为什么我说测试策略才是测试的核心

引子：从测试行业的危机论开始

狄更斯在《双城记》一开始就写到“这是个最好的时代，也是个最坏的时代”。我觉得这句话正好说出了目前中国的软件测试的环境：

“这是个最好的时代”，是因为经过这么多年的发展，测试体系已日趋成熟，有丰富的测试资源，从入门到高级都有，测试者可以快速学习快速成长。

但“这也是个最坏的时代”。

如果我们留意各种测试大会，各种内部的外部的分享交流，就会发现这几年测试几乎没有新的思想或技术提出，缺少创新和突破。而其他领域，正在进行着翻天覆地的变化。“测试已死”不再只是提提而已，事实上，真的有公司已经不再招聘测试岗位。还有的公司把“测试”的title改为了“测试开发”，要求测试者转型。最近还有明确不要6年以上工作经验测试者的事情出来，因为担心测试者工作年限太久可塑性变低，而不能适应新的要求。在这样的背景下，测试人员的代码能力被拔高到几乎是核心能力的要求，这让很多测试者感到无所适从，要知道，就算测试者能够写一些脚本或者工具，但是术业有专攻，他们的代码能力和相同年限的开发还是不能比，更何况那些一直专注于手工测试的测试人员。测试的未来之路到底在哪里？

最近正好看到了栾江义在华为开发者社区发表的《测试的行业危机》，我觉得文中对当前测试行业的危机的原因分析得比较客观和深入，总结起来是两点：

1. 软件开源社区的快速发展使得业务开发变得便捷的同时提升了组件的质量。
2. 软件工程方法的演进，团队角色分工变得模糊，要求开发具备全流程全角色的处理能力，以达到更快更及时的响应的目的。

这些做法对测试最深的影响，就是使得测试的团队的独特价值，如质量守护、缺陷预防，变得不再独特。这些职责不再是测试的职责，而是所有角色都要承担和面对的职责。**当一个角色在团队中失去了独特价值，转型就是必须的，这是每个测试者需要接受和正面的现实。**

为什么我说测试策略才是测试的核心

测试者面临转型已是不争的事实。如何转型，或者说如何为下一步的转型做好准备，是我们接下来需要思考的问题。

通过前面的分析我们已经知道，测试者需要转型的原因是在新的开发形式、技术的影响下，传统测试团队的独特价值正在消失，但这绝不等于“测试”会消失，相反，测试会分散到产品开发活动的各个阶段，由不同的角色在各自层面来展开测试，例如产品经理可能要进行验收测试，开发可能要进行功能测试等，各种不同的角色都需要具备测试思想，都要懂测试方法。另一方面，开发方式的演进，会降低产品研发的门槛，更多的人都可以来做软件，这不仅会加剧市场竞争，还会提高用户对产品质量的期望。**质量 = 符合要求，而探索和评估质量最有效的方式就是测试，所以团队依然需要有可以端到端的去评估和控制质量的人存在，20年前如此，20年后还是如此。**只是在传统的方式下，**这个人是指导测试团队要测什么和怎么测**，让测试团队更好的工作起来，同时想办法去尽量让测试提前，去做缺陷的预防；而**在新的方式下，这个人是让不同的角色明白要测什么和怎么测**，可以更好地协作起来，高效的探索产品质量，确认产品是否达到我们的质量预期，当然缺陷预防的说法也不存在了，因为新方式下各个角色需要真正做到对自己的输出负责，这是新技术下组织变革带来的红利，本文不展开讨论。**而测什么和怎么测，正是测试策略。**不过这时测试策略的制定，不一定还是由测试这种角色来制定，可能是由研发经理或是新的角色来进行，但对我们现在测试来说，不正是一种新的发展的机会吗？

新的方式下对做产品测试策略的人需求可能并不大，很有可能是一个团队或者几个团队只需要一位具备这样技能的人员。但是只要我们分析新方式下的测试策略，就可以预判出哪些内容可能会是大多数测试者的转型的方向，例如：

- 对测试工具或平台的支撑。
- 承担那些对测试技能要求非常高的测试。
- 测试方法的研究和改进。

我们还可以推测出专职的测试人员的工作形式：他们可能不会是某个团队的专有的，他们会以服务于多个团队，为团队提供测试的解决方案。

顺着这条线索，我们不仅能推测出测试者在面临这样的转型条件下，需要重点关注的技术，还能推测出这些技术需要的深度或广度。比如前面我们提到的“代码能力”，如果测试者想继续在产品团队中做一些和功能特性相关的自动化测试工作，他可能需要考虑未来可能会和开发会有更深入的融合，比如达到可以结对编程的效果，那他势必要考虑在现阶段做更多的编程实现包括算法等方面的积累。如果测试者想在未来往测试工具或者平台方向发展，那么他可能就要更多的积累一些代码架构方面的知识，并兼顾对测试方法的总结和研究。这样测试者就可以根据自己的现状、感兴趣的方向，去做技能规划，梳理自己的知识结构，建立符合未来发展的知识结构，能够在未来的转型竞争中保持优势。

当然，上面的内容只是我按照我的经验来做出的预判，不一定可以符合所有的情况。大家完全可以从“测什么和怎么测”，即测试策略的角度，结合自己所处的行业的未来发展趋势，来分析出应对自己领域危机的方式和策略。

事实上，对测试来说，测试策略，不仅可以作用于未来，更能帮我们做好当下的测试。

无论我们用的是敏捷还是瀑布，对测试来说，产品都不亚于一个迷宫。无论测试用的黑盒的方式还是白盒的方式，测试就是在探索这座产品迷宫。**对被测对象的准确把握，就是探索迷宫的罗盘。**关键因素包括代码复杂度，开发的技能和经验准备度，需求的质量

等。有了这些，就可以判断风险的所在，**在把握失效规律和失效影响的前提下，有的放矢的来开展各种测试活动。而实际的情况却是，我们太缺少对被测对象的分析和思考，这让我们做了很多看起来必要实际却很低效的事情。**

举一个回归测试的例子。如果开发修改问题改得非常好，我们的bug 99%都可以回归通过，那么我认为测试没有必要再做回归测试，因为投入产出比非常小。我们可以这样算一笔账，假设一个周期为两个月的产品有200个bug，测试人员一天可以验证10个bug，这就是20人天的工作量，这还不算回归测试中的测试设计或者是发散测试的工作量。我们完全可以把这个时间放在测试其他的更值得测试的地方，或者干脆把发布周期提前一周又为何不可呢。

再举一个自动化的例子。某测试团队加班加点把某个功能接口全部自动化了，大概有1000多个自动化脚本，运行却只发现了2个问题，后面反复运行都再也没有发现过问题了。换句话说，上面提到的这部分系统，可能本来就是不太需要测试的系统，就算是完美的回归流程、测试设计、自动化测试，意义又有多大呢?

所以对相对于自动化测试，测试设计，流程等相对固定的东西之外，测试还需要一个可以审时度势，动态的分析和调整的技术，让测试可以聪明的工作，关键是，我们要能意识到，**测试最核心的东西是动态的，是和整个研发过程和市场定位紧密捆绑的。不管测试的竞争力如何变化的，唯一不变的是我们的应对原则：根据被测对象和测试团队制定合适的测试策略。**

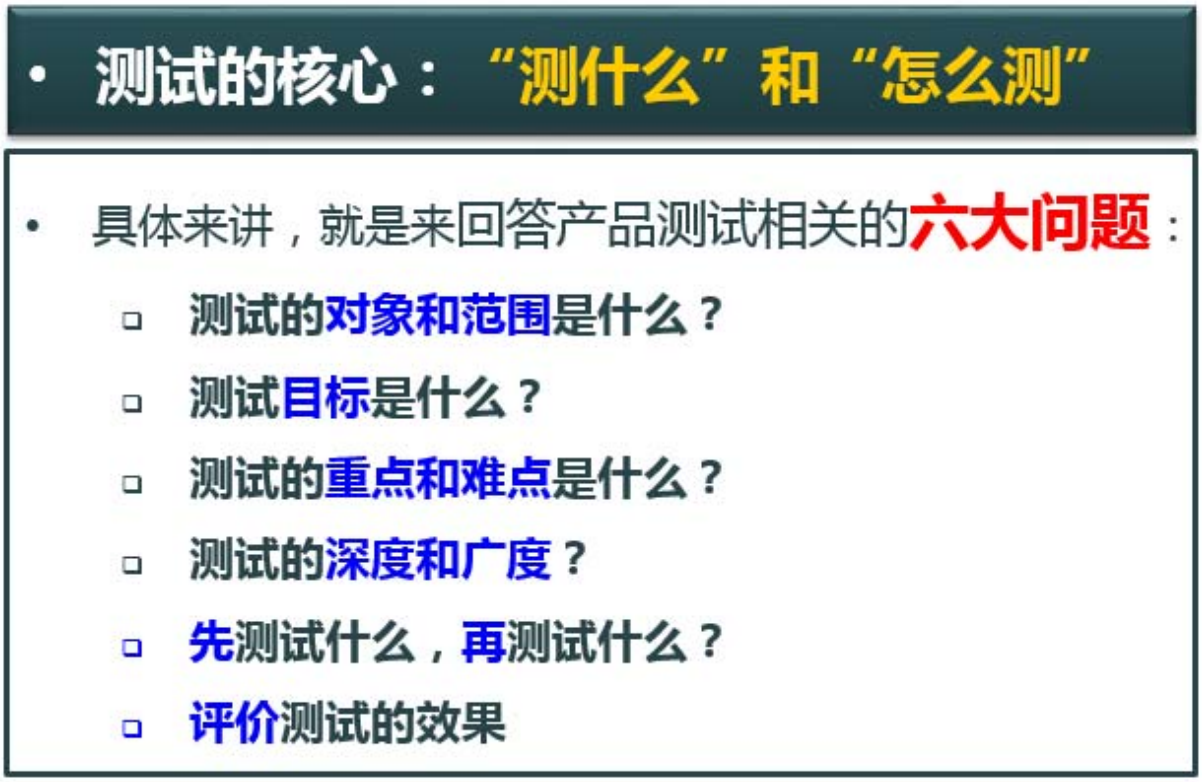
所以，我个人认为，**测试策略是测试技术中最核心的部分**。测试，简单来说就是，测什么和怎么测，这都是策略的范畴。**相比自动化、工具、测试设计等，测试策略往往更能更举足轻重地影响测试的质量和效率**——测什么和怎么测决定了质量，实际也决定了效率。如果能在深入分析的基础上大幅度减少测试用例，测试效率提升一定比自动化还高。然后再对最值得自动化的部分进行自动化，而不是为了自动化而自动化，测试效率又会大幅度提高。实际上，系统越来越复杂，我们总有分析不清楚的时候，如果自动化平台足够强大，我们多测试一些也没有关系，可以避免一些低级遗漏。所以，策略是重要的，但是毕竟是基于经验的，总会有不完善的地方，工具和自动化可以很好的补充。测试还没有达到科学或者工程定量的程度，但也不是只有少数人才会的艺术，而是一门基于经验，可复制性的工艺学科，只是还没有达到可以给出明确标准的批量复制程度。也正是因为如此，才更需要我们去做深入的思考和分析。

再认识测试策略

遗憾的是，目前大家对测试策略的关注度非常不够，市面上有很多讲如何操作测试工具的书，做测试设计或者是测试管理的内容，但测试策略相关的资料确少之又少。所以在正式讨论测试策略技术之前，我想先对测试策略相关的概念做一下讨论。

朱少明在微信公众号软件测试质量报告中的文章《究竟什么是软件测试策略》中指出，测试策略就是“为了以最低的成本完成最大程度的揭示（/降低）产品的质量风险或尽早地完成测试所选择的（或制定的）最合理或最合适的方式、方法、过程等”。我觉得朱老

师的这个定义略显复杂。我认为“测试策略”通俗来讲就是六个字：“测什么”和“怎么测”，具体来讲如何来解答好产品测试的六大问题：



这让制定测试策略看起来像是要写一道论述题一样。但是如果你真的想按照这个思路去给产品写一份测试策略，你会发现其中最大的一个问题就是，我该在何时开始制定测试策略？

容易想到的是我们需要在项目一开头就来制定测试策略，但这时项目中很多和测试策略相关的内容都不明了，测试策略容易写得很空虚，如果到测试执行的时候再写测试策略，测试策略的意义好像又没有那么大了。

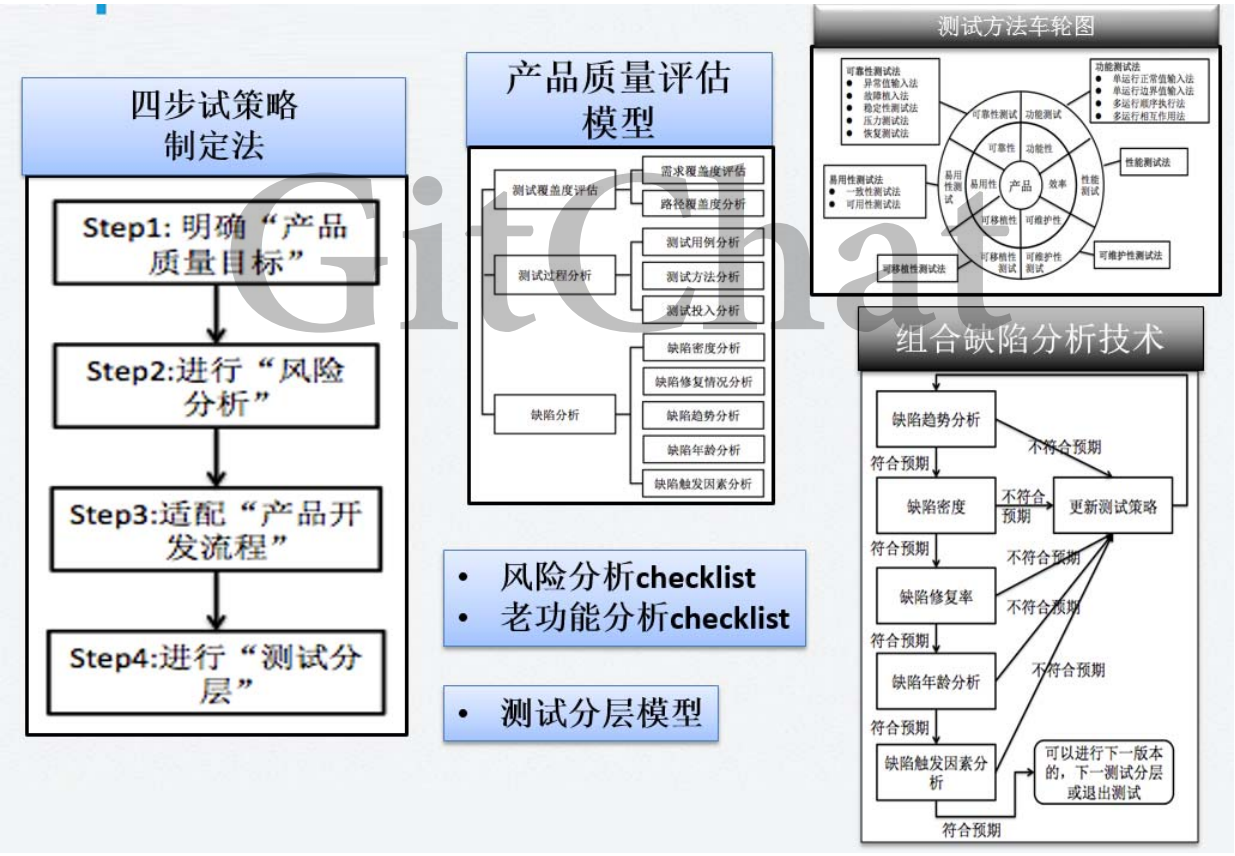
这个问题也给了我们一个提示：测试策略不是一开始，一下子就可以就写出来的，需要分阶段来完成，随着项目的深入，测试策略是需要调整和更新的。

另外一个难点是，这些问题看起来都不难，但是要答得很全面，并且深入到可以指导团队的程度却觉得很难。这就说明我们需要一些模型、工具或者checklist来帮助我们系统的分析思考。

四步策略制定法

四步测试策略制定法是我们制定测试策略的步骤。四步测试策略法的内容和核心理念如下图所示：

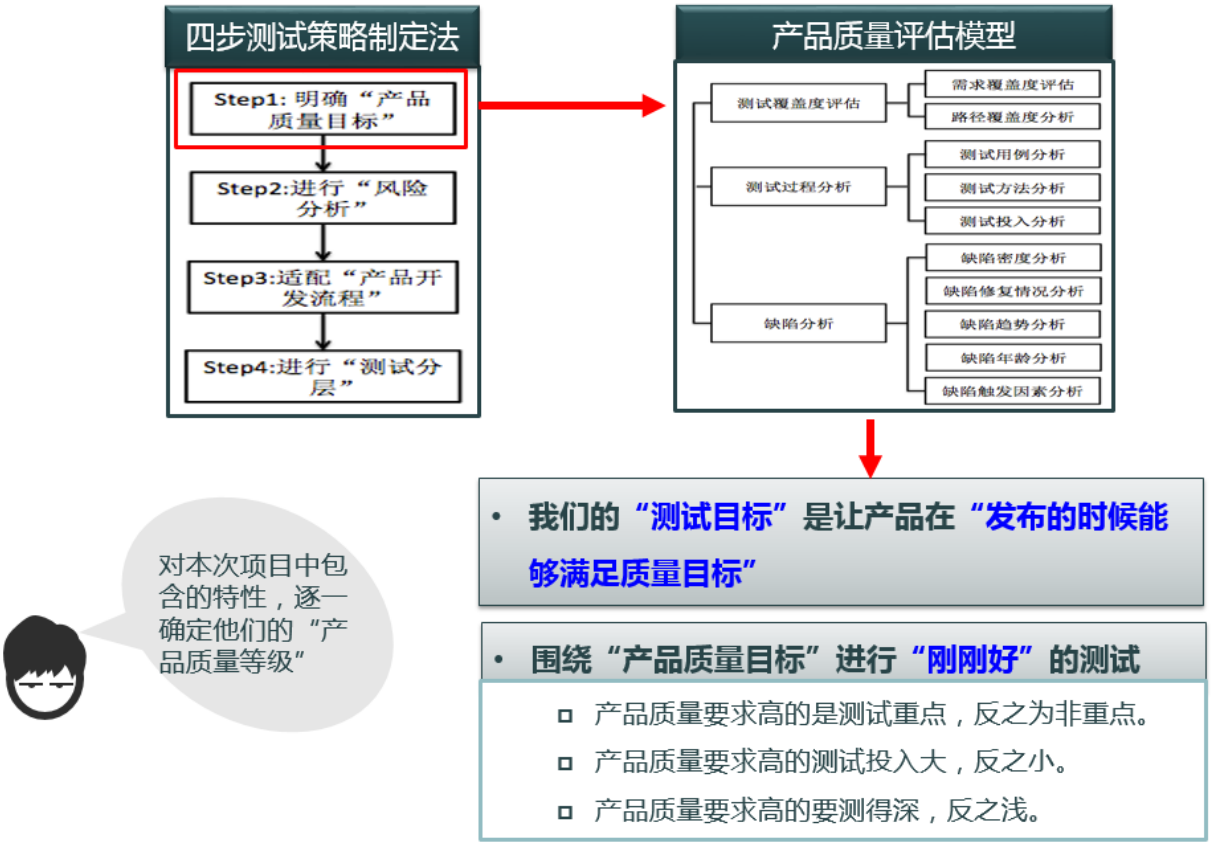
每个步骤又会有相关的模型、工具或者是checklist，如下图所示:



接下来我们将逐一进行介绍.

step1：明确“产品质量目标”

明确“产品质量目标”是我们在制定测试策略过程中十分关键的一个步骤。对我们而言，不仅需要关注操作层面的具体方法，更要理解其中蕴含的测试策略思想，即我们测试的目标是要围绕“产品质量目标”来进行“刚刚好”的测试，如下图所示：



请特别注意我们对“刚刚好”的理解：

- 产品质量要求高的是测试重点，反之为非重点。
- 产品质量要求高的测试投入大，反之小。
- 产品质量要求高的要测得深，反之浅。

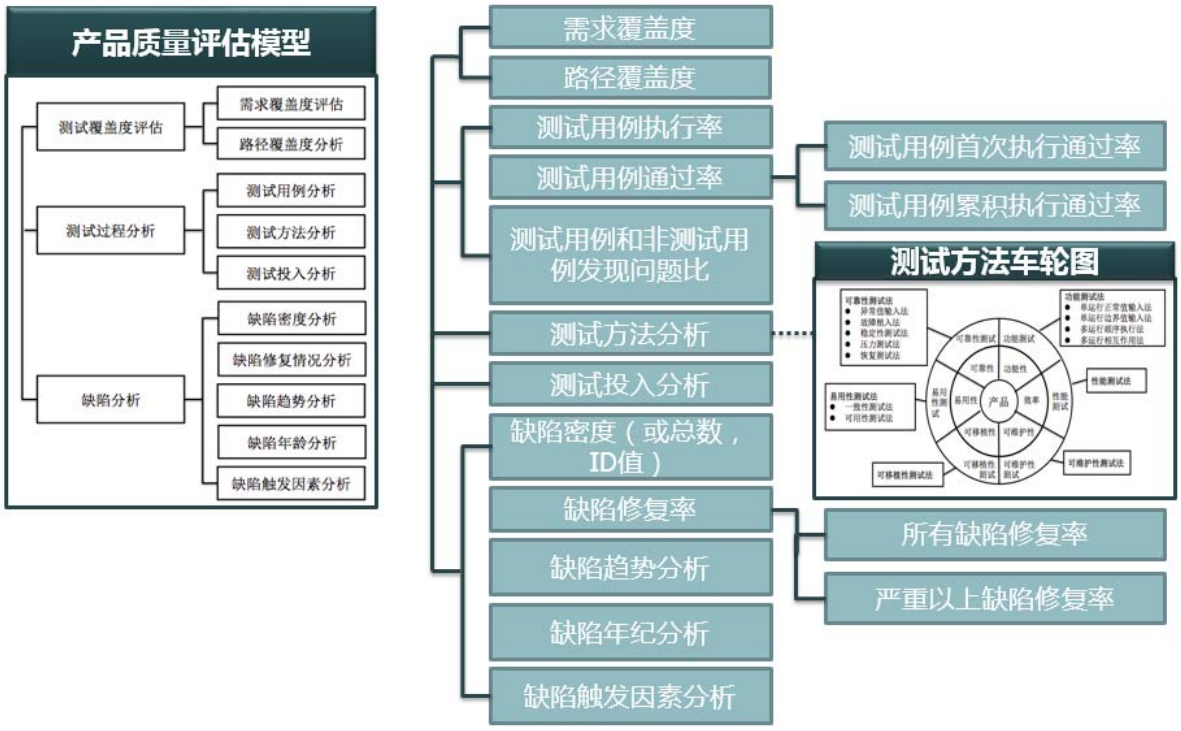
与之对比的是：

- 这是一个新开发的特性，大家都不熟悉，要作为重点好好测试一下。
- 这个特性，感觉没有什么用把，随便测试一下就好了。
- 这个特性，使用的技术还比较新，要作为重点好好测试一下。
- 这个特性还蛮有意思的，好好测试一下。
- 我想在这个特性中试试XX测试方法。

大家可以自己感受一下在这两种不同的思路下，对测试重点的把握，以及对测试质量和效率的影响。

此外，需要特别注意的是“刚刚好”的解释中的那三个“反之”。对大多数测试来说，投入更多，做更深入的测试是容易做到的，有意识的减少测试却是相当难的。但如果我们只是一味的加测试活动，而不敢于减掉一些测试，也不可能真正做得前面期望的测得全面和深入。原理很简单，全是重点，也就等于没有重点。

我们如何明确质量目标呢，在这里我们使用的是产品质量评估模型。产品质量评估模型的框架如下图所示：



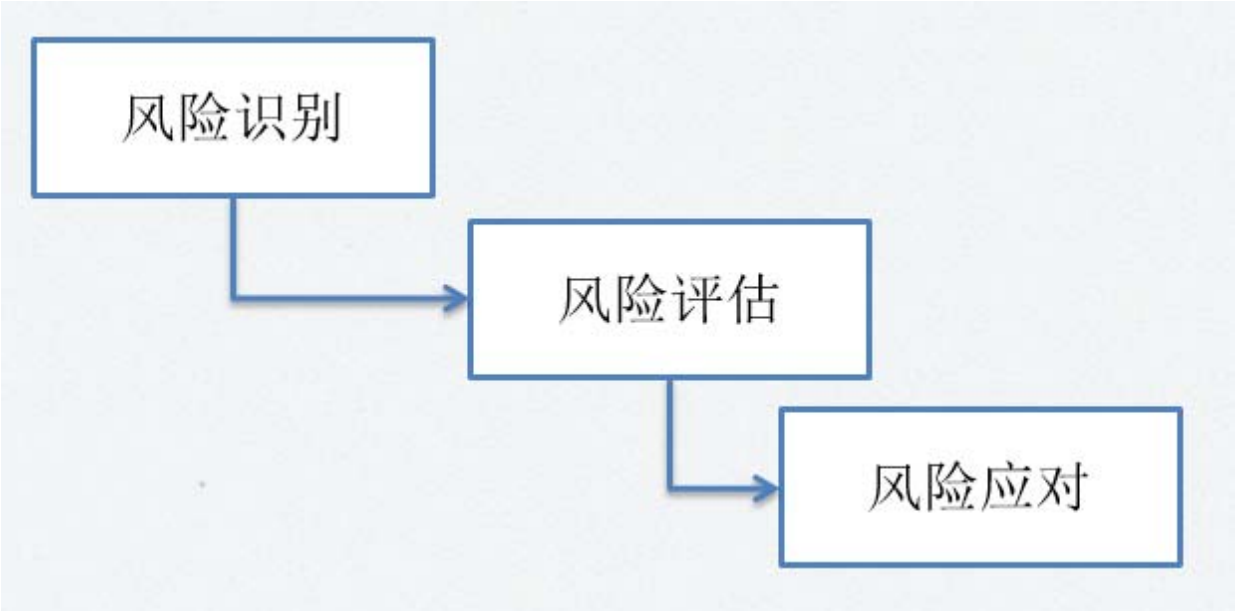
本文鉴于篇幅原因，无法提供详细的产品质量评估模型的说明。感兴趣的读者可以直接向我索取，我会以附件的方式提供给您。

step2：进行“风险分析”

即便是质量目标相同的功能特性，不同项目中的失效情况也可以会不一样，这就需要我们进行风险分析，并基于风险来进行测试。

我们前面讲到，测试最核心的东西是动态的，风险正是这种动态性最直接的体现。**我们对风险进行分析，目前就是来识别项目中那些可能会阻塞测试顺利进行的因素，然后加强对高风险地方的投入，同时减弱风险低的地方的投入。**

风险分析分为“风险识别”，“风险评估”和“风险应对”三个步骤：



内容上，我们不仅要对项目整体进行风险评估，还需要对老功能进行“差异性分析”和“历史测试情况分析”。

笔者从“需求”、“设计”、“流程”、“变更”、“组织和人”和“历史”六个维度，整理了《风险分析checklist》，从测试方法分析，缺陷分析和组织和人三个维度提供了差异性分析和历史测试情况分析表，限于篇幅太长，需要的朋友可以直接向我索取.

step3. 适配“产品研发流程”

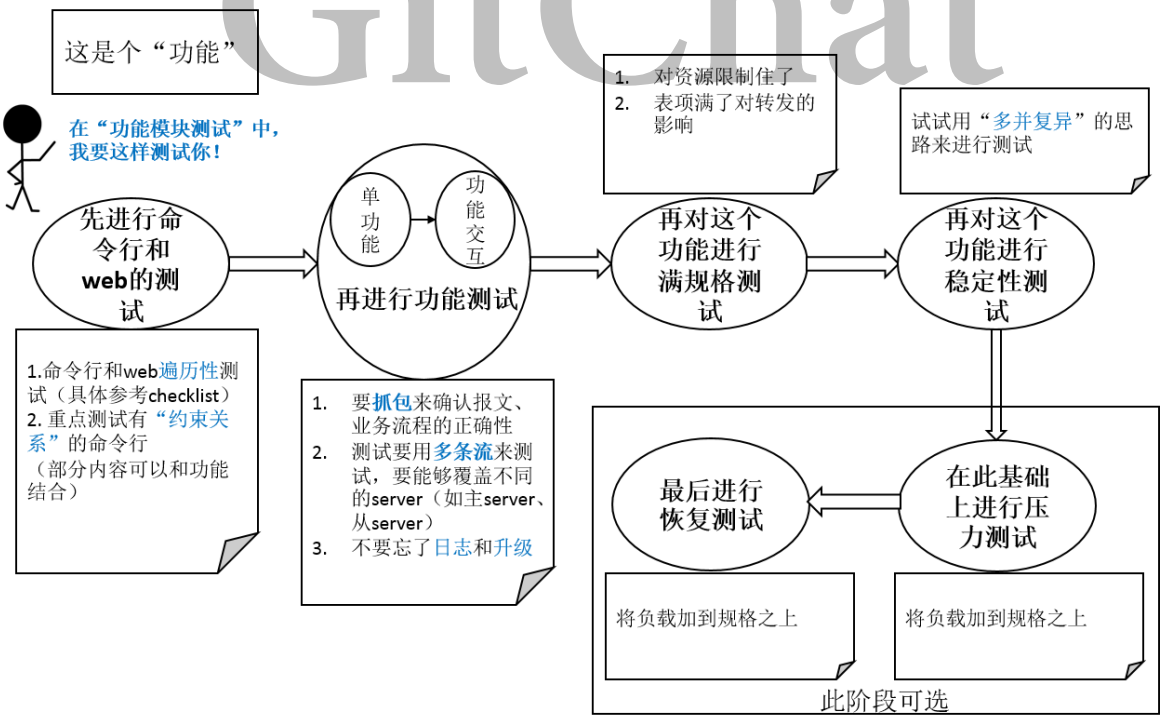
无论我们用的是传统的瀑布还是敏捷，都会有产品开发的节奏。产品开发也是随着这个的节奏逐渐展开的，产品的细节、问题也会随着产品开发的深入而逐渐暴露出来，测试策略也需要遵循这样的节奏来逐渐展开，确定在整个项目中要开展几次测试策略的制定活动，每次活动中测试策略要写到怎样的程度，这叫“确定测试策略的结构”。

这样操作的好处是可以在产品开发过程中不断的去回顾、总结和调整测试策略，避免测试策略变成“假大空”的形式，不能真正起到指导测试的作用。

第二个好处是，我们的测试活动和产品研发流程是有直接关系的。适配产品研发流程也是帮我们确定有哪些测试活动需要在后面的工作中进行。

step4. 进行“测试分层”

测试分层，就是把前面我们确定的活动进行归类，然后再在最合适的研发阶段来执行相应的测试活动。最经典的一个测试分层的例子，就是V模型下的“单元测试”、“集成测试”、“系统测试”和“验收测试”。敏捷下的“单元测试”、“功能测试”、“非功能测试”、“探索式测试”也可以看做是一个测试分层的例子。



测试策略编写举例

接下来我们将通过一个实际的例子来看如何四步测试策略法来制定测试策略。

在正式做测试策略评估之前，我们还需要确定一项内容，就是产品的质量等级。

产品的质量等级

产品的质量等级是我们在实际项目中判断质量好坏的“刻度”，因此产品质量等级完全是站在最终用户的角度来进行定义的。下面是一个产品质量等级的例子（注：这个例子也是和产品强相关的例子，并不是通用，大家在实践中需要结合自己的产品销售方式、最终用户的质量要求来制定符合自己产品实际需求的质量等级）：

- 第1级 完全商用：特性完全满足用户的需求，有少量（或者无）遗留问题，用户使用时无任何限制。
- 第2级 受限商用：特性无法满足用户的某些特定场景，有普通以上的遗留问题，但有规避措施。
- 第3级 测试、演示或小范围试用：特性只能满足用户部分需求，有严重以上的遗留问题，且无有效的规避措施，问题一但出现就会影响用户的使用，只能用于测试（例如Beta）或演示，或者小范围试用。
- 第4级 不能使用：特性无法满足用户需求，存在严重以上的遗留问题，会导致用户基本功能失效，且无规避措施，产品根本无法使用。

为项目中的每个特性确定质量等级

接下来我们就可以对项目中的每个特性来确定质量等级了。

我们以下面的这个实际例子为例：

特性	子特性	需求描述
文件还原	**文件还原功能**	文件还原功能
		文件还原配置
	协议支持	支持HTTP协议
		支持FTP协议
		支持POP3协议
		支持IMAP协议
		支持SMTP协议
	文件类型支持	普通文件类型支持
		压缩文件类型支持
静态检测	**文件解析**	文件解析功能
		普通文件类型支持
	静态检测	无差别shellcode检测
		JS/AS脚本引擎
	静态告警详情	静态告警详情

我们按照上一节中的质量等级的定义，来对这些特性来确定质量等级，假设如下所示（说明，这里质量目标和特性的对应关系只是假设示意，请大家在实践时按照自己产品的实际情况来进）：

特性	子特性	需求描述	质量目标
文件还原	文件还原功能	文件还原功能	A
		文件还原配置	C
	协议支持	支持 HTTP 协议	A
		支持 FTP 协议	A
		支持 POP3 协议	A
		支持 IMAP 协议	A
		支持 SMTP 协议	A
	文件类型支持	普通文件类型支持	B

按照质量等级，来对质量模型确定不同的质量目标

接下来我们需要按照质量等级的要求，来对质量模型中的评估项目，确定不同的质量目标如下所示：

产品质量评估模型	产品质量评估项目	完全商用（目标）	部分商用（目标）	测试、演示或小范围试用（目标）
测试覆盖度				
	需求覆盖度	100%	100%	100%
	路径覆盖度	>=75%	>=60%	不涉及
测试过程				
	测试用例执行率	100%	100%	85%
	测试用例首次执行通过率	>=75%	>=70%	60%

产品质量评估维度	产品质量评估项目	完全商用（目标）	部分商用（目标）	测试、演示或小范围试用（目标）
测试过程				
	测试方法	需要使用功能、性能、可靠性和易用性中所有的测试方法	需要使用功能测试的所有测试方法，可靠性中故障植入法和稳定性测试法	只需要使用功能测试方法即可

对不同的特性确定相应的质量要求

我们对质量评估模型按照质量等级确定了不同的目标后，接下来我们就可以对不同的特性，根据他们不同的质量等级，来确定不同的质量目标。以上面的例子为例：

特性	子特性	需求描述	质量目标	目标分解
文件还原	文件还原功能	文件还原功能	A	测试覆盖度 测试过程 缺陷
		文件还原配置	C	测试覆盖度 测试过程 缺陷
	协议支持	支持 HTTP 协议	A	测试覆盖度 测试过程 缺陷
		支持 FTP 协议	A	测试覆盖度 测试过程 缺陷
		支持 POP3 协议	A	测试覆盖度 测试过程 缺陷
		支持 IMAP 协议	A	测试覆盖度 测试过程 缺陷
		支持 SMTP 协议	A	测试覆盖度 测试过程 缺陷
	文件类型支持	普通文件类型支持	B	测试覆盖度 测试过程 缺陷

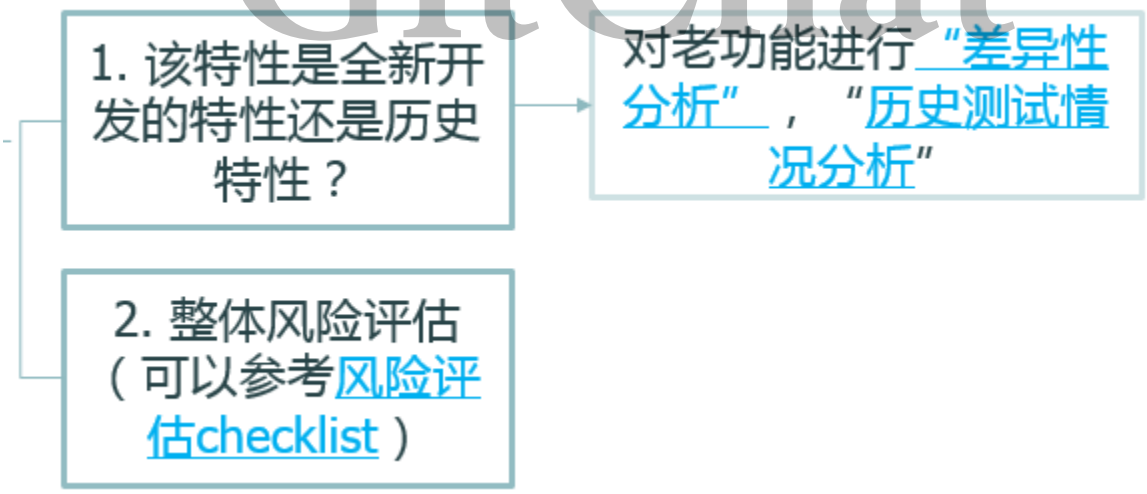
这样我们就完成了四步测试策略法中的第一步，明确产品的质量目标。

其实此时，我们已经对不同的特性按照质量等级做出了一个大致的分类，并确定了在这个分类下，这个特性需要测试的深度和广度，测试过程中要测试活动和最后希望的测试结果，都有了整体的认识。如下图所示：

接下来我们再来考虑风险因素。

考虑各个特性的风险因素

我们可以通过风险分析checklist，差异性分析和历史测试情况分析表，来对每个特性的风险来来进分析。



这样我们可以对特性再进行一次更细致的分类，并得到一些风险应对措施，如下图所示：

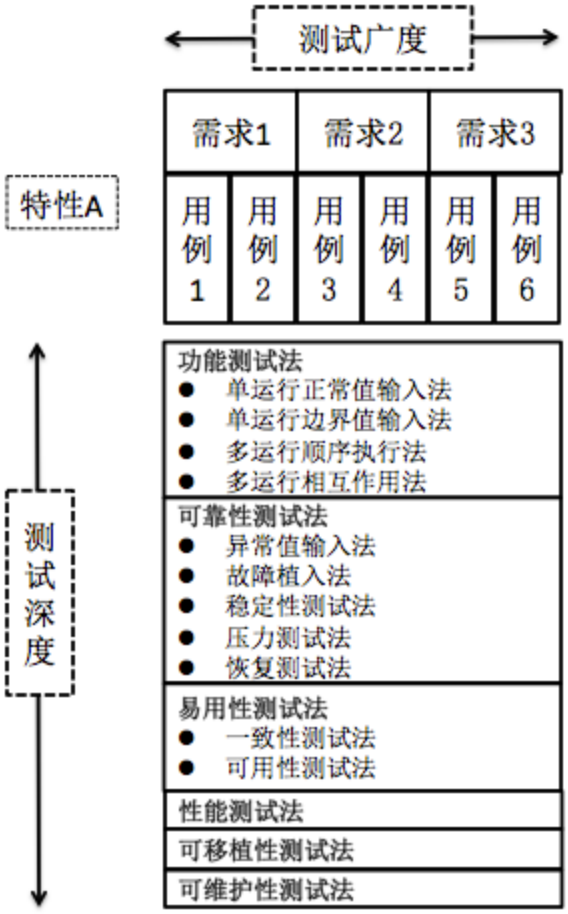
特性	子特性	需求描述	质量目标	目标分解	分类	风险应对
文件还原	文件还原功能	文件还原功能	A	测试覆盖度 测试过程 缺陷	老特性加强	更新测试设计文档
		文件还原配置	C	测试覆盖度 测试过程 缺陷	老特性不变	
		支持 FTP 协议	A	测试覆盖度 测试过程 缺陷	老特性不变	
		支持 SMTP 协议	A	测试覆盖度 测试过程 缺陷	老特性不变	
		普通文件类型支持	B	测试覆盖度 测试过程 缺陷	老特性加强	更新测试设计文档； 加强概要设计评审
	静态检测	无差别 shellcode 检测	A	测试覆盖度 测试过程 缺陷	老特性变化	加强概要设计评审
		JS/AS 脚本引擎	A	测试覆盖度 测试过程 缺陷	全新特性	加强需求评审； 加强概设评审

基于质量和风险分析来确定测试的深度和广度

接下来我们就可以根据前面的质量和风险分析结果来确定测试的深度和广度。

所谓测试的深度，是指在测试过程中会使用的测试方法。使用的测试方法越多，我们可以认为测试得越深入。

测试的广度是指测试的范围。下图比较直观的给出了测试深度和测试广度的示意：



在这个例子中，我们得到的测试深度和广度如下：

特性	子特性	需求描述	质量目标	目标分解	分类	风险应对	测试深度	测试广度
文件还原	文件还原功能	文件还原功能	A	测试覆盖度 测试过程 缺陷	老特性加强	更新测试设计文档	使用功能测试的所有测试方法，可靠性中故障植入法和稳定性测试法	❑ 对于新增的文件类型，进行全面测试； ❑ 对于之前支持的文件类型，进行回归测试+探索性测试； ❑ 稳定性测试
		文件还原配置	C	测试覆盖度 测试过程 缺陷	老特性不变		只需要使用功能测试方法即可	回归测试
		支持 FTP 协议	A	测试覆盖度 测试过程 缺陷	老特性不变		使用功能测试的所有测试方法，稳定性测试法	回归测试+探索性测试
		支持 SMTP 协议	A	测试覆盖度 测试过程 缺陷	老特性不变		使用功能测试的所有测试方法，稳定性测试法	回归测试+探索性测试
		普通文件类型支持	B	测试覆盖度 测试过程 缺陷	老特性加强	更新测试设计文档； 加强概要设计评审	需要使用功能、性能、可靠性和易用性中所有的测试方法	❑ 对于新增的文件类型，进行全面测试； ❑ 对于之前支持的文件类型，进行回归测试+探索性测试；
	静态检测	无差别 shellcode 检测	A	测试覆盖度 测试过程 缺陷	老特性变化	加强概要设计评审	需要使用功能、性能、可靠性和易用性中所有的测试方法	全面测试
		JS/AS 脚本引擎	A	测试覆盖度 测试过程 缺陷	全新特性	加强需求评审 加强概设评审	需要使用功能、性能、可靠性和易用性中所有的测试方法	全面测试

实际项目中，我们可以将测试深度和广度合并，并给出在这个这个测试深度和广度下，需要分析的内容和方法。下面内容摘抄自我的一个实际项目（只是部分摘抄），供大家参考：

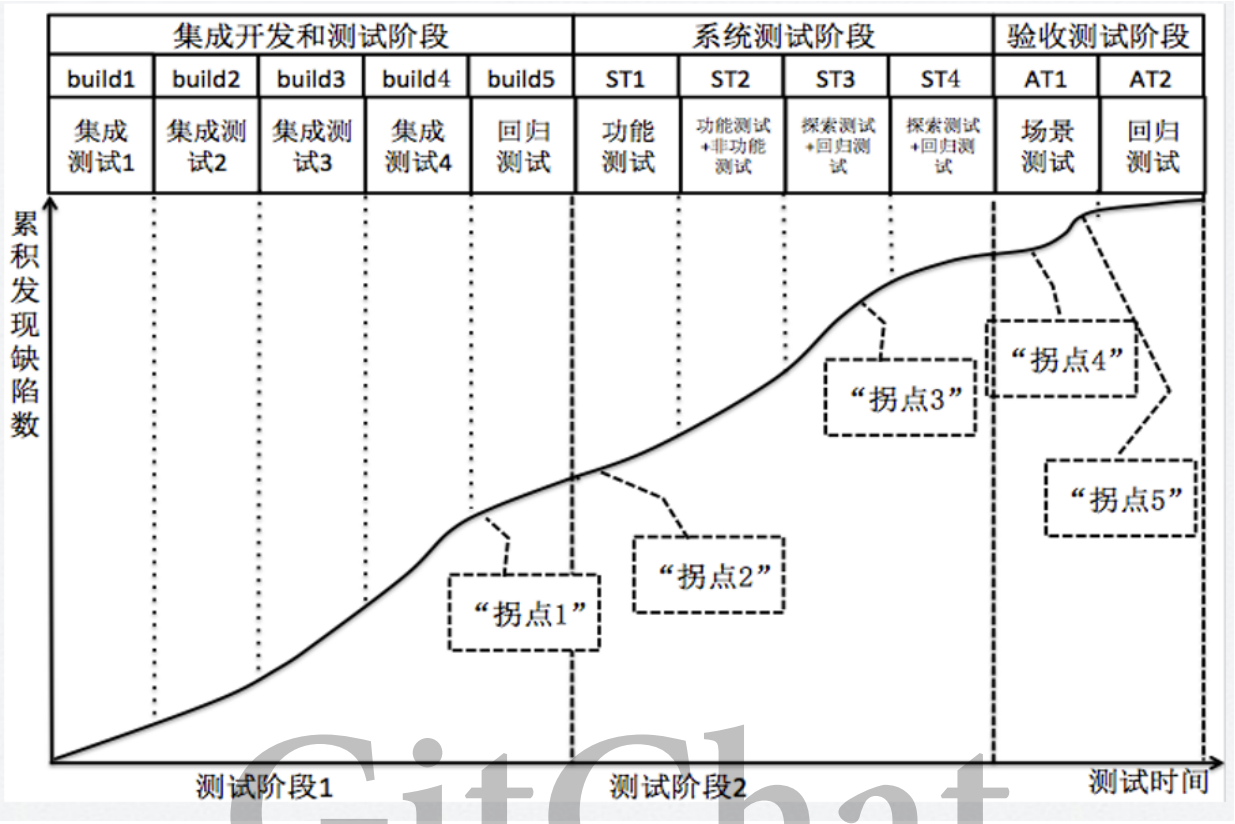
版本标识	功能类别	被测对象分类	说明	测试深度和广度	需要分析的内容和方法
X	A	**新增功能模块**	全新功能点	全面测试。	1. 明确需求的场景和业务流程。需要针对场景和业务流程来设计测试用例 2. 对开发实现流程进行覆盖（至少保证最小线性无关覆盖） 3. 进行功能交互分析 4. 进行测试类型分析 5. 是否支持并行化？如果支持，考虑和并行化相关的测试设计 6. 是否支持虚拟化？如果支持，考虑和虚拟化先关的测试设计 7. 考虑命令行、web（详见 D 类中的分析）
	B	**老功能加强**	老功能加强，新增小的功能点	对新增的功能进行全面测试。 分析加强部分的功能对老功能的影响，针对影响进行测试。	1. 明确需求的场景和业务流程。需要针对场景和业务流程来设计测试用例 2. 对开发实现流程进行覆盖（至少保证最小线性无关覆盖） 3. 进行功能交互分析 4. 进行测试类型分析 5. 进行新增部分对老功能的影响分析 6. 是否支持并行化？如果支持，考虑和并行化相关的测试设计 7. 是否支持虚拟化？如果支持，考虑和虚拟化先关的测试设计 考虑命令行、web（详见 D 类中的分析）

适配产品研发流程

很多时候，我们的产品研发流程并没有太大的变化，这样测试策略的框架，测试活动在整体上应该不会有太大的变化。在这一步中，需要我们重点考虑的是，沟通并安排好我们在风险识别中识别出来的那些，额外要做的一些质量保证活动。比如测试的同学要对某某新特性做一下技术学习储备，新的测试工具的学习等。

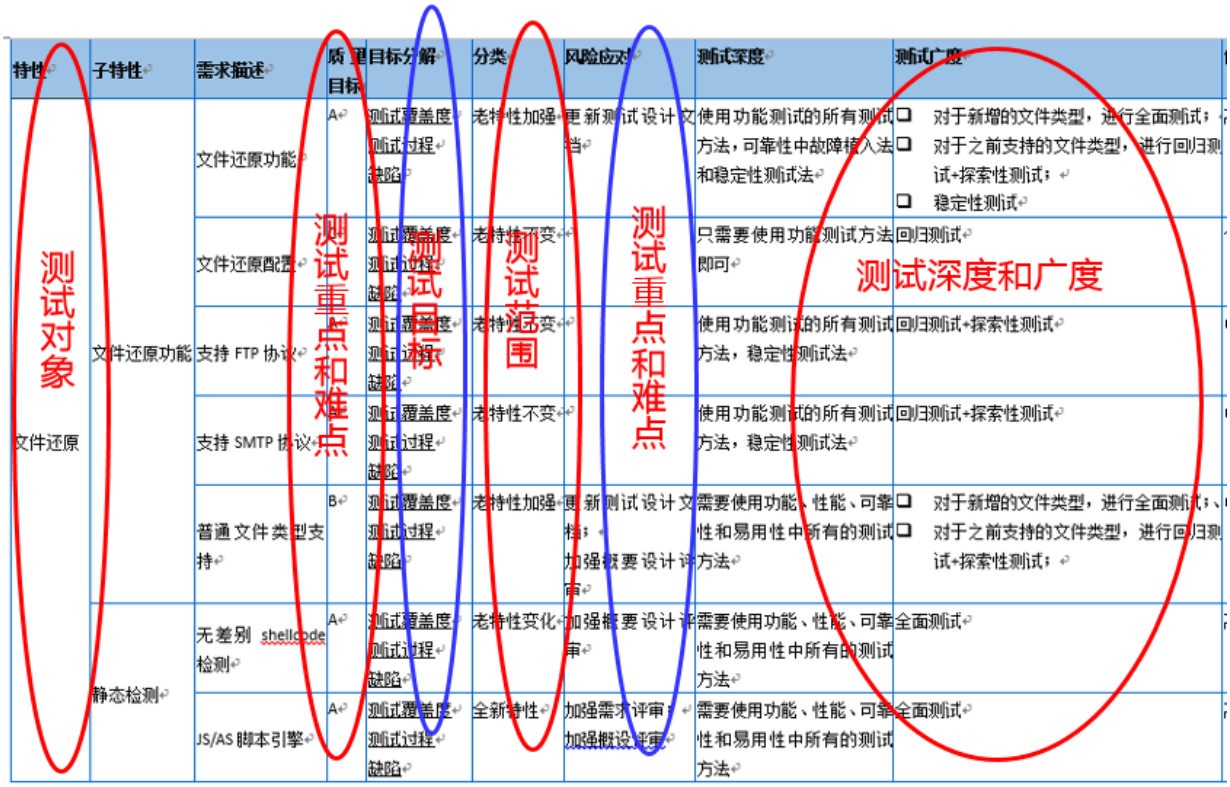
确定测试分层，预估测试结果

最后我们可以根本版本计划，确定测试分层，并对测试分层做更细粒度的划分，确定每个测试层次中要做的测试活动和这些测试活动的顺序。除此之外，我们还可以结合质量目标，版本的历史测试情况等信息来估计测试的结果（注：这里使用了组合缺陷分析技术，可以参考质量评估模型中对组合缺陷分析技术的说明）：



回顾：到目前为止我们是否成功回答了产品测试的六大问题

还记得前面我们提到的产品测试的六大问题么。下面我们就来总结回顾一下，我们是如何使用四步测试策略制定法，来系统的分析和解答相关的问题的。



特性	子特性	需求描述	质量目标	目标分解	分类	风险应对	测试深度	测试广度
文件还原	文件还原功能	文件还原功能	A	测试覆盖度 测试过程 缺陷	老特性加强	更新测试设计文档	使用功能测试的所有测试方法，可靠性中故障植入法和稳定性测试法	❑ 对于新增的文件类型，进行全面测试； ❑ 对于之前支持的文件类型，进行回归测试+探索性测试； ❑ 稳定性测试
		文件还原配置		测试覆盖度 测试过程 缺陷	老特性不变		只需要使用功能测试方法即可	回归测试
		支持 FTP 协议	A	测试覆盖度 测试过程 缺陷	老特性不变		使用功能测试的所有测试方法，稳定性测试法	回归测试+探索性测试
		支持 SMTP 协议		测试覆盖度 测试过程 缺陷	老特性不变		使用功能测试的所有测试方法，稳定性测试法	回归测试+探索性测试
		普通文件类型支持	B	测试覆盖度 测试过程 缺陷	老特性加强	更新测试设计文档； 加强概要设计评审	需要使用功能、性能、可靠性和易用性中所有的测试方法	❑ 对于新增的文件类型，进行全面测试； ❑ 对于之前支持的文件类型，进行回归测试+探索性测试；
	静态检测	无差别 shellcode 检测	A	测试覆盖度 测试过程 缺陷	老特性变化	加强概要设计评审	需要使用功能、性能、可靠性和易用性中所有的测试方法	全面测试
		JS/AS 脚本引擎	A	测试覆盖度 测试过程 缺陷	全新特性	加强需求评审， 加强概设评审	需要使用功能、性能、可靠性和易用性中所有的测试方法	全面测试

当然“先测什么，再测什么”和“测试评估”的内容，我们也在前面的讨论中涉及到了。

测试策略如何落地

前面我们分析了很多内容，大家可能会有个担心，就是测试策略的这些内容要如何落地，而不会变成一个写完之后大家都不管了，很空的一个文章。这个问题展开讨论可以包含很多内容，。限于篇幅的原因，接下来我们就来简单讨论一下，测试策略是如何指导测试设计和测试执行的。

测试策略是如何指导测试设计的

测试策略指导测试设计，其实非常简单，只要测试团队按照测试策略的分类，按照需要分析的内容和方法，对具体的特性进行测试分析就可以了。

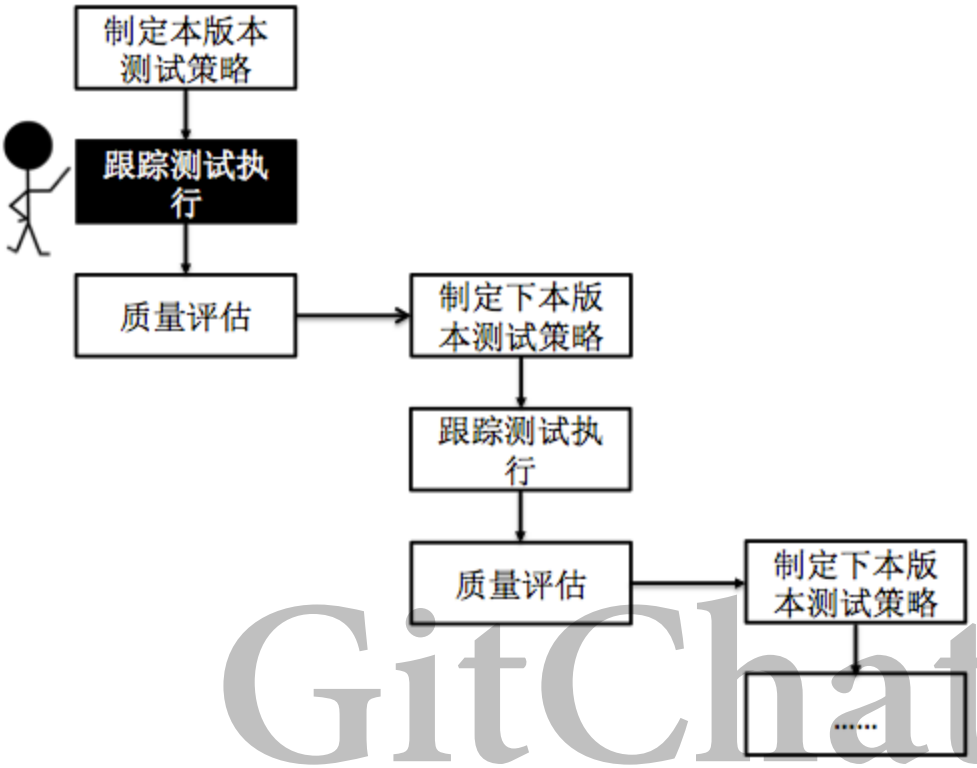
例如，上面我们的举例中，对A类功能，测试人员需要全面测试，并至少从7个方面来进行分析，那测试人员在做测试分析的时候，就需要根据测试策略的指引，从这7个方面来对被测对象进行分析，得到测试点。下图是上面的例子中，对应的A类特性的被测对象分析的示意，从这部分截图中，我们可以看到测试策略和测试设计的呼应关系。

被测对象类别	*A 类：新增功能*
测试目标：	支持客户端通过 telnet 登陆 CLI，进行设备管理（默认关闭）
测试场景：	测试场景： 1、 241 网段---（管理口）NF（配一条能通向工作 PC 的路由） 2、 241 网段---（工作口）NF（配一条能通向工作 PC 的路由）
被测对象分析：	**1. 功能实现流程** 带外管理（M 口和 H 口）：数据包不经 server，直接斤内核进行处理 带内管理（业务口）：数据包由 server 收入，进行判断，如果是发给本设备的，则送入内核进行处理 **2. 并行化分析** *（如果有）* 带外管理与并行化无关，带内管理支持并行化。 ✓ 多 server 运行的情况下，通过业务口进行带内管理（telnet 管理通过业务口 IP 登录设备 CLI）时，多个 server 从各接口收集数据包，并发给内核进行处理。 **3. 虚拟化分析** *（如果有）* 支持虚拟化 ✓ 根系统管理帐号新建 telnet 管理帐号时可指定虚拟系统； ✓ 虚拟系统帐号（操作员、审计员）只可管理自己的虚拟系统，也只能见到本虚拟系统的资源，无新建帐号权限； ✓ 根系统帐号（操作员、审计员）权限最大，可以配置和查看各虚拟系统的资源； ✓ 缺省管理帐号——缺省操作管理员 1 个（admin），缺省审计管理员 1 各（auditor），均属于根系统； **4. 功能交互分析** ✓ telnet 管理帐号和系统帐号一致； ✓ telnet 登录 cli 后，权限与 web 界面帐号权限一致； ✓ 同一账号，可以多用户同时登录；（web 和 cli） **5. Web 和命令行分析** Web 界面： ✓ 增加 telnet 管理开关——开启/关闭； ✓ 新建 telnet 帐号时选择所属虚拟系统； CLI 命令行： ✓ 可以开启/关闭 telnet 管理；

测试策略是如何指导测试执行的

到了测试执行阶段，测试设计可以保证测试执行的内容，前期识别出来的需要补充开展的活动可以起到质量预防和风险应对的左右，测试策略中考虑的测试执行顺序，可以帮助大家按照最合适的方式来进行测试。

实际项目中，测试执行往往和计划有偏差，这还需要我们根据测试执行情况，跟踪测试执行情况，评估当前的质量，然后根据版本的实际情况来动态的调整接下来的测试策略，保障在项目结束的时候可以达成测试的目标，如下图所示：



一般来说，我们在制定版本测试策略时，需要考虑的内容包括：

- 测试范围已经和计划相比的偏差
- 本版本的测试目标
- 需要重点关注的内容
- 测试用例的选择
- 测试执行顺序
- 试探性的测试策略
- 接收测试策略
- 回归测试策略
- 探索测试策略
- 自动化测试策略

跟踪测试执行时，需要关注：

- 跟踪测试用例的执行情况，特别是那些被阻塞的测试用例。
- 缺陷跟踪和分析，确定必须要在下个版本解决的问题，确定测试中发现缺陷的趋势是否符合预期，确定是否存在缺陷修改引入的问题。

最后我们需要根据评估结果来调整测试策略，调整测试策略的整体思路如下：