利用马克思主义方法分析，软件危机发生，客观上是由软件本身的特点造成的，主观上是软件开发与维护的方法不正确造成的。

**1.现阶段软件设计文档不规范，无法验证、分析、证明，导致产品可能存在错误、不能很好满足用户需求，造成后期的反复修改**

软件开发是将抽象构思具体化、可见化的工业化作业。它包括开发、测试、维护和升级等环节，但难以量化每个步骤的指标。不同阶段的语言和程序员风格差异大，团队化需要高可读性文档。常见的设计文档有可行性分析和项目计划，但国家标准化管理委员会官网上很多文件缺失。

**2.模块内聚性，逻辑内聚性模块的代码，内聚性低，但又无法避免，导致软件产品质量，增加维护难度**

如果一个模块执行多个完全不相关的动作，那么这个模块就有偶然内聚性。当一个模块执行一系列相关的动作，且其中一个动作是作为其他动作选择模块，称其有逻辑内聚性。

对于前者，这些模块降低了产品的可维护性，不能重用，与软件开发工业化、标准化冲突。对于后者，接口部分难于理解，多个动作的代码可能会缠绕在一起，导致严重的维护问题，难以在其他产品中重用这样的模块。各个动作之间没有逻辑关系，却又无法避免

**3.程序代码无法证明其正确性，只能通过测试验证**

计算机科学中存在“停机问题”这个不可解问题。停机问题指出不存在一个算法可以判断任意给定的程序是否会停止运行。换句话说，我们无法编写一个通用的程序来检查其他程序是否总是产生正确的结果。

此外，程序的正确性还取决于输入数据的完整性和正确性。即使程序本身是正确的，如果输入数据存在问题，也可能导致错误的结果。因此，在软件开发过程中，我们需要对程序进行充分的测试，以确保其在各种情况下都能产生正确的结果。

总之，由于停机问题的存在以及输入数据的影响，我们无法证明程序代码总是正确的。相反，我们需要通过测试来验证程序的正确性。

**4.客户的需求是具体并具有浓厚的个性化色彩的、且随时随地在变化,很难用某一种"标准的,通用的"软件就能满足全部客户需求。**

首先，用户对软件产品的质量要求是满足需求、可靠、可操作性好、可维护性强。软件产品是厂家适应客户的模式。是否正确分析客户的需求是着手开发软件的第一步,如果分析不恰当,那么在此基础上研制的软件也就报废了。

另外,客户的需求是动态的,即使依靠着标准文档和各种法律手段，需求的变化也是客观存在着，和不同的需求和程序实现之间也有不少的冲突。

为此，改变软件功能的难度远大于用户改变需求的难度。

**5.基于软件标准化的构件技术和对人工智能的期待**

随着软件规模和复杂度的增加，软件开发流程可以实现模块化。模块化和复用是软件开发工业化的重要特征，提高了软件生产率，缓解了软件危机。构件是构成软件系统的独立可装配的软件单元，能实现特殊需求，并与其他构件通过接口等协同工作。通过一系列标准化的范式步骤实现软件开发的系统化。这种模块化软件系统具有高灵活性和易维护性。

很可惜，软件标准化只是一种裱糊式的手段，针对形式而非内容,不能从根本上解决软件危机。提高软件生产率需要更高层次的自动化设计手段，而非单纯依靠软件标准化。只有依靠一种比编制代码更高层次的开发方式，才能大幅度提高软件生产效率。

这种目前还留在蓝图上的全程自动化设计开发，可能需要希冀于人工智能的发展。目前的人工智能可以读取理解程序员的大纲式指令和对每个细节的简单指令。我们可以预测随着人工智能逐步完善，它们能实现自动化开发、智能优化、预测缺陷并及时修复等，真正解决软件危机。