提高代码测试覆盖率

一、覆盖率不高的原因

1. 测试用例在撰写的时候容易表面上把页面所有信息的测试都考虑到，但实际上遗漏了大量测试覆盖点，导致测试出的程序比较脆弱。测试用例的覆盖度比较低，系统测试用例要想做到100%覆盖是很难的，按设计功能划分是远远不够的，还有大量的内部处理、转换、业务逻辑、相互影响的关系都是设计和需求中没办法点到的。

2. 测试数据设计方法的不足。

二、方法

*针对**测试用例**的设计，提出以下几点设计方法。*

A.测试用例的切面设计

所谓测试切面设计，其实就是测试用例大项的划分。测试用例划分的经典方法是瀑布模型，也就是从上到下，逐渐细分，大模块包括小模块，小模块包括更小的模 块。但仅仅如此是不够的，我们还要从更多的角度切入系统，从不同的角度把系统切分成一块一块的，来进行测试，从而确保测试大项的完整性。

1、功能点切面

这是最常见的切面，通常我们认为页面上的一个按钮就是一个功能点。然后我们可以根据功能的复杂程度，按每个功能；或一个功能点分多页；或多个功能点合成一页来进行用例的撰写。

2、特定切面

除此以外，还有一种特定切面的划分方法，也是用例撰写时经常会用到的。所谓的特定切面，就是忽略掉表面上的功能点，而关注测试对象的某一个面。比如我们 的内部管理系统提供了销售录入导入、注册录入导入等功能，从菜单划分上对应了七八个功能点。但这些功能处理后台有个共同的处理项就是授权记录的生成，这时 我们就可以把“授权记录生成”单独拿出来做一个测试项，而在其它测 试项中涉及这一部分的用例就不必再一一撰写。此外象一些界面共通的操作用例单独写成一页，也是一种特定切面。所以如果说将用例按功能点划分是一种纵向划分 法，那么特定切面就是从横向的角度分析所得到的切面。在普通功能点划分上再根据实际情况设计特定切面，可以使我们的用例阅读性、理解性、操作性更强。

3、隐含切面

这类用例是最容易被忽略的。它往往不是明显的某个功能项，可能是功能项后台的隐含处理，也可能是多个功能项之间的关联处理，甚至可能是在某种特定情形下的处理。这都需要测试人员通过对软件的学习了解，来进行挖掘。

（1）、后台功能

常见的如一些定时自动启动的服务；以及某种特定情况下自动执行的操作等。它们在界面上往往是不体现的，但许多在需求设计中还是会提到，也有一些比较细小 的功能可能会被忽略，就需要测试人员根据对项目的了解程度来进行挖掘。所以说一个熟悉项目的和一个不熟悉的测试人员，写出来的用例就完全是两个层次的。

（2）、完整业务流程的测试

我们都知道测试用例的设计是从点、线、面三个层次去考虑的。完整的一个功能项是线，其中的某个按钮是点，多个相关功能结合成完整业务流就是面。从实际来看这类用例往往被我们忽略。

事实上目前公司的软件本来都是业务型应用软件，将各种功能从业务流中切割出来单独写用例，肯定也会有涉及到整体流程的情况。若不加以区分，将细节与全局 搅在一起，不仅思路混乱，也容易考虑不周。因此在系统测试阶段，建议用例设计要有分有合，针对具体功能的就只围着这个功能转：而在业务流程测试项中，再完 全从整体的业务流角度出发去考虑用例，这样不仅不容易产生疏漏，用例阅读与执行也更清楚。

（3）、某种特定情况下的系统运行

这类用例的设计往往与系统实际业务情况密不可分。比如**软件，通常需要在月尾一天、月头一天、年尾一天、年头一天，对所有相关功能中的日期处理进行测试；又比如 WIN 2000 环境开发测试的系统，要测试在 98、XP、2003 等操作系统下是否能运行自如；再有如存在大量动态图片视频等的网页，在普通网速下的展现速度等等。总之就是要尽可能从实际应用的角度出发考虑，来进行测试补充。

（4）、其它相关系统

即指在当前项目中直接使用的其它成果，包括公司自有的系统模块、组件、函数；以及**或免费得到的一些功能组件。对这些内容需要预先与开发组长等讨论清 楚，是否需要测试。若时间紧张或其它原因决定不测的，应在测试计划中说明。若需要测试的，则具体可根据实际情况来设计，可以是通过系统某个功能的测试来涉 及，此时就不需要单独划分测试项；若相对比较独立的，也可以通过单独的测试项来对其专门进行测试。

（5）、除功能测试外的其它测试类型

包括可靠性、安全性、恢复性、配置安装测试等等，这些测试类型都是一个单独的测试项。

所谓好的开始是成功的一半，保证测试项划分的完整、合理、正确，会直接影响到本次测试的成效。通常建议该阶段工作要花 1-2 天的时间来考虑，并要在测试过程中随着对软件的深入了解，不断进行调整补充。可千万不要认为把分析设计中的功能模型图搬搬过来就可以了。

B.详细用例的设计

划分好了测试项，接着就是针对各个测试项，考虑具体的测试用例了。根据测试项的特点，测试用例的设计角度也有所不同。下面我们就来看看通常的功能点测试用例，该从哪些角度出发来进行设计：

1、功能切面表面用例设计

（1）、具体功能测试

根据需求分析设计，按页面提供的各个功能项，采用黑盒测试的各种方法，设计用例。比如页面提供了增、删、改、查功能，那么这四个功能是否正确实现就是我要验证的。这是最简单、最基本，同时也是必须的测试用例，通常我们的编码人员自测也就是做到这个程度。^_^

（2）、组合操作的测试

这是从上一角度扩展出来的，相对而言也是编码人员不会去测试的，所以需要测试人员多作考虑。

所谓组合操作测试，也就是选择某几个操作项，按一定的顺序进行操作，验证系统不会出现意外错误。当然要将所有功能项排列组合一遍来测试不仅不必 要，也是不可能的。所以具体要将哪些功能项进行结合，要按怎样的步骤来操作，还是需要测试人员根据实际情况来作设计（所以说在 IT 业人才就是一切呀，呵 呵：）。

一般来说我们会考虑功能项之间的数据是否会存在关联，若有就需要考虑这种组合了。常见的如查询功能，需要将各条件逐一累加进行测试；增完的数据 能否改，改完能否删，删完能否再增，这之间能否查询到正确结果；按钮的连续多次点击会否出现异常；有严格前后顺序要求的几个操作，尝试颠倒顺序去操作，系 统能否控制等等。

不仅在某功能内部，扩展到有关联的多个功能项之间，同样有组合操作测试的存在。如申报完了能才反馈；如申报成功或失败后再尝试申报等。当然对于 这类用例既可以写到某个功能切面中，也可以单独写到完整业务流程的切面中，这就取决于可能涉及用例的数量了，若关系比较复杂，当然是单独写比较好；若也就 是三五个用例数，那就直接在某个功能的用例中补充好了。

（3）、GUI 界面的测试

这类测试是测试人员的强项，具体测试项目如限长、非法输入等等，就不必赘述了。要提醒的是在测试时，一定要从实际使用者的操作习惯出发。要知道 界面原型所能确定的也只是页面的摆放显示，而实际操作时的控制实现仍是编码人员自行实现的，即使有编码指南，其所及范围也是十分有限。而许多编码人员在用 户操作方便性上的考虑往往差强人意。所以测试人员就必须要把好这一关。

（4）、数据初始化情况测试

不该为空的数据是否有校验；该有默认值的数据默认值是否正确；引用其它功能生成的数据，是否会实时刷新；页面关闭或系统重启后，数据的初始化设置等都是这类用例。

（5）、业务需求实现是否正确

这类问题往往是由于我们的需求说明欠详细，而编码人员的需求了解程度又较低造成。作为测试人员自然要对需求进行深刻研究，来对软件实现进行把关。这里常见的一些关注点有：

●数据的长度、类型控制是否合理（比如控制纳税人识别号只能为数字，但实际业务中是会有字母出现的）；

●业务逻辑控制是否合理（比如某数据项不提供修改，但实际业务中该数据项经常会需要改动）；

●提供的实现方式是否合理（比如只在某一页面提供某数据的获取功能，但根据业务划分有些人员不能操作此页面，却必须要能看到该数据）；

●所做的数据控制是否合理（比如必须在 A 功能中新增数据，然后才能在 B 功能中操作，但实际业务中有可能会出现相反操作）；

●所做的数据控制是否完整（如授权的方式有普通按月、有买断、有按数量控制，那么当同一企业尝试同时存在以上几种授权方式时，系统是否能有必要的控制）；

●还有其它一些操作细节上的满足（如业务上需要批量操作的数据有否提供批量操作功能、导入失败的结果文件是否能修改后直接再导入等）。

对于不满足的需求，经开发组长、需求经理等确认不作修改的，就要作为软件的缺限或限制在测试报告中进行说明民。

2、功能切面隐含测试项用例设计：

（1）、数据完整性的测试

当某数据被其它功能引用；或当前功能要引用其它来源的数据，就会涉及到数据完整性的测试。最常见的如被引用的数据删除了，或关键字被修改了，引 用的数据会否出错；两个途径进入的数据会否冲突或重复；此外还有因为相关的几个功能由不同人员编码，从而导致彼此的控制不一致，如 A 功能进入的数据在可允 许的**情况下，到 B 功能中引用会否异常（最常见如用户名录入时允许长度 10，但引用到某个单子填写时允许长度是 8，此时就会异常了）。

（2）、后台的特殊处理

是指某功能除了表面所见以外的程序处理。比如订单录入，表面所见的就是订单的保存，但后台还会有重复数据的判断、非法数据的处理、业务逻辑上冲 突情况的处理以及其它种种根据需求设计所特有的处理。又比如备份功能，在备份前可能有数据的清空、备份目录的清空、备份目标是否存在的校验、备份文件重复 时的处理等等。类似这些在分析设计中就未必会写全了，还是要测试人员多花心思去思考挖掘。

（3）、功能业务之间的关联与转换

相关联的几个功能之间数据的传递，会否产生影响。比如新增录入的某种特殊字符，要查询时会引起查询 SQL 语句异常；又如某下载文件名中存在中文 等字符，下载时由于编码问题导致乱码的出现；再有报表填写时到小数点后四位，生成报文时会不会被忽略成两位了等等。象这种问题，通常只能是在每个功能设计 用例时，尽量保证用例中的数据能涉及到允许范围的各种情况，即充分运用等价类划分+边界值的方法设计出各种“稀奇古怪”的数据，并需验证这些数据从头流到 尾，都还是能保持其正确性，而不仅仅是在当前功能中正确。

（4）、从设计实现发掘测试点

这个就是我们测试中最难捉的 BUG 了，它往往是由编码人员自己在编码时创造出来的，连设计人员都不会知道。

比如内部管理系统中，正常的产品，其类别通常是 2 位数字；如果是模块，其类别就以产品代码来取代。这时如何来判断该产品是模块呢？最保险的当然 是校验其产品类别字段的值能否在产品表中找到；也有比较简单的方法就是直接判断类别代码大于 2 位还是小于等于 2 位。此时若能确切知道采用的是哪种实现方 法，就可以直接找到其漏洞所在。比如采用后一种方法，当产品类别长度变化时，明显系统会出错。那么即使确认该实现方式不改，测试人员也应将其作为限制写入 测试报告，。让大家知道这个产品类别长度是不能随意变化的。

而让人郁闷的是，类似这样的实现，有太多的编码人员都是随性处理的，它们细而隐蔽，在系统数据正常情况下根本不会被发现；而在漫漫的软件使用道 路中，由于需求变更等原因对原有一些设计做维护变化，这种问题就会突然暴发出来让人措不及防。所以要杜绝这类漏洞，除了测试人员要做土拨鼠，不停地对软件 各功能的实现细节进行挖掘外，也要多给编码人员灌输完美实现的理念，多用复杂但抗压性高的代码，来替代简单但依赖性强的代码。

（5）、并发操作时的测试

即两个或多个用户同时操作同一功能时，会否引起数据的混乱。通常在 C/S 结构下，如果有同时操作的可能，是需要作此测试的；而在 B/S 结构下由 于其特殊性，此问题通常难以解决。除非就是某用户一旦使用过某功能后，在一定时间内锁定不允许再用，但这也会带来实际应用中的不便，所以除非是特别核心的 数据，一般我们也不会去做此控制，当然对于可能出现的并发冲突也就作为系统的限制进行遗留了。

3、特定切面用例设计

所谓特定切面，其实就是从另一个角度切割出来的用例面，所以具体的用例撰写方式其实与功能切面是一致的。

4、隐含切面用例设计

隐含切面分以下几种情况：

（1）、无界面的后台功能

对这类测试项，需要通过参数设置、代码调用等方式来实现测试，但具体的测试设计其实与普通功能测试并无二致。这里要注意，因为测试时往往前台、 后台是分开来分别进行的，而实际运行时两者很可能是交集的，所以测试时要多注意后台功能的执行与前台的一些功能执行会否产生冲突？比如后台有个文件搬运的 服务，那有没有可能在前台文件生成过程中，后台执行文件搬运了？若有可能就要注意会否出现问题了。

（2）、与业务流相关的测试

这类测试用例的设计，就要从完整业务角度来设计数据了。从理论上来讲，应该要将各个功能可能出现的各种数据排列组合到一起，按业务流程逐一进行 测试。但实际上我们不可能去做全覆盖。所以设计这类用例时，最好有一张草稿，将所有相关功能按业务流程逐一列示，然后再将每个功能可能出现的特定数据一一 标上，最后将图中最可能出现的、最可能出错的、最核心的数据取出来，分别组合成一个个完整的业务数据用例，来进行测试。这样就可以按清晰的思路，找出最实 用、最有效的测试数据。

（3）、其它测试类型

这一类的测试通常都有其特定的方法。如要测可靠性就准备大量数据不停地执行；要测安全性就考虑数据的加密、数据的传输、数据的破坏；恢复性一般 从网络、电源方面着手；配置安装则根据系统可支持的配置，搭建相应环境进行功能验证，此处的验证也要掌握技巧，即要多测试那些涉及到：数据库读写、磁盘文 件读写、文件上传下载、文件加解密、数据统计、图表展现、打印等方面的功能。

*针对**测试数据**的设计，提出的方法。*

每一个测试思路最终都要转化成具体的数据才能来执行。关于测试数据设计的方法也不外乎那几种，就不再赘述了。此处单就一些经常易犯的错误，提出一些注意点，作为用例数据设计时的参考：

1、尽量避免可能出现歧义测试结果的数据：即你设计的数据必须能唯一正确地反映出你所希望测试的结果。比如一组测试数据，有可能得到结果 A 或结果 B，此时单用此数据来测试预期结果为 A 的用例，那明显就产生了歧义。

2、对于不便具体列示的数据，则必须详细描述其各项特性：有时我们在设计用例时为节约时间，不一定要到具体的一个数值，这也是允许的，但前提是 你必须要详细描述清楚你要测试的数据特性。比如数据库字段限长 20，要测试超长数据时，可以描述为：尝试输入长度为 21 位的半角英文字符；尝试输入长度为 19 位的半角英文字符，然后切换到中文全角再输入一位全角字符等。千万不能写成：尝试输入超长字符，因为这只能是测试方案，作为方案是可以这样写，但到用 例阶段，必须要是具体的、明确的、可操作的。

3、测试数据的设计必须有明确目的性：即测试数据是从测试方案衍生而来的。如上例测试方案是测超长字符输入控制，所以测试数据就要根据具体字段 长度来录入超长数据，如果一味录入长 15 位、长 16 位的数据那就没意义了。好的测试数据是可以同时针对多个测试方案的，此时可以在用例边注明一下该数据的 测试目的，因为随着时间推移，对着具体的数据你也许会忘了它到底是测什么的，而这对你最后总结测试，查验测试覆盖率是非常不利的，所以随时记下你的思路想 法吧，好记性不如烂笔头。

4、测试数据可省略描述：测试数据描述以能让人看懂为准则。所以写用例时当碰到连续几个用例，仅某几个关键数据值改动，其余均是一样的情况下， 不必每个用例都要重复描述所有数据，可以在第一个用例描述完整之后，其余用例中仅列示不同的数据，并标明其余数据同上第 X 个用例，即可。这样测试时仍能复 原测试数据，且该用例的测试目的一眼就明，增加了用例的清晰性。

总结：盲目追求快速提高覆盖率其实意义不大，但下次迭代的时候，仍可以从这三个层面：测试用例的切面设计、详细测试用例设计、测试数据设计出发去提高测试用例的覆盖度。