

# 软件测试-测试计划

来源	来自于 Javatpoint 网站
翻译	Andrew (火龙果软件)
说明	本文描述了测试策略、目标、测试时间表、所需资源 (人力资源、软件和硬件)、测试估算和测试可交付成果, 希望对您的学习有所帮助。



MBSE with 火龙果

## 目录

测试计划 .....	1
一、 测试计划的类型 .....	4
1. 主测试计划 .....	4
2. 阶段测试计划 .....	4
3. 具体测试计划 .....	5
二、 如何编写测试计划 .....	5
三、 测试计划组件或属性 .....	5
1. 测试方法 .....	8
2. 方法 .....	9
3. 假设 .....	11
4. 风险 .....	11
5. 缓解计划或应急计划 .....	11
6. 角色与责任 .....	12
7. 附表 .....	<b>错误!未定义书签。</b>
8. 缺陷跟踪 .....	16
9. 测试环境 .....	17
10. 进入和退出标准 .....	19
11. 测试自动化 .....	21
12. 工作量估算 .....	23
13. 测试可交付成果 .....	23

14. 模板 .....	29
四、 测试计划指南 .....	33
五、 测试计划的重要性 .....	33

测试计划是描述软件测试领域和活动的详细文档。它描述了测试策略、目标、测试时间表、所需资源（人力资源、软件和硬件）、测试估算和测试可交付成果。

测试计划是每个软件测试的基础。这是确保按适当顺序提供所有计划活动清单的最关键活动。

测试计划是一个模板，用于将软件测试活动作为由测试经理完全监视和控制的已定义过程进行。测试计划由测试主管（60%）、测试经理（20%）和测试工程师（20%）准备。

## 一、 测试计划的类型

测试计划有三种类型

- 主测试计划
- 阶段测试计划
- 测试类型特定的测试计划

### 1. 主测试计划

主测试计划是一种具有多个测试级别的测试计划。它包括一个完整的测试策略。

### 2. 阶段测试计划

阶段测试计划是一种测试计划，用于解决测试策略的任何一个阶段。例如，工具列表、测试用例列表等。

### 3. 具体测试计划

针对安全测试、负载测试、性能测试等主要类型的测试设计的特定测试计划。换句话说，为非功能测试设计的特定测试计划。

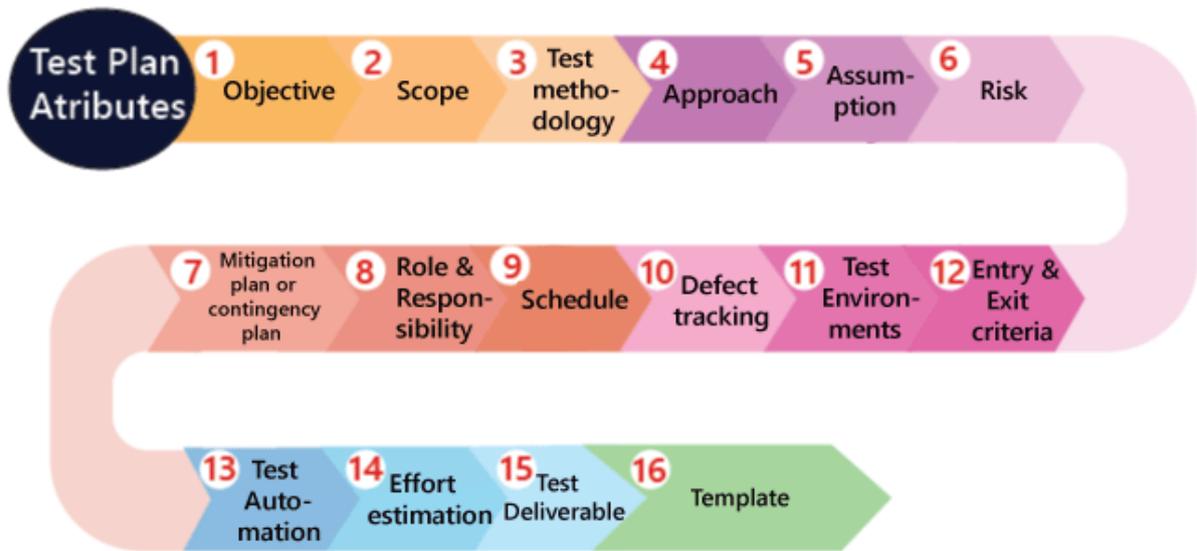
## 二、 如何编写测试计划

制定测试计划是测试管理过程中最关键的任务。根据 IEEE 829，按照以下七个步骤准备测试计划。

- 首先，分析产品结构。
- 设计测试策略。
- 定义所有测试目标。
- 定义测试区域。
- 定义所有可用资源。
- 以适当的方式安排所有活动。
- 确定所有测试可交付成果。

## 三、 测试计划组件或属性

测试计划由各个部分组成，这有助于我们得出整个测试活动。



**目标：**它由有关模块、功能、测试数据等的信息组成，这些信息表明应用程序的目的是指应用程序的行为、目标等。

**范围：**它包含需要使用相应应用程序进行测试的信息。范围可进一步分为两部分：

- 在范围内
- 超出范围

**在范围内：**这些是需要严格（详细）测试的模块。

**范围：**这些是模块，不需要严格测试。

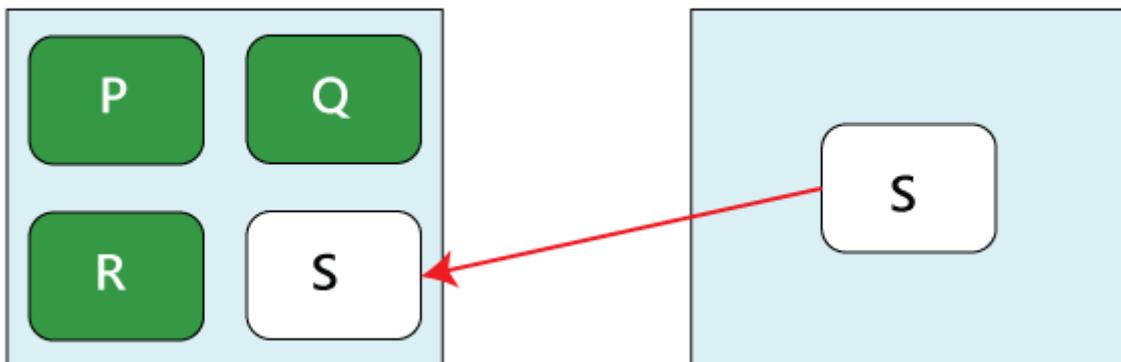
**例如，**假设我们有一个要测试的 Gmail 应用程序，其中要测试的功能（例如撰写邮件，已发送邮件，收件箱，草稿）和未测试的功能（例如“帮助”）等，这意味着在计划阶段，我们将根据产品中给出的时间限制来决定是否必须检查哪些功能。

现在**我们如何决定哪些功能不被测试？**

我们可以在以下方面决定不测试哪个功能：

- 正如我们在上面看到的，**帮助**功能不会被测试，因为它是由技术作家编写和开发的，并由另一位专业作家审查。
- 假设我们有一个具有 **P、Q、R 和 S** 功能的应用程序，需要根据需求进行开发。但在这里，**S** 功能已经被其他公司设计和使用的。因此，开发团队将从该公司购买 **S**，并与 **P、Q 和 R** 等附加功能集成。

现在，我们不会对 **S** 特性执行功能测试，因为它已经被实时使用。但是我们将在 **P、Q、R 和 S** 功能之间进行集成测试和系统测试，因为新功能可能无法与 **S** 功能一起正常工作，如下图所示：



- 假设在产品的第一个版本中，已经开发的元素，例如 **P、Q、R、S、T、U、V、W.....X、Y、Z**。现在，客户将在第二版本中提供改进产品的新功能的要求，新功能是 **A1、B2、C3、D4 和 E5**。

之后，我们将测试计划期间的范围写为

### 范围

烟雾测试	功能测试	集成测试
------	------	------

## 要测试的功能

A1、B2、C3、D4、E5 (新功能)

P, Q, R, S, T

## 不测试的功能

W.....X, Y, Z

因此, 我们将首先检查新功能, 然后再继续使用旧特征, 因为添加新特征后可能会受到影响, 这意味着它也会影响影响区域, 因此我们将对 P、Q、R...、T 特征进行一轮回归测试。

## 1. 测试方法

它包含有关在应用程序上执行不同类型的测试 (如功能测试、集成测试和系统测试等) 的信息。在这种情况下, 我们将决定哪种类型的测试; 我们将根据应用程序要求执行各种功能。在这里, 我们还应该定义我们将在测试方法中使用什么样的测试, 以便每个人, 如管理层、开发团队和测试团队都可以轻松理解, 因为测试术语不是标准的。

**例如,** 对于 **Adobe Photoshop** 等独立应用程序, 我们将执行以下类型的测试:

冒烟测试 → 功能测试 → 集成测试 → 系统测试 → 临时测试 → 兼容性测试 → 回归测试 → 全球化测试 → 可访问性测试 → 可用性测试 → 可靠性测试 → 恢复测试 → 安装或卸载测试

假设我们必须测试 <https://www.jeevansathi.com/> 应用程序, 因此我们将执行以下类型的测试:

系统测试	临时测试	兼容性测试
回归测试	全球化测试	辅助功能测试
可用性测试	性能测试	

## 2. 方法

此属性用于描述执行测试时的应用程序流，并供将来参考。

我们可以借助以下几个方面来了解应用程序的流程：

- **通过编写高级方案**
- **通过编写流图**

### 4.1 通过编写高级方案

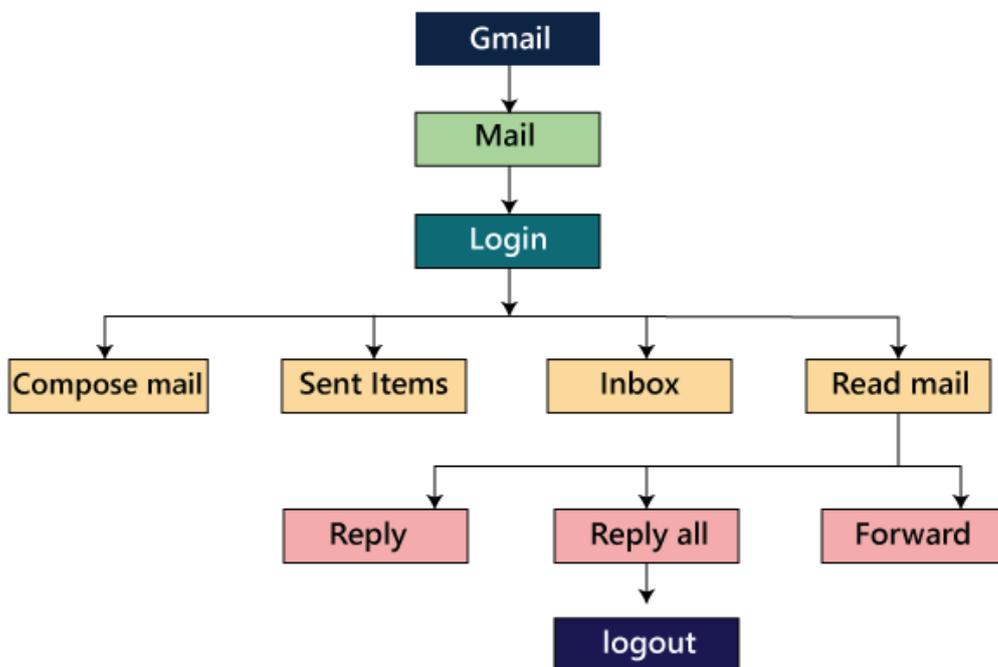
**例如**，假设我们正在测试 **Gmail** 应用程序：

- 登录到 Gmail-发送电子邮件并检查它是否在“已发送邮件”页面中
- 登录 ....
- .....
- .....

我们写这篇文章是为了描述测试产品时必须采取的方法，并且只针对我们将编写高级场景的关键功能。在这里，我们不会专注于涵盖所有场景，因为可以由特定的测试工程师决定哪些功能必须测试或不测试。

#### 4.2 通过编写流程图

编写流程图是因为编写高级方案是花费时间的过程，如下图所示：



我们正在创建流程图，有以下优点，例如：

- 覆盖面简单
- 合并容易

该方法可分为以下两部分：

- 自上而下的方法

- 自下而上的方法

### 3. 假设

它包含有关在测试过程中可能发生的问题或问题的信息，当我们编写测试计划时，将做出有保证的假设，例如资源和技术等。

### 4. 风险

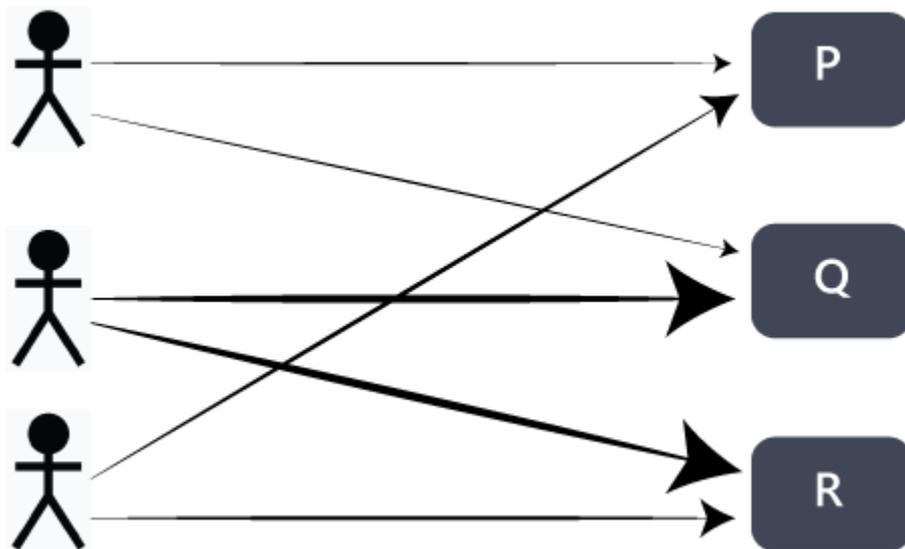
这些是我们在当前版本中测试应用程序时需要面对的挑战，如果假设失败，则涉及风险。

例如，对应用程序的影响，发布日期将推迟。

### 5. 缓解计划或应急计划

这是一个准备克服风险或问题的备用计划。

让我们一起看一个假设、风险和应急计划的例子，因为它们是相互关联的。



在任何产品中，我们将做出的**假设**是，有 3 名测试工程师都将在那里，直到产品完成，并且他们每个人都被分配了不同的模块，例如 P、Q 和 R。在这种特殊情况下，如果测试工程师在项目中途离开项目，则**可能存在风险**。

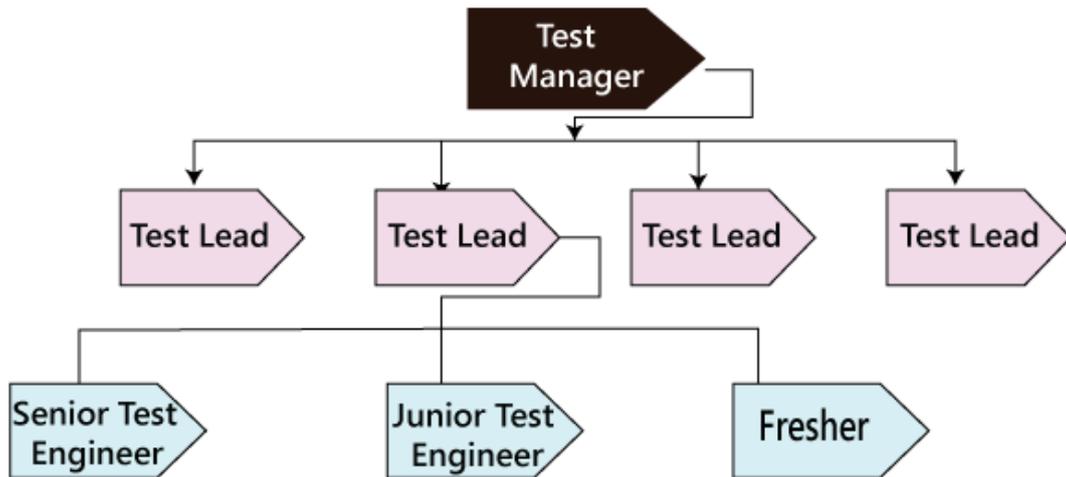
因此，将为每个功能分配**应急计划**的主要所有者和从属所有者。因此，如果一位测试工程师离开，下属所有者将接管该特定功能，并帮助新的测试工程师，以便他/她能够理解他们分配的模块。

假设、风险和缓解或应急计划始终对产品本身是精确的。各种类型的风险如下：

- 客户视角
- 资源方法
- 技术观点

## 6. 角色与责任

它定义了需要由整个测试团队执行的完整任务。当一个大型项目到来时，测试**经理**是编写测试计划的人。如果有 3-4 个小项目，则测试经理会将每个项目分配给每个测试主管。然后，测试负责人为项目编写测试计划，并为其分配测试计划。



让我们看一个示例，我们将了解测试经理、测试主管和测试工程师的角色和职责。

**角色：测试经理**

**姓名：Ryan**

**责任：**

- 准备（撰写和审查）测试计划
- 与开发团队进行会议
- 与测试团队进行会议
- 与客户进行会议
- 每月举行一次站立会议
- 签署发行说明
- 处理升级和问题

**角色：测试主管**

**姓名：Harvey**

**责任:**

- 准备（撰写和审查）测试计划
- 每天召开站立会议
- 审查和批准测试用例
- 准备 RTM 和报告
- 分配模块
- 处理时间表

**角色: 测试工程师 1、测试工程师 2 和测试工程师 3**

**姓名: Louis, Jessica, Donna**

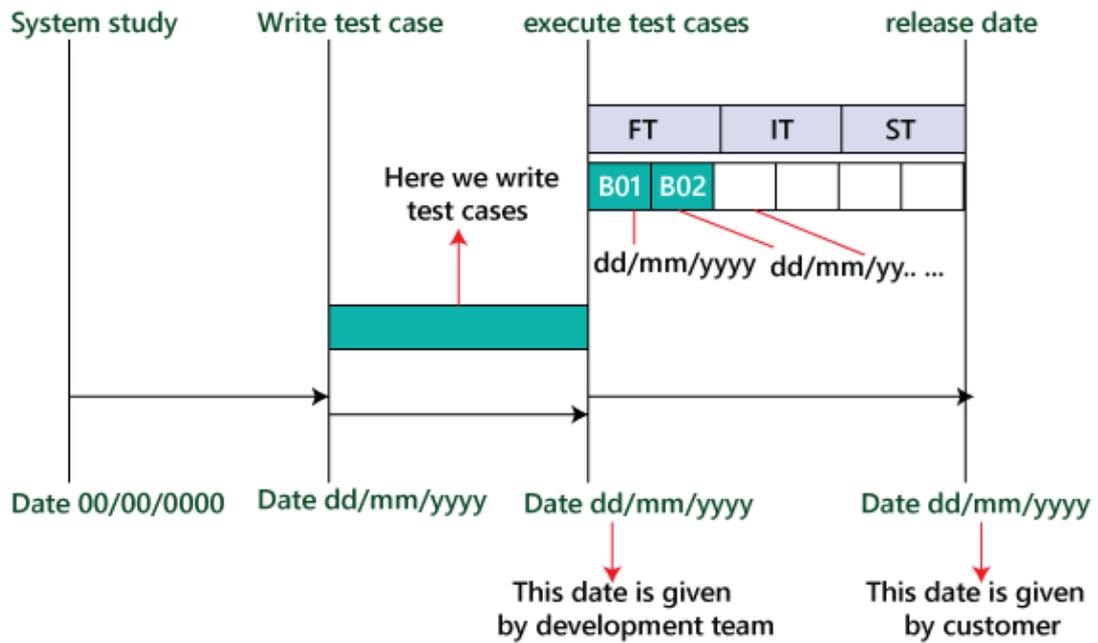
**分配模块: M1、M2 和 M3**

**责任:**

- 编写、查看和执行由测试用例和测试场景组成的测试文档
- 阅读、审查、理解和分析需求
- 编写应用程序的流程
- 执行测试用例
- 各个模块的 RTM
- 缺陷跟踪
- 准备测试执行报告并将其传达给测试主管。

## 7. 时间表

它用于解释工作的时间，需要完成哪些工作，或者此属性涵盖每个测试活动应该何时开始和结束？并且还提到了特定日期的每个测试活动的确切数据。



因此，正如我们在下图中看到的那样，对于特定活动，将有一个开始日期和结束日期；对于特定版本的每个测试，都会有指定的日期。

### 例如

- 编写测试用例
- 执行过程

## 8. 缺陷跟踪

它通常是在工具的帮助下完成的，因为我们无法手动跟踪每个错误的状态。我们还评论我们如何传达在测试过程中发现的错误并将其发送回开发团队以及开发团队将如何回复。在这里，我们还提到了高、中、低等错误的优先级。

以下是缺陷跟踪的各个方面：

- **追踪 bug 的技术**

.....

.....

.....

- **Bug 跟踪工具**

我们可以评论工具的名称，我们将使用它来跟踪错误。一些最常用的 bug 跟踪工具是 Jira, Bugzilla, Mantis 和 Trac 等。

- **安全**

严重程度可能如下：

被阻止

.....

..... (在测试计划中用示例解释)

**例如**，模块中会出现缺陷;我们不能进一步测试其他模块，因为如果错误被阻止，我们可以继续检查其他模块。

**关键...**

..... (在测试计划中用示例解释)

在这种情况下，缺陷会影响业务。

### 专业

...

(在测试计划中用示例解释)

.....

..... (在测试计划中用示例解释)

这些缺陷会影响应用程序的外观和感觉。

### ○ 优先级

#### 高-P1

.....

#### 中-P2

.....

#### 低 P3

.....

.....

#### 小四

因此，根据 bug 的高、中、低优先级，我们将它分为 P1、P2、P3 和 P4。

## 9. 测试环境

这些是我们测试应用程序的环境，这里我们有两种类型的环境，即**软件配置**和**硬件配置**。

**软件配置**意味着有关不同**操作系统**（如 Windows, Linux, UNIX 和 Mac）以及各种**浏览器**（如 Google Chrome, Firefox, Opera, Internet Explorer 等）的详细信息。

**硬件配置**意味着有关不同大小的 **RAM、ROM 和处理器**的信息。

例如

- **本软件**包括以下内容：

### 服务器

<b>操作系统</b>	Linux 目录
<b>网络服务器</b>	Apache Tomcat
<b>应用服务器</b>	Websphere
<b>数据库服务器</b>	Oracle or MS-SQL Server

注意：上述服务器是测试团队用于测试应用程序的服务。

### 客户

<b>操作系统</b>	Window XP, Vista 7
<b>浏览器</b>	Mozilla Firefox、Google Chrome、Internet Explorer、Internet Explorer 7 和 Internet Explorer 8

注意：以上详细信息提供了测试团队将在其中测试应用程序的各种操作系统和浏览器。

- **硬件**包括以下内容：

**服务器**：Sun StarCat 1500

测试团队可以使用此特定服务器来测试其应用程序。

客户:

它具有以下配置，如下所示:

处理器	Intal2GHz
RAM	2GB
....	....

注意: 它将提供测试工程师 (即测试团队) 的系统配置。

- 软件的**安装过程**...

.....

.....

开发团队将提供如何安装软件的配置。如果开发团队尚未提供该过程，那么我们将在测试计划中将其编写为基于任务的开发 (TBD) 。

## 10. 进入和退出标准

这是必要条件，在开始和停止测试过程之前需要满足。

### 10.1 进入标准

进入标准包含以下条件:

- 应完成白盒测试
- 了解和分析需求并准备测试文档或测试文档准备就绪。

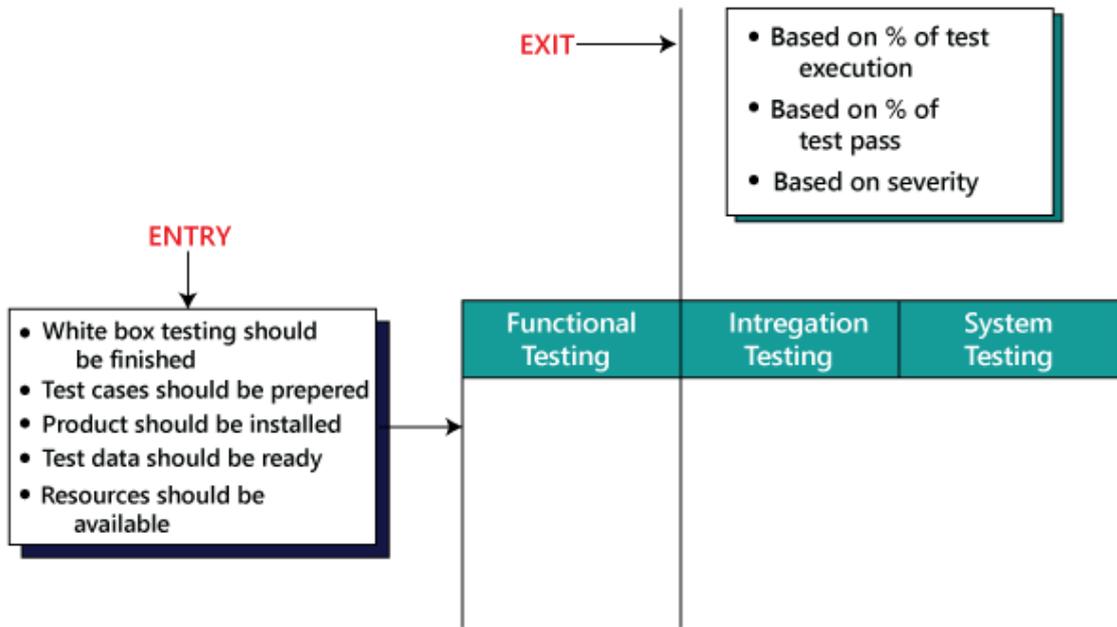
- 测试数据应准备就绪。
- 必须准备生成或应用程序
- 模块或功能需要分配给不同的测试工程师。
- 必须准备好必要的资源。

## 10.2 退出标准

退出条件包含以下条件：

- 当执行所有测试用例时。
- 大多数测试用例必须**通过**。
- 取决于错误的严重性，这意味着不得有任何阻止程序或主要错误，而存在一些小错误。

在我们开始执行功能测试之前，应遵循上述所有**进入标准**。在我们执行功能测试之后，在我们进行集成测试之前，应遵循功能测试的退出标准，因为退出条件的百分比是由与开发和测试经理的会议决定的，因为他们的协作可以达到百分比。但是，如果不遵循功能测试的退出标准，那么我们就无法进一步进行集成测试。



在这里，**根据错误的严重性**，测试团队将决定在下一阶段进一步进行。

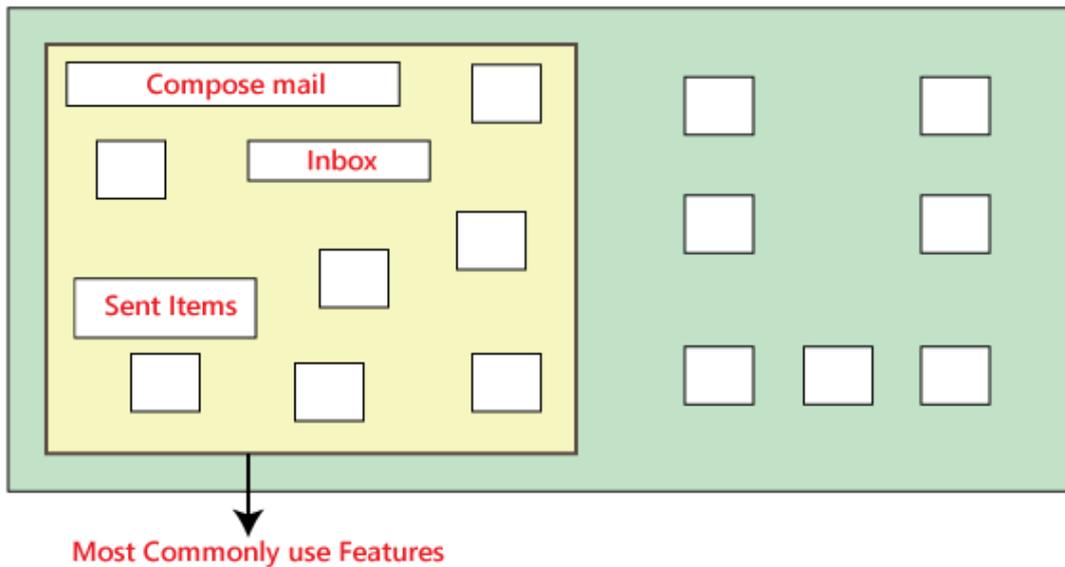
## 11. 测试自动化

在此，我们将决定以下内容：

- 哪些功能必须自动化而不是自动化？
- 我们将在哪个自动化框架上使用哪种测试自动化工具？

我们仅在首次发布后自动执行测试用例。

这里出现了一个问题，**我们将在什么基础上决定必须测试哪些功能？**



在上图中，我们可以看到最常用的功能需要一次又一次地测试。假设我们必须检查 Gmail 应用程序，其中的基本功能是**撰写邮件，已发送邮件和收件箱**。因此，我们将测试这些功能，因为在执行手动测试时，它需要更多的时间，并且它也变成了一项单调的工作。

现在，**我们如何决定哪些功能不会被测试？**

假设 Gmail 应用程序的**帮助功能**没有一次又一次地测试，因为这些功能不经常使用，所以我们不需要经常检查它。

但是，**如果某些功能不稳定并且有很多错误，这意味着我们不会测试这些功能**，因为在进行手动测试时必须一次又一次地测试它。

如果**有一个功能必须经常测试**，但我们预计该功能的需求会发生变化，所以我们不检查它，因为与更改自动化测试脚本相比，更改手动测试用例更舒适。

## 12. 工作量估算

在这方面，我们将计划每个团队成员所做的工作。

## 13. 测试可交付成果

这些是测试团队输出的文件，我们将其与产品一起交给客户。它包括以下内容：

- 测试计划
- 测试用例
- 测试脚本
- RTM (需求可追溯性矩阵)
- 缺陷报告
- 测试执行报告
- 图表和指标
- 发行说明

### 13.1 图形和指标

在此，我们将讨论我们将发送的图形类型，我们还将提供每个图形的示例。

如我们所见，我们有五个不同的图表来显示测试过程的各个方面。

**图 1：**在这里，我们将显示每个模块中已经识别了多少缺陷以及修复了多少缺陷。

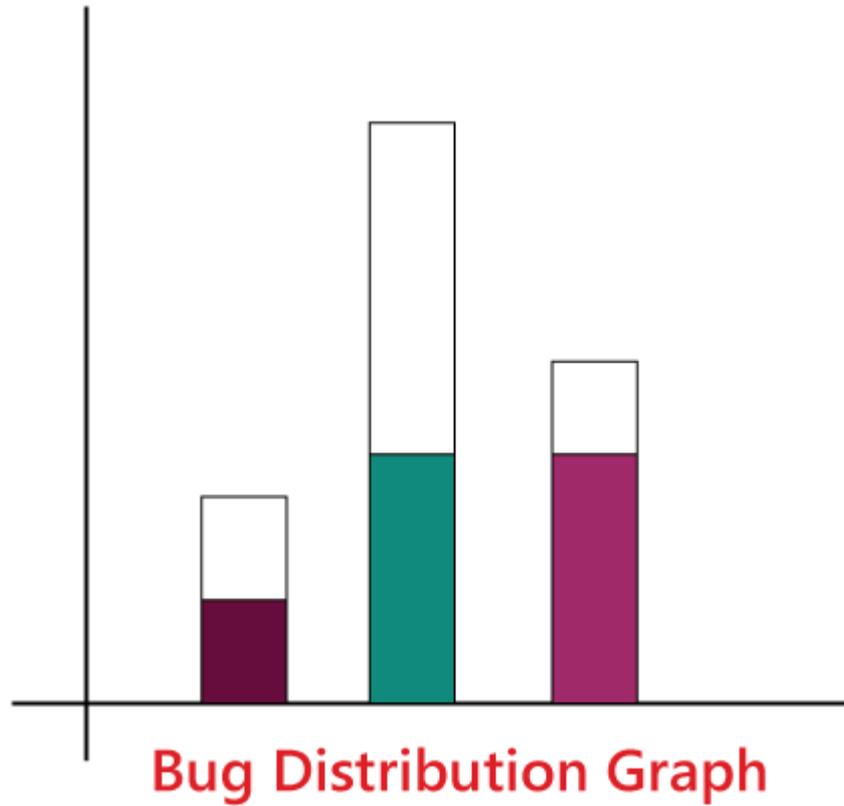


图 2: 图 1 显示了每个模块已识别出多少个关键缺陷、主要缺陷和次要缺陷, 以及已修复其各自模块的缺陷数量。

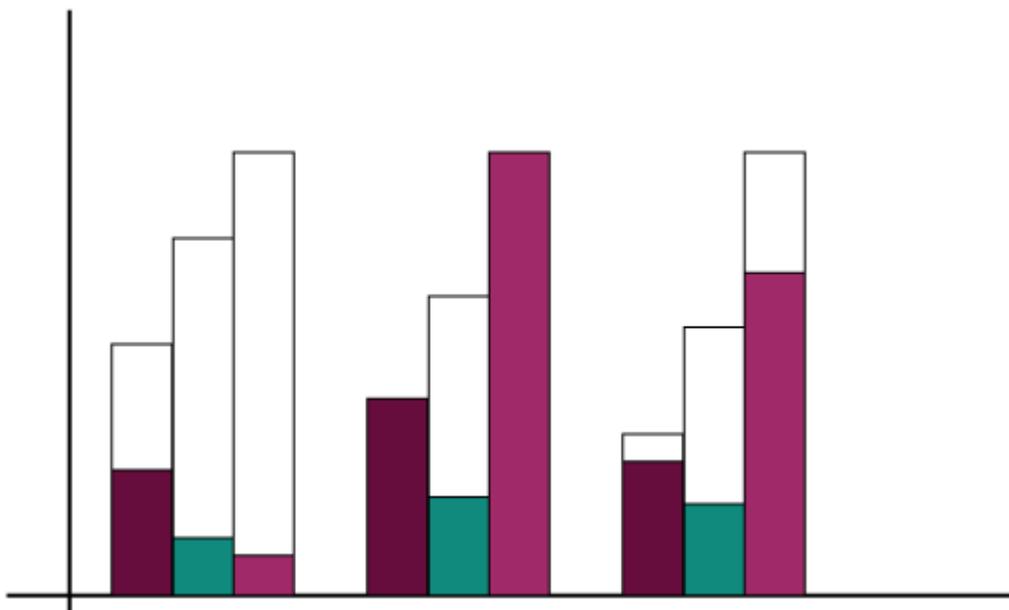


图 3: 在这个特定的图中, 我们表示构建智能图, 这意味着在每个构建中, 每个模块都已识别并修复了多少缺陷。根据该模块, 我们已经确定了错误。我们将添加 R 以显示 P 和 Q 中的缺陷数, 我们还添加 S 以显示 P、Q 和 R 中的缺陷。

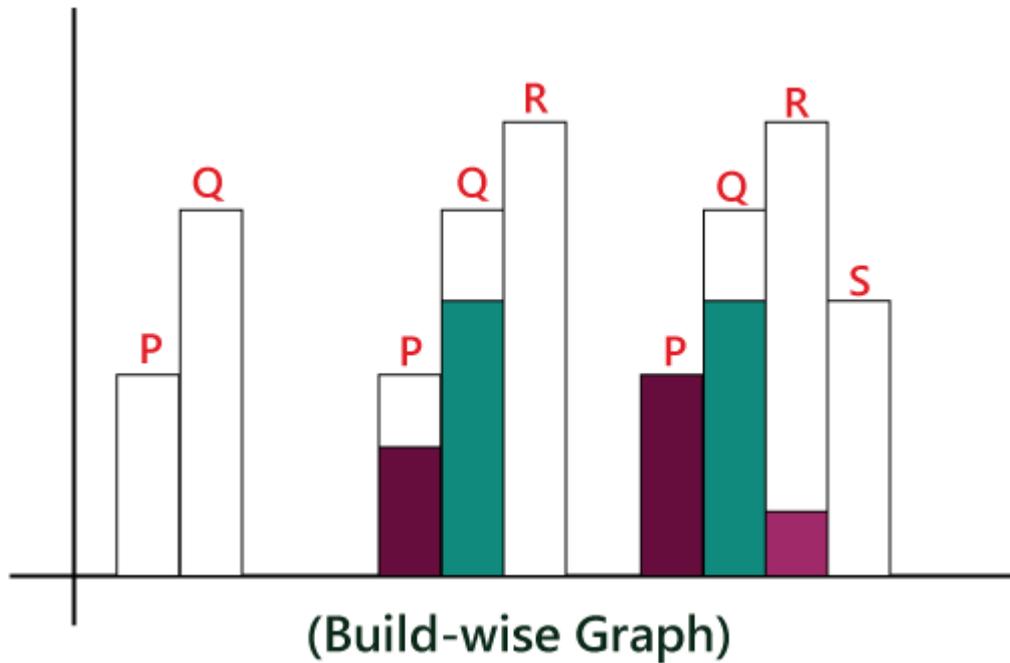


图 4: 测试负责人将设计每月创建的 Bug 趋势分析图, 并将其发送给管理层。这就像在产品结束时完成的预测一样。在这里, 我们还可以对错误修复进行评分, 因为我们可以观察到弧在下图中具有上升趋势。

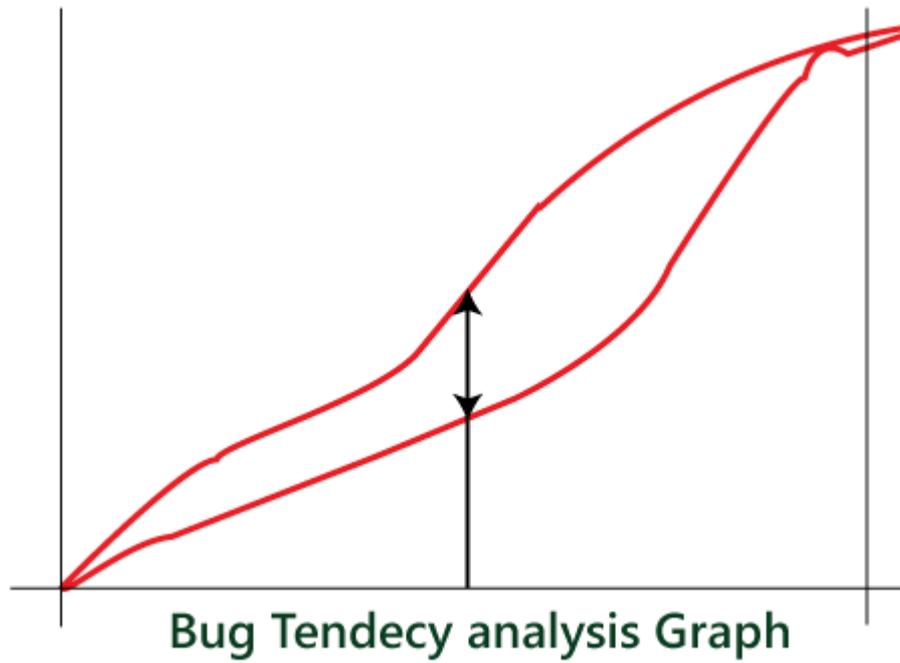
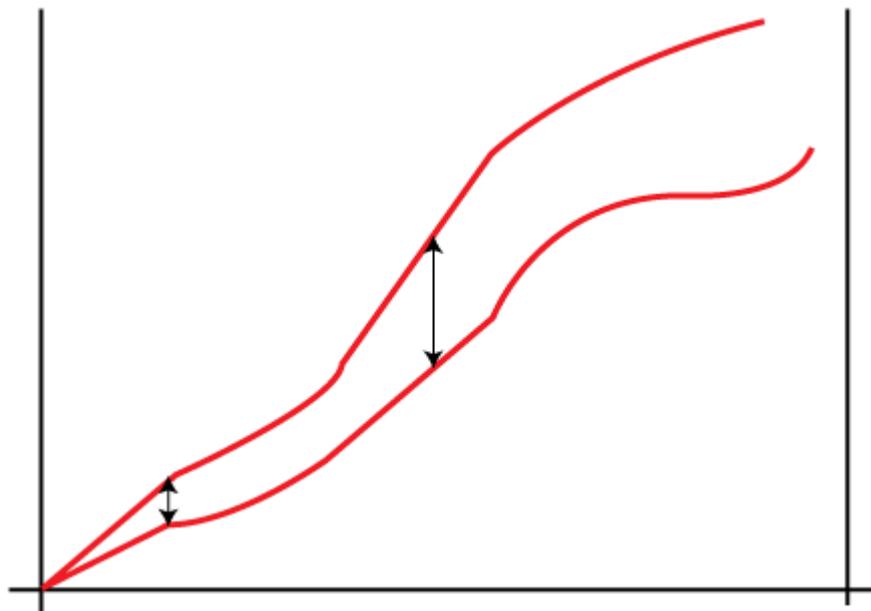


图 5：测试管理器设计了这种类型的图形。此图旨在了解 bug 评估中的差距和已发生的实际 bug，并且此图还有助于改进将来对 bug 的评估。



指标

Module Name	Critical		Major		Minor	
	Found	Fixed	Found	Fixed	Found	Fixed
Purchase	50	46	60	20	80	10
Sales	..	...	...	...	...	...
Asset Survey	...	...	...	...	...	...

如上所述，我们创建了 bug 分布图，如图 1 所示，借助上述数据，我们还将设计指标。

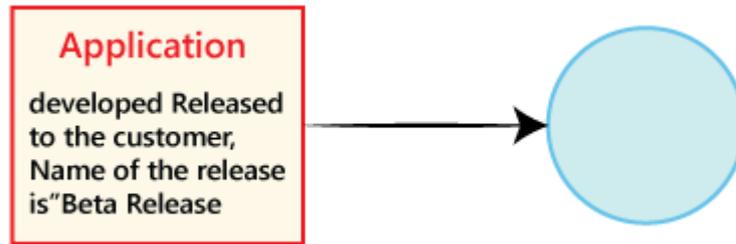
例如

Test Engineer Names	Critical		Major		Minor	
	Found	Fixed	Found	Fixed	Found	Fixed
John	50	46	60	20	80	10
James	..	...	...	...	...	...
Sophia	...	...	...	...	...	...

在上图中，我们保留了特定项目中所有测试工程师的记录，以及已识别和修复的缺陷数量。我们还可以将这些数据用于将来的分析。当有新要求出现时，我们可以根据他们之前根据上述指标发现的缺陷数量来决定谁提供具有挑战性的功能进行测试。我们将处于更好的情况，知道谁可以很好地处理有问题的特征并找到最大数量的缺陷。

**发行说明：**它是在产品发布期间准备并由测试经理签名的文档。

在下图中，我们可以看到最终产品已开发并部署到客户，最新版本名称为 **Beta**。



**发行说明**包括以下内容：

- 用户手册。
- 待处理和未解决的缺陷列表。
- 已添加、已修改和已删除要素的列表。
- 测试产品的平台（操作系统、硬件、浏览器）列表。
- 未在其中测试产品的平台。
- 当前版本中修复的 bug 列表，以及先前版本中已修复的 bug 列表。
- 安装过程
- 软件的版本

**例如**

假设 **Beta** 是发布第一个版本 **Alpha** 之后应用程序的第二个版本。在第一个版本中发现的一些缺陷，这些缺陷已在后续版本中修复。在这里，我们还将指出从 alpha 版本到 beta 版本的新添加、修改和删除功能的列表。



## 14. 模板

这部分包含将在产品中使用的文档的所有模板，所有测试工程师将在项目中仅使用这些模板来维护产品的一致性。在这里，我们在整个测试过程中使用了不同类型的模板，例如：

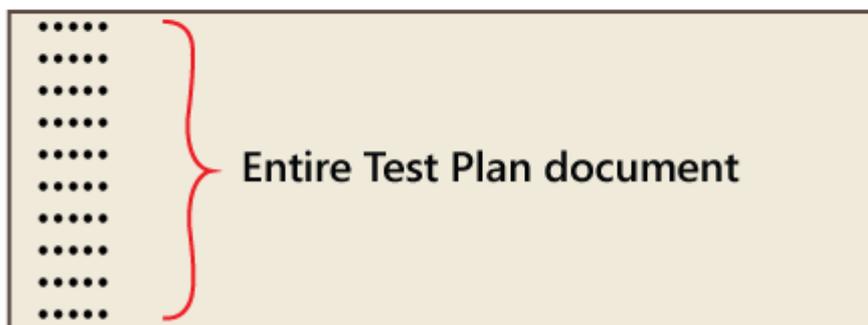
- 测试用例模板
- 测试用例审查模板
- RTM 模板
- 错误报告模板
- 测试执行报告

让我们看一个测试计划文档示例

### Testplan

Version	Author	Reviewed By	Approved By	Comments	Approval date
1	...	...	Name of manager	Version 1.0 is developed	dd/mm/yyyy
1.1	...	..	..	Version 1.1 is developed. PQR feature is added	dd/mm/yyyy
.....	.....	.....	.....	.....	.....
.....	.....	.....	.....	.....	.....

TABLE OF CONTENTS	
• Objective	pg 1
• Scope	pg 2
• Approach	pg 3
• ..	...
• ..	...
• ..	...
• ..	...



## REFERENCES

- 1) Customer requirement specification(CRS)
- 2) Software requirement specification(SRS)
- 3) Functional specification(FS)
- 4) Design Documents

....  
....  
....

在第 1 页中，我们主要仅填写“版本”、“作者”、“注释”和“审阅者”字段，在经理批准后，我们将在“批准截止日期”和“批准日期”字段中提及详细信息。

大多数情况下，测试计划由测试经理批准，测试工程师只对其进行审查。当新功能到来时，我们将修改测试计划并在**版本**字段中进行必要的修改，然后再次发送以供经理进一步审查、更新和批准。每当发生任何更改时，都必须更新测试计划。在第 20 页上，**参考文献**指定了我们将用于编写测试计划文档的所有文档的详细信息。

**注意：**

### 谁编写测试计划？

- 测试主管→60%
- 测试经理→20%
- 测试工程师→20%

因此，从上面我们可以看到，在 60%的产品中，测试计划是由测试负责人编写的。

### 谁审查测试计划？

- 测试主管

- 测试经理
- 测试工程师
- 客户
- 开发团队

测试工程师根据其模块角度审查测试计划，测试经理根据客户意见审查测试计划。

### **谁批准测试计划？**

- 客户
- 测试经理

### **谁编写测试用例？**

- 测试主管
- 测试工程师

### **谁审查测试用例？**

- 测试工程师
- 测试主管
- 客户
- 开发团队

### **谁批准测试用例？**

- 测试经理

- 测试主管
- 客户

## 四、 测试计划指南

- 取消测试计划
- 避免重叠和冗余。
- 如果您认为您不需要上面已经提到的部分，请删除该部分并继续。
- 要具体。例如，当您 will 将软件系统指定为测试环境的一部分时，请提及软件版本，而不仅仅是名称。
- 避免冗长的段落。
- 尽可能使用列表和表格。
- 需要时更新计划。
- 请勿使用过时和未使用的文档。

## 五、 测试计划的重要性

- 测试计划为我们的思维提供了方向。这就像一本规则手册，必须遵守。
- 测试计划有助于确定必要的工作，以验证被测软件应用程序的质量。
- 测试计划可帮助这些人了解与开发人员、业务经理、客户等外部相关的测试详细信息。
- 测试计划中记录了测试计划、测试策略、测试范围等重要方面，以便管理团队可以对其进行审查并将其重用于其他类似项目。