TURING 图灵原创

学习微软资深MVP
如何颠覆程序设计思维

基于大型互联网系统架构
解析.NET高级设计艺术

.NET 框架设计

模式、配置、工具

王清培◎著

人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵原创

.NET框架设计

模式、配置、工具

王清培◎著

人民邮电出版社

北京

图书在版编目（CIP）数据

.NET框架设计 ：模式、配置、工具 / 王清培著. --
北京 ：人民邮电出版社，2015.1
（图灵原创）
ISBN 978-7-115-38028-9

Ⅰ. ①N… Ⅱ. ①王… Ⅲ. ①计算机网络－程序设计
Ⅳ. ①TP393.09

中国版本图书馆CIP数据核字(2014)第298586号

内 容 提 要

本书总结了框架设计的整体思路和经验，包含了常见应用框架设计的模式、框架灵活性的配置和框架工具的支持，有助于读者了解框架设计的核心思想，加深对框架设计的理解，快速掌握框架设计的技巧，并在研究其他框架时能够做到举一反三。

本书适用于应用层开发者、框架学习者和对框架设计感兴趣的读者。

◆ 著　　王清培
责任编辑　王军花
执行编辑　张　霞
责任印制　杨林杰

◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路11号
邮编　100164　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京天宇星印刷厂印刷

◆ 开本：800×1000　1/16
印张：13.5
字数：319千字　　2015年1月第1版
印数：1－3 000册　　2015年1月北京第1次印刷

定价：49.00元

读者服务热线：(010)51095186转600　印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京崇工商广字第0021号

业界推荐

“这本书最大的价值就在于告诉你如何在实战中运用平时学到的知识，如何站在不同的角度分析和解决问题。与市面上其他的图书不同，这本书中的内容都是清培在工作中遇到实际问题后分析得出的经验。对我而言，里面的各种设计都具有独到的见解，往往能将复杂的问题简化成优雅的模式。我最喜欢的书中的一句话是：“毫不夸张地讲，这一节的内容将直接决定什么叫程序设计，什么叫代码可以改变世界，没有做不到，只有想不到。”

——刘星亮

携程旅行网攻略社区事业部技术经理、架构师

“这本书字里行间都充满了高大上的气息，但又有一种低奢内的大家风范。实在是一本在架构设计方面实用的经验之书、智慧之书。”

——袁永福

微软MVP、南京都昌信息科技有限公司创办人

“本书融入了作者丰富的实践经验，既有原理性的说明，又有指导性的介绍；内容由浅入深，搭配精心设计的例子，更易于读者理解和掌握，且实用性极强；不但适合作为高校教材使用，而且也适合程序开发爱好者自学提升，在.NET开发技术图书当中是一本难得的好书。”

——冯瑞涛

《Windows 8 应用开发权威指南》作者

微软MVP、微创软件公司开发经理兼架构师

“本书通过平易近人的语言和大量实例代码，基于作者多年的框架编程经验，对.NET框架模式进行了细致深入的分析，是.NET框架程序员不可多得的教材之一，对于想要写出高效、稳定、健壮代码的程序员而言，本书也将大有裨益。”

——Wind Sang

新蛋信息技术（中国）有限公司Team Leader

“很早之前就在博客园看过作者写的博文，他的博文不管是在条理上还是在内容上，都可以说是无可挑剔的，当时就非常希望作者的文章可以整理成书，现在终于看到本书出版了，心中感到无比喜悦。本书包含了作者多年来对框架设计的思考，读者可以通过本书对.NET的框架设计产生新的理解。”

——李志

微软MVP

前　言

其实这本书能面市十分不易，在写作过程中我经历了很多次犹豫和矛盾，我自己所理解和认为的框架模式是通往一个框架本质的捷径，难道大家也是同样认为的吗？在没有得到大家广泛认可之前，我真的不敢保证这些模式都是客观的、有意义的，并且是可以被传递交流的，所以我尝试着在博客上发表了一篇，看看大家对框架设计背后的模式是否真的感兴趣。结果出乎我的预料，大家都非常有兴趣，都支持我多去写点这类文章，我也被博友们的鼓励所感动，连续写了几篇这方面的文章。事实证明我的犹豫是没有必要的，所以我决定将自己这些年来积累的对框架学习的经验分享给大家，但是通过博文的方式进行整理并不能随着时间的推移很好地沉淀下去，可能过不了多久就会石沉大海，而且写成博文或是随笔并不能很好地定义一些关键概念，也不能规范某些专业用词，容易导致理解上的错误。所以只有整理成书，才能写得详细并且便于查询学习，最重要的是我能对自己积累的这些东西有一个完全可靠的肯定，从而会觉得这一切都是有价值的。

每当我们准备买一本技术书时，都会在第一时间非常投入地阅读一下目录，看是否是自己喜欢的，我敢肯定当你仔细观察本书的目录时，你会惊奇地发现这些看似熟悉但是又陌生的概念，都是在以往.NET图书当中无法找到的。有这种感觉一点都不奇怪，因为书中的这些设计技巧或者说编程经验，都是我点点滴滴积累而得的，每当我看到、听到、领悟到一些设计思想，我都会及时在本子上记录下来，然后再详细地分析实现。虽然这些可能是不常见的，但是却是很重要的。这些经验真的很难得，无数次的犯错、吃亏，最终才能换来一个具有实际意义的经验，而本书正是将大量的设计经验毫不保留地奉献给你。

说到框架设计，有好多朋友都不太能够准确地理解，可能认为一个Project就是一个未来的框架，整理点Function、Help Class就认为是框架了。其实并不是这么简单的，框架本意是肩负着架构的直接着陆，框架存在的意义不是为了帮助我们快速开发，而是能够在架构设计之后将这张蓝图通过各种不同的框架，让架构能从一个理想化的抽象概念变成一个能够真正运转起来的核心代码块，而这些核心代码块是模式在背后支撑着，这也是本书的重点。我们并不需要给框架设计下一个准确的定义，因为它的形态、作用、模式都会很多，如果非要根据这些给框架下定义，显然是很不明智的。我们只要能够明白框架设计其实是非常复杂的过程，只有经过这个复杂过程的锤炼之后才能算是合格的框架，通过这个复杂的过程来来验证你的设计是否是合理的。通过这种方式来理解框架设计会比下一个不负责的定义来理解准确得多。

框架设计不仅仅是代码模式（传统设计模式），不是只有学会了设计模式就能开发一个很好的框架，通过配置文件可以很方便地避开强编码，动静结合可以将开发时、编译时、运行时3个不同的阶段产生的代码最终合并在一起动态运行，这些高级应用跟你能熟练使用设计模式关系不是很大。

那么框架与组件又有什么区别呢？其实框架与组件的区别在于框架应该是基础组件，组件化拆分所有的业务模块，分隔关注点。比如，你的框架在引用第三方库的时候要考虑到你的客户是否愿意接受你的“私人恩怨”，所以需要抽象出对外部的依赖，尽量做到给你的框架使用者选择的权利，如果你的框架将一堆依赖的第三方库都强耦合地引用进来，就会导致你的框架使用者在使用的时候也被迫地使用这些第三方库，所以我们经常会觉得某个框架很臃肿，很重。

那么框架与Library、API的区别又是什么呢？其实区别还是比较明显的，也很好理解。Library是一个相对比较小的API的封装，API可以理解成目录式的方法，就好比Win32 API那样，而框架显得比较强大，能解决某一类复杂的问题集，这是由于它本身就肩负着重要的任务，所以框架的示例代码很重要，需要用很简短的方式来表达框架的作用，方便入门，让框架使用者有兴趣了解下去。当我们提供给别人某个功能的时候，要考虑好到底以什么样的形式提供，而这将直接决定你后期的维护、升级以及复杂的向下兼容等一系列成本。

本书将以大型互联网系统架构为背景，基于当下最流行的SOA架构，针对最切合当前的互联网业务来展开框架设计，保证框架设计的众多实践都具有科学依据。

最后可以肯定的是，能够买本书的读者对代码是有追求的，视为艺术品，远远超于UI界面。让我们在本书中一起来领略框架设计背后的真正精髓，观赏框架设计中的另一番美景。

本书结构

本书共分为6章。

第1章首先介绍了框架设计的一些基本概念，然后介绍了企业应用框架的大致模型，让读者能够在应用框架上形成正确的认识，并对本书所讲的框架设计三元素（框架模式、框架配置和框架工具）有一个简单的了解。有了这三个核心概念的铺垫，读者就会容易理解后面具体的讲解。

第2章讲述了重要的C#设计技巧和重要语法。首先回顾了C#的核心语法和重要的设计技巧。由于我们将使用C#来设计应用框架，所以有必要回顾一下该部分的核心内容，同时也对设计模式、单元测试、稳定性设计等相关概念进行了详细的讲解。因为在设计重要的代码时，需要很多设计思想加以辅助才能为将来循环迭代起到防护墙的作用。

第3章介绍了本书的核心元素之一——框架设计模式。这一章是本书的重头戏，通过循序渐进的方式给读者展现了10个流行的框架设计模式，这些模式都是.NET应用框架中的常客，熟练

掌握这些模式对我们日后的框架设计是非常重要的，同时对我们学习第三方框架来说也是非常有价值的。

第4章介绍了本书的核心元素之二——框架配置。这同样是本书的重点，介绍了框架设计中的核心配置，通过学习相关配置技术可以很好地改善我们设计框架的灵活性和扩展性。

第5章介绍了本书的核心元素之三——框架的工具。这一章讲解了框架设计中经常被忽视的工具部分，一个框架在使用方便性方面将直接取决于框架本身提供的相关工具，以及这些工具是否支持二次开发。当然具体工具的开发不是这一章的重点，重点是要让我们能够引起对框架工具的重视。

第6章对本书的核心框架设计三元素进行了简单的梳理和总结。

目标读者

作为.NET开发群体中的一员，我深知.NET开发人员在技术的发展中会遇到一些门槛，而这些门槛通常也是非.NET开发人员小瞧.NET开发人员的有力证据。别人之所以这么看确实是有原因的，因为确实存在这样的现状。

.NET程序员不应该总是浮于工具的表面，应该沉下去细心地研究你平时所使用的这些工具、框架的设计原理，以及自己是否能够开发出一些工具或者框架来，从而慢慢地走出拖拉工具的圈子。让自己真正成为一名软件工程师，而不是非要加上一个.NET在你的技术简介里。

本书是我多年来潜心研究应用框架的总结，不能说这本书是多么高深的书，但是本书是非常实惠的一本书，里面的部分章节是比较抽象的、模式化的，通过认真地学习本书将大大提高.NET开发人员的技术功底。毕竟.NET程序员首先要面对的就是快速地应用框架更新，我们要在很短的时间内掌握某个应用框架，从而能够在.NET市场上有一个好的发展机会。

本书可以说是帮助.NET程序员晋级的非常稀有的一本书，因为目前市面上的图书要么就是讲框架如何使用，要么就是讲某个框架的原理，而没有跳出框架来看框架的设计模式和架构思想，不能让我们通过学习这些东西举一反三。

本书将帮助你稳定地提升到.NET高级层次，书里面的很多实践经验是只有亲身经历过一些大型项目才会有的，所以通过本书你会吸收到一线大型项目的开发经验。

勘误

由于时间和水平都比较有限，书中难免会出现一些纰漏和错误。如果大家发现了一些问题，请在图灵社区本书页面（http://www.ituring.com.cn/book/1603） 提交勘误，也可以直接联系我。我会尽快在本书的后续版本中加以改正，万分感谢。

致谢

这本书能顺利地出版，首先我要感谢的就是陈冰老师，没有他耐心的指导，这本书是不可能出版的。他细心地指导我进行整本书的前期规划和后期格式的调整，这些细节让我明白他是多么负责的一个人，也让我看到了一个顶尖的编辑是何等的专业，对于我们初次写书的新人来说，能与陈冰老师合作是一个可遇不可求的机会。

写一本书所要花费的精力和时间是无法计算的，作为家庭中的一员，我占用了太多本该陪家人的时间，这里谢谢我的爱人邓爽，谢谢你帮我处理了很多生活上的琐事，让我一心一意地写书。还要感谢我的父母，没有你们前期辛苦地培养和教育，我也不可能有能力写这本书，祝你们永远健康快乐。

在写这本书的过程中，金源帮了我很大的忙。他作为第一个试读者，帮我检查出了很多细节问题，感谢你抽出自己私人的时间来帮我校对本书的内容。

能有今天的成果，永远离不开我的恩师魏强当年对我的辛苦栽培，千言万语感谢恩师，感谢您在那漫长的岁月里耐心无私地教育我，祝恩师全家健康幸福，您的恩情我永远铭记于心。

最后还需要感谢曾经帮助过我的各位朋友和领导，我相信人世间的所有事情都是有因果关系的，正所谓喝水不忘打井人，感谢武双、罗亮、庄金东、阮明、Peter、Wind、刘星亮、李锦国、袁勇福、冯瑞涛等，谢谢你们，祝愿你们身体健康，家庭幸福。

王清培

2014年6月28日 于上海

目　录

第 1 章
框架设计的基本概念

要知其然，更要知其所以然。

在进行框架设计学习之前，我们有必要对框架及框架设计相关的概念加以解释，从而方便我们对本书后面细节部分的理解，不至于产生很多疑问。

本章将通过一系列的解释说明来帮助读者快速地理解何为框架，何为框架设计。通过讲解一个简单的实例，让读者清晰地明白本书所归纳的框架设计核心三元素——模式、配置、工具到底在框架设计中扮演着什么样的角色，从而能为后续的学习做一个整体的铺垫。

1.1　框架

本节内容

- 框架的概念及作用；
- 框架的生命周期及复用。

本节将简单介绍在软件设计中举足轻重的元素——框架，包括什么是框架，如何理解框架，它所发挥的作用是什么，以及为什么需要框架来辅助系统设计及开发。

找一个能准确解释框架的定义很难，给框架直接下一个定义更难，一个很抽象的东西无法用概念来直接描述，只有从不同角度、不同层面来看软件设计，只能从侧面反映出这个不被作为核心功能性需求的元素（框架），它是如何肩负起系统的设计责任的，又是如何保证功能性需求安全着陆的，只有这样，每个人才能对框架有一个自己的定义。虽然每个人的定义可能各不相同，但是这样能保证大家对框架作用的理解是一致的。

上述内容都将在本节解释清楚，读完之后，你将会对框架有基本的认识。

1.1.1　框架的通常作用及层面

既然框架不是作为功能性需求而存在的，那么为什么在系统开发中还需要引入各种各样的框架呢？按理说这些框架应该不是软件开发的重点，为什么我们还需要加倍关心该使用何种框架呢？

首先，软件开发要满足用户的功能性需求，只有在满足了功能性需求的前提下，非功能性需求才算是有价值的。我们可以归纳出软件开发的两大需求：第一是功能性需求，第二是非功能性需求，前者优先级高。

懂软件开发的人都知道，一个功能性需求可以用很多种方式去实现，如果这个时候不考虑任何非功能性需求，只管完成目的，那么可以说不需要任何框架支撑，完全可以使用最为原始、最为简陋的方法去完成。

在满足了用户的功能性需求后，就需要去完成非功能性需求。非功能性需求形态各异：安全、性能、稳定性、易维护、易伸缩。在完成了功能性需求后，这些将变成你重点考虑的范围。

那么，当你已经使用最为原始的方法完成了功能性需求后，现在又提出一系列的非功能性需求，你该如何完成呢？

这个时候，框架就是那个默默无闻地肩负着功能性需求和非功能性需求安全落地的保障者。在开发期间，如果你正确使用了某个框架来解决某个特定的问题域，那么后续的非功能性需求完

成起来将会变得很平滑，不会因为一些非功能性需求使你的项目延期，或是变得很脆弱。框架的基本作用及层面如图1.1所示。

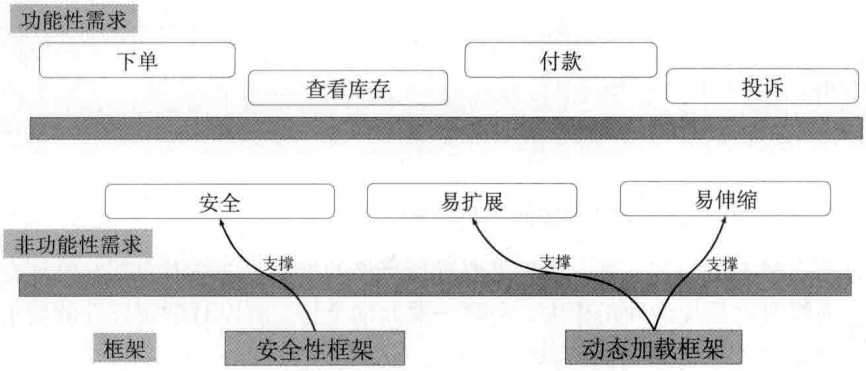

图1.1 框架的基本作用及层面

以上就是框架的基本作用及层面，其实框架的主要作用就是让我们更好地实现非功能性需求，因为非功能性需求直接影响着功能性需求的使用。良好的用户体验、良好的视觉效果，这些都是现代软件所必需的。

1.1.2 框架的生命周期

现代应用软件的需求已经发生了翻天覆地的变化，从功能性需求到非功能性需求都在飞快地发展和创新，那么框架所在的位置就必然要求它能够快速满足这些不断变化的上层需求。能够快速地支撑上层需求变化是考验框架的整体指标之一。

非功能性需求的一个特点就是不变性。这些需求可能在任何一个软件系统当中都会需要，比如，上述例举的几个非功能性需求在绝大部分系统中都是需要的。

但是功能性需求是各不相同的。每个软件系统都有着自己的业务需求，尽管大体相似，但是在业务细节的处理上还是会有很大的差别。

也就是说，非功能性需求是完全可以在多个系统之间复用的。既然非功能性需求可以复用，那么用来支撑的框架也就是可以复用的了。

框架在其生命周期中，为了满足各种需求，需要不断地维护。在多系统之间复用框架，必然会出现很多定制化需求，所以框架在其生命周期中经常要维护修改，只有这样框架才会一直保持着重要的价值。框架的生命周期过程和任何一个非功能性需求的生命周期过程在特征上基本一致，都会面临着各种修改。因此，我们要认识到：框架并不是固定不变的。

1.2　框架设计

本节内容

- 框架设计及一般设计过程；
- 框架设计对架构的要求。

框架固然是好东西，那么我们该如何得到它呢？

框架设计有着复杂的过程，每一步都需要进行严格的把关。这是因为框架设计在系统架构中举足轻重，它的修改对系统的冲击很大，会牵一发而动全身，所以对框架设计的要求要高出一般业务功能，只有这样才能保证框架的最终质量。

本节将详细介绍框架设计的一般过程，以及每个过程中的重点。

1.2.1　确定问题域和识别变化点

首先需要确定问题域，问题域也就是你将要用框架来解决的问题范围，有了范围之后我们才可以针对每个点进行详细的分析。

例如通信框架，首要是确定业务范围，它允许哪些类型的消息通过该框架，你在脑海中要有一个大致的范围，此时可以使用思维导图勾画出一个大概的业务模型来，问题域模型如图1.2所示。

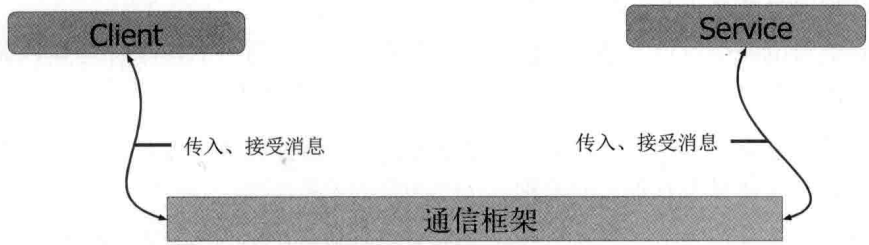

图1.2　框架的问题域模型

确定了大概的问题域之后，就可以进一步地分析它的变化点（如图1.3所示），这对框架的整体设计起着铺垫的作用。

变化点的整理可以让我们更细致地了解架构，这对后面的架构选择很重要，因为架构模型是由架构模式驱动出来的，正所谓模式驱动模型，模型驱动设计。模式是对某一类固定问题的求解方式和方法，是经验的总结。确定了问题域之后，我们就可以选择相应的模式来驱动出符合当前问题域的框架架构。

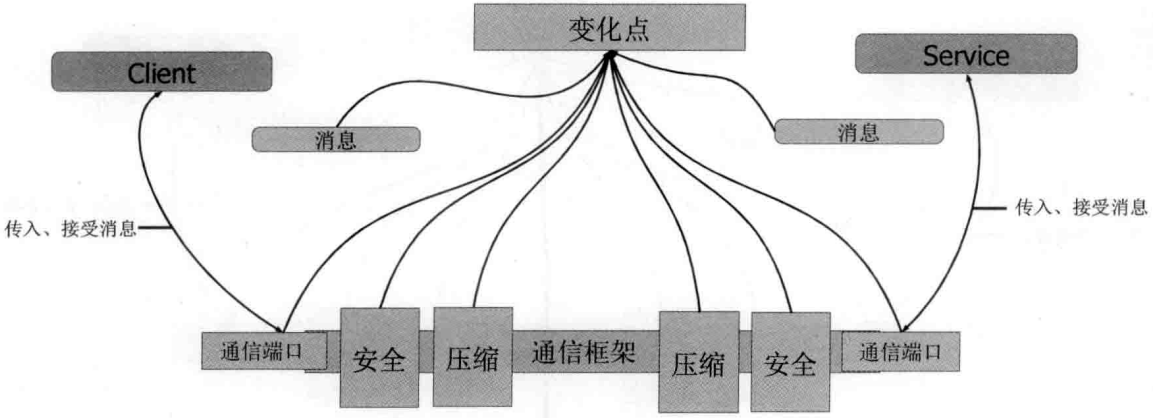

图1.3 框架的变化点模型

1.2.2 选择合适的架构模式、配置变化数据、可视化管理

选择架构模式的前提是你已经确定了问题域并且识别出了大部分的变化点，这样我们才能够准确地选择架构模式。通常一种模式对应一种问题的解决方法，不过在必要情况下需要组合多个模式来搭建一个符合当前问题域的模式出来。如果你识别出来的变化点够详细，就可以考虑是否需要组合多个架构模式来完成一个架构模式。

根据上面的问题域和变化点，可以大概得知将使用到如下几个模式。

- 通信框架：管道模式
- 消息：契约式设计
- 通信端口：异步消息+事件驱动
- 安全：链式编程
- 压缩：IOC注入第三方压缩算法

设计好架构之后，就需要将变化点配置起来，以便在需要的时候动态地切换变化点，如图1.4所示。

配置的方式基本上有两种，一种是在本地通过静态文件的方式，另一种是通过远程服务动态配置的方式。两种方式对应不同的应用场景，静态配置通常用在无需及时更新的时候，而远程动态配置是需要在运行时动态地调用随时会变化的配置，如促销的折扣率和促销价格，这些时刻都会变化的配置是不能够用静态文件的方式配置的，需要能够及时地修改和访问。

框架设计中另一个很重要的部分就是可视化。大部分框架都会忽略可视化部分的设计，其实是不正确的。一个完整的框架必须配备可视化工具来帮助我们方便地使用框架和管理框架（如图1.5所示）。

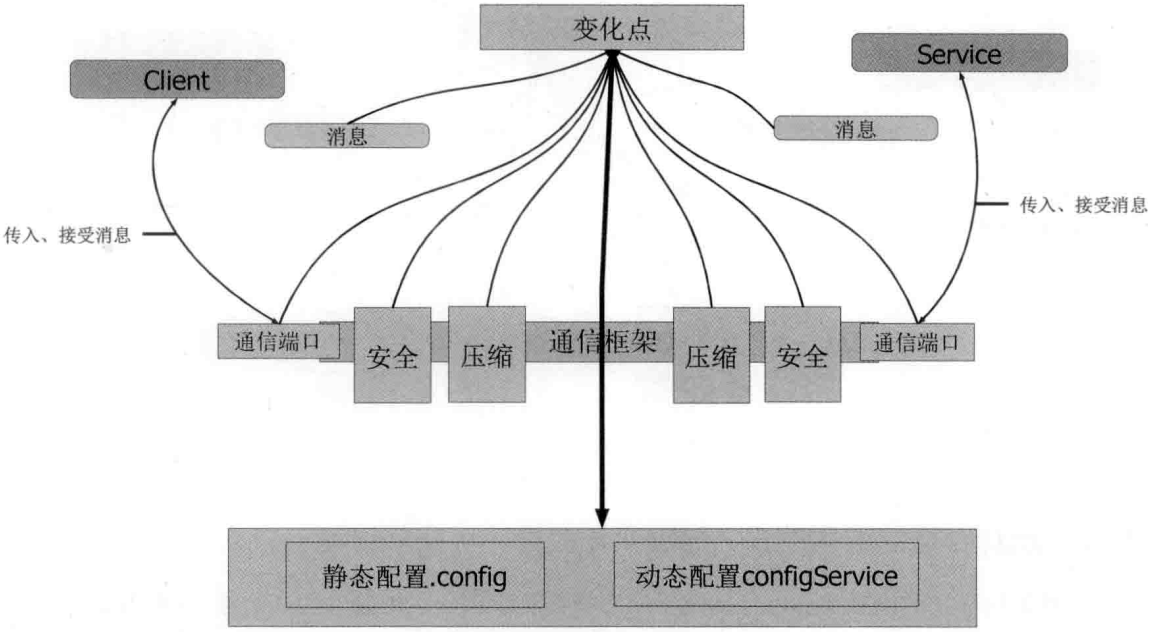

图1.4　框架的变化点模型

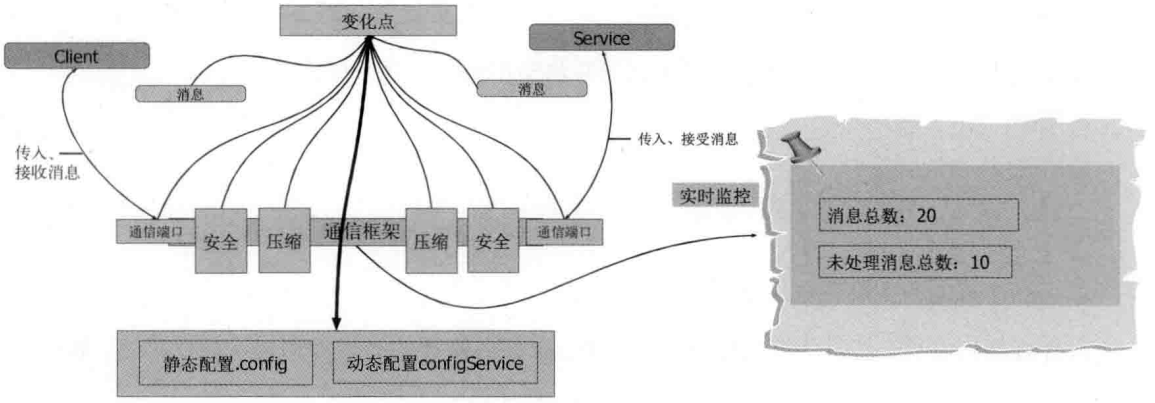

图1.5　可视化辅助管理、调试、监控

这里可能就需要一个可视化工具，从而跟踪、查看通信框架中的消息，便于我们在测试的时候查看消息的状态，修改消息的内容。

作为一个完善的框架，不仅需要在运行时很稳定，而且也要保证该框架是可测试的。若能做到在测试时框架内部是透明的，那么将大大提高框架的受欢迎度。

到目前为止，你应该对框架设计有了基本的理解，上面所讲的都是框架设计过程中的核心内容，当然还有一些辅助性的过程，都是为了尽可能地保证框架的设计按期完成，避免漏掉某些重要的点。