Testability:

指软件可以很容易地通过测试证明其错误。具体来说，可测试性是指假定软件至少有一个错误，它在下一次测试执行中失败的概率。如果系统中出现故障，那么我们希望它在测试期间尽快失败。

要使系统可以正常测试，必须能够控制每个组件的输入（并可能操纵其内部状态），然后观察其输出（以及可能的内部状态）。

Testability Summary:

确保系统易于测试，可以在测试成本和系统可靠性方面获得回报。

控制和观察系统状态是一类可测试性策略。

复杂系统很难测试，因为它们的计算发生的状态空间很大，并且由于系统元件之间的互连数量较多。 因此，保持系统简单是另一类支持可测试性的策略。

了解：可测试性一般场景。

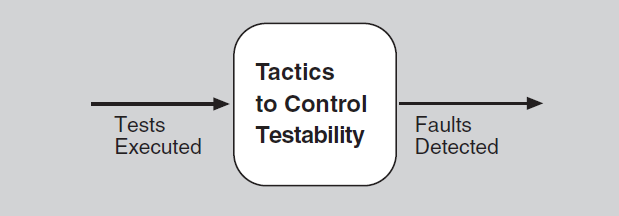
|  |  |
| --- | --- |
| **Portion of Scenario** | **Possible Values** |
| **Source** | 单元测试人员，集成测试人员，系统测试人员，验收测试人员，最终用户，手动运行测试或使用自动化测试工具 |
| **Stimulus** | 由于完成编码增量（例如类，层或服务），执行一组测试; 完成子系统的集成; 完整的系统实施; 或者将系统交付给客户。 |
| **Environment** | 设计时间，开发时间，编译时间，集成时间，部署时间，运行时间 |
| **Artifacts** | 正在测试的系统部分 |
| **Response** | One or more of the following:  执行测试套件并捕获结果; 捕获导致故障的活动; 控制和监控系统的状态 |
| **Response  Measure** | One or more of the following:  努力寻找故障或类故障，努力实现状态空间覆盖的给定百分比; 下一次测试揭示出错的概率;进行测试的时间; 努力发现故障; 在测试中最长依赖关系链的长度; 准备测试环境的时间长度; 减少风险暴露（规模（损失）\*概率（损失）） |

Example: 单元测试器在开发期间完成代码单元并执行测试序列，其结果被捕。并且在测试的3小时内提供85％的路径覆盖。

掌握：可测试性战术。可测试性设计清单。

Goal of testability tactics:

在软件开发增量完成时允许更容易的测试。架构师可以采取的任何措施来降低测试的高成本，以达到显着的benefit。可测试性有两类策略：系统增加可控性和可观察性；限制系统设计的复杂性。



Testability tactics:



1. **Control and Observe System State**
2. **Build-in monitor:** 组件可以基于内部状态实施策略以支持测试过程；组件可以通过接口维护状态，性能负载，容量，安全性等；当监视状态激活时记录事件。
3. **Specialized Interfaces**：通过测试工具或正常执行，以控制或捕获组件的变量值。
4. **Record/Playback**：捕获跨接口的信息，并将其作为进一步测试的输入。
5. **Localize State Storage**：为了使测试在任意状态下，启动处系统、子系统或模块。如果将状态存储在单一的位置，则最convenient。
6. **Abstract Data Sources**：抽象接口可以更轻松地替换测试数据。
7. **Sandbox(**沙盒**)**：将系统与现实世界隔离开来，使实验不受担心实验的后果的约束。Ex: executable Assertions.
8. **Limit Complexity**
9. **Limit Structural Complexity**
10. **Limit Non-determinism(非确定性)**

Design for testability

1. 测试驱动开发
2. 隔离能力

没有看看就好系列呢。。。。（no case study ）