防性容错。