第五章　软件可追踪生成工具的设计与实现

在本章中，针对结合代码依赖和用户反馈的软件可追踪生成问题，我们设计了面向用户的工具。该工具集成了我们结合代码依赖紧密度分析和用户反馈的软件可追踪生成方法。我们介绍了软件可追踪生成工具的应用场景，并详细阐述了工具的设计与实现。此外，我们结合一个案例说明工具的使用流程。

5.1　应用场景

软件测试是软件上线之前需要完成的一个重要步骤。如果测试人员发现软件没有正确完成某个需求，此时，测试人员会将与之相关的异常信息进行整理并向软件负责人反馈。由于缺乏需求到代码的追踪矩阵，项目负责人很难快速获取与给定需求存在相关性的所有代码元素。接下来，我们将介绍需求到代码可追踪生成工具如何支持这一需求可追踪查询的过程。

软件可追踪生成工具结合代码依赖紧密度分析和用户反馈两部分信息，对通过信息检索方法形成的候选追踪线索列表进行优化调整，图 **5.1** 中展示了软件可追踪生成辅助工具在软件可追踪建立场景中的作用效果。 接下来我们将简要介绍软件可追踪生成工具给用户（项目负责人）带来的益处：

1. 该工具类似一个搜索引擎（如图 **5.3**左侧窗体所示）根据输入需求返回与需求存在相关性的代码元素列表（如图 **5.5**最右侧窗体），工具界面精简，操作简单。

2. 在用户对少量有代表性追踪线索相关性进行验证的阶段（如图 **5.3**右侧窗体所示），工具提供了大量的额外信息（如图 **5.4**所示）辅助用户完成判断。此举能够有效减少用户付出的时间成本并有效提升用户判断的正确率。

利用软件可追踪生成工具，用户只需判断少量的候选追踪线索相关性。之前 **4.2**实验表明，当有**3.5**　候选追踪线索相关性交由用户验证时，最终得到的候选追踪列表的精度即可明显优于基线方法。在用户验证阶段，工具通过提供代码依赖图拓扑图、特殊单词高亮等多种手段（详见 **5.3**）保证用户能够高效的做出给出正确的判断结果。综上所述，通过该工具用户只需很少的参与即可得到相对高精度的需求到代码的追踪列表。

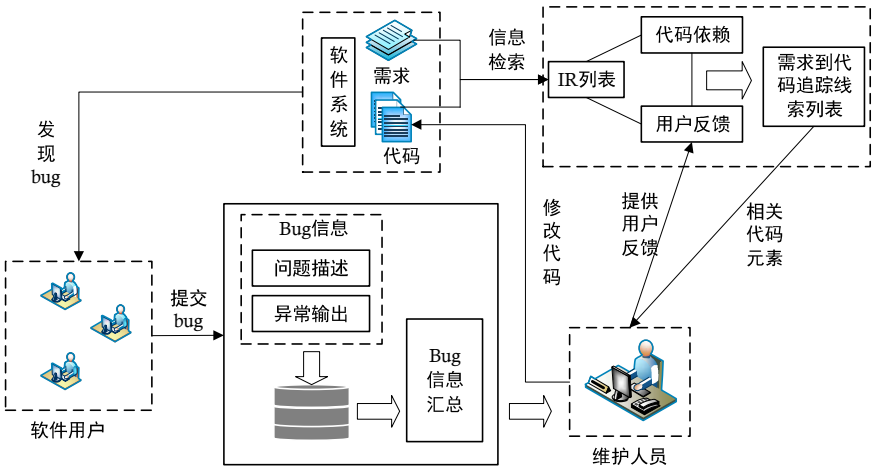

图 **5.1:** 结合用户反馈和代码依赖紧密度分析的软件可追踪生成技术应用场景

5.2　工具的体系结构

该工具的核心是需求到代码可追踪生成部分，图 **5.2** 展示该部分的体系结构，其中包括数据准备、信息检索、代码依赖捕获、用户交互以及候选线索列表优化五个模块。在本小节中，我们将对以上五个模块的设计与实现进行说明。

1. 数据准备模块：软件可追踪生成辅助工具的输入是需求和代码，这里的代码数据包含针对项目的测试集。输出是需求到代码的追踪线索列表。数据准备模块主要是提供数据导入接口。数据导入模块是其它所有模块的基础，在代码依赖捕获模块，工具会通过数据准备模块中导入的测试代码集合得到代码依赖；在用户交互模块，工具会展示需求和代码的文本内容；在检索模块，工具对导入的需求和代码做文本相似度计算并生成候选追踪列表。

2. 信息检索模块：该模块是对数据准备模块中导入的需求和代码数据进行处理。对需求文本进行文本预处理，包括移除停用词、词形还原和词干提取等操作。对于代码文本，首先需要根据命名规则进行分词，然后与需求文本进行同样的文本预处理；接下来基于信息检索技术，计算需求文本和代码文本集合之间的文本相似度。我们实现了　**M**、　**I**与　这三个被广泛使用的信息检索模型，以　**M**（向量空间模型）为例，将需求文本和代码文本用高维向量　、**r**表示，向量中每个维度**w**对应一个单词的权重，权重**w**可用**TF-I**　**F**公式计算，对于　、**r**两个高维向量，两者组成一条候选追踪线索，可利用余弦距离计算向量之间的余弦相似度，我们将其定义为这

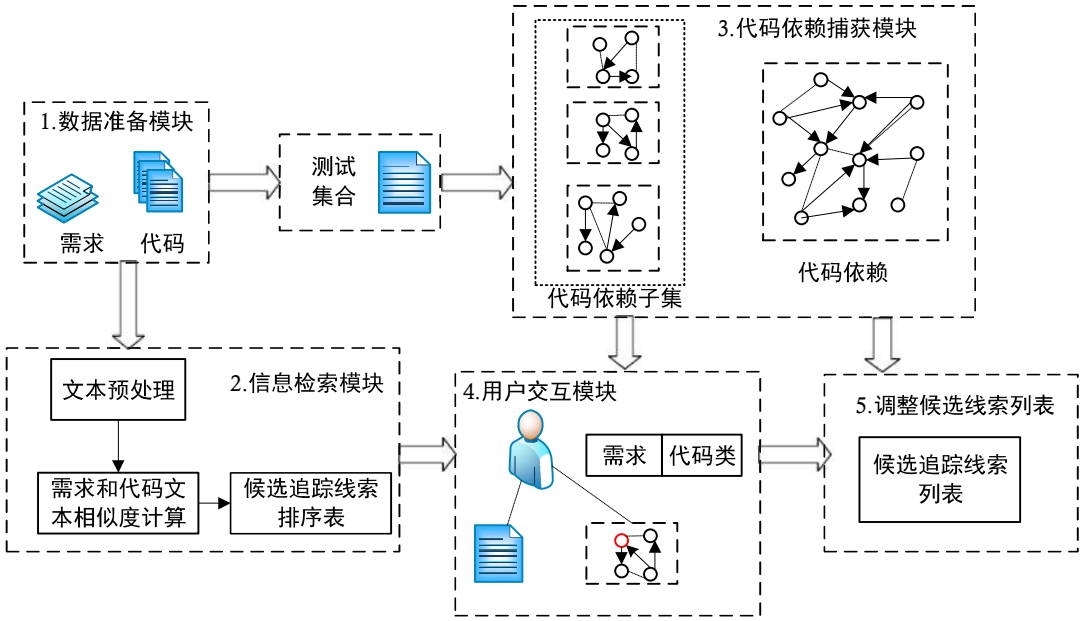

图 **5.2:** 软件可追踪生成辅助工具的系统体系结构图

条追踪线索的相似度值。然后我们根据追踪线索的相似度值，对候选追踪线索按照相似度值从大到小的顺序排序，形成候选追踪列表。

3. 代码依赖捕获模块：该模块主要负责捕获代码依赖关系，这里的代码依赖包括直接依赖（类之间的调用、使用和继承）和数据依赖（类之间的数据共享）。虽然像**T T**（**The Eclipse Test erformance Tools latform**）和 **F**（**ava lug-in Framework**）等工具都能在软件系统运行过程中捕获系统中存在的方法之间的调用依赖。但是根据我们的调研，并没有一个现成的动态分析工具可以捕获方法之间的数据依赖。因此我们基于 **M**提供的接口，注册函数调用数据访问等事件，并在其回调函数中存储方法进入、退出和数据访问记录，最终对该数据进一步处理得到代码依赖关系。根据项目使用的构建工具（**Maven**、**Ant**）我们采用不同的方式运行测试集，并在此过程中通过不同的方式插桩我们的代码依赖捕获工具，得到各测试用例对应的代码依赖子集。然后我们对代码依赖子集进行合并得到代码依赖。

4. 用户交互模块：该模块首先会对代码依赖模块捕获的代码依赖进行紧密度分析，并通过代码依赖紧密度阈值生成代码域。对于每个需求，根据各域内类与需求相似度最大值，对用户为判断域按照相似度值自大到小的顺序重排序。对于排名第一的代码域，取其域内与需求相似度值最大的类交由

用户判断与需求的相关性，如图 **5.3**右侧窗体所示。一方面在对话框左栏用户可以选择相关、不相关、跳过该判断和结束整个判断过程，右栏是当前需求的文本内容。另一方面用户可以点击帮助查看与该类代码依赖关系紧密的其它类，以及它们之间的拓扑结构，如图 **5.4**所示，通过点击类，用户可以查看该类的文本内容，与当前类相关的需求，出现在需求中的单词等多种辅助用户判断信息。工具会根据用户的判断结果调整候选追踪列表，继续向用户推荐下一个需要用户判断的追踪线索或者生成最终结果。

5. 候选追踪列表优化模块：该模块会根据对代码依赖紧密度分析得到的代码域，和在用户交互模块中得到的用户反馈信息对候选追踪列表进行调整。当用户判断给定的候选追踪线索具有相关性时，对于该候选线索中的类，工具会通过不同的方式提升与该类在一个域中的其它类，和在域外但是和该类存在之间或数据依赖的类对应候选追踪线索的相似度值。该过程会迭代多次直到用户交互模块中，用户选择结束判断为止。此时工具会将候选追踪列表数据持久化到磁盘中，方便软件利益相关者使用。

5.3　案例介绍

在本小节，我们将结合一个案例说明工具的使用方式。 对于一个即将上线

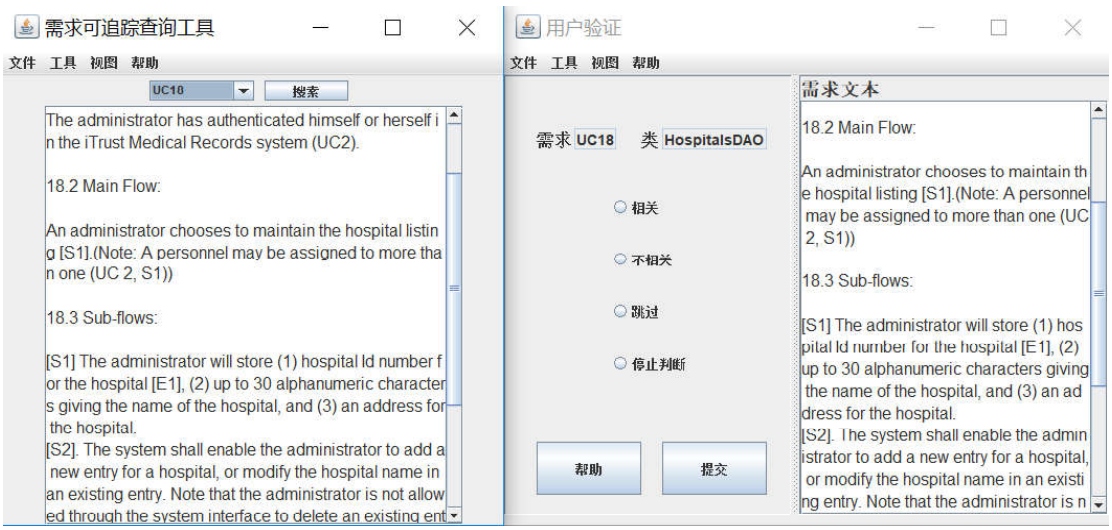

图 **5.3:** 需求可追踪查询

的医疗管理软件**iTrust**，测试组发现该系统的需求**UC18**没有通过测试，测试人员会整理问题相关信息以及可能的异常输出信息等向软件负责人反馈。软件负责人收到该反馈信息后需尽快修复造成该问题的代码段以免造成项目延期。然

而，因为项目组并没有维护需求到代码的追踪矩阵。这使得很难快速的找出与需求**UC18** 具有相关性的代码元素。此时，我们的工具能帮助用户（项目负责人）快速解决这个难题。

首先用户需要导入需求和代码等数据。接下来进入图 **5.3**左侧所示界面，

我们的工具类似一个搜索引擎，向用户返回与指定需求相关的代码元素信息。点击搜索后会进入用户交互界面（图 **5.3** 右侧所示界面），工具选择少量有代表性的追踪线索交由用户验证。本例**Hospitals　AO**是其所在代码域与需求**UC18**相似度最大的类。因此，工具用**Hospitals　AO**与**UC18**的相关性来代表代码域中其它类与**UC18**的相关性。为了使用户能够高效的给出正确的验证结果。工具提供大量信息辅助用户完成验证，用户只需通过 帮助按钮即可获得大量辅助信息（如图 **5.4**所示）。

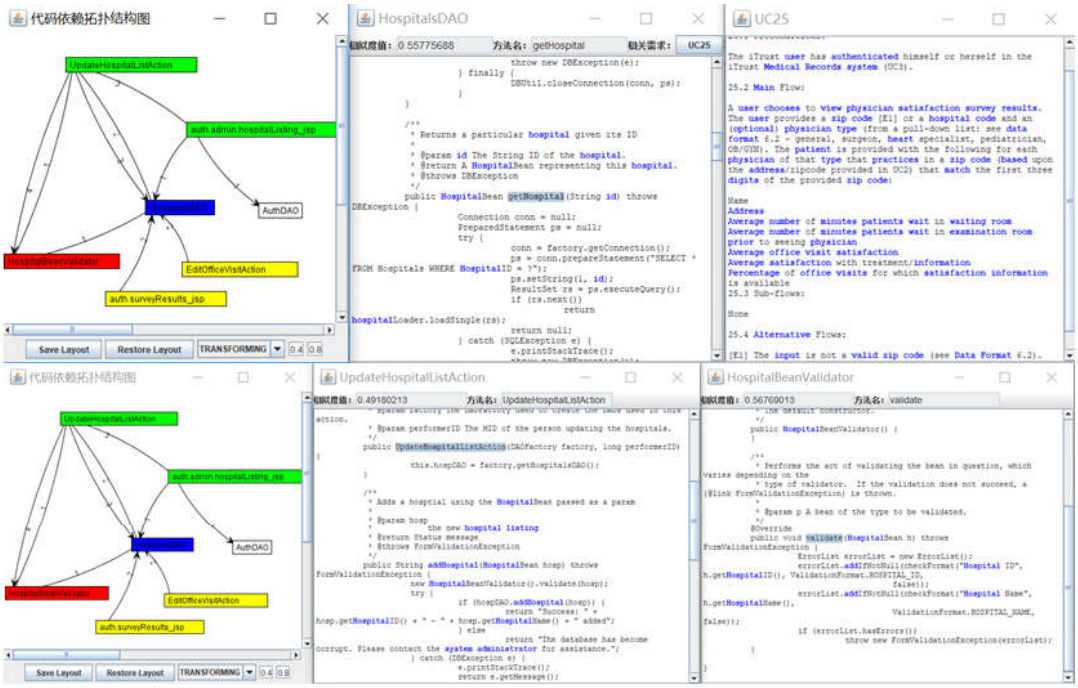

图 **5.4:** 辅助用户验证信息

图 **5.4**最左侧为代码依赖拓扑图，蓝色矩形代表的代码元素为用户需要判断的类，绿色代码元素为与当前类位于同一个代码域中的类。黄色矩形代表与当前类存在代码依赖关系的域外类。红色矩形代表与当前需求相似度最大的类。白色为与域内其它类存在代码依赖的域外类。用户通过点击这些矩形框可以很方便的查看这些类与需求的相似度值，当前查看的方法的方法名等信息。在需求中出现过的单词会在类文本中高亮显示（如图 **5.4**所示）。工具也会向用户提供已经用户验证的与当前类存在相关性的需求信息（如图 **5.4**右侧的窗体所示，

Hospitals　AO和**UC25**具有相关性），同时用户也能很方便的查看该需求文本内容（如图 **5.4**右侧窗体所示）。用户通过阅读**UC18**的文本内容（如图 **5.3** 所示）知该需求描述了管理员维护医院列表信息的功能。

一方面，从图**5.4**查看与**Hospitals　AO** 位于同一代码域的**UpdateHospital　istAction** 和与**UC18** 文本相似度最大的**HospitalBean　alidator** 的文本内容，里面出现大量 **hospital**、**administrator** 等在**UC18** 中出现的单词。因此，用户很容易得出**UpdateHospital　istAction** 大概率与**UC18**具有相关。另一方面，根据代码命名规则，**Hospitals　AO**是负责与数据库交互。根据该类与周边类的代码依赖关系，用户很容易得出该类可能负责添加、查询医院信息等操作。与**Hospitals　AO** 存在相关性**UC25** 所描述的功能为，根据用户提供医院编码和医生类型等数据，返回患者对医生的满意度信息。这验证了用户对**Hospital　AO** 负责与数据库交互读取与与医院相关数据表的猜测。综合与**Hospitals　AO** 存在代码依赖的类和与其存在相关性的需求信息，用户很容易得出该类与**UC18** 存在相关性。接下来进入图 **5.5** 所示界面，用户给出自己的判定结果。此时工具会更新候选列表排序并可能推荐一条新的候选追踪线索交由用户判断，如图 **5.5**中间窗体所示，

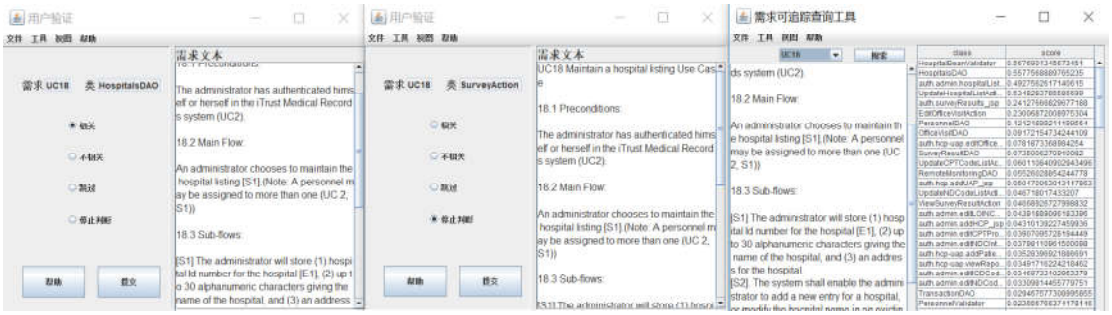

图 **5.5:** 代码依赖结构展示界面效果图

直到用户终止判断，工具给出图 **5.5**右侧所示结果。在排序表中越靠前的类，与需求存在相关性的概率越大。通常情况下，软件负责人通过**review**排在列表顶部的几个类，即可完成对问题代码段的修复任务。

5.4　本章小结

在本章中，我们介绍了需求到代码可追踪生成辅助工具的设计实现与应用场景，并辅以一个具体案例解释了工具的使用流程。需求到代码可追踪生成辅助工具能减少用户参与并且提高追踪列表的准确率和完全率。该工具可以帮助用户快速得到与指定需求存在相关性的代码元素。