# Requirement

**令人惊讶的是，很少有研究关注需求可测性问题，尽管它也被IEEE软件需求规范830 1998[1]推荐在实践时需要考虑。例如，七个选定的指标在可读性文献[2]中是众所周知的，并且是：(1) Average Grade Level, (2) Flesch Kincaid Reading Ease, (3) Flesch Kincaid Grade Level, (4) Gunning Fog Score, (5) Simple Measure of Gobbledygook (SMOG Index), (6) Coleman Liau Index, 和(7) Automated Readability Index。 然后，他们使用机器学习和统计分析开发了需求可测试性模型，并评估了是否可以学习需求可测试性模型并将其应用于其他需求。**

**1.1** Measuring design testability of a UML class diagram

它提出了一种测量需求质量以预测可测试性的方法。 作者选择了一组文档可读性指标，用于表征需求的可理解性和质量，并评估它们是否表征了可测试的需求。

具体的，该文献描述了测量UML类图的设计可测试性方法（针对面向对象系统的UML类图）。

为了指导测试任务，并且类图具有继承和动态绑定关系，因此类图可作为检测和掌握广泛的隐式控制依赖关系的良好基础。

**1.2** Requirement Decomposition and Testability in Development of Safety-Critical Automotive Components

它侧重于安全关键汽车软件背景下的需求可测试性。 作者将需求Ri定义为逻辑表达式Li：<Object X>shall<Action Y>[apply to]<Subject Z>。需求被映射到对象X上，对象X将动作Y应用到对象Z上。 Ri的可测试性是Ri的一个属性，它可以验证该逻辑表达式Li。 他们建议，为了实现可测试性，需求必须包括Object，Action和Subject以及Object，Action和Subject必须在系统内可识别并存在。 在这些条件有效的情况下，可以验证该需求，并且需求是可测试的。

* 1. Sizing Software with Testable Requirements

标题为“具有可测试需求的大小调整软件”是另一项有趣的研究，由一名从业者撰写。 该工作提出了可测试需求作为一种新的软件测量范例。 通过在工业环境中应用它，该论文指出，可测试需求是一种直观，灵活的措施，是向用户和管理层传达问题的有用工具。

**1.4** Testable Requirements for Offshore Outsourcing

它考虑了离岸外包（全球软件工程）背景下的需求可测试性。 作者认为，离岸外包需要一套可测试的需求，这是承包商和客户之间法律协议质量保证的核心。 他们为该环境中可测试需求的模型提供了语义。

* 1. Testable Use Cases in the Abstract State Machine Language

它通过使用抽象状态机语言（ASML）指定它们来检查可测试性用例，ASML是Microsoft Research开发的可执行规范语言。 然后，作者通过描述如何从ASML中的用例规范生成测试用例和测试oracles来演示该方法的优点

* 1. Using the Testability Analysis Methodology for the Validation of AIRBUS Systems

它提供了Airbus系统验证需求可测试性的案例研究。 实验表明，可测试性分析可以简化系统验证活动。

# Design

**设计一个便于测试的系统非常重要。 可测试性设计（DfT）通常在SDLC（Software Development Life Cycle）的设计和编码阶段进行。**

**2.1** Improving Web Application Testing Using Testability Measures

糟糕的设计（例如，陷入设计反模式的陷阱）可能会对测试阶段的系统或其单元的可测试性产生负面影响。

**2.2** Quantitative Analysis of Testability Anti-patterns on Open Source Java Applications

可测试性反模式也是另一项工作的焦点，它将可测试性反模式定义为“已知使测试变得困难和/或增加要执行的测试用例数量的设计方法”。

**2.3** A Study of the Relationship between System Testability and Modularity

它的作者使用system-dynamics建模来研究模块性和可测试性之间的关系。 作者关注模块化的成本和收益相关的可测试性。 使用来自system-dynamics的模拟数据，作者发现采用适当的模块化和适当的测试架构可以提高系统测试的效率和有效性。

**2.4** A Study on Design for Testability in Component-Based Embedded Software

它在欧洲电信行业的两家大型公司的基于组件的嵌入式软件环境中报告了一项关于DfT的研究。 根据访谈和技术文档，讨论了不同DfT方法（侧重于可观察性和可控性）的差异和益处。 该论文提出了几个关于DfT的建议，例如，（1）特别是在嵌入式系统的情况下，良好的主机测试环境能够实现高可测试性。 当此环境尽可能与目标系统匹配时，可以进行有效的主机测试; （2）在系统中包含测试支持功能可以进行更有效的测试，包括分析长时间运行的测试和部署的系统。

**2.5** Design for Testability

由从业者撰写的论文为DfT提供了实用的建议。 根据作者的经验，它提供了用于测试各种软件产品的可测试性功能的示例，包括事件记录，断言，诊断，资源监视，测试点，故障注入钩子和支持软件安装和配置的功能。 它还探讨了可测性问题如何影响GUI测试自动化以及测试人员在缺乏可测试性时必须提供哪些规定。 本文最后讨论了测试人员如何与产品开发人员合作，以获得内置于软件产品中的重要可测试性功能。

**2.6** Design for Testability in Object-Oriented Systems

这是一篇有影响力的1994年论文题为“面向对象系统的可测试性设计”（撰写本文时引用次数为378次），该论文的专家罗伯特·宾德尔（Robert Binder）。 作者介绍了DfT的重要概念和不同架构。 它最后说：“几乎所有用于实现高可测试性的技术和技术都已确立，但需要财务承诺，计划和有意识的努力”。

**2.7** On Building Testable Software Components

它专注于构建可测试的软件组件，并引入了可测试（Java）“bean”的概念，并提出了一种基于可测试体系结构和良好定义的内置接口构建可测试bean的方法。

# Measurement

# Improving

# Other

# References

**[1]** IEEE, IEEE Recommended Practice for Software Requirements Specifications 830-1998, Accessed: 25 November 2017. Last Last <http://standards.ieee.org/findstds/standard/830-1998.html>.

**[2]** K. Kirk, Writing for Readability, American Society for Training and Development, 2010.