IEEE软件工程标准术语表将需求定义为：

1. 用户所需的解决某个问题或达到某个目标所要具备的条件或能力。
2. 系统或系统组件为符合合同、标准、规范或其他正式文档而必须满足的条件或必须具备的能力。
3. 上述第一项或第二项中定义的条件和能力的文档表述。

RUP将需求定义为：

需求描述了系统必须满足的情况或提供的能力，可以直接来自客户需要，也可以来自合同、标准、规范或其他有正规约束力的文档。

1、需求捕获、需求分析、系统分析

从软件过程全局看，需求分析是承上启下的阶段，上乘愿景，下接设计。

需求捕获是获取知识的过程，采集者必须理解用户所从事的工作，并且了解用户和客户希望软件系统在哪些方面帮助他们。

需求分析是挖掘和整理知识的过程，在已掌握知识的基础上进行。致力于搞清楚软件系统要“做什么”。

系统分析是针对系统所要面临的问题，搜集相关资料，以了解产生问题的原因所在，进而提出解决问题的方法与可行的逻辑方案，以满足系统的需求，实现预定的目标。已经开始涉及“怎么做”的问题。

- 需求捕获、需求分析以及系统分析是相互伴随、交叉进行的。
- 需求分析与系统分析不能混淆。

2、需求成果

2.1、需求捕获

需求采集卡，包括项目、时间、地点、
需求类型、需求编号、用户关注度
描述、背景或原因、相关材料、
提出者/受访者、记录者/采访者

需 求 采 集 卡					
项目		时间		地点	
需求类型		需求编号		用户关注度	
描述					

描述			
背景或原因			
相关材料			
提出者/受访者		记录者/采访者	

2.2、需求分析

应交付一份明确、规范的需求定义--《软件需求规格说明书》（SRS），《SRS》精确地阐述了一个软件系统必须提供的功能、必须达到的质量属性指标，以及必须遵守的约束。其中，当前最常用的是用例（Use Case）技术，用例图刻画系统能为外部用户或系统提供的服务，用例规约刻画系统应提供的具体行为。SRS通常包括如下内容：

1. 前言
    a. 目的
    b. 范围
    c. 定义、缩写词、略语
    d. 参考资料
2. 需求概述
    a. 用例模型　//此节，归档“用例图”等。
    b. 限制与假设
3. 具体需求
    a. 用例描述　//此节，归档“用例规约”等。
    b. 外部接口需求
        i. 用户接口
        ii. 硬件接口
        iii. 软件接口
        iv. 通信接口
    c. 质量属性需求
        i. 性能
        ii. 易用性
        iii. 安全性
        iv. 可维护性...
    d. 设计和实现约束
        i. 必须遵循的规则

ii. 硬件的限制...

2.3、系统分析

通过结构化分析得到的工作成果是数据流图，面向对象的系统分析方法得到的工作成果主要是分析类图、鲁棒图、序列图-其中分析类图描述设计的静态方面，而鲁棒图和序列图描述设计的动态方面。

关注功能需求、质量、约束等各种需求类别。

3、需求完整性

3.1、二维需求观与ADMEMS矩阵

首先，需求是分层次的。将需求划分为三个层次：

- **组织级需求：包含客户或出资要达到的业务目标、预期投资、工期要求，以及要符合哪些标准、对哪些遗留系统进行整合等约束条件。**
- **用户级需求：用户使用系统来辅助完成哪些工作？对质量有何要求？用户群及所处的使用环境方面有何特殊要求?**
- **开发级需求：开发人员需要实现什么？开发期间、维护期间有何质量考虑？开发团队的哪些情况会反过来影响架构?**

其次，需求还必须从不同方面进行考虑。实践一再表明，忽视质量属性和约束性属性，常常导致架构设计最终失败。

- **功能需求：更多体现各级直接目标要求。**
- **质量属性：运行期质量+开发期质量。**
- **约束需求：业务环境因素+使用环境因素+构建环境因素+技术环境因素。**

ADMEMS矩阵又称“需求层次-需求方面矩阵”，可以作为需求梳理和需求评审的工具。

	广义功能	**质量**	**约束**
组织需求	业务目标	快好省	技术性约束
			标准型约束
			法规性约束
			遗留系统集成
			技术趋势
			分批实施
			竞争因素与竞争对手
用户需求	用户需求	运行期质量	用户群特点
			用户水平
			多国语言
开发需求	行为需求	开发期质量	开发团队技术水平
			开发团队磨合程度
			开发团队分布情况
			开发团队业务知识
			管理：保密要求

			管理：产品规划
			安装
			维护

3.2、功能

如果采用比较正规的方式，可以在《软件需求规格说明书》中将每项功能的用户类、用户输入或系统外激励、系统动作（本项和前一项相当于用例规约的主事件流）、业务规则、例外及相应处理（本项相当于用例规约的备选事件流）、特殊需求或限定、相关功能、注释和说明等内容进行展开说明。

功能名称 描述							
功能编号				功能名称			
创建者		最后更改者		版本		优先级	
用户类							
用户输入或系统外激励				系统动作			
业务规则							
例外及相应处理							
特殊需求或限定							
相关功能							
注释和说明							

3.3、质量

推荐将软件质量属性划分为运行期质量属性和开发期质量属性两大类：

运行期质量属性：性能（Performance），安全性（Security），易用性（Usabiliity），持续可用性（Availability），可伸缩性（Scalability），互操作性（Interoperability），可靠性（Reliability），鲁棒性（Robustness）。

开发期质量属性：易理解性（Understandability），可扩展性（Extensibility），可重用性（Reusablility），可测试性（Testability），可维护性（Maintainability），可移植性（Portability）。

1. **性能（Performance），是指软件系统及时提供相应服务的能力。**
    a. 吞吐量通过单位时间处理的交易数来度量；
    b. 速度往往通过平均响应时间来度量；

c. 而持续高效性是指保持高速处理速度的能力。

2. 安全性（Security），指软件系统同时兼顾向合法用户提供服务，以及阻止非授权使用的能力。

3. 易用性（Usabiliity），指软件系统易于使用的程度。

4. 持续可用性（Availability），指系统长时间无故障运行的能力。

5. 可伸缩性（Scalability），指当用户数和数据量增加时，软件系统维持高服务质量的能力。

6. 互操作性（Interoperability），指本软件系统与其他系统交换数据和相互调用服务的难易程度。

7. 可靠性（Reliability），软件系统在一定的时间内无故障运行的能力。

8. 鲁棒性（Robustness），指软件系统在以下情况下仍能够正常运行的能力：用户进行了非法操作；相连的软硬件系统发生了故障，以及其他非正常情况。

9. 易理解性（Understandability），尤指设计被开发人员理解的难易程度。

10. 可扩展性（Extensibility），指为适应新需求或需求的变化为软件增加功能的能力，也成为灵活性。

11. 可重用性（Reusablility），指重用软件系统或其一部分能力的难易程度。

12. 可测试性（Testability），指对软件测试以证明其满足需求规约的难易程度。

13. 可维护性（Maintainability），指为了达到下列三个目的之一而定位修改点并实施修改的难易程度：修改Bug；增加功能；提高质量属性。

14. 可移植性（Portability），指将软件系统从一个运行环境转移到另一个不同的运行环境的难易程度。

3.4、约束

约束需求 = 业务环境因素 + 使用环境因素 + 构建环境因素 + 技术环境因素。

第一，业务环境因素（来自客户或出资方的约束性需求）

- 架构师必须充分考虑客户对上线时间的要求、预算限制、以及继承需要等非功能需求。
- 客户所处的业务领域为哪些？有什么业务规则和业务限制？
- 是否需要关注相应的法律法规、专利限制？
- ...

第二，使用环境因素（来自用户的约束性需求）

- 软件提供给何阶层用户？
- 用户的年龄段及使用偏好是哪些？
- 用户是否遍及多个国家？
- 使用期间的环境有电磁干扰、车船移动等因素吗？

- ....

第三，构建环境因素（来自开发者和升级维护人员的约束性需求）

- 开发团队的技术水平如果有限、磨合程度不高、分布在不同城市，会有何影响？
- 开发管理方面、源代码保密方面，是否需要顾及？
- ...

第四， 技术环境因素（业界当前技术环境本身也是约束性需求）

- 技术平台、中间件、编程语言等的流行度、认同度、优缺点等。
- 技术发展的趋势如何？
- ...