基于对以上相关工作的研究，我们形成如下重要研究思路：

（1）通过代码依赖识别系统中功能紧密的代码元素。

（2）引入用户反馈来防止直接使用代码依赖可能带来的列表污染问题。

具体而言，我们综合利用了代码依赖和用户反馈以提升信息检索方法所产生候选追踪线索列表的精度。

首先，我们通过使用代码依赖信息，将通过代码依赖交互紧密的代码元素放到同一个代码域（code region）中。我们认为同一代码域中的代码元素在系统功能上紧密相关。然后，%为了衡量代码元素之间的代码依赖紧密程度，我们引入代码依赖紧密度的概念，通过设定紧密度阈值得到代码域。

我们使用代码域作为用户反馈的切入点。对于一个给定需求，每个代码域中只需要取出一个有代表性的代码元素交由用户判断与需求的相关性，用这个结果表示该代码域内所有代码元素与需求的相关性，从而有效减少了方法所需要的用户反馈数量。最终，我们针对域内和域外的代码元素设置了不同的优化策略，从而改善整个候选列表的排序。

为了验证我们方法的有效性和实用性，我们需要高质量的基于日常生活实践的软件可追踪数据集。然而，当前领域内缺乏此类数据集。

当前领域内公开的需求到代码的追踪数据大多是一些缺乏基于日常实践的可追踪数据集。因此，我们整理了由代码托管平台管理的开源软件系统的数据集。

因此，我们整理了一些被广泛应用于日常实践的开源软件系统数据集。

我们的实验系统由一个被领域内广泛用于软件可追踪方法验证的高质量软件系统数据集和三个被广泛用于日常实践的开源软件系统组成。

对于实验系统中由代码托管平台管理的开源软件，我们通过分析其在issue-tracking工具中的软件行为信息，整理出了这些开源系统需求到代码的可追踪数据集。

此外，我们通过运行开源软件系统自带的用于验证自身功能的测试用例，动态捕获到了我们方法中需要的高质量代码依赖信息。实验表明，

我们的方法只需要少量的（约3.5\%）用户反馈信息即可显著优于基线方法。

为了将我们的方法应用到日常实践中，我们实现了软件可追踪生成工具并集成了结合代码依赖和用户反馈的软件可追踪生成方法。

一方面，该工具操作简单、用户学习成本低。该工具类似一个搜索引擎，返回与指定需求具有相关性的代码元素列表。另一方面，在用户验证追踪线索相关性阶段，工具提供大量信息辅助用户高效完成验证。通过使用该工具，用户可以快速得到高质量的需求到代码追踪列表。

综上所述，本文工作概括如下：

基于代码依赖紧密度分析和用户反馈的可追踪性生成方法。我们提出了一种结合了代码紧密度分析和用户反馈的软件可追踪生成方法。一方面通过设置代码紧密度阈值划分代码域，使得功能紧密的代码元素位于同一个代码域中；另一方面，对于给定需求，将各代码域中有代表性的代码元素交由用户判断与该需求相关性，根据用户反馈信息调整相关代码元素对应候选线索的相似度值。

实验数据组织及方法验证。我们用一个被领域内广泛用于可追踪方法验证的高质量数据集和三个被广泛应用于日常实践的开源系统验证了我们方法的有效性和实用性。并且，我们通过对开源软件在issue-tracking工具上的行为信息进行分析整理，组织了其需求到代码的追踪关系。此外，我们通过运行开源系统自带的用于验证系统功能的测试用例得到了我们方法所需的代码依赖。

基于代码依赖和用户反馈的软件可追踪生成工具的设计与实现。为了将我们的方法应用到日常实践中，我们设计并实现软件可追踪生成工具，并集成了我们基于代码依赖紧密度分析和用户反馈的软件可追踪生成方法。

{需求可追踪性，代码域，信息检索，紧密度分析，用户反馈，代码依赖}