[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 9/44 (2006.01)
G06F 17/00 (2006.01)

[12] 发明专利申请公布说明书

[21] 申请号 200880017725.4

[43] 公开日 2010 年 3 月 31 日

[11] 公开号 CN 101689111A

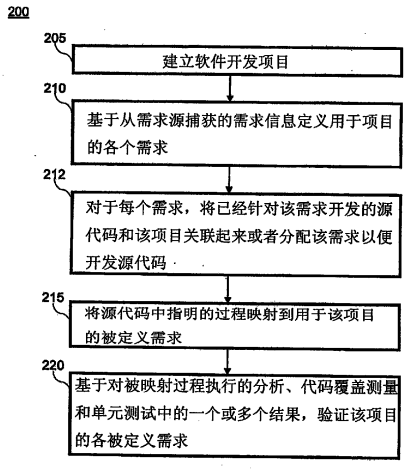
[22] 申请日 2008.4.3
[21] 申请号 200880017725.4
[30] 优先权
[32] 2007. 4. 3 [33] US [31] 60/907,474
[86] 国际申请 PCT/US2008/004349 2008.4.3
[87] 国际公布 WO2008/124038 英 2008.10.16
[85] 进入国家阶段日期 2009.11.27
[71] 申请人 LDRA 技术公司
地址 英国威勒尔
[72] 发明人 W·G·圣克莱尔
S·A·圣克莱尔
[74] 专利代理机构 北京亿腾知识产权代理事务所
代理人 陈 霁
权利要求书 8 页 说明书 40 页 附图 26 页

[54] 发明名称

软件需求验证的自动化管理

[57] 摘要

一种以电子方式管理软件开发所用需求的示例性系统包括：项目模块，需求模块，映射模块和验证模块。 项目模块被配置为建立软件开发项目。需求模块，被配置为基于从需求源捕获的需求信息定义项目的需求。 对于每个需求，项目模块被配置为将为该需求而开发的源代码和项目关联起来或者将该需求分配用于开发源代码。 映射模块配置用于将源代码中识别出的过程映射到被定义需求。 验证模块，配置为基于针对映射过程而开展的分析、代码覆盖测量和单元测试中的一或多个的结果验证项目的被定义需求。



1. 一种以电子方式管理软件开发的需求的方法包括:

建立软件开发项目;

基于从需求源捕获的需求信息定义该项目的需求;

对于每个需求，将为该需求开发的源代码和该项目关联起来或者将该需求分配用于源代码的开发;

将源代码中标明的过程映射到所述项目的被定义的需求; 并且

基于针对所映射过程而进行的单元测试、系统测试、代码覆盖测量和分析中的一个或多个的结果验证该项目的被定义需求。

2. 如权利要求 1 所述的方法，其中定义项目的各需求的步骤包括:基于需求信息捕获用于该项目的高级需求; 并且，基于所述需求信息定义用于该项目的至少一个额外需求。

3. 如权利要求 2 所述的方法，其中定义用于该项目的至少一个额外需求的步骤包括从由下列构成的组中选择定义用于该项目的至少一个额外需求: 满足该项目的一个捕获高级需求的一部分的低级需求，从该项目的捕获的高级需求或定义的低级需求中推断出的衍生需求; 作为该项目的高级、低级或衍生需求的副本的同级需求,所述项目定义了并能够执行一个验证任务。

4. 如权利要求 3 所述的方法，其中建立软件开发项目的步骤包括:识别项目的用户; 将项目的被定义需求分配给用户，以便建立相应的子项目。

5. 如权利要求 4 所述的方法，还包括: 为用户产生汇总被分配给用户的被定义需求的状态报告，并且在项目期间更新状态报告，以纳入分析、代码覆盖测量和单元测试的任何相应结果。

6. 如权利要求 4 所述的方法，还包括: 基于被分配执行类似动作的用户组创建多个用于项目的角色; 将权限分配给所述角色，以定义用户可以完成的动作; 并且，从所述多个创建角色中选择用于用户的角色。

7. 如权利要求 6 所述的方法，其中创建多个角色的步骤包括: 创建项目管理者角色，它具有对所述以电子方式管理软件开发用需求的

方法实施控制的权限。

8. 如权利要求 7 所述的方法，其中所述创建多个角色的步骤包括：从由下列角色组成的组中创建至少一个额外角色：开发者角色、测试工程师角色、独立验证和确认角色和质量保证角色。

9. 如权利要求 6 所述的方法，其中对角色分配权限的步骤包括：分配从下列权限组成的组中选择出的至少一个权限：创建低级需求权限，创建衍生需求权限、生成子项目权限、编辑线程特性权限、和人工验证权限。

10. 如权利要求 1 所述的方法，其中映射所述过程的步骤包括：调用静态分析工具来分析源代码以识别源代码中的过程。

11. 如权利要求 10 所述的方法，其中映射所述过程的步骤还包括：调用静态分析工具以便对源代码进行分析，包括向源代码应用项目编码规则和质量度量。

12. 如权利要求 11 所述的方法，其中验证项目的被定义需求的步骤包括：如果分析结果表明源代码和项目编码规则和质量度量一致，则验证通过所述的项目的被定义需求。

13. 如权利要求 12 所述的方法，还包括：基于分析结果跟踪不一致性；报告所述不一致性，包括提供有关所述不一致性的处理的信息。

14. 如权利要求 10 所述的方法，其中所述映射过程的步骤还包括：调用单元测试工具对所映射的过程运行测试用例。

15. 如权利要求 14 所述的方法，其中验证项目的被定义需求的步骤包括：如果测试用例产生期望结果，则验证通过项目的被定义需求。

16. 如权利要求 1 所述的方法，还包括提供项目的被定义需求的状态，指明每个被定义需求是已获验证或未获验证的。

17. 如权利要求 16 所述的方法，还包括：产生集成需求跟踪和验证矩阵，以便动态跟踪被定义需求的实现和被定义需求的状态。

18. 如权利要求 17 所述的方法，还包括：通过将项目的被定义需求分配给参加项目工作的用户来建立项目的子项目；并且将子项目和集成的需求跟踪和验证矩阵同步。

19. 如权利要求 18 所述的方法，还包括：允许用户通过通信网络访问项目；并且通过监视用户何时经通信网络签入和签出相应的子项目来维持需求跟踪和验证矩阵的完整性。

20. 一种以电子方式管理软件开发的需求的系统，包括：项目模块，被配置为建立软件开发项目；需求模块，被配置为基于从需求源捕获的需求信息定义项目的需求，其中对于每个需求，项目模块被配置为将为该需求而开发的源代码和项目关联起来或者将该需求分配用于开发源代码；映射模块，配置用于将源代码中识别出的过程映射到被定义需求；以及，验证模块，配置为基于针对映射过程而开展的分析、代码覆盖测量和单元测试中的一或多个的结果验证被定义需求。

21. 如权利要求 20 所述的系统，其中需求模块经配置为捕获用于该项目的高级需求；并且，基于所述需求信息定义用于该项目的至少一个额外需求。

22. 如权利要求 21 所述的系统，其中需求模块被配置为定义从由下列需求构成的组中选出的用于该项目的所述至少一个额外需求：满足该项目的一个捕获高级需求的一部分的低级需求，从该项目的捕获的高级需求或定义的低级需求中推断出的衍生需求；作为该项目的高级、低级或衍生需求的副本的同级需求。

23. 如权利要求 20 所述的系统，其中项目模块被配置为识别项目的用户；并且将项目的被定义需求分配给用户，以便建立相应的子项目。

24. 如权利要求 23 所述的系统，其中项目模块被配置为基于被指定执行类似动作的用户组创建多个用于项目的角色；将权限分配给所述角色，以定义被分配给该角色的用户可以完成的动作；并且，将角色分配给用户。

25. 如权利要求 24 所述的系统，其中所述角色包括具有对所述以电子方式管理软件开发用需求的系统实施控制的权限的项目管理者角色，和从由下列角色组成的组中选择出的至少一个额外角色：开发者角色、测试工程师角色、独立验证和确认角色和质量保证角色。

26. 如权利要求 25 所述的系统，其中所述角色具有由拥有创建角色的权限的用户之一定义的角色。

27. 如权利要求 24 所述的系统，其中所述权限从下列权限组成的组中选择出：创建低级需求权限，创建衍生需求权限、生成子项目权限、编辑线程特性权限、和人工验证权限。

28. 如权利要求 20 所述的系统，其中映射模块被配置为调用静态

分析工具来识别源代码中的过程。

29. 如权利要求 28 所述的系统，其中映射模块被进一步配置为调用静态分析工具以便对测试中的源代码的代码覆盖进行测试和度量。

30. 如权利要求 29 所述的系统，其中验证模块被配置为在代码覆盖测量产生所需结果的情况下，则验证通过所述的项目的被定义需求。

31. 如权利要求 28 所述的系统，其中映射模块被进一步配置为，通过向源代码应用项目编码规则和质量度量，调用静态分析工具以便对源代码进行分析。

32. 如权利要求 31 所述的系统，其中验证模块被配置为，如果分析结果表明源代码与项目编码规则和质量度量一致，验证通过所述的项目的被定义需求。

33. 如权利要求 32 所述的系统，其中验证模块进一步配置为基于分析结果跟踪不一致性；报告所述不一致性，提供有关所述不一致性的处理的信息。

34. 如权利要求 28 所述的系统，其中映射模块被配置为调用单元测试工具对所映射的过程运行测试用例。

35. 如权利要求 34 所述的系统，其中验证模块被配置为，如果测试用例产生期望结果，则验证通过项目的被定义需求。

36. 如权利要求 20 所述的系统，其中项目模块被配置为：跟踪项目的被定义需求的状态，该状态指明每个被定义需求是已获验证或未获验证的。

37. 如权利要求 36 所述的系统，其中项目模块被进一步配置用于产生集成需求跟踪和验证矩阵，以便动态跟踪被定义需求的实现和被定义需求的状态。

38. 如权利要求 37 所述的系统，其中通过将项目的被定义需求分配给参加项目工作的用户来建立项目的子项目；并且将子项目和集成的需求跟踪和验证矩阵同步。

39. 如权利要求 1 所述的系统，其中该项目模块被配置用于基于该跟踪和验证矩阵产生状态报告。

40. 如权利要求 37 所述的系统，还包括网络接口模块，它配置用于允许用户通过通信网络访问项目；其中项目模块被配置用于通过监

视用户何时经网络接口模块签入和签出相应的子项目来维持需求跟踪和验证矩阵的完整性。

41. 一种利用可扩展标记语言以电子方式管理软件开发需求的方法，包括：对于为软件开发项目而定义的多个需求的每个需求创建一个 XML 线程，其中每个线程包括描述该线程的各种特性的多个元素，和允许该项目的各被定义需求可跟踪的唯一标识符；实施 XML 包装器，以允许应用和 XML 线程接口；其中每个 XML 线程的各元素包括该项目的一个对应的被定义需求，被映射为所述对应被定义需求的项目源代码的过程，以及针对被映射过程而进行的单元测试、代码覆盖测量和分析中的一个或多个的结果。

42. 如权利要求 41 所述的方法，还包括：将每个 XML 线程转换为由多个容器对象构成的分层树，其中每个容器对象对应于该 XML 线程的一个元素并且具有唯一识别号。

43. 如权利要求 42 所述的方法，还包括：实现一个 XML 类，以帮助访问容器对象。

44. 如权利要求 42 所述的方法，还包括：实现一个 XML 类，以帮助顺序搜索容器对象构成的该分层树。

45. 如权利要求 42 所述的方法，还包括：将 XML 线程组织成多个文件类型，以促进对该项目的处理。

46. 如权利要求 45 所述的方法，其中所述处理包括：更新被定义需求的一个跟踪和验证矩阵，该矩阵指明每个所述被定义需求是经验证或未获验证的，并且跟踪不一致性。

47. 如权利要求 41 所述的方法，还包括：将 XML 线程转换为 SQL 格式，供经网络存储和访问。

48. 一种以电子方式管理软件开发的需求的方法，包括：建立软件开发项目；基于从需求源捕获的需求信息定义项目的需求；对于每个需求，将为该需求开发的源代码一个或多个部分和该项目关联起来或者将该需求分配用于源代码的开发；将源代码中标明的过程映射到所述项目的被定义要求；并且根据一或多个验证规则基于一个或多个测试的结果验证该项目的被定义需求。

49. 如权利要求 48 所述的方法，其中被定义需求的同级需求是通过将该同级需求和验证规则关联而创建。

50. 如权利要求 48 所述的方法，其中被定义需求的同级需求是通过将该同级需求和用户关联而创建。

51. 如权利要求 48 所述的方法，还包括：创建一个组；将验证规则的一个或多个和该组关联起来；并且将被定义需求的一个或多个和该组关联起来。

52. 如权利要求 51 所述的方法，其中对于和该组关联的一个需求，为和该组关联的每个所述验证规则产生一个同级需求。

53. 如权利要求 48 所述的方法，其中验证规则包括代码审查、质量审查、系统测试、单元测试、子系统测试和集成测试。

54. 如权利要求 48 所述的方法，其中验证规则规定被分配给一个被定义需求的用户是否被要求执行该需求的验证，或被分配给所述被定义需求的用户以外的用户被要求执行验证。

55. 如权利要求 48 所述的方法，其中项目的被定义需求包括基于需求信息的类型和格式解析需求信息。

56. 如权利要求 55 所述的方法，其中需求信息的类型和格式由一个用户规定。

57. 如权利要求 55 所述的方法，其中需求信息的类型和格式预先定义。

58. 一种以电子方式管理软件开发的需求的系统，包括：

项目模块，被配置为建立软件开发项目；

需求模块，被配置为基于从需求源捕获的需求信息定义项目的需求；

映射模块，将源代码的一个或多个部分和一个被定义需求关联起来；和，

验证模块，被配置为根据一或多个验证规则基于一个或多个测试的结果验证该项目的被定义需求。

59. 如权利要求 58 所述的系统，其中被定义需求的同级需求是通过将该同级需求和验证规则关联而创建。

60. 如权利要求 58 所述的系统，其中被定义需求的同级需求是通过将该同级需求和用户关联而创建。

61. 如权利要求 48 所述的系统，其中验证模块创建一个组；将所述验证规则的一个或多个和该组关联起来；并且将所述被定义需求的

一个或多个和该组关联起来。

62. 如权利要求 61 所述的系统，其中对于和该组关联的一个需求，为和该组关联的每个所述验证规则产生一个同级需求。

63. 如权利要求 58 所述的系统，其中需求模块基于需求信息的类型和格式解析该需求信息。

64. 如权利要求 58 所述的系统，其中需求的类型和格式由一个用户规定。

65. 如权利要求 58 所述的系统，其中需求的类型和格式预先定义。

66. 如权利要求 58 所述的系统，还包括将一或多个测试用例分配给约束（circumscribe）期望功能的每个需求。

67. 如权利要求 48 所述的系统，其中被定义需求的同级需求是通过将该同级需求和确保与进程对象或质量评估一致的一个对象关联而创建的。

68. 如权利要求 67 所述的系统，其中进程对象或质量评估包括需求确认、计划或归档（documentation）。

69. 如权利要求 48 所述的系统，其中被定义需求的同级需求是通过将同级需求和被定义需求的一个故障项关联而创建的。

70. 如权利要求 58 所述的系统，其中被定义需求的同级需求是通过将该同级需求和确保与进程对象或质量评估一致的一个对象关联而创建的。

71. 如权利要求 70 所述的系统，其中所述进程对象或质量评估包括需求确认、计划或归档（documentation）。

72. 如权利要求 58 所述的系统，其中被定义需求的同级需求是通过将同级需求和被定义需求的一个故障项关联而创建的。

73. 如权利要求 58 所述的系统，其中被定义需求的同级需求是针对一个和该被定义需求关联的测试用例而创建的。

74. 如权利要求 48 所述的方法，其中被定义需求的同级需求是针对一个和该被定义需求关联的测试用例而创建的。

75. 如权利要求 1 所述的方法，其中额外需求包括和一个被定义需求相关联的测试用例。

76. 一种以电子方式管理软件开发所用需求的系统，包括：项目模块，被配置为建立软件开发项目；需求模块，被配置为基于从需求源

捕获的需求信息定义项目的需求；建模模块，被配置用于包括构架性工件和执行工件；以及，验证模块，配置为根据一或多个验证规则基于一或多个测试的结果验证项目的被定义需求；其中需求模块将被定义需求和建模模块的相应构架性工件与执行工件链接起来。

77. 如权利要求 76 的系统，其中构架性工件和执行工件自动产生用于所述被链接的被定义需求的源代码。

78. 如权利要求 76 的系统，还包括一个映射模块，被配置用于将人工产生的源代码的一个或多个部分和被定义需求关联。

79. 一种以电子方式管理软件开发的需求的方法，包括：建立软件开发项目；基于从需求源捕获的需求信息定义项目的需求；建立包括构架性工件和执行工件的建模模块；根据一个或多个验证规则，基于一个或多个测试的结果验证项目的被定义需求；其中被定义需求和相应的构架性工件与执行工件链接起来。

80. 如权利要求 79 的方法，其中构架性工件和执行工件自动产生用于所述被链接的被定义需求的源代码。

81. 如权利要求 76 的系统，还包括一个映射模块，被配置用于将人工产生的源代码的一或多个部分和被定义需求关联。

软件需求验证的自动化管理

技术领域

[0001]本发明涉及软件开发需求的管理。具体地说，本发明涉及分布式需求跟踪（traceability）和软件测试/验证整合的自动化技术和工具。

背景技术

[0002]在许多安全性要求严格（safety-critical）和任务性质关键（mission-critical）的行业，比如航空电子、医疗、国防和核工业里，与软件开发项目有关的需求跟踪和验证任务可以占据相当数量的项目预算。现有一些技术可用于从各种需求信息源自动地将用于软件开发项目的需求信息捕捉成为需求跟踪（traceability）矩阵（RTM）的形式，该矩阵可以用于跟踪这些需求在源代码中的实现。需求跟踪矩阵通常包含工程级、用户定义的需求（指定软件产品应该如何操作）以及底层设计约束（不包含代码级）和需求验证（指定软件产品应该如何验证）。此外，也存在一些技术，用于人工测试实现这些需求的源代码以及用于验证这些需求。因此，所需要的是一种将分布式软件开发项目的需求跟踪和测试/验证任务进行集成和自动化，并引入统一的用户接口以帮助管理高度复杂和高技术含量的软件开发项目的技术。

发明内容

[0003]本发明支持高度分布且异步的操作。本发明的多个示例性实施方案采用了中央集成库（depository），它们实质上（inherently）是集中化（centralize）的并且可以由管理者管理。本发明的各示例性实施方案将用户工作区和开发工件（artifact）按动态管理项目树链接（link）起来，从而综合各个进程（process）并且确保了它们的可重复性。用户将软件开发工件返回给中央集成库。因此，本发明的各个示例性实施方案建立并且维持了可证明的可跟踪性。本发明的示

例性实施方案可以解决相关领域系统中的一些问题，这些领域系统中所执行的进程相当程度上是人工进行的、易发生错误并且一致性不好。请留意，本发明不限于或不被要求解决相关领域的这些问题。

[0004]一种以电子方式管理用于软件开发的需求的示例性方法包括：建立软件开发项目；基于从需求源捕获的需求信息定义用于该项目的需求；对于每个需求，将为该需求开发的源代码和该项目关联起来或者将该需求分配以便开发源代码；将源代码中标明的过程映射到用于所述项目的所述被定义需求；并且基于针对所映射过程而进行的单元测试、代码覆盖测量和分析中的一个或多个的结果，验证该项目的被定义需求。

[0005]利用可扩展标记语言（XML）以电子方式管理用于软件开发的需求的一种示例性方法包括对于为软件开发项目而定义的多个需求中的每个需求创建一个 XML 线程。每个线程包括描述该线程的各种属性的多个元素，和实现用于该项目的各被定义需求的可跟踪性的唯一标识符。该方法还包括实现 XML 包装器（wrapper），以允许各应用程序和 XML 线程互联。每个 XML 线程的各种属性包括该项目的一个对应的被定义需求，被映射到所述对应被定义需求的用于该项目的源代码过程，以及针对被映射过程而进行的单元测试、代码覆盖测量和分析中的一个或多个的结果。

[0006]一种以电子方式管理在用户网络上进行的软件开发需求的示例性系统包括项目模块、需求模块、映射模块和验证模块。项目模块被配置为建立软件开发项目。需求模块被配置为基于从需求源捕获的需求信息来定义该项目的需求。对于每个需求，项目模块被配置将为该需求开发的源代码和该项目关联起来，或者将该需求分配用于开发源代码。映射模块配置为将源代码中标明的过程映射到被定义的需求。验证模块配置为基于针对被映射过程而进行的单元测试、代码覆盖测量和分析中的一个或多个的结果来验证被定义的需求。

[0007]一种以电子方式管理在用户网络上进行的软件开发需求的示例性系统包括项目模块，被配置用于建立软件开发项目；需求模块，被配置为基于从需求源捕获的需求信息定义用于该项目的需求；建模模块，配置为包括构架性工件和执行工件；验证模块，配置为根据一或多个验证规则，并基于一或多个测试结果来验证该项目被定义的需

求。该模块将被定义的需求和建模模块的构架性工件与执行工件链接起来。

[0008]一种以电子方式管理软件开发需求的示例性方法包括建立软件开发项目；基于从需求源捕获的需求信息来定义该项目的需求；建立包括构架性工件和执行工件的建模模块；根据一或多个验证规则，基于一或多个测试的结果验证用于该项目的被定义需求。被定义需求和相应的构架性工件与执行工件链接起来。

附图说明

[0009]结合附图阅读下文对优选实施方案的详细描述之后，本领域的普通技术人员将对本发明的其它目的和优点一目了然；附图中相似的参考标号用于指代相似的元素；并且，附图中：

[0010]图 1A 和 1B 分别示意了一个其中可以实现自动管理软件需求的系统的示例性环境和自动需求验证工具的示例性实施方案；

[0011]图 2 示意一个处理流程图，它提供自动管理软件需求验证的方法的各示例性步骤；

[0012]图 3A-3E 示意了自动管理软件需求验证的系统的一种示例性可扩展标记语言（XML）实现；

[0013]图 4A-4J 示意了自动管理软件需求验证的系统的示例性图形用户接口的屏捕获；

[0014]图 5 示意了选择，创建，重新命名或者删除组视图的例子；

[0015]图 6 示意了编辑用于给定组的规则的一个例子；

[0016]图 7 示意了基于类型解析的例子；

[0017]图 8 示意了在微软 MS Word 文档下的需求的例子；

[0018]图 9 和图 10 示意了类型编辑器的例子；

[0019]图 11 示意了未编组需求和编组需求的例子；

[0020]图 12 示意了同级需求的例子；以及

[0021]图 13 示意了同级需求的另一个例子。

具体实施方式

系统总揽

[0022]本发明应当结合示例性实施方案加以描述，但不是必需受

限于此。对于本领域的普通技术人员来说在不偏离后附权利要求书所定义的发明范围的情况下会出现许多变化和变更。绝对的语句（开始以“必须（must）”）和有关各种优点或其它方面的语句适用于特定的示例性实施方案，而不是必然地应用于权利要求书所覆盖的所有实施方案。

[0023]这里给出了对软件需求进行自动管理的技术，这些技术可以在分布式用户工作区（例如，本发明的各实例所允许的工作目录）的网络上缩小高级需求跟踪工具和低级源代码测试工具之间的差距。下面结合示例性实施方案进行解释，但本发明不受限于此。

[0024]图 1A 示意了实施用于对软件需求验证进行自动管理的系统 100 的示例性环境。如图 1A 所示，系统 100 可以结合基于计算机的系统使用，在该基于计算机的系统中各元素可以按硬件、软件、固件或其组合的方式实施。如图 1A 所示，系统 100 包括需求信息源 105、需求捕获工具 110、自动化需求验证工具 115、静态分析&代码覆盖工具 120、单元/集成测试工具 125、网络 130 和用户 135。软件开发项目（例如医院的账单系统、商店的库存跟踪系统，航天电子显示系统，等）可以由需求定义。所述需求可以指明软件产品在其实施时应当如何操作。需求信息源 105，如图 1A 所示，可以代表任何需求信息源，其中包括需求管理工具（例如 Telelogic DOORS®和 IBM Rational RequisitePro®），分析表程序（spread sheet programs）（例如 Microsoft Excel®），文字处理程序（例如 MS Word®）。需求捕获工具 110 可以从需求信息源 105 捕获需求信息。该信息可以包括需求标识符和需求描述以及测试例标识符和测试例描述。

[0025]在一个实施方案中，需求捕获工具 110 可以使用自然语言解析需求信息。例如，需求捕获工具，比如来自 Geensys(网址 geensys.com)的 Reqtify™ 工具，采用这样的技术从需求信息产生 RTM。也可以采用诸如 LDRA TBreq™的产品或者定制的手工编码系统来实现需求捕获工具 110。对需求的解析可以基于类型来进行，下文将在段落"基于类型视图进行解析（Parsing Based On Types View）"处详细描述。

[0026]例如，如果为在线杂货店开发软件，那么需求信息源可能包括 MS Word™文件，该文件的一条需求规定“在线杂货店应当维持一

个消费者账户”。在该例子中，需求捕获工具 110 可能从 MS Word™文件的文本中捕获需求信息“维持账户”。在另一个例子中，需求信息源可能包括 DOORS™模块，该模块的一条需求规定“软件必须为所有故障类型提供故障隔离”。在该例子中，需求捕获工具 110 可以从 DOORS™模块的文本中捕获需求信息“隔离故障”。

[0027] 在一个示例性实施方案中，软件建模模块 190 可以结合需求捕获工具 110 使用。软件建模模块 190 可由软件建模工具，比如 Telelogic Rhapsody™, Mathworks Simulink™或 National Instruments Labview™实现。在该示例性实施方案中，需求捕获工具 110 将各需求与构架性工件和执行工件在软件建模工具中 190 链接在一起。由此，这些构架性和执行工件被隐式映射到由建模工具所产生的源代码中。由此，每个需求和一或多个软件构架性模型元素关联起来。这些模型元素被传递到分布的验证模块，以便执行、测试和分析。

[0028] 此外，可以使用建模工具或软件构架性模型来自动产生源代码，其中需求和源代码之间建立映射关系。换句话说，与构架性工件和执行工件链接在一起的各需求被隐式映射到建模工具所产生的源代码。

[0029] 如果软件实现是人工编码的，那么需求到源代码的映射将利用基于建模工具所产生的执行工件的自动化需求验证工具 115 实现。下文将对自动需求验证工具 115 进行详细说明。

[0030] 静态分析&代码覆盖工具 120 可以用于分析源代码文件(该文件此前由软件开发工具编写)，以便识别源代码中的过程并且评估源代码与项目编码规则和质量度量的一致性。这里，“过程”描述的是编写用于实现某需求的一部分代码。例如，过程可能包括 C 编程语言中的函数和 C ++编程语言下的类和成员函数。可以采用各种自动化软件分析和测试产品，比如 LDRA Testbed™; Coverity(网址 coverity.com) 的 Coverity Prevent Coverity Extend; Programming Research (网址 programmingresearch.com)的 QA-C, QA-C++和 QA-MISRA; Gimple Software(网址 gimpel.com)的 PC-lint for C/C++和另一基于 Lint 的产品; Klocwork (网址 klocwork.com)的 inSpect, inSight, inTellect and inForce; Parasoft (网址 parasoft.com)的 C++ Test, C++ Test for Embedded Systems, Code Wizard with Test

Case Sniffer; McCabe & Associates(网址 mccabe.com)的 McCabe IQ; Telelogic 的 Telelogic Logiscope, TAU/Tester 和 TAU/Logiscope TestChecker; 以及 Testwell(网址 testwell.fi) 的 CMT++, Complexity Measures Tool for C/C++, 可用来实现静态分析&代码覆盖工具 120 的静态分析能力。

[0031]就在线杂货店的例子而言，用于实现捕获需求“维持账户”的示例性源代码文件“MaintainAcount.c”可能包括如下代码段:

```
float MaintainAccountFunction ( )
{
   long accntNum;
   float balance, avgBalance, ratio;

   accntNum = GetSessionAccountNumber ( ) ;
   if (accntNum == 0) { return -1; }
   balance = GetAccountBalance (accntNum) ;
   if (balance <= 0) { FatalError ( "Balance too low"); }
   avgBalance = GetAverageAccountBalance (accntNum) ;
   ratio = balance / avgBalance;
   if (ratio < .2)
   {
      UserMessage ( "Balance is running low");
   }
   return 0;
}
```

在该例子中，静态分析&代码覆盖工具 120 可能识别下面的源代码基本块(块 4-6):

```
60  4   balance =
61  4     GetAccountBalance (
62  4     accntNum ) ;
63  4   if
64  4    (
65  4     balance <= 0
```

```
66  4     )
67  5     {
68  5
69  5       FatalError (
70  5       "Balance too low" ) ;
71  5     }
72  6   avgBalance =
73  6     GetAverageAccountBalance (
74  6     accntNum ) ;
75  6   ratio = balance / avgBalance ;
76  6   if
77  6     (
78  6     ratio < .2
79  6     )
```

在分析源代码和项目编码规则的一致性之后，静态分析和代码覆盖工具 120 可能鉴定下面的代码为不一致性（M）命令：

```
56  2     {
57  2     return
58  2     - 1 ;
(M) STATIC VIOLATION: 101 S : Function return type
inconsistent
```

```
67  5     {
68  5
69  5     FatalError (
70  5     "Balance too low" ) ;
(M) VIOLATION :1 J : Unreachable Code found.
71  5     }
```

```
72  6     avgBalance =
73  6     GetAverageAccountBalance (
```

```
74  6     accntNum ) ;
75  6    ratio = balance / avgBalance ;
76  6    if
77  6     (
78  6     ratio < .2
(M) STATIC VIOLATION : 96 S : Use of mixed mode arithmetic:
float double ratio < .2
79  6     )
```

以航空电子显示系统为例。一个用于实现捕获需求“维持账户”的示例性源代码文件“CheckAvionicsDisplay.c”可能包含下列代码段:

```
int CheckAvionicsDisplay ( )
{
// 试图启动各模块，如果一个模块和它的备份失败
// 返回一个负数，表明该模块发生故障。
//
if ( IBITestModule1 () ) { if ( ! BackupModule1 ( ) ) return
-1; }
if ( !BITestModule2 () ) { if ( ! BackupModule2 ( ) ) return
-2; }
if (!BITestModule3() ) { if ( ! BackupModule3 ( ) ) return
-3; }
// 所有模块均已启动。返回 1 表明成功。
return 1;
}
```

在此例子中，静态分析&代码覆盖工具 120 可能识别下列源代码基本块（块 1-7）:

```
37  1   int
38  1    CheckAvionicsDisplay ( )
39  1    {
40  1   //试图启动各模块。如果一个模块和它的备份失败
```

```
41  1   // 返回一个负数，表明该模块发生故障
42  1
43  1     if
44  1      (
45  1      !
46  1        BITestModule1 ( )
47  1      )
-------------------------------------------------------
48  2     {
49  2       if
50  2       (
51  2       !
52  2         BackupModule1 ( )
53  2       )
-------------------------------------------------------
54  3       {
55  3        return
56  3        - 1 ;
-------------------------------------------------------
57  4       }
-------------------------------------------------------
58  5    }
-------------------------------------------------------
59  6    if
60  6     (
61  6      !
62  6       BITestModule2 ( )
63  6      )
-------------------------------------------------------
64  7     {
65  7      if
66  7         (
```

```
67  7          !
68  7          BackupModule2 ( )
69  7        )
```

在分析源代码和项目编码规则的一致性之后，静态分析&代码覆盖工具120可能鉴定下列代码为不一致性（M）命令：

```
37  1    int
38  1      CheckAvionicsDisplay ()
(M) STATIC VIOLATION: 63 S Empty parameter list to procedure/
funct .
39  1      {
40  1    //试图启动各模块。如果一个模块&它的备份失败
41  1    // 返回一个负数，表明该模块发生故障
42  1    //
43  1        if
44  1          (
45  1            !
46  1              BITestModule1 ( )
(M) STATIC VIOLATION : 114 S : Expression is not Boolean.
47  1            )
-------------------------------------------------------------
48  2            {
49  2              if
50  2                (
51  2                !
52  2              BackupModule1 ( )
(M) STATIC VIOLATION : 114 S : Expression is not Boolean.
53  2                )
-------------------------------------------------------------
54  3                {
(M) STATIC VIOLATION : No brackets to then/else (added by
analysis)
55  3                  return
```

```
56  3                 - 1 ;
-------------------------------------------------------
57  4               }
-------------------------------------------------------
58  5           }
-------------------------------------------------------
59  6        if
60  6          (
61  6          !
62  6           BITestModule2 ( )
(M) STATIC VIOLATION : 114 S : Expression is not Boolean.
63  6           )
67  7               !
68  7                 BackupModule2 ( )
(M) STATIC VIOLATION : 114 S : MISRA-C:2004 12.6 13.2:
Expression is not Boolean.
69  7               )
-------------------------------------------------------
```

此外，静态分析&代码覆盖工具120也可以用于测试和度量外部运行检验中的代码覆盖。静态分析&代码覆盖工具120对静态分析和代码覆盖（也称为动态分析）能力的集成为测试验证的三角测量和软件抽象的结构化覆盖提供了关键保障。

[0032]单元/集成测试工具125可以用于测试源代码是否符合需求。在一个实施方案中，单元/集成测试工具125可以运行测试用例，所述测试用例定义源代码各部分的输入和期望的输出，并且产生描述代码覆盖的结果。例如，源代码可以实现为多个函数。如果在单元/集成测试期间某个函数被调用并且执行，那么该函数可以描述为“被覆盖”。这样，通常希望被覆盖的源代码所占百分比比较高。另外，在单元/集成测试期间，可以对各种级别的覆盖进行测试，比如路径覆盖、语句覆盖、分支覆盖和变更的条件/判定覆盖（MC/DC）。可以采用各种自动化和集成测试软件产品来实现单元/集成测试工具125，比如LDRA TBrun™; Parasoft(网址parasoft.com)的C++ Test, C++

Test for Embedded Systems, Code Wizard and Jest with Test Case Sniffer; Vector Software (网址 vectorcast.com)的 VectorCAST/Ada, VectorCAST/C, VectorCAST RSP for Real-time Embedded Testing 和 VectorCAST/Cover; IBM Rational (网址 www-306.ibm.com/software/rational/) 的 Test RealTime; IPL (网址 ipl.com) 的 Adatest 95 和 Cantata ++; Free Software Foundation (网址 gnu.org) 的 Cunit, CPPunit 和 Junit; 以及, Testwell (网址 testwell.fi)的 CTA++, C++ Test Aider, CTC++, Test Coverage Analyzer for C/C++。

[0033]以在线杂货店为例。一个示例性测试用例可能测试映射到需求“维持账户”的过程(比如浮点 MaintainAccountFunction())是否符合该需求。在该例子中，测试用例可能规定：如果 GetAccountBalance(accntNum)被调用时输入值=3201，则测试用例将返回值=1。在该例子中单元测试的覆盖结果可能表示如下：

— 覆盖—

- 语句-

期望的覆盖率(%)： 80

映射过程的通过率 (%)： 89

-分支判定-

期望的覆盖率(%)： 60

被映射过程的通过率(%)： 63

- 路径-

期望的覆盖率 (%)： 70

被映射过程的通过率(%)： 72

[0034] 以航空电子显示系统为例，一个示例性测试用例可能测试映射到需求“隔离故障”的过程(比如 int CheckAvionicsDisplay())是否符合该需求。在该例子中，测试用例可能规定如果函数 BITestModule3 ( )被设置为 1 并且 BackupModule 返回 1。在该例子中的单元测试的覆盖结果可能表示如下：

--覆盖--

- 语句-

期望覆盖率 (%)： 80

被映射过程的通过率 (%): 61

- 分支判定-

期望覆盖率(%): 60

被映射过程的通过率 (%): 51

- 路径-

期望覆盖率 (%): 50

被映射过程的通过率 (%): 48

[0035] 对需求是否已经在源代码中实现的跟踪和对所实现的源代码是否按期望方式运作的验证可以人工方式进行，但是对大型软件开发项目来说可能有数以千计的需求，对需求的人工跟踪和验证很快便会变得不切实际并且无法管理。因此，如图 1A 所示，自动化的需求验证工具 115 可以实现，以缩短需求捕获工具 110 和静态分析和单元/集成测试工具 120 和 125 之间的差距。图 1B 示意了自动化需求验证工具 115 的一种示例性实现，它包括需求模块 140、项目模块 145、映射&分析模块 150、验证模块 155 和网络接口模块 160。本文中结合图 4A-4J 描述的 LDRA TBmanager™工具可以用来实现自动化需求验证工具 115。

[0036] 需求模块 140 可以配置为基于需求捕获工具 110 从需求信息源 105 捕获的需求信息定义用于该软件开发项目的需求。被捕获的需求信息可以表示用户定义的，项目级的软件开发项目需求，这里称之为“高级”需求。在线杂货店的例子中，被捕获的需求“维持账户”是示例性高级需求。但是，高级需求并不描述实际实现该软件所需要的代码级设计需求。如此，需求模块 140 可以被配置为提供定义未被需求捕获工具 110 所捕获的其它需求的选项。

[0037] 例如，需求模块 140 可以用于定义满足某高级需求的一部分的需求，本文称为“低级”需求。低级需求可以定义成规定软件的一部分应当如何实现。例如，一个低级需求可能集中在特定函数、目标或目的。以在线杂货店为例，可以为高级需求“维持账户”定义一种被称为“账户名”的低级需求，以便为该账户建立一个名字。在另一个例子，即航空电子显示系统中，为高级需求“隔离故障”定义一个低级需求“启动被命名的冗余系统”，以确保只有被识别为 Built In Test 的冗余模块被启用。

[0038] 需求模块140也可以用于定义从高级需求或低级需求推断或衍生而来的需求，这里称之为“衍生”需求。因此，衍生需求可能不会由需求捕获工具110捕获，但是可以后续由自动化需求验证工具115创建并且管理。以在线杂货店为例，可以为高级需求“维持账户”定义一个被称为“取消账户”的衍生需求，以取消一个被维持的账户。在另一个例子即航空电子显示系统中，可以为高级需求“隔离故障”定义一个称之为“损失评估”的衍生需求，以帮助恢复过程通知。

[0039] 需求模块140还可以用于定义作为高级、低级或衍生需求的副本的需求，这里称之为“同级”需求。同级需求可以是出于需求多样化和任务管理（比如，用户135之一可能写源代码来实现某个低级需求，而用户135中的另一个可能验证所实现的源代码）的目的而创建的。如这里所述，同级需求可以由唯一标识符区别，尽管它在其它方面和作为它母体的基础的高级、低价或衍生需求并无不同。这样，需求模块140可以用于给RTM填充（populate）来自需求信息源105的被捕获的高级需求、以及其它的低级、衍生和同级需求，以便允许自动化需求验证工具115提供向下到源代码级的自动化需求跟踪工作。

[0040] 同级需求可以被称为验证任务。一种额外的同级类型，或者验证任务，被称为目标（objective）。目标被定义为确保同处理目标或诸如需求验证或恰当计划和归档之类的量化评估的一致性。

[0041] 另一种同级需求类型和缺陷报告生成及解决方案有关。作为需求的对立面，缺陷报告可以由用户使用静态分析&代码覆盖工具120或单元/集成测试工具125的自动产生。缺陷报告的分配导致产生一个同级或验证任务，该任务的唯一目的便是跟踪缺陷报告到其关闭为止。

[0042] 项目模块145可以配置用于建立项目。如这里所述，项目模块145所建立的“项目”描述了真实世界软件开发项目的电子实现，其中被建立的项目可以提供一种电子机制使得现实世界的许多方面有机化和自动化，其中一些方面包括定义用于项目的需求，分配用户参加项目的工作、跟踪各需求的实现、分析和验证所开发用于实现所述需求的源代码文件、跟踪缺陷。所建立的项目可以包括相关的需求信息和为实现这些需求由开发者编写的源代码文件。在一个实施方案

中，先前针对某需求开发的源代码可以和所述项目关联起来。如果该需求不存在已开发好的源代码，那么项目模块 145 可以推进基于需求的开发，其中在源代码开发之前将该需求分配给用户。

[0043] 项目模块 145 可以配置为对已建立的项目的相关需求和源代码文件进行的分层显示，并且可以提供某需求的状态信息（比如，该需求是否已验证或未获验证，被分配参加该需求工作的用户，等等）。例如，图 4C（这里结合一种示例性用户接口描述）示意了一种在线杂货店例子下的项目树 412，图中给出了需求和源代码文件的分层体系。通过将高级需求和其低级实现进行关联，项目模块 145 可以允许项目管理者看到逐个需求的分解图示，了解哪些需求的实现符合编码约定（convention）、代码覆盖水平和函数测试准则。如此，项目模块 145 可以提供集成的需求跟踪和验证矩阵。在一个实施方案中，项目模块 145 可以产生状态报告，总结用于某个用户的需求信息，该状态报告可以随着项目的进展加以更新，以纳入诸如单元测试之后的代码覆盖结果之类的其它信息。

[0044] 项目模块 145 可以配置为添加用户 135 到项目之中。用户 135 可以包括参加项目工作的任何人，比如项目经理、软件开发者、测试工程师等。在一个实施方案中，项目模块 145 可以基于被指定完成类似行为的用户 135 组为项目创建角色”，并且将所创建的角色分配给用户 135。例如，可以创建开发者角色、测试工程师角色、独立验证和确认（validation）（IVV）角色和质量保证（QA）角色，以及其它角色。在一个实施方案中，项目管理者角色可以被创建，使得被分配项目管理者角色的用户可以对需求验证处理的几乎所有方面实施控制，并且可以创建其它角色，并且向用户 135 分配角色和需求。

[0045] 在一个实施方案中，可以给角色分配各种权限，以定义被分配该角色的用户可以执行的动作。权限的例子包括创建低级需求权限、创建衍生需求权限、生成子项目权限（例如，当一个用户被分配了某个需求的时候可以建立一个子项目）、编辑线程属性权限（例如，可以给每个需求创建 XML 线程）、人工验证权限（比如，允许用户访问验证模块 155，以审视本文所述的单元测试、代码覆盖和静态分析结果。）以及其它权限。

[0046] 项目模块 145 也可以配置为允许用户具有必要的权限去向

用户 135 分配需求模块 140 所定义的各种需求（即高级、低级、衍生和同级需求）。在一个实施方案中，向用户分配需求可以建立相应的子项目。这里所提到的“子项目”描述了一个用户可以访问执行和他们被分配角色有关的指定任务（例如，写源代码、分析源代码文件、测试源代码文件等等）的工作区。

[0047] 在一个实施方案中，项目模块 145 可以调整子项目的操作状态，要求用户在对某子项目执行任何任务之前从项目模块 145 “签出”该子项目，并且在完成所述任务之后将子项目“签入”项目模块 145。换句话说，为了签出某个子项目，用户可以出于远程完成任务的目的制作它们的子项目文件的副本，并且为了将子项目签入，用户可以将他们子项目文件的修改后内容复制回已立项项目的目录树中。如此，项目模块 145 可以通过在每次用户签入和签出它们的子项目的时候将子项目和 RTM 的当前状态进行同步，保持 RTM 的完整性。另外，如这里所述，可以分配一种创建子项目的权限，以允许用户通过项目模块 145 更新子项目。例如，受影响的子项目应当在它们相应的需求被更改（例如由于添加或删除用户的缘故）的时候加以更新。

[0048] 如图 1B 所示的网络接口模块 160 可以配置为允许项目的全球（globally）分布，并且提供基于通信网络 130（例如 LAN、WAN、互联网等等）的集成处理。在一个实施方案中，网络接口模块 160 可以允许用户 135 从远方将它们的子项目签入和签出。可以实施一些安全措施，比如建立安全连接以及要求用户 135 登录并且提供密码。此外，验证状态报告可以做到全球可用。通过这些方式，网络接口模块 160 可以用于执行同步或异步操作并且实现基于全球动态更新的需求矩阵（跟踪和验证矩阵）。

[0049] 图 1B 所示的映射&分析模块 150 可以配置用来调用静态分析&代码覆盖工具 120。如本文所述，映射&分析模块 150 可以调用静态分析&代码覆盖工具 120，将源代码中指明的过程映射到用于项目的各个被定义的需求。以在线杂货店为例，在示例性源代码文件 "MaintainAccount.c"中指明的过程 float MaintainAccountFunction(), long GetSessionAccountNumber(), float GetAverageAccountBalance(long accntNum), float GetAccountBalance(long accntNum), void FatalError(const char

* msg), void UserMessage(const char * msg) 可以被映射到高级需求“维持账户”。以航空电子显示系统为例，过程有 int CheckAvionicsDisplay0; int BITestModule1(); int BITestModule2(); int BITestModule3(); int BackupModule1(); int BackupModule2(); int BackupModule3()。图 4G 示意了一种将过程映射到需求的示例性图形用户接口。

[0050] 此外，映射&分析模块 150 可以允许用户通过指明分析需求验证方法和/或测试需求验证方法，来验证某个需求。例如，在一个实施方案中，映射&分析模块 150 可以用于调用静态分析&代码覆盖工具 120 来提供集成分析能力，包括分析被映射的过程是否符合项目编码规则和质量度量。此外，在另一个实施方案中，映射&分析模块 150 可以用于调用单元/集成测试工具 125 来测试被映射的过程是否符合需求，办法是运行测试用例并且产生描述代码覆盖的结果。在又一个实施方案中，映射&分析模块 150 可以用来装备代码，这些代码将被在外部执行，并且其执行结果（即代码覆盖结果）可以后续返回给映射&分析模块 150 以便进行事后代码覆盖分析。

[0051] 如图 1B 所示的验证模块 155 可以配置为基于由静态分析&代码覆盖工具 120 对被映射过程分析的结果以及由单元/集成测试工具 125 对被映射过程进行单元测试的结果，来验证所述被定义的需求。例如，在一个实施方案中，具有必要权限的用户可以利用验证模块 155 来选择需求并且观看该选定需求的期望代码覆盖率和所获得的代码覆盖率（例如，获得期望覆盖率的被映射过程所占百分比）。如果用户对选定需求所取得的覆盖率满意，用户可以将选定需求的当前状态维持为已验证。但是，如果用户对选定需求所取得的覆盖率不满意，那么用户可以改变选定需求的当前状态为未经验证。由于未对某需求做验证即更改了该需求线程（如同本文结合示例性 XML 实现进行描述的那样），用户应当后续更新和该需求有关的任何子项目。

[0052] 验证模块 155 也可以配置为在一个用户下创建多组，并且将需求和相应组管理起来。各组分别分配有验证规则，使得当一个需求被分配给某组时，为该（基本）需求产生用于每个验证规则的一个同级需求。对于测试验证规则，为每个和基本需求相关的测试用例创建一个同级或验证任务。所有需求可以在初始时是未编组的需求。然

后，需求可以和一个或多个组关联起来。

[0053] 在该示例性实施方案中，验证规则规定哪些验证任务必须在该需求可以认定为已验证之前完成。某个需求必须满足的内容由该需求所处的验证组决定。验证规则不仅规定什么验证任务必须完成，也规定谁应当去做。验证规则可以确定该需求是否通过代码审查（review），也可以进一步规定代码审查由特定用户或者特定角色（例如，“测试者”或“QA”）的用户来完成。各规则可以排序，排序将促使各规则按照验证组中列出的顺序完成。

[0054] 当验证规则被定义的时候，可以应用一个或多个质量模型。一种质量模型可以设立一种标准，通过该标准用于某给定项目的软件的质量可以测量并且可视化。质量模型由代码审查和质量审查验证任务强制执行。

[0055] 代码审查标准可以定义在一对文件中，报告文件（<language>report.dat）和处罚文件（<language>pen.dat）。报告文件包含完整的一组预先定义的编码规则。该组规则可以扩展，以便包容某个项目。该组规则的各个子组由公布的编码标准定义，它们是预先定义的，使得用户可以使用这些标准，而不必精通该标准的各个细节。通过报告文件，人们可以看到这些标准到规则的映射及其实施。处罚文件可以是代码审查配置的遗留（legacy）机制。

[0056] 质量审查标准可以定义在'metpen.dat'中。该文件包含一系列的分析测量，这些分析测量是和容限（tolerance）阈值配对的。例如，秩（cyclomatic）复杂度测量、代码复杂度测量、不可达代码行测量、注释比例测量。如果复杂度超过某个预设的阈值，则质量审查报告将相应地给予标记。如同用于代码审查的'report.dat'文件那样，'metpen.dat'可以被定制以满足某个项目的需要。

[0057] 对不同质量模型的选择可以通过对分配给各组的验证规则的组合实现。当调用同级需求进行代码或质量审查时，采用适当的质量模型进行测试。

[0058] 如上文所提及，对于基本需求必须满足的每个验证规则可以产生某个验证任务的某个同级需求。例如，如果某需求的验证规则明确要求它需要通过代码审查，那么产生一个同级，并且如果该同级通过了代码审查，那么该需求被认为通过了代码审查。

[0059] 相应地，可以在某个软件开发项目中采用自动化需求验证工具 115，以便允许基于需求的开发和验证进程的自动化集成，并且为项目团队的所有成员（包括管理在内）提供对动态更新的跟踪和验证需求矩阵的全球访问。如本文所述，在一个实施方案中，自动化需求验证工具 115 可以利用可扩展标记语言（XML）实现。

[0060] 为了使需求跟踪可重复处理自动化，必须支持一种对测试验证处理的三角测量（triangulation）。这种三角测量包括三个矢量：被映射的需求（包括某过程的静态关联、各过程的聚集体、类或者任何执行工件）、结构性覆盖（代码覆盖的动态度量和测试用例执行）以及测试用例（待执行的功能性）。需求跟踪的实现由于软件编程抽象（software programming abstractions）（比如重载、多重继承，通用参数化（generic parameterizations），和动态调度表）而极大地复杂化。本发明实现这一跟踪的办法是集成所需要的功能性规定的静态和动态矢量。本发明所实现的跟踪，包括对静态分析&代码覆盖工具 120 的利用，扩展到软件及其运行时间工件的各个用例。除了被映射的需求之外，测试用例（或使用例）必须明确映射为供任何待执行的基于测试的验证任务之用的谓词（predicate）。

处理回顾

[0061] 图 2 示意了以电子方式管理软件开发需求的处理 200 示例性步骤。并不是图 2 的所有步骤都必须按照图示顺序出现，本领域的普通技术人员在阅读了本文的教导之后对这一点应当是一目了然的。结合下文的讨论，其它的操作性和结构性实施方案对本领域的普通技术人员来说也是一目了然的。这些步骤将在下文详细讨论。

[0062] 在步骤 205，建立软件开发项目。例如，结合图 1B 描述的项目模块 145 可以用来实现步骤 205。在一个实施方案中，步骤 205 包括为项目识别各用户。在另一个实施方案中，步骤 205 包括基于被指定执行类似动作的用户组创建该项目的多个角色。例如，项目管理者角色可以创建为具有对处理 200 实施控制的权限，可以产生至少一个额外的角色，比如开发者角色、测试工程师角色、独立验证和确认（validation）角色、和质量保证角色。在该实施方案中，步骤 205 可以包括向各角色分配权限，以定义用户可以完成的各项动作，比如

创建低价需求权限、创建衍生需求权限、生成子项目权限、编辑线程属性权限和人工验证权限，步骤 205 也可以包括为用户选择各角色。

[0063] 在步骤 210，基于从需求源捕获的需求信息来定义用于项目的各个需求。例如，结合图 1B 描述的需求模块 140 可以用来实现步骤 210。在一个实施方案中，步骤 210 包括基于需求信息捕获用于该项目的高级需求，基于需求信息来定义该项目至少一个额外需求。所述的额外需求可以定义完成用于该项目的被捕获和分配的高级需求中一部分的低级需求、从用于该项目的被定义高级需求或者低价需求推断而得到的衍生需求，和/或作为用于该项目的被定义高级、低级或衍生需求的副本的同级需求。额外需求信息可以包括任何给定需求相关的测试用例。在被定义为原始需求信息的一部分测试用例不存在的情况下，在执行基于测试的验证任务之前用户必须输入一个测试用例。

[0064] 在步骤 212，对于每个需求，将已经针对该需求开发的源代码和该项目关联起来。例如，结合图 1B 描述的项目模块 145 可以用于实现步骤 212。此外，如果还没有为某个需求开发源代码，则步骤 212 可以促进基于该需求的开发工作，其中需求分配在源代码实现之前完成。在该实施方案中，将源代码和项目的关联直到步骤 215 才会出现。在步骤 212，相应的子项目可以在项目被定义的需求分配给用户的时候建立。

[0065] 可选地，处理 200 可以包括为用户产生状态报告，该用户概括被分配给用户的被定义需求，并且在项目期间更新状态报告，以纳入分析、代码覆盖和单元测试相应结果的各个额外步骤。

[0066] 在步骤 215，源代码中指明的过程被映射到用于该项目的被定义需求。例如，结合图 1B 描述的映射&分析模块可以用于实现步骤 215。在一个实施方案中，步骤 215 包括调用静态分析工具，比如结合图 1A 描述的静态分析&代码覆盖工具 120，以分析源代码来识别源代码中的各个过程。在该实施方案中，静态分析工具也可以被调用来对源代码进行分析，包括对相关的源代码文件应用项目编码规则和质量度量。此外，该静态分析&代码覆盖工具 120 可以用于测试和度量外部运行检验中的代码覆盖。在另一个实施方案中，步骤 215 包括调用单位测试工具，比如结合图 1A 描述的单元/集成测试工具 125，

以便对被映射过程进行用例测试。

[0067] 在步骤220，基于对被映射过程执行的分析、代码覆盖测量和单元测试中的一个或多个结果，来验证该项目的各被定义需求。例如，可以采用结合图1B描述的验证模块155来实现步骤220。在一个实施方案中，步骤220包括：如果分析的结果表明源代码和项目编码规则和质量度量一致的话，则验证该项目的被定义的需求。在该实施方案中，可以基于分析结果对未一致情况进行追踪，并且可以就未一致的情况提出报告，包括提供有关不一致的分布信息。在另一个实施方案中，步骤220包括如果测试用例产生所需的结果，则验证该项目的被定义需求。

[0068] 在另一个实施方案中，步骤220包括下列额外步骤：提供用于该项目的被定义需求的状态，指明每个被定义需求是否被验证或未获验证；并且产生集成的需求跟踪和验证矩阵，以便动态跟踪被定义需求的实现情况和被定义需求的状态。在该实施方案中，项目的子项目可以通过将项目的被定义需求分配给参加项目工作的用户而建立，并且子项目可以与集成的需求跟踪和验证矩阵同步。

[0069] 可选地，处理200进一步包括允许用户通过通信网络访问项目的步骤。例如，结合图1B描述的网络接口模块160可以用于实现该联网步骤。在一个实施方案中，需求跟踪和验证矩阵的完整性可以通过监视用户何时经通信网络签出和签入其相应的子项目的方式而保持。换言之，处理200可以支持一种单元测试工作流场景，该场景包括通过高级、低级和衍生需求而进行的需求跟踪和测试验证并且有助于将这些需求和源代码过程或方法映射在一起。映射后的需求可以后续提供给开发者或测试者进行测试规范（specification)创建和测试验证。处理200也可有助于从这些测试规范创建测试用例。处理200可以自动跟踪该单元测试场景的结果而追溯到需求源，以确保其调度中的需求跟踪矩阵（RTM）完整性和用户工作区（例如，项目/子项目）安全性和灵活性。在一个实施方案中，用户可以就“签入”的子项目展开工作，其中用户所执行的所有任务可以被动态跟踪并且与RTM的更新同步，或者用户可以就“签出”的子项目展开工作，其中可以在任何地点执行验证任务，并且可以后续在用户返回工作（签入的子项目时）将结果和当前RTM同步。

示例性 XML 实现的详细描述

[0070] 正如本文所述，自动化需求验证工具 115 和自动化需求管理处理 200 可以用在软件开发项目中，以缩短高级需求跟踪工具和低级代码测试工具之间的差距，并且在一个实施方案中，它们可以利用 XML 结构实现。

[0071] 例如，需求管理数据可以按被称为“线程”的 XML 构造方式维持，使得每个被跟踪的需求具有一个相应的 XML 线程。按照此种方式，如本文所指，术语“线程”和“需求”实质上是可以互换的，只是线程更准确地描述了基于需求的任务。例如，如图 3A 所示，线程 305 可以包括描述该需求以及和其它需求 310 之间关系的元数据，以及源代码文件中被映射到需求 315 的过程的有关原型信息和与该需求验证有关的分析与测试结果信息 320。所有原型信息可以保持在线程自身中，包括利用<Prototypex/Prototype> XML 块的代码覆盖信息。

[0072] 除分层信息（即，线程关系数据可以放置在线程自身中，但是所有的线程一起构成有结合力的层次结构）外，一个线程的所有相关信息均可以放置在线程自身中。如此，一个线程可以在很大程度上成为一个自包含的对象，允许程序能够明智（sensibly）地使用线程信息，而无需关于线程所属项目的任何知识，包括用户、角色和权限信息。例如，图 3C 示意了一种项目机制 340，并且可以看到线程 XML 文件 345 可以独立于项目 XML 文件 350、355 和 360，而项目 XML 文件 350、355 和 360 可以用于创建有结合力项目并且将线程信息以有益和直观的方式呈现给用户。

[0073] 线程的定义可以规定数据聚合称为“线程”适量（proper）和必须存在的字段（在 XML 特定范畴下也称为“元素”）最小数量及其排列。放入到这些字段中的信息是否和真实世界数据有任何相关性超出了线程定义的范围。线程的定义仅和唯一参考识别符（例如 ref=” ”元数据）有关，而和线程可读所采用的实际字符串无关。

[0074] 图 3B 示意了线程 XML 文件 335 可以由调用应用/程序 325 通过被称为 CXML Wrapper 的 XML 包装器类加以操控的情形。尽管可以直接从/往线程中读出/写入，CXML Wrapper 可以用于强制执行 ref=” “ XML 惯例（convention），由此确保线程按一致的符合标准

的方式写和重写。

[0075] 图 3D 示意了线程在存储器中的示例性表示。CXML Wrapper 类是基于 CGenericXMLWrapper XML 包装器类的，后者可以实现用于将 XML 文件转换为被称为 ReqDataContainer365 的分层对象树。概念上来看，ReqDataContainer 365 可以维持关于其自身的分层信息（例如，其父母和子女是谁）以及关于自己数据的分层信息（例如，名字、唯一参考识别符和数据）。CGenericXMLWrapper 类可以为读取 XML 文件时遭遇到的每个元素创建一个 ReqDataContainer。该元素的名字和唯一参考号可以存储，有别于任何数据。分层信息可以基于相关因子（其中一个因子比如是，是否该元素具有数据和父元素）从上下文方面加以确定。如此，在 ReqDataContainer 的树中，每个 ReqDataContainer 都可以类似于一个具有指向树上同级的指针的节点。

[0076] 在遭遇到不具有参考号的元素的情况下，可以给该元素分配一个参考号，以一个合理高的“未知界限”作为起点。如此，希望检索该元素的数据的程序必须提前了解未知界限的值（该值可以定义在 XMLWrapper.h 类中），以及有关多少未设参考号的元素正被读取以及已经被读取的其它因素。实际上，可以更有效地定义线程树，为线程的每个元素使用唯一参考号（或其它唯一识别符），而不是依赖于自动分配参考号。但是，自动分配参考号可以有助于避免由于程序的故障部分不向其新的元素分配参考号而导致整个线程（或 _trace.xml 文件）不能被访问的情形。

[0077] 一旦 CGenericXMLWrapper 类创建了一个 ReqDataContainer 树，如图 3D 所示，调用程序可以完全自由地修改 ReqDataContainer。例如，调用程序可以添加或去除 ReqDataContainer，并且修改已有 ReqDataContainer 中的数据。在调用程序完成对 ReqDataContainer 树的修改之后，它可以调用该包装器类的一种被称为 WriteAsXMLQ 的 XML 方法，将 wrapper class 转换 ReqDataContainer 树返回 XML 文件，并且将 XML 文件写入存储设备。

[0078] ReqDataContainer365 在设计时可以尽可能多地模仿 XML。ReqDataContainer 并不必需基于它们是否有数据、子容器、同级或任

何其它上下文属性（property）而加以分类。但是，在实际中，ReqDataContainer之间应当有所区别，使得数据可以相应地加以组织。

[0079] ReqDataContainer树实现方案的灵活特性可以允许不同的程序将其自身的数据添加到线程中，而不会彼此干扰。例如，在一个实施方案中，缺陷跟踪信息和函数原型XML块可以添加到线程“范围”，而不会混淆其主要的载荷、个体需求。这些是随着个体跟踪需要增加（在此情况下，验证向下延伸到个别函数）而添加到线程中的数据的例子，并且在此方式下，将和需求有关的所有数据及其跟踪信息装入单个线程中的做法可以实现用户程序的灵活性和自由度。ReqDataContainer树实现也可以定义对数据编码的标准途径。如此，希望利用其它程序提供的特征程序仅需要理解所使用的XML块协定（convention），以便围绕它建立其自身功能性。

[0080] 在一个实施方案中，可以采用被称为RDCPtr, iRDCPtr, 和CwrapperLibrary的类，（1）将重复的XML块组织成“垂直数据”的“堆”，以避免和父节点接触的需要，（2）允许有顺序地访问同级，而不必和父节点通信，（3）提供ReqDataContainer和所存储的XML文件的单个元素之间的一对一关联。

[0081] 避免依靠父节点获得子层次关系信息的做法可以改善代码的完整性。例如，程序通常需要访问某个XML块的所有实例，比如从某个线程中获得原型信息。不使用RDCPtr类便能从线程获得原型信息的函数的伪代码表示可以表达如下：

```
ReqDataContainer prototype_parent =
      current_thread.GetElement(REF_TEST_SPECIFICATION) ;
ReqDataContainer prototype =
    prototype_parent.GetElement (REF_PROTOTYPE) ;
 while (prototype != NULL) {
    PrintPrototypeInfo (prototype) ;
    prototype = prototype_parent .NextElement (prototype) ;
 }
```

使用RDCPtr类从线程获得原型信息的函数的伪代码表示可以表达如下：

```
RDCPtr prototype = current_thread
    .GetElement (REF_TEST_SPECIFICATION)
    .GetElement (REF_PROTOTYPE) ;
while (prototype != NULL) {
    PrintPrototypeInfo (prototype) ;
    prototype = prototype. GetNext ();
}
```

[0082] 两个实现之间无代码变动（cleanliness）方面的改进可以是显著的，如果考虑到仅需传递一个线程所具有的原型“列表”中的第一原型的 RDCPtr，则可以将这一列表作为一个整体到处传递，这是因为该列表中的第一原型知道其它原型的位置。如此，不再需要由一个函数将两个 ReqDataContainer 指针（一个用于该元素，一个用于它的父元素）到处传递，相反仅需要将一个 RDCPtr 对象到处传递，便可以使每个函数访问所有的原型。此外，wrapper 类仅需要服务（servie）一个应用调用（即得到第一原型的调用），而不必服务其它调用来获得任何额外的原型信息。如此，重复的 XML 数据块可以表示为“垂直”数据，如图 3D 和 3E 所示。第一原型可以和任何其它的 XML 块一样对待，而后续读取的原型块可以“堆”在第一原型的上面。RDCPtr 类体现了 ReqDataContainer 堆这一理念。

[0083] 如此，调用程序不必和 ReqDataContainer 直接接触。所有的数据访问可以由 RDCPtr 完成，后者可以实现如图 3D 和 3E 所示的唯一的“垂直数据”堆机制。由于它们可以充作智能指针，RDCPtr 可以提供专有的访问 ReqDataContainer 的手段。即使某个元素没有垂直数据（即，只有该元素的一次迭代），它仍然可以采用 RDCPtr 的形式返回。这样，函数不必被重写来接受 ReqDataContainer 和 RDCPtr，所有的函数可以简单地接受一个 RDCPtr。通过传递一个较小的 RDCPtr，整个调用程序可以随机或顺序地访问整个垂直数据堆。该特征可以促成更容易、较无变动（cleaner）且更准确的代码实现。此外，通过采用 RDCPtr，调用程序不必向/从存储器创建或删除 ReqDataContainer，如本文所述该过程可以完全自动地进行。

[0084] iRDCPtr 类可以提供将 ReqDataContainer“左和右”编排进列表的机制。iRDCPtr 类不是由调用程序使用，但是可由后端（例

如，用于和 XML 文件接口的包装器）使用来实现通过同级和写递归算法的顺序搜索法搜索 ReqDataContainer 树（ReqDataContainer 不在 ReqDataContainer 树记录其同级）。调用程序可以通过索引（即通过唯一参考识别符）访问 ReqDataContainer 的子容器，但不能顺序地遍历 ReqDataContainer 的子容器以搜索找到它所需要的子容器，但是对于内部进程来说，CGenericXMLWrapper 可以使用 iRDCPtr 来顺序地遍历 ReqDataContainer 的子容器。

[0085] 调用程序应当使用 CWrapperLibrary 类来得到用于 XML 文件的包装器。出于存储器管理的目的，自动化需求验证工具 115 及其后端可以使用一个类来管理资源的分配和删除。例如，CXMLWrapper 类可以用于创建所需要的 ReqDataContainers、RDCPtrs 和 iRDCPtrs。调用程序可以基于请求获得 RDCPtrs，并且由 CXMLWrapper 表示的整个 XML 文件可以由 CXML Wrapper 创建和删除。继而，CXMLWrapper 可以注册利用 GlobalRDCPool 对象创建的所有对象（比如 ReqDataContainers），GlobalRDCPool 对象可以由调用程序在退出的时候清空。

[0086] 此外，自动化需求验证工具 115 及其后端可以利用被称作 CWrapperLibrary 的类来管理 CXMLWrappers，管理方式和 GlobalRDCPool 类管理 RDCPtrs 的方式相同。由于 CWrapperLibrary 的存在，调用程序根本不必创建包装器。相反，调用程序可以向 ReqDataContainer 树请求获得 XML 文件。如果还未为该请求的 XML 文件建立 ReqDataContainer 树，则可以使用 CWrapperLibrary 类来创建包装器，让包装器建立 ReqDataContainer 树，并且向调用程序返回一个指向所创建的包装器的指针。如此，CWrapperLibrary 可以组织调用程序正在使用的所有 CGenericXMLWrappers。例如，如果调用程序请求和“file1_trace.xml”对应的包装器，CXMLWrapper 可以搜索已经建立的包装器库来查找 file1_trace.xml 的 CXMLWrapper 对象。如果用于 file1_trace.xml 的 CXML Wrapper 对象已经被创建，则 CXMLWrapper 可以简单地将指向已经生成的 CXMLWrapper 的指针返回给调用程序，否则 CWrapperLibrary 可以为 file1_trace.xml 产生一个新的 CXMLWrapper 对象，并且将指向新对象的指针返回给调用程序。

[0087] 象 RDCPtrs 一样，CXMLWrapper 对调用程序隐藏存储器的分配，使得调用程序不负责释放 CWrapperLibrary 传递回来的对象。同样，当调用程序使用 CWrapperLibrary 的时候，该程序的所有部分可以确定是工作在相同的 ReqDataContainer 树上(ReqDataContainer 是 XML 文件中的单个元素，故此 ReqDataContainer 的树表示整个 XML 文件），并且调用程序并不必要遍历该程序实行同步。

[0088] 通常，调用程序将调用 GetFile() 方法来检索 RDCPtr 中的文件根元素，并且将不会再对包装器作任何进一步的调用。调用程序不删除被创建的包装器。相反，EmptyWrappers() 方法（它也将所有已被开启的包装器写入存储设备）可以用替代方式删除包装器。如此，CWrapperLibrary 类可以用于确保自动化需求管理程序的所有工作都在相同的 ReqDataContainer 树上并且有助于存储器管理。

[0089] 各个存储器管理类在一起使用的时候可以得到一个不间断存在的 RDCPtrs（即，调用程序在会话开始时接收的 RDCPtr 与调用程序在会话期间后续时间接收的 RDCPtr 相同），以此允许调用程序使用的相同 GlobalRDCPool 和 CWrapperLibrary 各个部分将 RDCPtrs 到处传递。因此，调用程序决定要求存储器管理类清空它们自己的时间是一个重要的决策点。出于充分利用存储器的原因，在某些情况下，强制清空包装器和 RDCPtrs 甚至可以是可取的。这种存储器管理技术可以导致在程序的生命周期中存储器利用率的提高。一旦存储器利用率达到一定的水平，它不太可能进一步增加，但是对于在其生命周期中可能打开数十个或甚至数以百计 XML 文件的程序来说，强制清空存储器管理类可以是有益的。但是应当注意对存储器管理类所管理的资源的引用在对这些资源清空后将不再需要。

[0090] 在一个实施方案中，线程可以转换为 SQL 格式，以便全局网络存储和访问。该进程可以由接受来自自动化需求验证工具 115 的上载请求的网络（web）服务支持。同一个网络服务可以基于来自网络用户的请求产生跟踪和验证报告。另外，该网络服务可以向自动化需求验证工具 115 提供下载请求。这些网络服务的结果是，可以促进全局项目同步。

示例性图形用户接口实现

[0091] 图 4A-4J 示意了用于由 LDRA™开发的软件需求管理工具 TBmanager™的图形用户接口（GUI）的示例性实现。TBmanager™可以用于促进软件需求的管理和分配，以便开发、测试和验证。本领域的普通技术人员将理解到，GUI 设计不必受限于图 4A-4J 中示意的设计，其它的 GUI 设计也可以采用。在该实施方案中，可以利用需求捕获工具（比如 Geensys™开发的 TBreq™）首先创建一个项目文件（例如，具有".tbp"扩展），需求捕获工具可以从诸如 Telelogic DOORS® 或 IBM Rational RequisitePro®, MS Excel®, MS Word®的需求信息源和其它需求信息源捕获用于软件开发项目的需求信息。

定义角色和用户

[0092] 图 4A 示意了在初始设置期间定义项目的角色的示例性 GUI401。角色可以用于将执行类似或有关任务的用户编组在一起；例如，可以对项目的所有开发者创建并应用一个角色。每个用户可以分配一个角色。一个用户可以采用项目管理者的角色。项目管理者角色可以对需求管理进程的每个方面进行控制，可以负责创建角色和用户以及委托需求。例如，项目管理者可以创建和分配角色，授权其他用户创建角色和委托需求的能力。

[0093] 如图 4A 所示，GUI401 的新角色字段 402 可以用于输入新角色的名字，并且可以按压 Add 按钮添加该新角色。示例性角色 403 可以包括开发者角色、GUI 设计者角色、独立验证和确认（IVV）角色、质量保证（QA）角色和测试工程师角色以及其它角色。如图 4A 所示，可以利用 Remove Selected（去除选定）按钮来去除某个已添加角色，并且可以选择 Import Roles From File（从文件引入角色）按钮来引进已经定义在文本文件或另一个项目中的角色。

[0094] 可以给各角色分配权限，为相应的用户赋予不同级别的功能，比如定义用户可以访问的需求信息和他们可以完成的动作。如图 4A 所示，可以利用 Assign Permission（分配权限）按钮将权限分配给角色 403。示例性权限 404 可以包括“Create Low Level (LL) Reqs”（创建低级需求），它可以被分配给某角色来允许相应的用户创建低级需求；"Create Derived (DV) Reqs"（创建衍生需求），它可以被分配给某角色来允许相应的用户创建衍生需求；“Make SubProjects（生成子项目）”，它可以允许相应用户为项目需求分配用户；"Edit

Thread Properties"（编辑线程属性），它可以分配给某角色以允许相应的用户通过本文描述的 Requirements View（需求视图）编辑可用的需求属性；以及，“Manual Verification”（人工验证），它可以被分配给某角色以便使相应用户访问本文描述的 Verification View（验证视图）。

[0095] 角色可以用于使用户之间的分配需求多样化。如此，相同的需求（通过同级需求）可以被分配给两或多个不同角色的用户，而同时保持对开发和验证任务的跟踪。例如，相同的需求可以分配给开发者进行编码，给 QA 进行测试，给 IVV 进行独立的验证和确认。

[0096] 类似地，图 4B 示意了为项目定义用户的示意性 GUI405。如图 4B 所示，新的用户字段 406 可以用于为新用户输入名字，为该新用户选择相应的角色，并且可以通过按压 Add 按钮添加新用户。添加的用户可以显示在用户字段 407。可以利用 Remove Selected（去除选定）按钮从项目中去除所添加的用户，可以选择 Import Roles From File（从文件引进角色）按钮来引进此前在另一文件定义的用户。

[0097] 当跟随有本文描述的子项目更新时，在设立之后添加新用户和角色可以安全进行。但是一旦如本文所述为用户分配了需求，则用户和角色将不能删除。例如，删除一个角色将导致某用户丢失在项目工作的必要权限，删除用户将导致需求未被分配。因此，在删除角色和用户时应当留意。

基于类型视图进行解析

[0098] 可以引进需求文档并且对之解析。解析工作可以基于类型进行。解析提要必须描述在类型文件中。如图 7 所示，许多类型文件可以预先定义，涵盖通常使用的工具，比如图表软件（Visio 702）；需求管理工具（DOORS 704 和 706，以及 Rational Requisite Pro 708）；字处理程序（比如 MS Word 710）；电子表格程序（Excel 712）；以及创作和出版软件（比如 Framemaker 714）。类型文件可以是一个转换器，使得它可以理解需求如何描述在参考文件中。

[0099] 图 8 示意了 MS Word 文件中的需求例子。如图 8 所示，必须给需求 ID 802 和需求文本 804 采用合适的样式（style），使得文件可以按作者希望的方式解析。

[0100] 图 9 和图 10 示意了类型编辑器的例子。文档中采用的 MS Word 样式，包括需求 ID 802 和需求正文 804 在内，必须描述在类型编辑器字段 902 和 1002 中，以便 MS 字文件可以被正确解析。

组视图

[0101] 项目的各验证组可以创建在一个用户下。图 5 示意了在“Verification Groups”对话框 500 下选择、创建、重新命名或删除组视图的例子。如图 5 所示，点击“New Group”（新组）按钮 502 创建验证组。当点击“New Group”按钮 502 的时候，对话框“Enter Group Information”（输入组信息）518 出现。可以在名字文本字段 514 输入验证组的名字。在此例子中，在名字文本字段 514 中输入组名字“High Priority（高优先级）”。此外，在描述文本字段 516 输入描述“Must be implemented in the current spiral”（必须在当前螺旋中实现）。当点击“OK”按钮的时候，一个验证组得以创建。

[0102] 一旦创建了验证组，用户可以突显该组并且点击“Rules >”(规则)按钮 508，为某个给定组编辑验证规则。图 6 示意了在“Edit Rules”（编辑规则）对话框 600 中为给定组编辑规则的例子。参看图 6，当点击“New”（新）按钮 610 时，“New Rule”（新规则）对话框 616 出现。在“New Rule”对话框下，规则分别在文本字段 602 和 604 规定需求必须满足“Unit Testing”（单元测试），并且必须由“The Requirement's Assignee”（需求受指派人）完成。可供选择的验证规则可以包括：代码审查、质量审查、系统测试、单元测试、子系统测试和集成测试。也可以判断被分配给该需求的用户是否被允许执行该测试，或者是否必须由另一用户执行该测试。当点击“OK”按钮 606 时，新规则便被创建在“High Priority”（高优先级）组下。“< Group”（组）608 按钮可以用于返回“Verification Groups”（验证组）对话框 500。“Save and Use”（保存和使用）按钮 612 可以用于保存这些规则，供将来之用。“Use Existing”（使用现有的）按钮 614 可以用于为给定组选择所保存的规则。

[0103] 返回图 5，“Rename Group”（重新命名组）按钮 504 可以用于对该组重新命名，如果它已经被创建的话。“Delete Group”（删除组）按钮 506 可以用于删除某个组。“Save and Use”（保存

和使用)按钮可以用于保存组信息设置,供将来之用。“Use Existing”(使用现有的)按钮512可以用于为给定组选择所保存的组信息设置。

[0104] 在按照上文对组定义之后,需求可以和该组关联起来。图11示意了未编组需求和编组需求的例子。在图11中, REQ 1 1102和REQ 2 1104是未编组需求。REQ 3 1106和REQ 4 1108被包含在命名为“Static Testing”(静态测试)的组别中。

同级需求视图

[0105] 图12示意了同级需求视图的例子。在图12中,两个同级需求1202和1204对应于需求1206。同级需求1202供代码审查用,同级需求1204供质量审查用。

[0106] 验证可以由该组同级需求驱动,并且每个同级需求都要求一个单独的测试运行被执行。需求只有在所有和该需求对应的同级需求被验证之后才算验证通过。

[0107] 图13示意了同级需求视图的另一个例子。在图13中,同级需求1306位于与命名为“Sam Brown”的开发者有关的命名为“Static Testing”(静态测试)1304的组别之下。同级需求1306表明验证任务是代码审查。在该例子中,同级需求的验证状态1308是“Passed”。它也表明分别命名为CashRegister.cpp和Backbeat.cpp 1310和1312源文件和同级需求1306有关。

项目视图

[0108] 如图4C所示,GUI可以有选项卡式的格式,为用户提供可供选择的不同视图(例如,项目、验证、定义需求、需求信息、分配用户、映射过程、以及过程)和可以基于其被分配角色而执行的动作。图4C示意了在选择Project View(项目视图)标签的时候可以显示的GUI 408。项目GUI 408可以显示被分配给用户和被用户分配的所有需求(例如,粗体的需求可以表示被分配给用户的需求,非粗体的需求可以表示为用户已经分配的需求)。

[0109] 项目GUI 408包括当前需求下拉菜单409,该菜单可以用于选择和显示项目的特定需求。GUI 408也包括一些按钮,比如可以用于选择供上载的项目或子项目文件的Open Project(打开项目)按钮410a,可以选择用于利用对项目需求所做的近期修改来更新/同步所有当前签入的子项目的Update Sub-Projects(更新子项目)按钮

410b,可以选择来基于项目的当前需求显示状态报告的Status Report（状态报告）按钮410c，可以选择用来对当前项目添加/上载代码源文件的Add Source（增加源）按钮410d，可以选择用来从当前项目去除当前选定的源代码文件的Remove Source（去除源）按钮410e，可以选择用来发布单元测试用的外部工具（比如LDRA Testbed™和TBrun™）的Unit Test（单元测试）按钮410f，以及如果当前用户有必要的权限的话可以选择来设立角色和用户的Setup Wizard（设置能手）按钮410g。项目GUI的其它元素分别包括可以选择用来显示用户指引的User Guide按钮411，以及可以选择用来分别退出工具和保存当前项目的Exit and Save（退出和保存）按钮413和414。

[0110] 项目GUI 408也可以显示项目树412，列明被分配给用户的各个需求和添加到项目的任何源代码文件。需求树412可以按号码、名字、参考标识符和类型显示每个需求，这里可以采用不同的图标表示不同类型的需求。此外，“X”号可以覆盖在需求图标上，以表明相应的需求是未被验证的。为了观看某个需求的更多信息，用户可以选择本文描述的Requirement Information（需求信息）标签。

[0111] 对子项目的更新可以进行，将对项目的各项需求的修改分发给受影响的子项目。为了更新子项目，用户应当首先签入他们的子项目。然后，在修改需求（例如重新分配需求，未验证需求等等）之后，可以选择Update Sub-Projects（更新子项目）按钮410b来散发该修改给所有受影响的签入子项目，使得当用户后来签出他们的子项目的时候，他们可以访问该修改。尽管只有工作在受需求修改影响的子项目的用户需要在更新子项目之前签入他们的子项目，但是所有用户应当签入它们的子项目以便完成对用户的修改（例如，当从/往项目去除/添加用户的时候）。

[0112] 在一个实施方案中，用户可以通过按压Status Report（状态报告）按钮410c产生状态报告，以显示所有和用户相关的需求的当前状态。状态报告可以提供被分配给用户或由用户分配的需求的汇总，可以显示详细的需求属性，比如需求名称、需求号、需求文件、需求体、分配给需求的用户和被分配用户的角色。随着项目的进行以及过程被映射到需求，如本文所述，状态报告可以被更新，以便纳入每个需求的被映射过程表。该表可以显示被映射过程的详细属

性，比如过程名字、源代码文件的名字和位置、所取得的路径覆盖、所取得的分支判决覆盖、所取得的 MC/DC 覆盖和所取得的语句覆盖。表中的取得覆盖字段可以保持为空，直至对被映射过程的测试已经完成，如本文所述。在单元测试完成之后，这些字段可以更新以显示所取得的覆盖。另外，在单元测试完成之后，状态报告可以指明取得所需覆盖的被测试映射过程的百分比。状态报告也可以配置为区分获验证和未获验证的需求被显示的方式，例如采用不同颜色的文本来加以显示。

[0113] 在对需求定义并且将用户分配给所被定义需求（如本文所述）之后，可以通过选择 Add Source（增加源）按钮 410d，将已经由用户开放的源代码添加到项目中。一旦添加之后，源代码可以显示在项目树 412 中。值得注意，源代码文件可以添加到项目而不是需求自身，这是因为源代码文件中的过程，而不是源代码文件自身，可以被映射到需求中，如本文所述。也可以通过选择拟在 Project Tree（项目树）412 中去除的源代码文件并且按压 Remove Source（去除源）按钮 410e，将源代码文件从项目中去除。

需求信息视图

[0114] 图 4D 示意了在 Requirement Information（需求信息）标签被选择时可以显示的 GUI 415。需求信息 GUI 415 可以显示有关当前选定需求的信息。具体地说，需求信息 GUI 415 可以将当前选定的需求的有关信息编组为三个关注区域：test configuration（测试配置）、test management（测试管理）和 requirement nomenclature（需求命名）。Test Configuration 下拉菜单 416 可以用于观看和编辑当前选定需求的测试配置特性的有关信息，Test Management 下拉菜单 418 可以用于观看和编辑当前选定需求的测试管理特性，而 Requirement Nomenclature 下拉菜单 417 可以用于观看和编辑当前选定需求的需求命名的有关信息。需求信息 GUI 415 包括 View/Edit（观看/编辑）区 419，其中可以显示和/或编辑选定需求。当不允许用户编辑和被选定需求信息有关的数值或描述时，View/Edit 区 419 可以变灰。

定义需求视图

[0115] 在项目的寿命周期期间，可能需要引入新的需求或改进现

有需求。为了定义某个需求，可以选择 Define Requirement（定义需求）标签。图 4E 示意了在选定 Define Requirement 标签的时候可以显示的 GUI 420。利用定义需求 GUI 420，用户可以根据被分配给用户的相应角色的权限，创建并且添加低级或衍生需求到项目中去。如本文所述， 高级需求包括被需求捕获工具所捕获并且引入项目中的需求。低级需求包括和高级需求有关的需求。衍生需求包括未被需求捕获工具捕获但在 TBmanager™中定义并且由高级或低级需求推知或衍生得到的需求。

[0116] 如图 4E 所示，定义需求 GUI 420 包括可以用于输入被定义需求的名字的 Requirement Name（需求名字）字段 421，可以用来选择被定义需求是否是衍生(DV)或低级(LL)需求的 Requirement Type（需求类型）单选按钮 422，可以在定义低级需求时使用来选择和该低级需求相关联的高级需求（否则可以选择 NONE（DV 需求））的 Reference Requirement（参考需求）下拉菜单 423。定义需求 GUI 420 也包括可以用来输入被定义需求的号码的 Requirement Number（需求号）字段 424。在一个实施方案中，项目的每个需求要求有唯一的识别号，它可以由 TBmanager™自动填写。定义需求 GUI 420 也包括可以用来输入到详细描述被定义需求的文件的路径的 Requirement Document Path（需求文件路径）字段 425（Browse 按钮可以用于找到该文件），可以用来输入被定义需求的简短描述的 Requirement Body（需求体）字段 426，可以用来清除定义需求 GUI 420 的所有字段的 Clear（清除）按钮 427，和可以在按压后创建新需求的 Create LL/DV（创建）需求按钮 428。在一个实施方案中，在向项目引入需求的时候，TBmanager™可以自动使用规定在模板中的数值作为该新需求的期望代码覆盖的数值。新定义的低级或衍生需求可以显示在项目树 412 中。

[0117] 在项目的寿命周期期间，可能有必要摒弃或忽略某个需求，以便它不再被考虑用于验证。用户可以通过项目 GUI 408 选择拟在项目树 412 中解除分配的需求，并且通过定义需求 GUI 420 利用 De-allocate/Re- allocate Current Requirement（解除/重新分配当前需求）按钮 429 解除对选定需求的分配。De-allocate/Re-allocate Current Requirement 按钮 429 可以配置为相对于选定需求

的上下文相关的按钮，使得它根据需要在“De-Allocate Current Requirement”（解除当前需求）和“Re-allocate Current Requirement”（重新分配当前需求）之间切换。如此，用户可以通过项目 GUI 408 选择项目树 412 中的被解除分配的需求，按相似方式重新分配该被解除分配的需求，并且可以通过定义需求 GUI 420 利用 De-allocate/Re-allocate Current Requirement 按钮 429 重新分配被选定需求。

分配用户视图

[0118] 已经建立项目的角色和用户之后，可以通过选择 Assign Users 标签，为用户分配要工作的需求。图 4F 示意了在 Assign Users 标签被选定的时候可以显示的 GUI 430。利用该分配用户 GUI 430，具有必要权限的用户可以分配各用户给各个需求，并且创建同级需求。如本文所述，可以出于需求多样化和任务管理的需要，在 TBmanager™中创建同级需求，并且代表项目的高级、低级或衍生需求的副本。拟分配的需求可以通过项目 GUI 408 从项目树 412 中选出，或者通过分配用户 GUI 430 从 Current Requirement（当前需求）下拉菜单中选出。

[0119] 分配用户 GUI 430 包括“Assigned to this Requirement”（分配给该需求）字段，它有一个可以显示被分配给当前需求的用户名字的 User 字段 431 和可以显示被分配给该用户的角色的 Role 字段 432。分配用户 GUI 430 也包括“Assign User”（分配用户）字段，它有一个可以用于基于项目的用户 435 的相应角色 434 识别哪些用户可供选择的 Filter by Role（按角色过滤）下拉菜单 433。可以选择 Assign to Requirement（分配给需求）按钮 436 来将识别出用户 435 中的一个选定用户分配给当前需求。

[0120] 为了完成该分配并且为该用户创建一个子项目，可以通过项目 GUI 408 然后由 Update Sub- Projects（更新子项目）按钮 410b 选择 Save（保存）按钮 414。在一个实施方案中，可以在项目目录中建立命名为“Sub-Projects（子项目）”的目录，以容纳所有该用户的子项目。每个用户可以具有单独的子项目目录，它可以在首次分配一个需求的时候自动创建。然后用户可以打开他们各自的子项目来就被分配的需求开展工作。用户可以将他们的子项目目录复制到不

同的机器（例如他们自己的安装有TBmanager™的工作站或笔记本）来开展所分配需求的工作。用户然后可以在其机器上启动TBmanager™，通过项目GUI 408选择Open Project（打开项目）按钮410，导航到子项目目录，并且选择适当的子目录来开展工作。用户可以被要求在登陆时输入名字。然后，项目GUI 408可以刷新自身，以显示用户项目和所分配的需求。

[0121] 分配用户GUI 430也可以用来创建同级需求，它是已有高级、低级或衍生需求的副本。同级需求可以，在除了被分配给该同级需求的唯一识别符以外，和相应的高级、低级或衍生基本需求相同。同级需求可以在被创建的时候复制基本需求线程。如此，同级需求可以允许用户将相同的需求分配给具有不同角色的多个用户（一个同级不能被分配给具有相同角色的两个用户）

[0122] 为了创建同级需求，通过选择Create Sibling Req + Assign（创建同级需求+分配）按钮437，经分配用户GUI 430显示的用户435之一可以被选定并且分配给同级需求。在一个实施方案中，被分配给同级需求的用户应当具有不同于被分配给基本需求的用户的角色。新创建的同级需求可以通过项目GUI 408观看，并且在项目树412中将其显示为基本需求的子需求。仅仅有制作子项目权限的用户应当被允许创建同级需求。

[0123] 由于同级需求可以被看成相应基本需求的严格副本，如果基本需求发生变化，则其所有的相关同级需求应当被解除分配，如本文所述。然后，修改后的基本需求的新同级需求可以被创建并且分配给适当的用户。如果基本需求有新过程分配给它，有它的需求体发生改变，或者任何其它的可能影响据以验证基本需求的条件的变化发生，那么应当创建新的同级需求。

映射过程视图

[0124] 图4G示意了在选择Map Procedures（映射过程）标签的时候可以显示的GUI 438。在向项目添加用户开发的源代码文件之后，如本文所述，映射过程GUI 438可以用于选择一个添加的源代码文件，为可用的过程分析该选定源代码文件，并且将选定的过程映射到一个当前选定需求。可以利用Project Source Files（项目源文件）下拉菜单439，来选择已经被添加到当前项目的源代码文件并且返回在选

定源代码文件中找到的过程列表 442。例如，可用调用 LDRA Testbed™ 工具来分析选定源代码文件并且将识别出的过程列表 442 返回给 TBmanager™。

[0125]可以通过分别选择 Analyze Procedures（分析过程）按钮 440 和 Analyze Interactively（交互分析）按钮 441，以静态和/或交互方式分析选定源代码文件与项目编码规则和质量度量之间的一致性。也可以调用 LDRA Testbed™来对选定源代码文件进行静态和/或交互的分析。如果希望采用交互分析，那么用户可以进行代码审查、质量审查和设计审查分析。完成分析之后，映射过程 GUI 438 可以显示在选定源代码文件中找到的识别过程 442。利用 Map to Current Requirement（映射到当前需求）按钮 444，用户可以将某个识别出的过程 442 映射到当前需求，后者可以通过项目 GUI 408 利用 Current Requirement（当前需求）下拉菜单或 Project Tree（项目树）412 选择。类似地，可以选择 Unmap Selected（解除选定的映射关系）按钮 443 来解除某个识别出的过程 442 和当前需求的映射关系。在将过程映射到需求之后，用户可以选择 Procedures（过程）标签来观看被映射到当前需求的过程的列表。

过程视图

[0126] 用户可以选择当前需求的映射过程并且通过过程 GUI 445 观看其相应的输入和输出（I/O），如图 4H 所示。过程 GUI 445 包括一个 Select Procedure 下拉菜单 446，它可以用于从映射到当前需求的各过程中选择一个过程。可以从 Current Requirement（当前需求）下拉菜单或通过项目 GUI 408 从 Project Tree（项目树）412 选择当前需求。作为选择映射后过程的结果，过程 GUI 445 可以被更新，以显示 I/O 变量字段 447 中的选定被映射过程的 I/O 变量。另外，过程 GUI 445 包括可以用来为当前需求修改期望代码覆盖值（例如，语句、分支、MC/DC、和路径覆盖值）的 Desired Coverage（所需覆盖）字段 449，和可以用来在单元测试之后观看所取得覆盖的 Coverage Achieved at Verification（验证所取得的覆盖）字段 450。可以选择 Unmap This Procedure （解除该过程的映射关系）448 来解除选定被映射过程和当前需求之间的映射关系。

[0127] 一旦各过程被映射给需求，就可以启动单元测试。首先，

可以从 Project Tree （项目树）412 或通过项目 GUI 408 从 Current Requirement（当前需求）下拉菜单 409 选择有待测试的被映射过程的需求。接下来，为了启动单元测试，在按压 Unit Test（单元测试）按钮 410f 后，包含被映射到选定需求的过程的源代码文件可以被选择。然后，可以调用外部单元测试工具来完成单元测试。例如，LDRA TBrun™是可以经 LDRA Testbed™工具调用的单元测试工具。利用该外部测试工具，可以创建用于被映射过程的测试用例。然后，可以选择待测试的被映射过程，可以输入测试用例的各个输入和期望输出，并且运行测试用例。

[0128] 如果测试用例的结果是令人满意的并且取得了期望的代码覆盖，那么应当确认需求验证结束。在确认需求验证的完成之前，对某个需求的每个被映射过程应当采用至少一个测试用例来进行测试。在关闭外部单元测试工具之后，TBmanager™可以刷新 Project Tree （项目树）412 来显示该需求为通过验证的（例如，相应的需求图标在显示时可以不带“X”符号）。项目管理者现在可以审查项目的单元测试信息，评估用户的工作。

验证视图

[0129] 一旦用户对某需求的各被映射过程进行了单元测试，就可以在 Project Tree 412 将该需求显示为“已获验证”，如项目 GUI 408 所示。但是，由于对映射过程的单元测试可能没有取得期望的覆盖，项目管理者或具有必要权限的其它用户应当审查单元测试的结果，并且判断是否该需求实际上应当被视为已获验证。

[0130] 取决于被分配给用户的权限，用户可以通过选择 Verification 标签来访问验证 GUI 451，验证某需求是否已经达到期望的代码覆盖，如图 4I 所示。为了验证某个需求，用户可以经项目 GUI408 或经验证 GUI 451 从 Current Requirement 下拉菜单，从 Project Tree 412 中选择拟验证的需求。验证 GUI 451 包括显示被选需求 452 的当前状态的 Details（细节）字段 453。Details 字段 453 指明被分配给选定需求 452 的用户及其角色、期望的代码覆盖、和取得的代码覆盖，后者可以根据取得期望覆盖的被映射过程的百分比加以显示。如果用户对所选定需求取得的覆盖满意，那么用户可以按压 Verify（验证）按钮 454 将选定需求的当前状态维持为已获验证。但

是，如果用户对选定需求所取得的覆盖不满意，那么用户可以按压Un-verify(未验证)按钮455，将选定需求的当前状态改为未获验证。由于将需求标记为未获验证的做法改变了该需求，那么用户应当更新任何和该需求相关的子项目。

[0131] 对于同级需求的验证，同级需求在相当程度上象低级需求。换言之，为了让基本需求被看作为已获验证，它的所有同级需求必须先获验证。例如，如果开发者用户已经分配了一个低级需求，它具有两个同级需求，一个用于IVV用户而另一个用于QA用户，开发者不能指明该低级需求是获验证的，直到IVV和QA用户将相应的同级需求标记为获验证。

[0132] 一个TBmanager™的示例性用途场景如下文所述。首先，项目管理者可以基于捕获的高级需求建立一个项目、创建低级、衍生和同级需求（如果恰当的话），将各需求分配给各个用户。用户可以签出它们各自的子项目，所述子项目是在项目管理者将需求分配给用户时建立的。换言之，用户可以出于在远程展开子项目工作的考虑制作他们的子项目的副本。用户可以通过映射过程、测试和分析源代码等操作，修改子项目。然后，用户可以签入它们的修改的子项目。换言之，用户可以将子项目的更新内容复制回项目的目录树。项目管理者可以在后续登录，并且获得项目所有方面的完整的当前视图。

缺陷视图

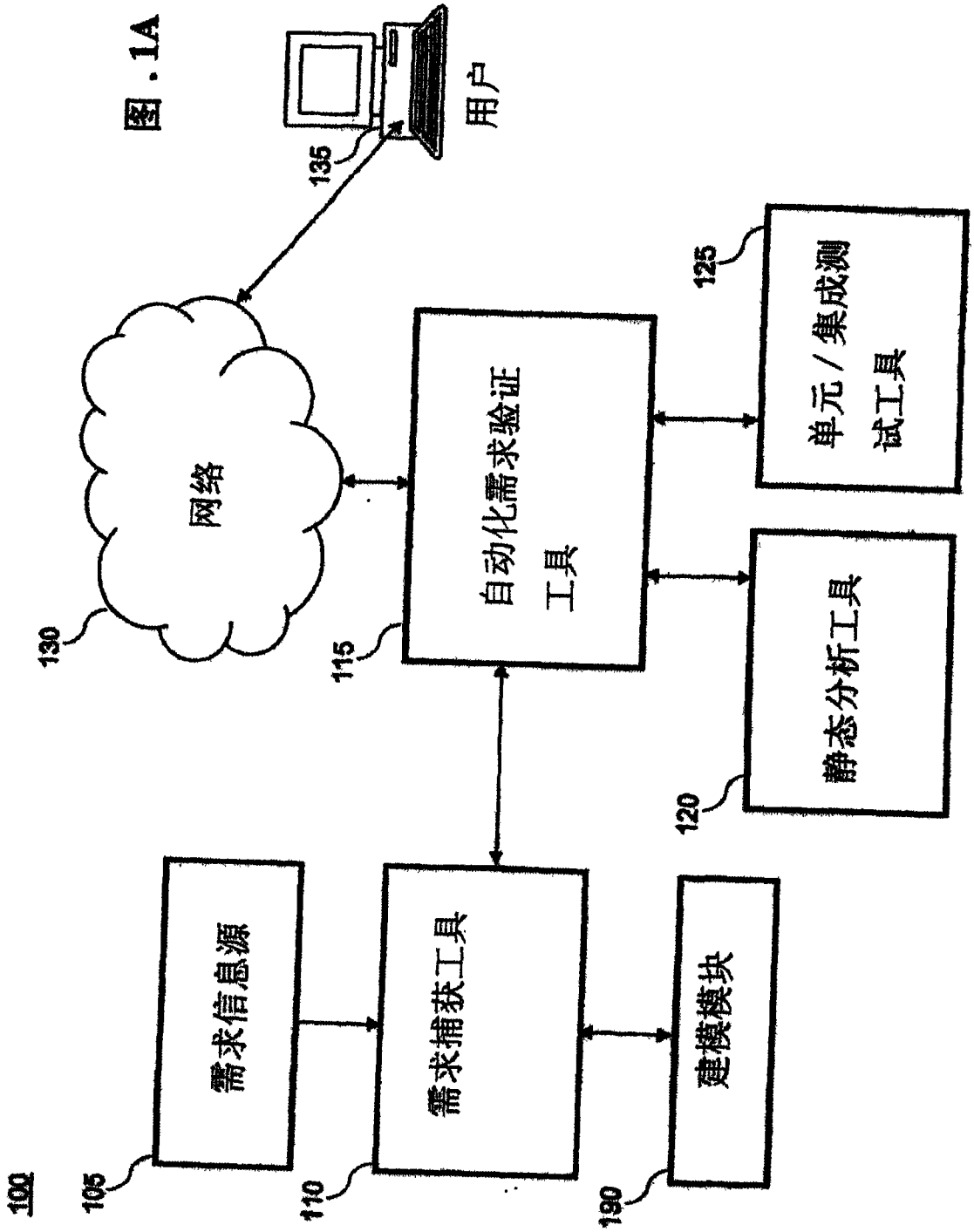
[0133] 图4J示意了当缺陷标签被选定时可以显示的GUI 456。在一个实施方案中，用户可以通过缺陷GUI 456跟踪项目源代码中的缺陷。例如，缺陷GUI 456可以用于基于静态分析工具所进行分析的结果，跟踪源代码中的缺陷（即不一致性）。如图4J所示，用户可以通过选择“Add Defect”（增加缺陷）按钮457，添加拟跟踪的缺陷。添加的缺陷458与该缺陷的有关信息以及缺陷处置信息一起显示，该有关信息包括例如缺陷号、创建日期、状态、和被分配给缺陷的用户。添加缺陷458也可以被选择用于编辑。

结论

[0134] 上文结合多个示例性实施方案对本发明进行了描述，但是相关领域的普通技术人员易于理解的是有可能采用不同于前文描述的示例性实施方案的方式来实施本发明。这在不偏离本发明范畴的

情况下是可行的。这些示例性的实施方案仅仅是示意性的，无论如何不应当看作为限制性的。本发明的范围由后附的权利要求书而不是前文的说明书限制，并且所有落入权利要求书范围之内的所有变化和等价物均被视为包含在内。

图 . 1A


用户
135
网络
130
自动化需求验证
工具
115
单元 / 集成测
试工具
125
静态分析工具
120
需求信息源
105
需求捕获工具
110
建模模块
190
100

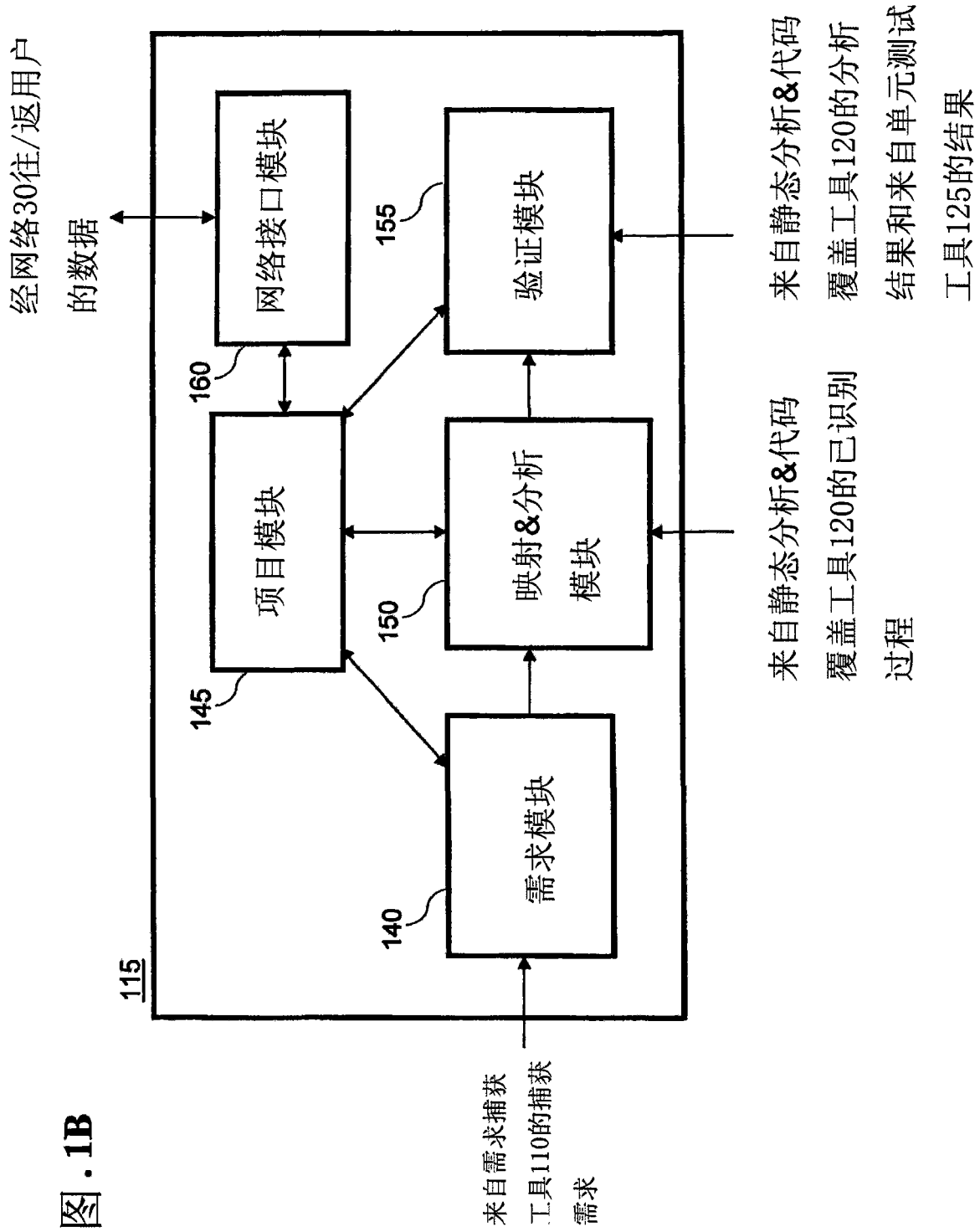

经网络30往/返用户
的数据
115
网络接口模块
160
项目模块
145
需求模块
140
映射&分析
模块
150
验证模块
155
来自需求捕获
工具110的捕获
需求
来自静态分析&代码
覆盖工具120的已识别
过程
来自静态分析&代码
覆盖工具120的分析
结果和来自单元测试
工具125的结果

图 . 1B

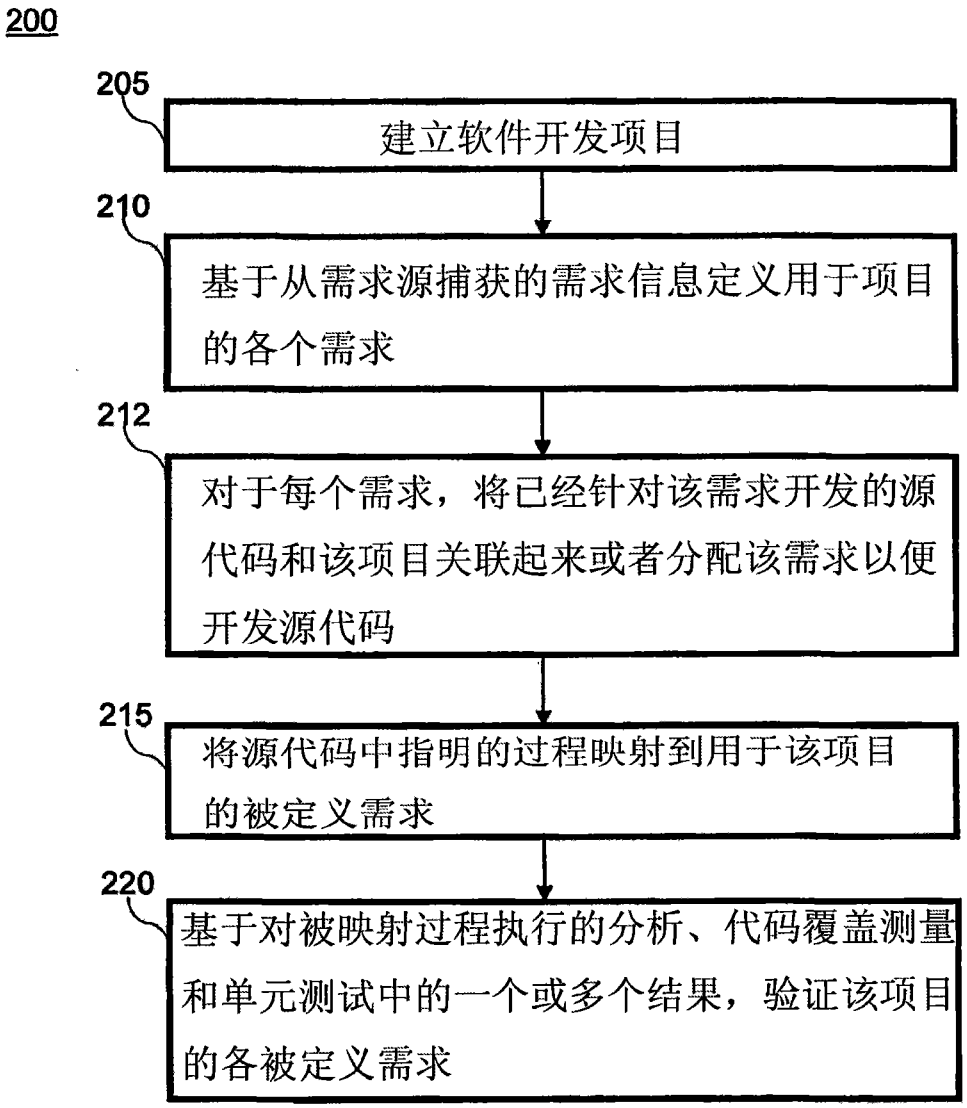

200
205
建立软件开发项目
210
基于从需求源捕获的需求信息定义用于项目的各个需求
212
对于每个需求，将已经针对该需求开发的源代码和该项目关联起来或者分配该需求以便开发源代码
215
将源代码中指明的过程映射到用于该项目的被定义需求
220
基于对被映射过程执行的分析、代码覆盖测量和单元测试中的一个或多个结果，验证该项目的各被定义需求

图 .2

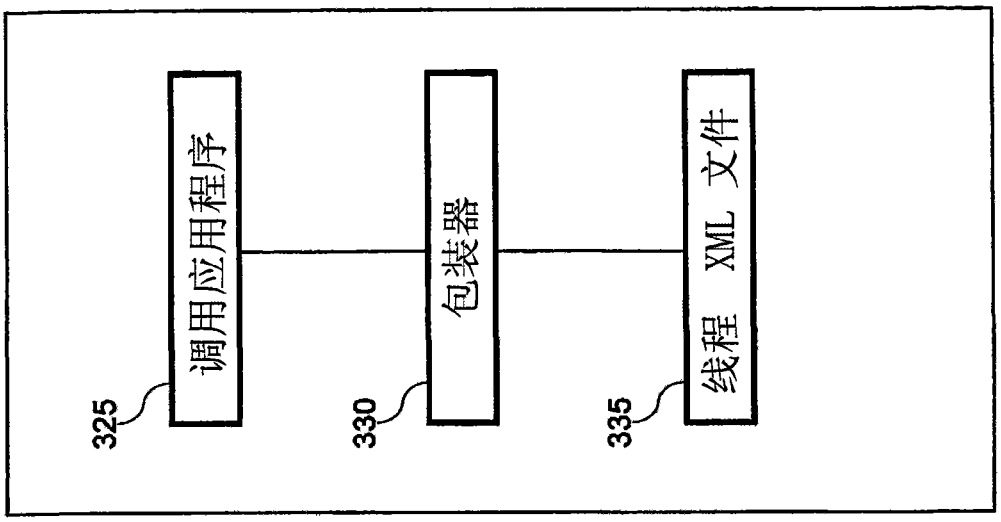

325
调用应用程序
330
包装器
335
线程 XML 文件

图 . 3B

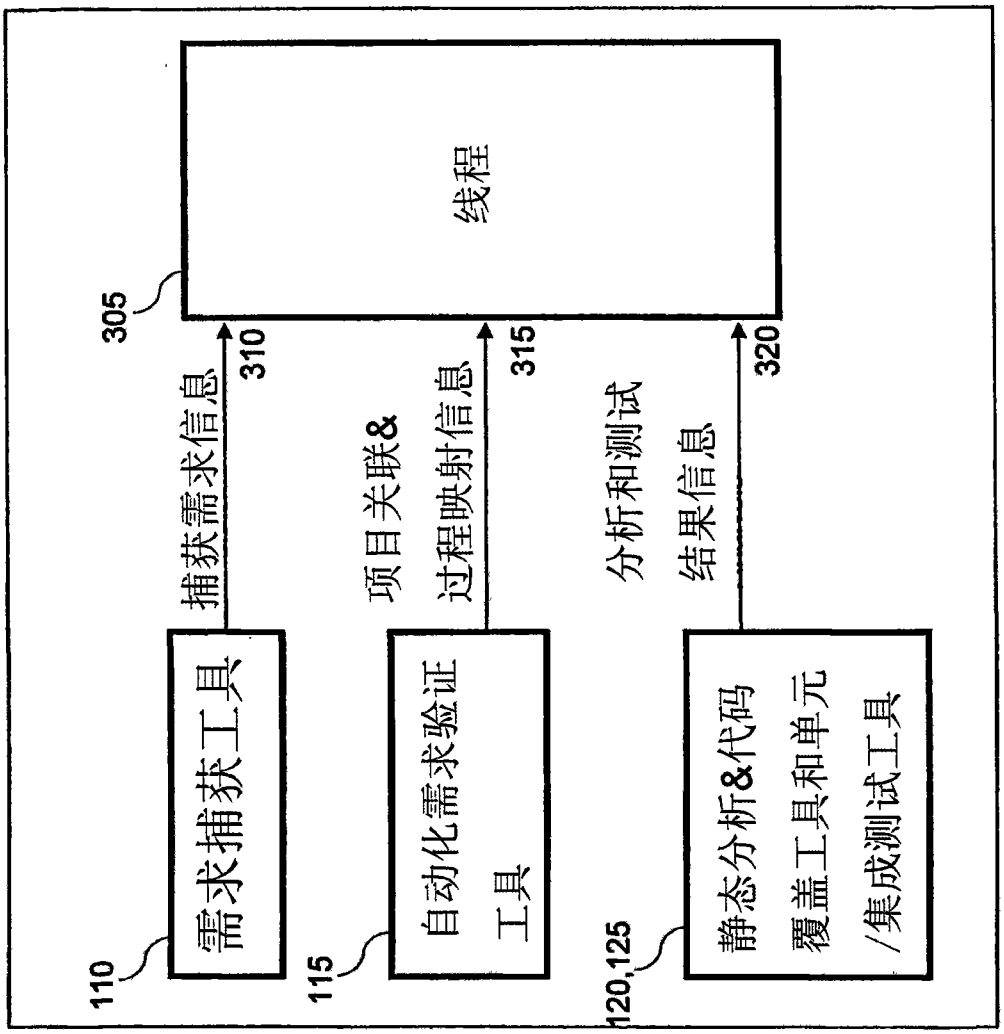

110
需求捕获工具
捕获需求信息
310
305
线程
115
自动化需求验证
工具
项目关联&
过程映射信息
315
120,125
静态分析&代码
覆盖工具和单元
/集成测试工具
分析和测试
结果信息
320

图 . 3A

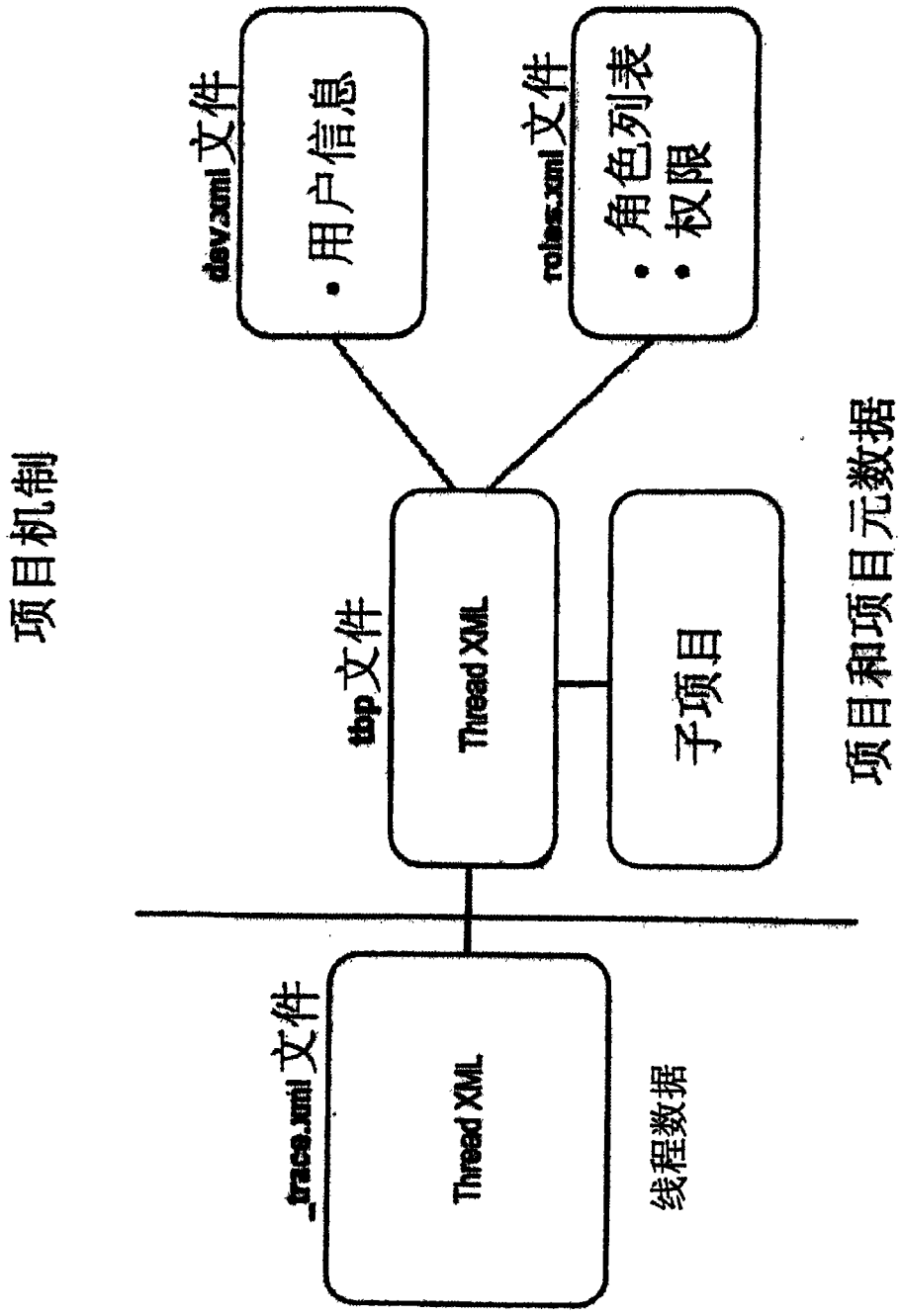

项目机制
dev.xml文件
• 用户信息
roles.xml文件
• 角色列表
• 权限
top文件
Thread XML
子项目
项目和项目元数据
_trace.xml文件
Thread XML
线程数据

图 . 3C

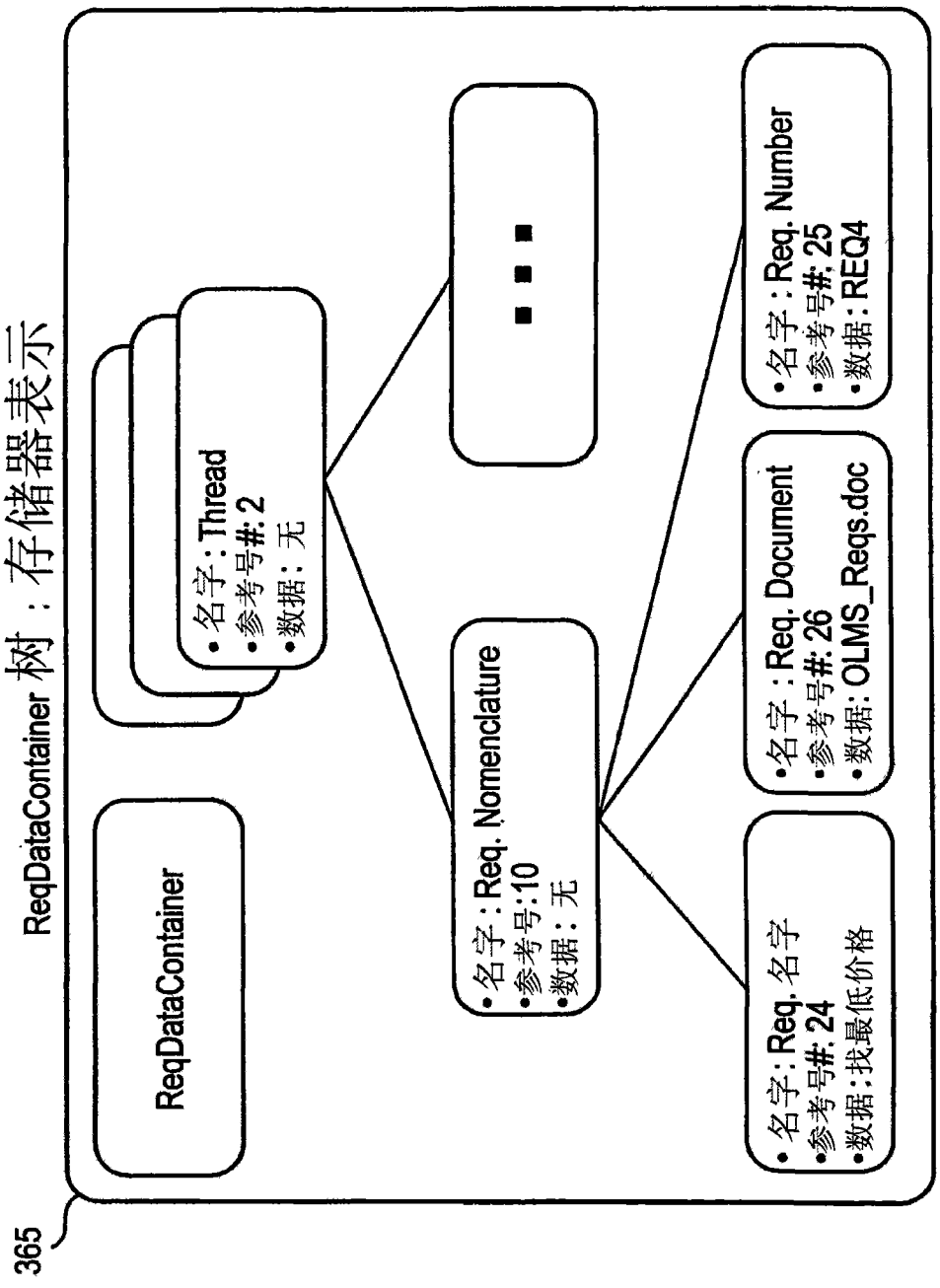

ReqDataContainer 树：存储器表示
365
ReqDataContainer
•名字：Thread
•参考号#: 2
•数据：无
•名字：Req. Nomenclature
•参考号:10
•数据：无
■ ■ ■
• 名字: Req. 名字
•参考号#: 24
•数据：找最低价格
•名字：Req. Document
•参考号#: 26
•数据: OLMS_Reqs.doc
•名字：Req. Number
•参考号#: 25
•数据：REQ4

图 . 3D

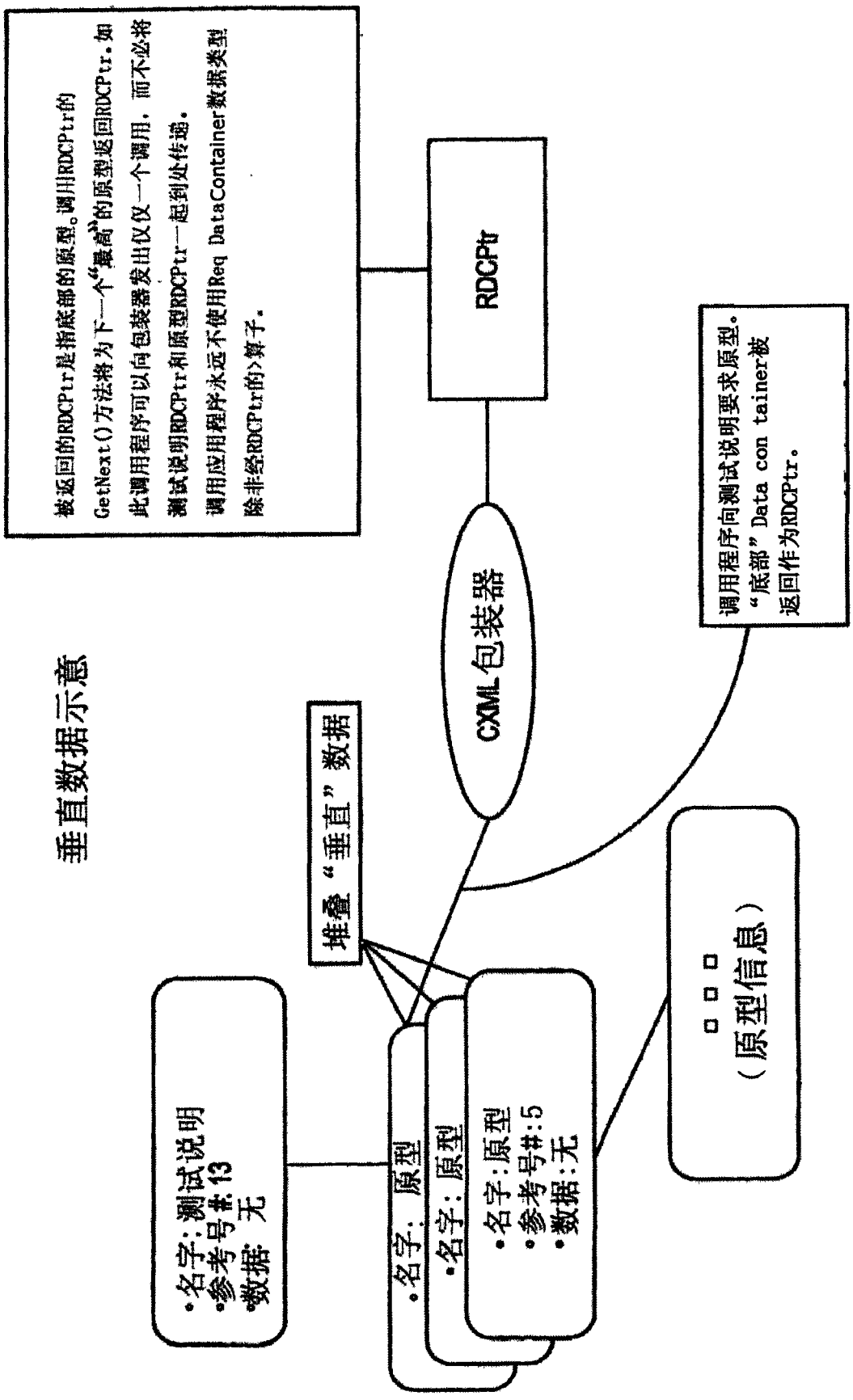

垂直数据示意
被返回的RDCPtr是指底部的原型。调用RDCPtr的GetNext()方法将为下一个"层高"的原型返回RDCPtr。如此调用程序可以向包装器发出仅仅一个调用，而不必将测试说明RDCPtr和原型RDCPtr一起到处传递。
调用应用程序永远不使用Req DataContainer数据类型除非经RDCPtr的>算子。
RDCPtr
CXML包装器
堆叠"垂直"数据
调用程序向测试说明要求原型。"底部"Data con tainer被返回作为RDCPtr。
•名字:测试说明
•参考号#:13
•数据:无
•名字:原型
•名字:原型
•名字:原型
•参考号#:5
•数据:无
□□□
(原型信息)

图. 3E

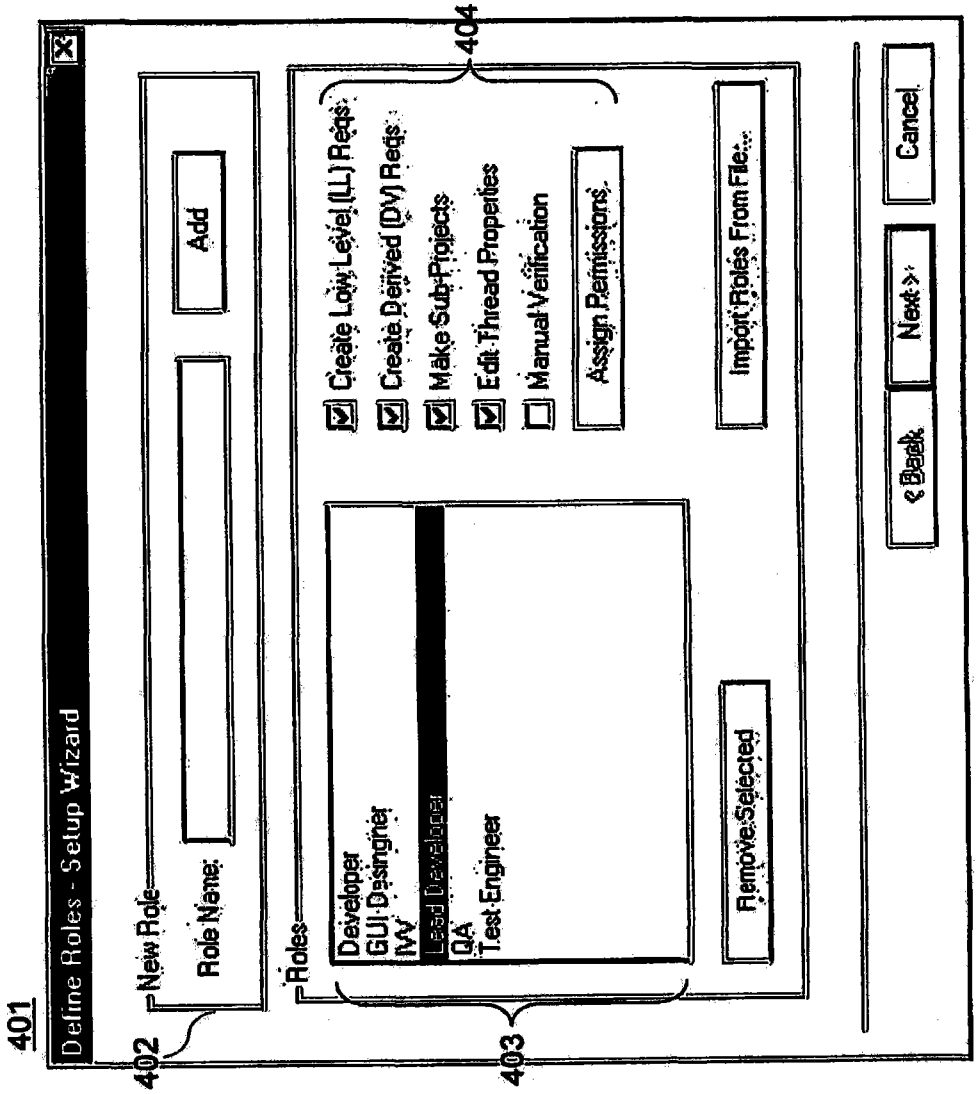

401
Define Roles - Setup Wizard
New Role
Role Name:
Add
402
Roles
Developer
GUI Designer
IVV
Lead Developer
QA
Test Engineer
403
Remove Selected
404
Create Low Level (LL) Reqs
Create Derived (DV) Reqs
Make Sub-Projects
Edit Thread Properties
Manual Verification
Assign Permissions...
Import Roles From File...
< Back
Next >
Cancel

图 · 4A

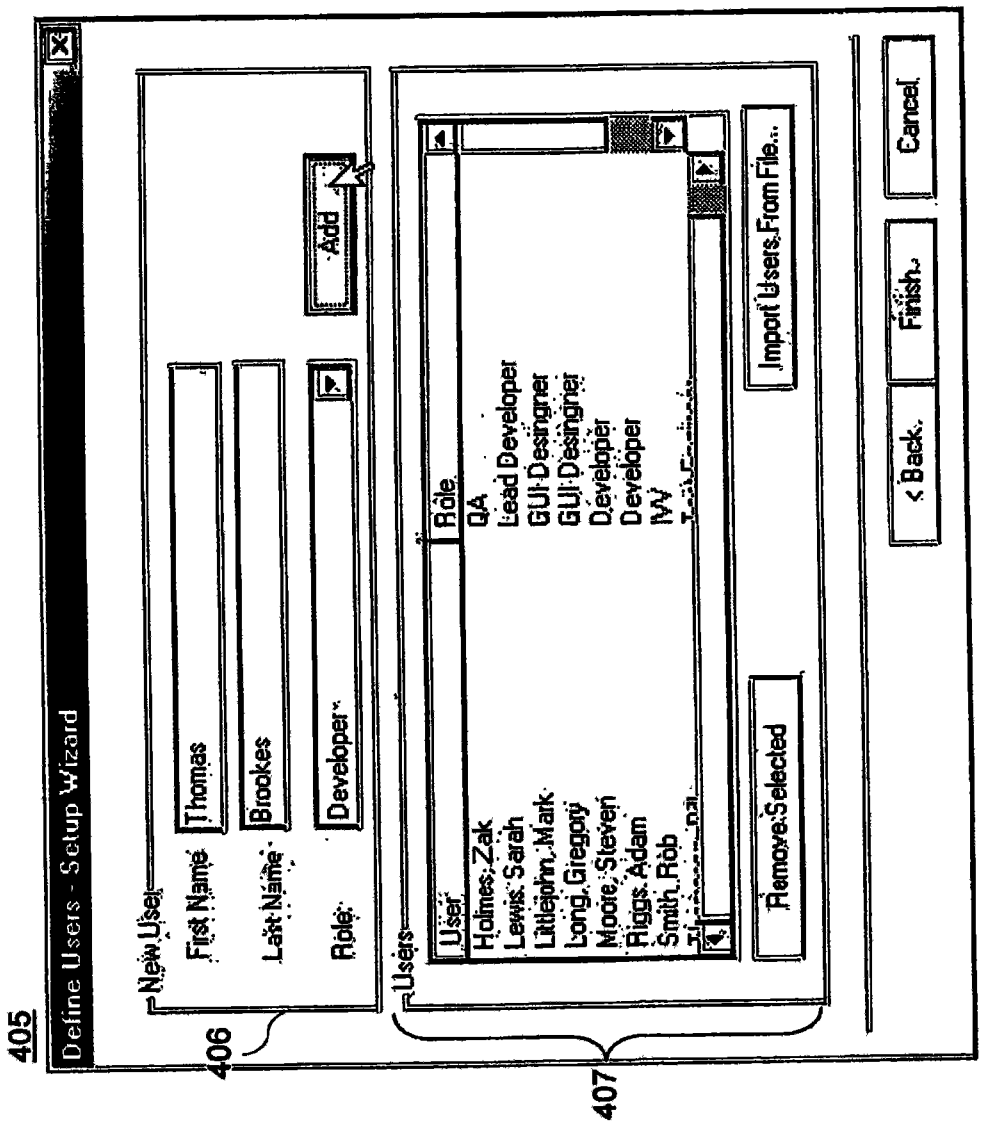

405
Define Users - Setup Wizard
New User
First Name
Thomas
Last Name
Brookes
Role
Developer
Add
406
Users
User
Role
Holmes, Zak
Lewis, Sarah
Littlejohn, Mark
Long, Gregory
Moore, Steven
Riggs, Adam
Smith, Rob
QA
Lead Developer
GUI Designer
GUI Designer
Developer
Developer
PM
Remove Selected
Import Users From File...
407
< Back
Finish
Cancel

图 . 4B

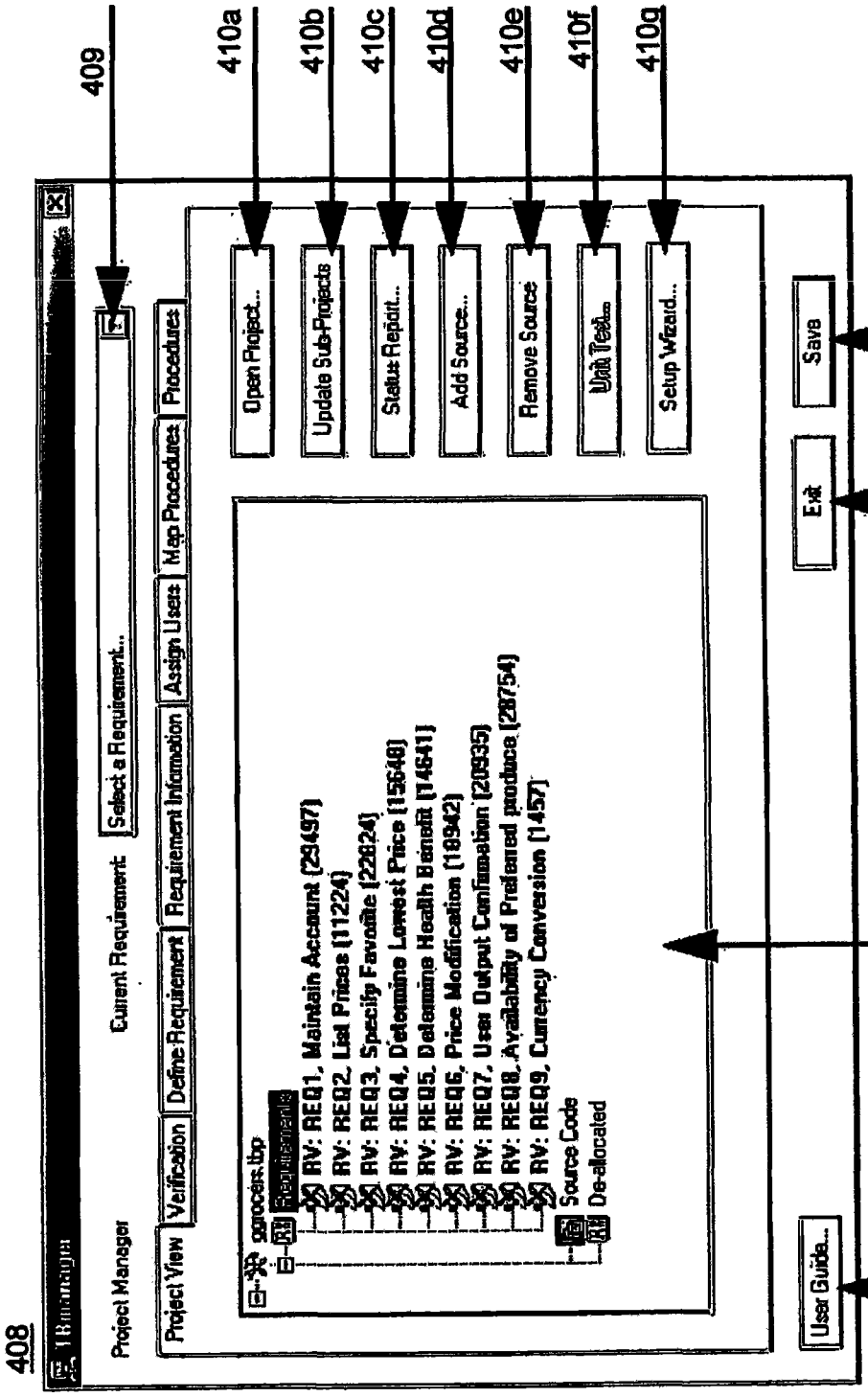

408
409
410a
410b
410c
410d
410e
410f
410g
Project Manager
Current Requirement:
Select a Requirement...
Project View
Verification
Define Requirement
Requirement Information
Assign Users
Map Procedures
Procedures
groceries.tbp
Requirements
RV: REQ1, Maintain Account [29497]
RV: REQ2, List Prices [11224]
RV: REQ3, Specify Favorite [22824]
RV: REQ4, Determine Lowest Price [15648]
RV: REQ5, Determine Health Benefit [14541]
RV: REQ6, Price Modification [18942]
RV: REQ7, User Output Confirmation [20935]
RV: REQ8, Availability of Preferred produce [28754]
RV: REQ9, Currency Conversion [1457]
Source Code
De-allocated
Open Project...
Update Sub-Projects
Status Report...
Add Source...
Remove Source
Unit Test...
Setup Wizard...
User Guide...
Exit
Save
411
412
413
414

图 . 4C

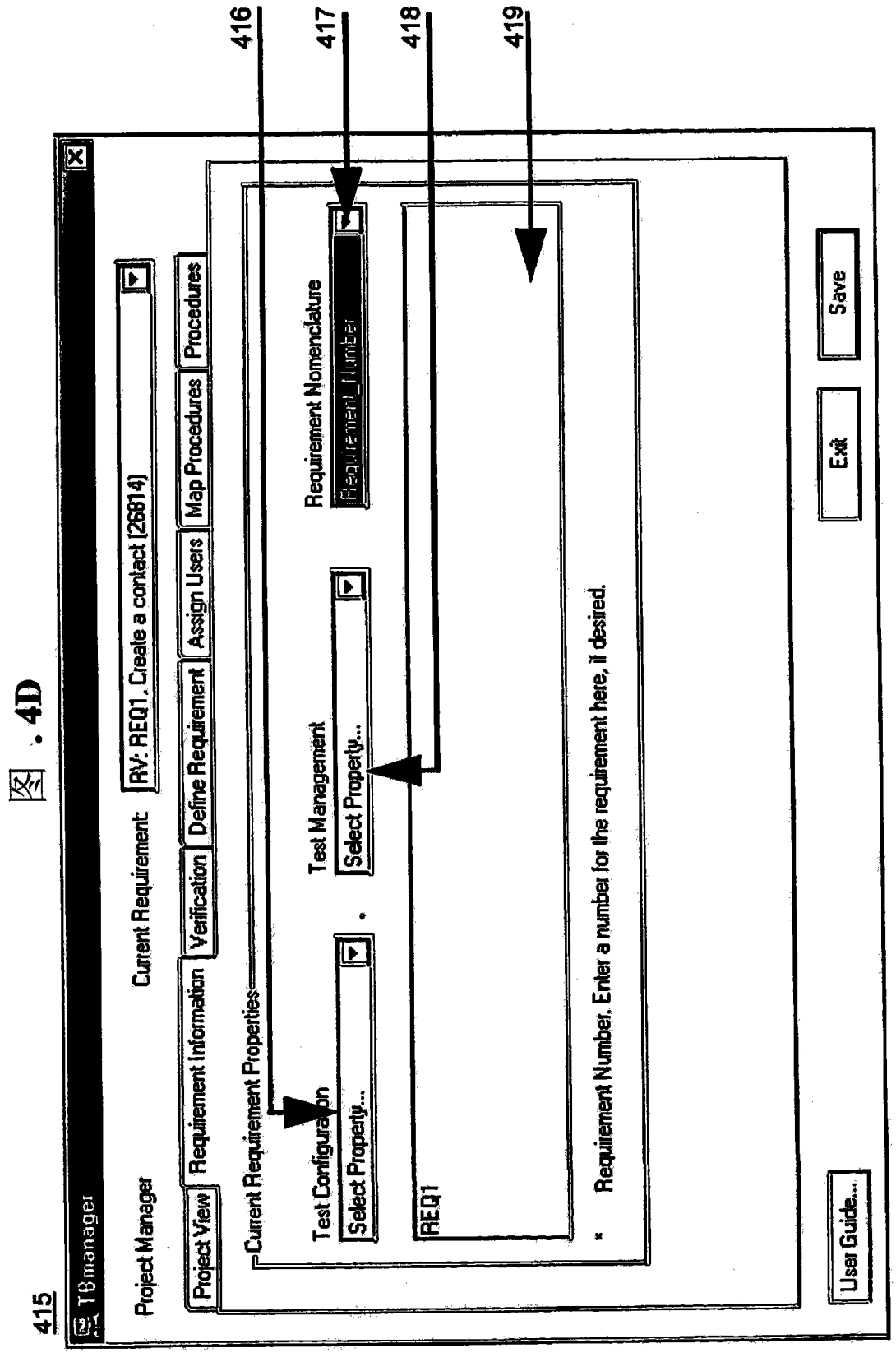

415
TBmanager
Project Manager
Current Requirement
RV: REQ1, Create a contact [26814]
Project View
Requirement Information
Verification
Define Requirement
Assign Users
Map Procedures
Procedures
Current Requirement Properties
Test Configuration
Select Property...
Test Management
Select Property...
Requirement Nomenclature
Requirement_Number
REQ1
* Requirement Number. Enter a number for the requirement here, if desired.
User Guide...
Exit
Save
416
417
418
419

图 . 4D

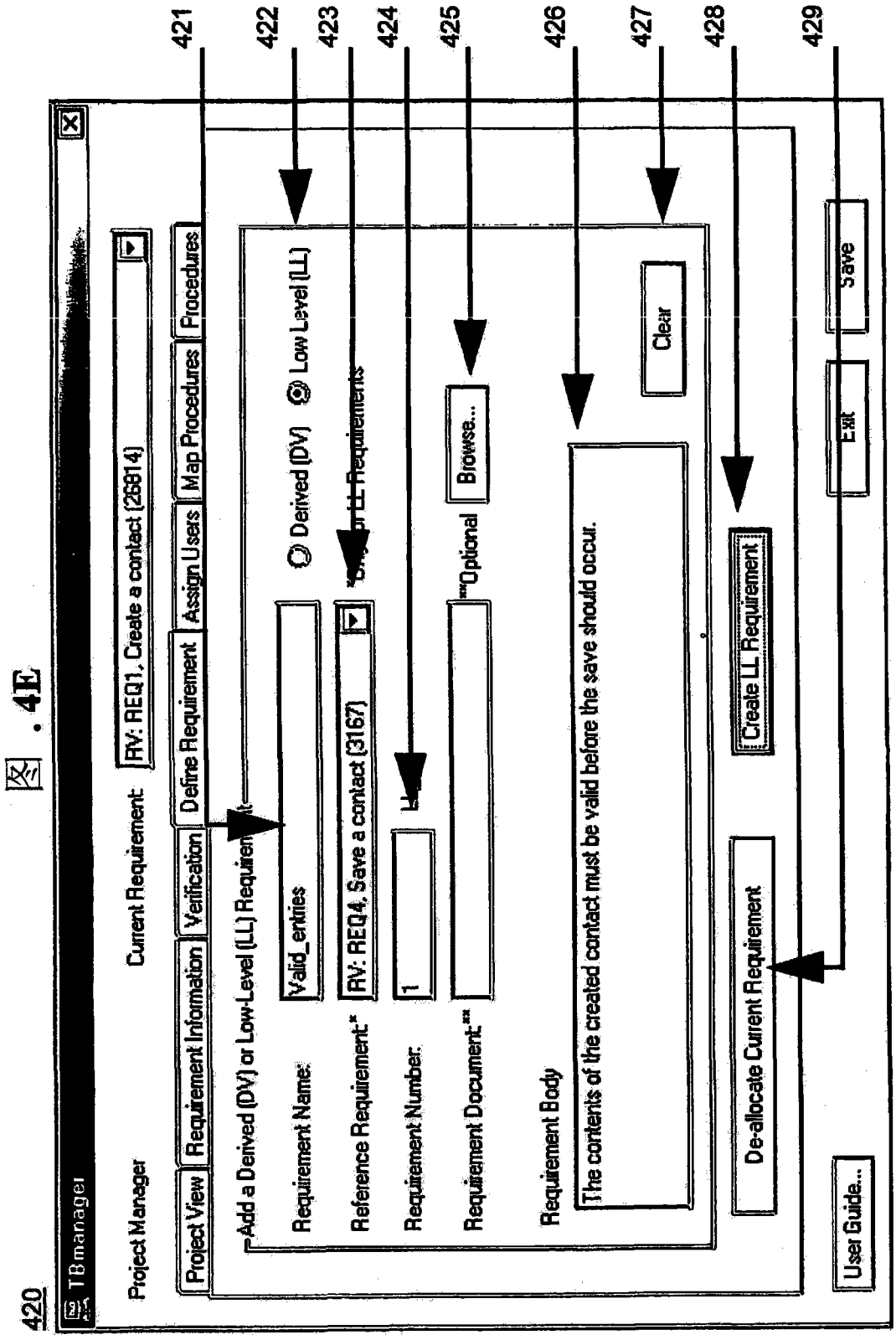

420
TBmanager
Project Manager
Current Requirement:
RV: REQ1, Create a contact (26914)
Project View
Requirement Information
Verification
Define Requirement
Assign Users
Map Procedures
Procedures
421
422
423
424
425
426
427
428
429
Add a Derived (DV) or Low-Level (LL) Requirement
Requirement Name:
Valid_entries
Derived (DV)
Low Level (LL)
Reference Requirement:*
RV: REQ4, Save a contact (3167)
Requirement Number:
1
Requirement Document:""
""Optional
Browse...
Requirement Body
The contents of the created contact must be valid before the save should occur.
Clear
De-allocate Current Requirement
Create LL Requirement
Exit
Save
User Guide...

图 .4E

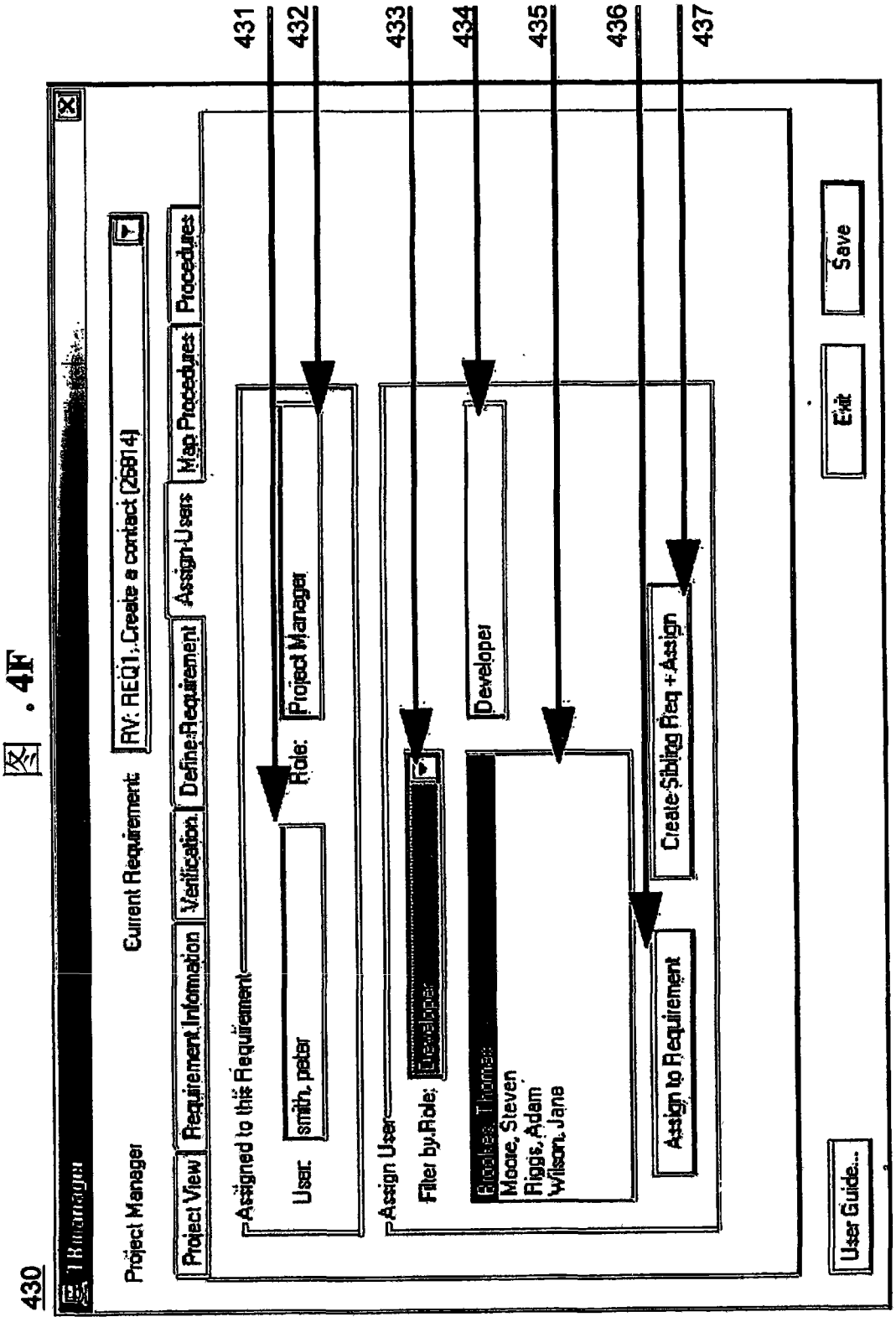

430
Project Manager
Current Requirement:
RV: REQ1, Create a contact (26814)
Project View
Requirement Information
Verification
Define Requirement
Assign Users
Map Procedures
Procedures
Assigned to this Requirement
User:
smith, peter
Role:
Project Manager
Assign User
Filter by Role:
Developer
Bradley, Thomas
Moore, Steven
Riggs, Adam
Wilson, Jane
Developer
Assign to Requirement
Create Sibling Req + Assign
Exit
Save
User Guide...
431
432
433
434
435
436
437

图 .4F

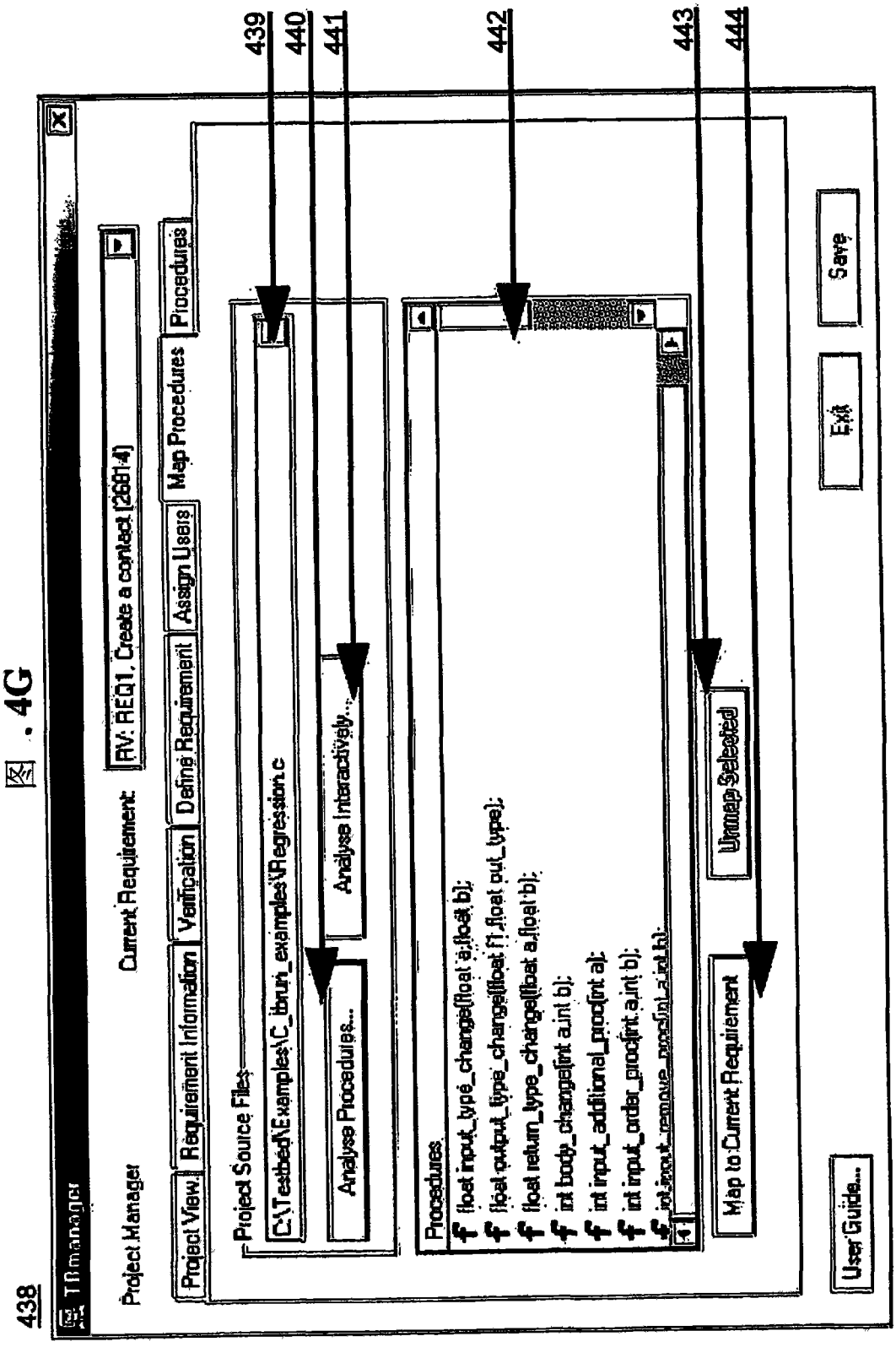

438
TBmanager
Project Manager
Current Requirement:
RV: REQ1, Create a contact (2681-4)
Project View
Requirement Information
Verification
Define Requirement
Assign Users
Map Procedures
Procedures
Project Source Files:
C:\Testbed\Examples\C_ibrun_examples\Regression.c
Analyse Procedures...
Analyse Interactively...
Procedures:
float input_type_change(float a,float b);
float output_type_change(float f1,float out_type);
float return_type_change(float a,float b);
int body_change(int a,int b);
int input_additional_proc(int a);
int input_order_proc(int a,int b);
int input_remove_proc(int a,int b);
Map to Current Requirement
Unmap Selected
User Guide...
Exit
Save
439
440
441
442
443
444

图 4G

图 . 4H

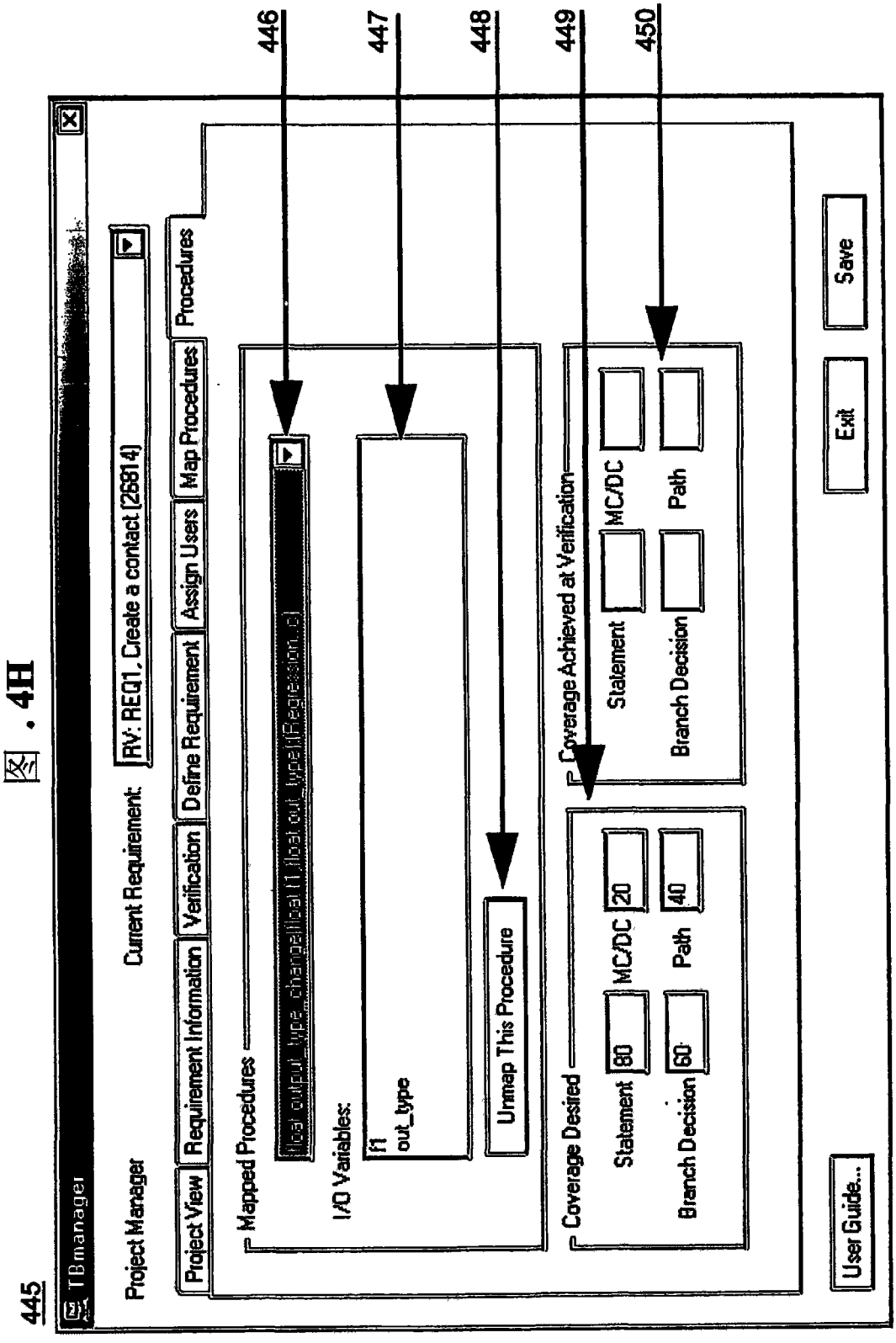
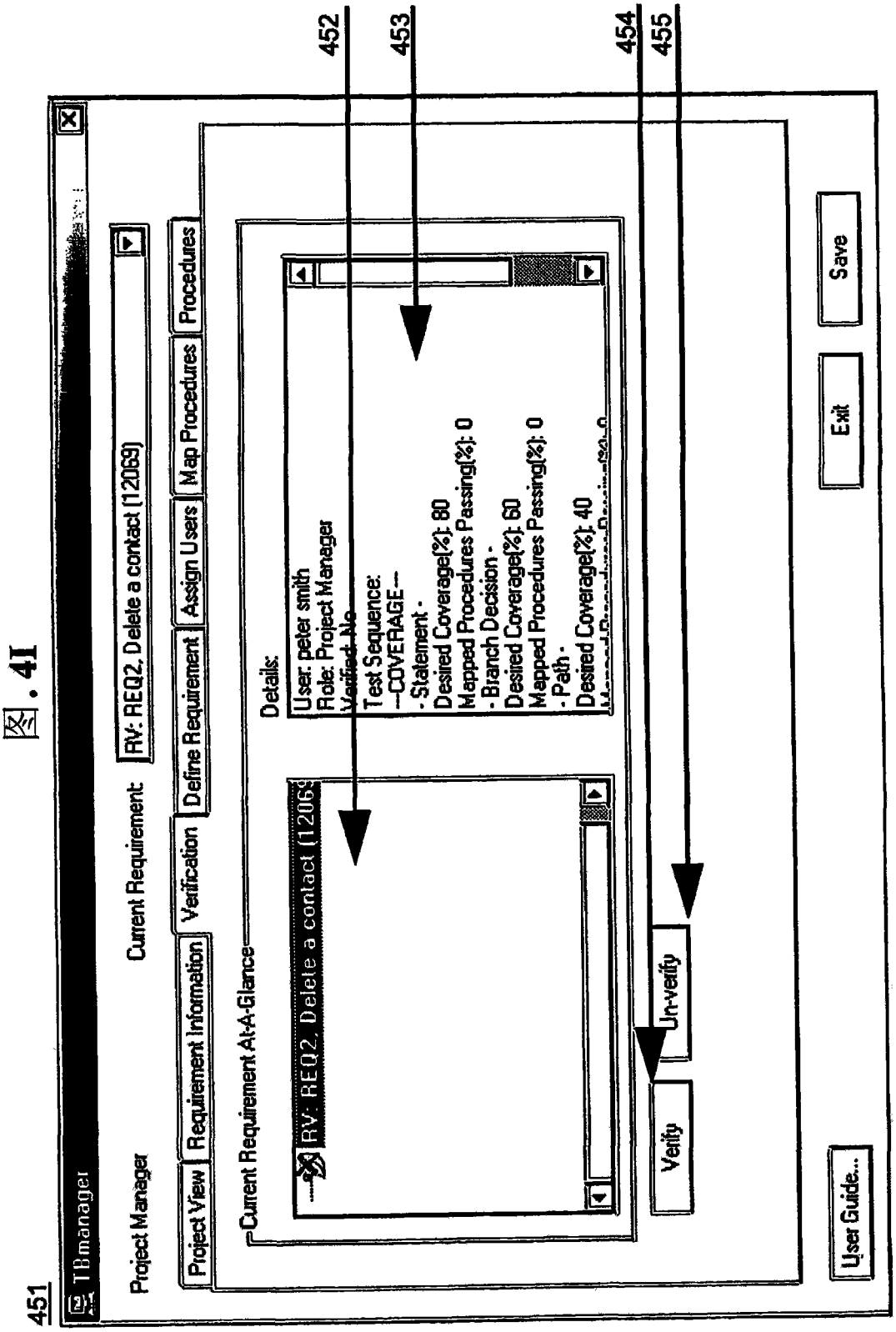

445
TBmanager
Project Manager
Current Requirement:
RV: REQ1, Create a contact (26814)
Project View
Requirement Information
Verification
Define Requirement
Assign Users
Map Procedures
Procedures
446
447
448
449
450
Mapped Procedures
I/O Variables:
f1
out_type
Unmap This Procedure
Coverage Desired
Statement
80
MC/DC
20
Branch Decision
60
Path
40
Coverage Achieved at Verification
Statement
MC/DC
Branch Decision
Path
Exit
Save
User Guide...

图.4I


451
TBmanager
Project Manager
Current Requirement
RV: REQ2, Delete a contact (12069)
Project View
Requirement Information
Verification
Define Requirement
Assign Users
Map Procedures
Procedures
Current Requirement At-A-Glance
RV: REQ2, Delete a contact (12069)
Details:
User: peter smith
Role: Project Manager
Test Sequence:
--COVERAGE--
- Statement -
Desired Coverage(%): 80
Mapped Procedures Passing(%): 0
- Branch Decision -
Desired Coverage(%): 60
Mapped Procedures Passing(%): 0
- Path -
Desired Coverage(%): 40
452
453
454
455
Verify
Un-verify
Exit
Save
User Guide...

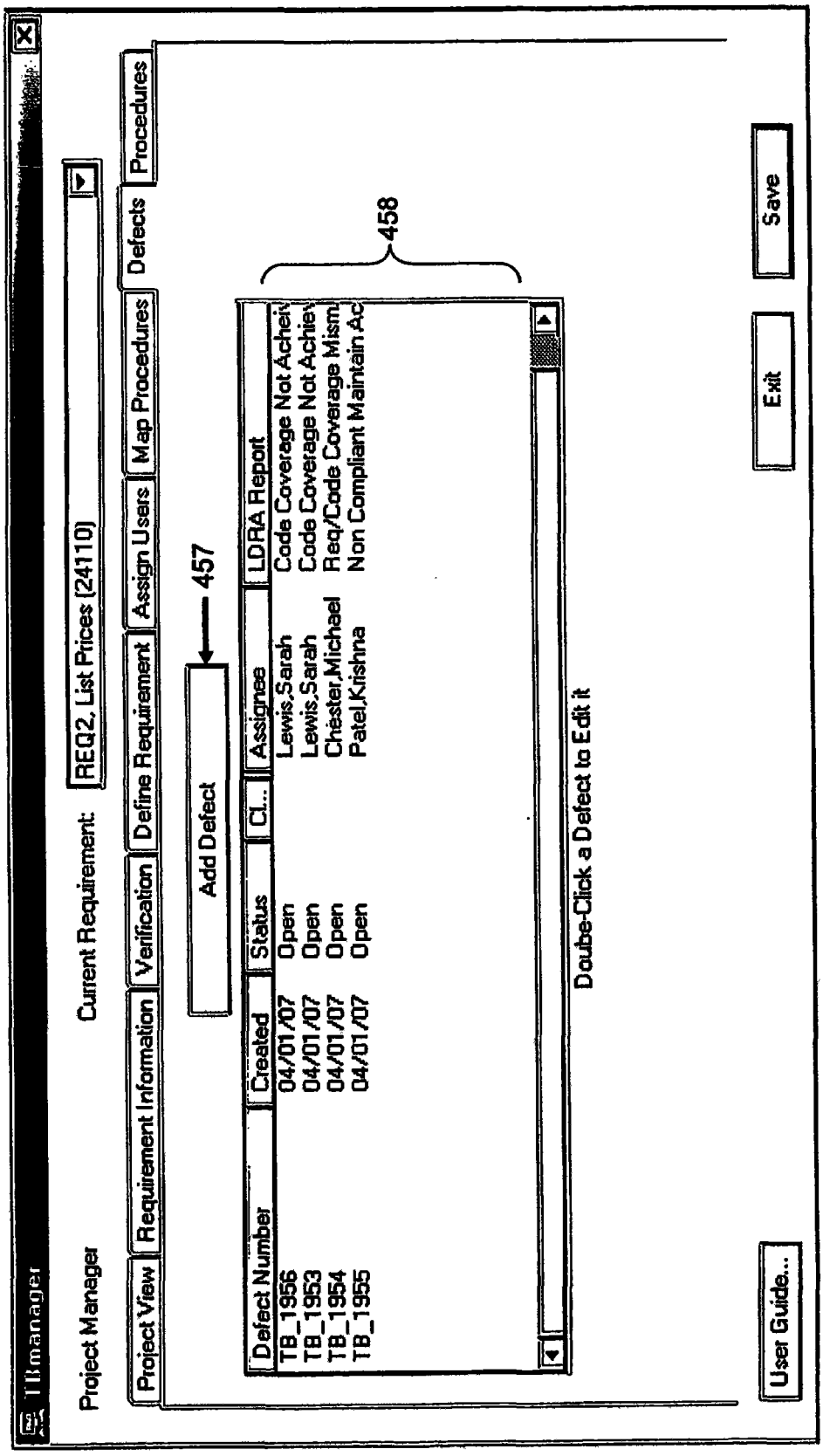

456
TBmanager
Project Manager
Current Requirement:
REQ2, List Prices (24110)
Project View
Requirement Information
Verification
Define Requirement
Assign Users
Map Procedures
Defects
Procedures
Add Defect
457
458
Defect Number
Created
Status
Cl...
Assignee
LDRA Report
TB_1956
04/01/07
Open
Lewis,Sarah
Code Coverage Not Achie
TB_1953
04/01/07
Open
Lewis,Sarah
Code Coverage Not Achie
TB_1954
04/01/07
Open
Chester,Michael
Req/Code Coverage Mism
TB_1955
04/01/07
Open
Patel,Krishna
Non Compliant Maintain Ac
Doube-Click a Defect to Edit it
User Guide...
Exit
Save

图 . 4J

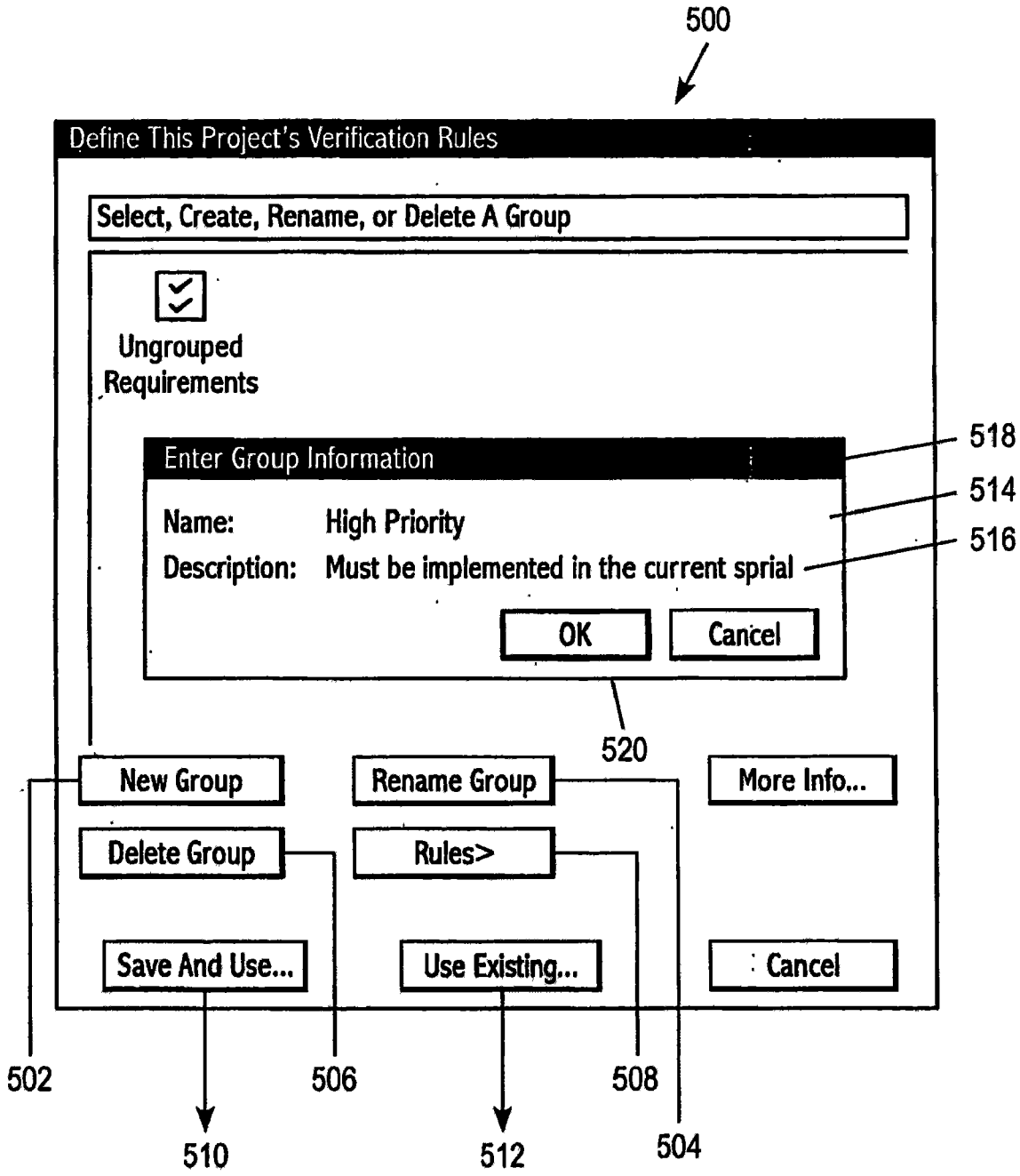

500
Define This Project's Verification Rules
Select, Create, Rename, or Delete A Group
Ungrouped Requirements
Enter Group Information
Name: High Priority
Description: Must be implemented in the current sprial
OK
Cancel
518
514
516
520
New Group
Rename Group
More Info...
Delete Group
Rules>
Save And Use...
Use Existing...
Cancel
502
506
508
510
512
504

图 .5

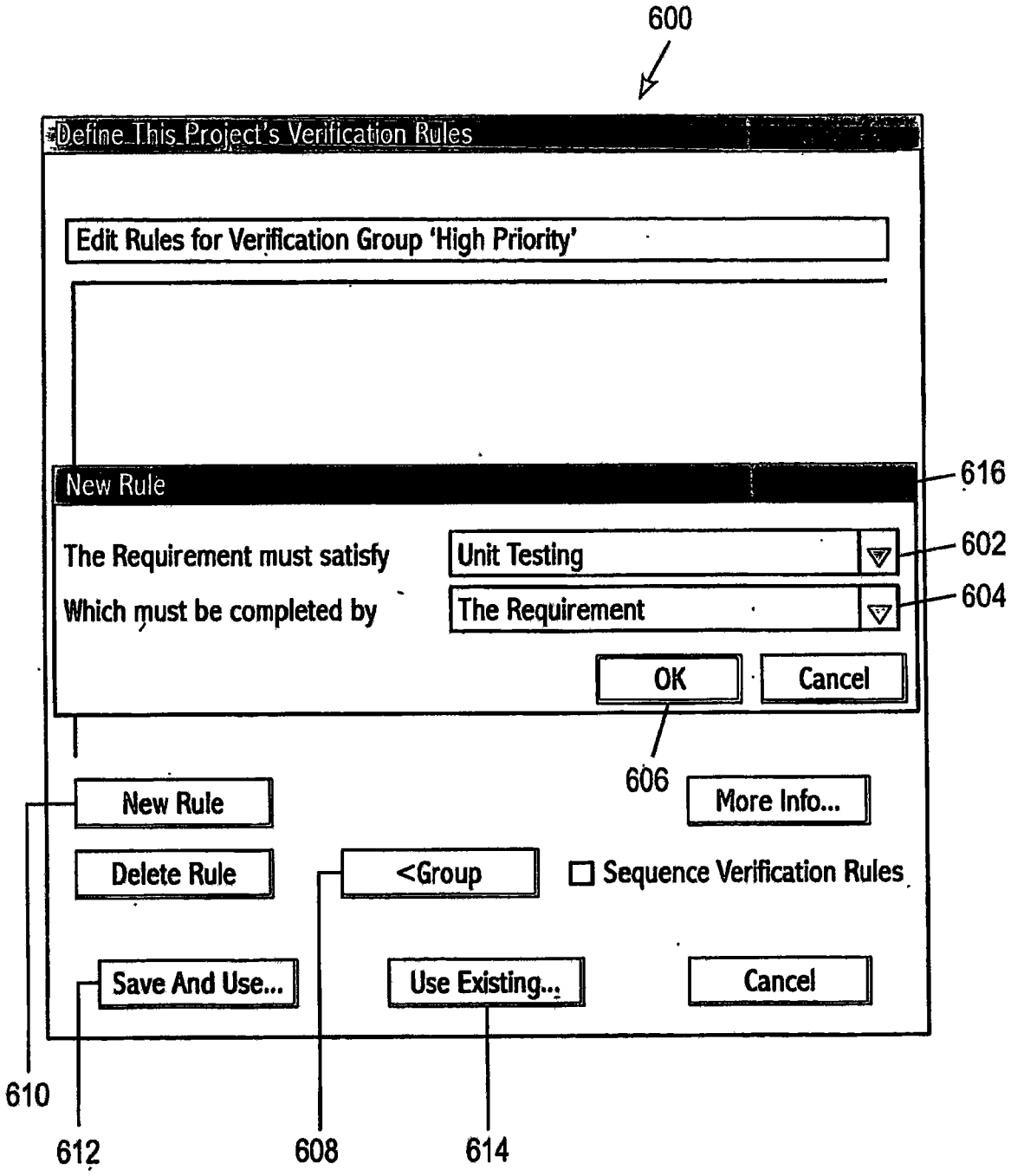

600
Define This Project's Verification Rules
Edit Rules for Verification Group 'High Priority'
New Rule
616
The Requirement must satisfy
Unit Testing
602
Which must be completed by
The Requirement
604
OK
Cancel
606
New Rule
More Info...
Delete Rule
<Group
Sequence Verification Rules
Save And Use...
Use Existing...
Cancel
610
612
608
614

图 . 6

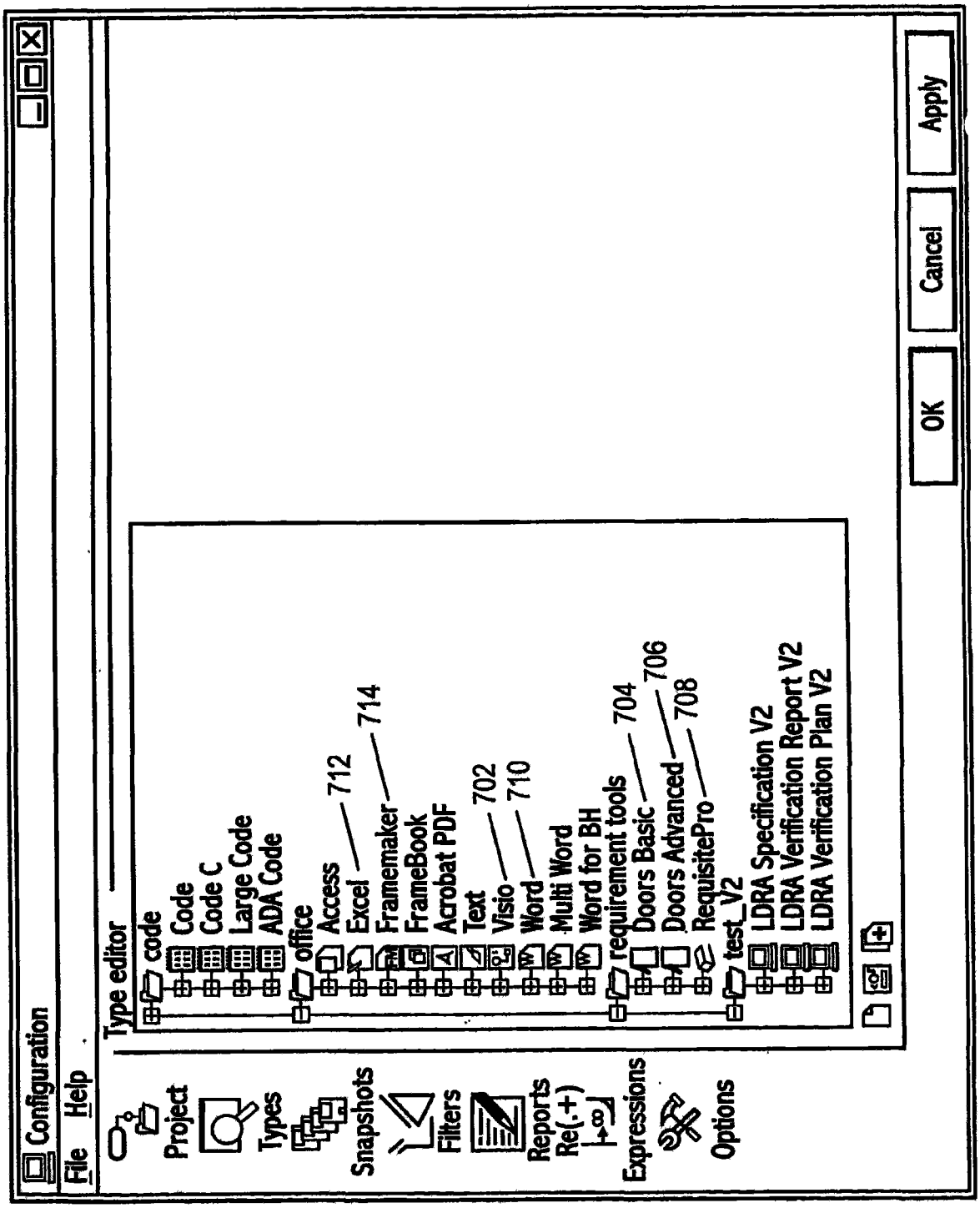

Configuration
File Help
Project
Types
Snapshots
Filters
Reports
Re(.+)
Expressions
Options
Type editor
code
Code
Code C
Large Code
ADA Code
office
Access
Excel
Framemaker
FrameBook
Acrobat PDF
Text
Visio
Word
Multi Word
Word for BH
requirement tools
Doors Basic
Doors Advanced
RequisitePro
test_V2
LDRA Specification V2
LDRA Verification Report V2
LDRA Verification Plan V2
712
714
702
710
704
706
708
OK
Cancel
Apply

图 . 7

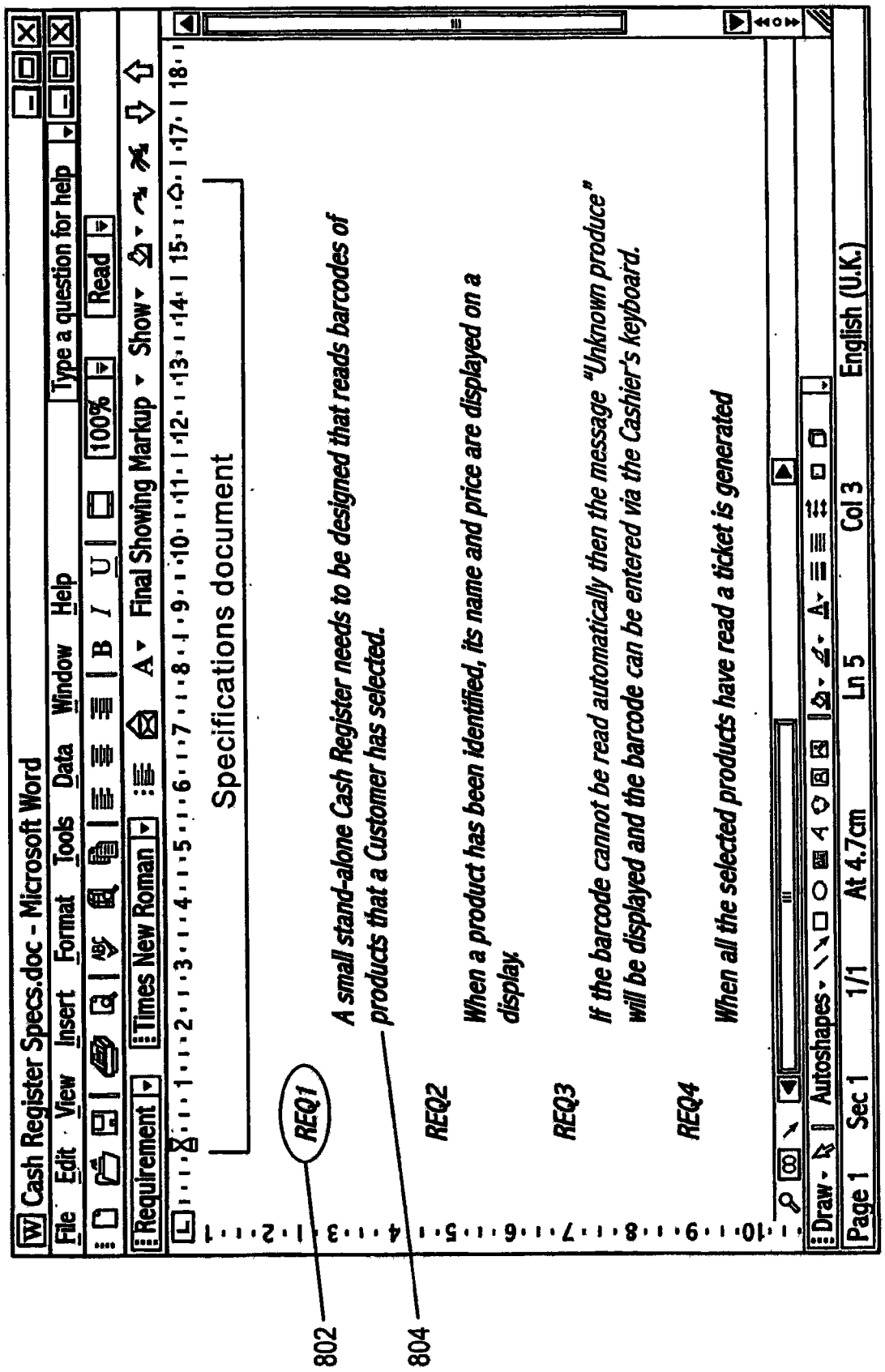

Cash Register Specs.doc - Microsoft Word
File Edit View Insert Format Tools Data Window Help
Type a question for help
Requirement
Times New Roman
Final Showing Markup
Show
Read
Specifications document
REQ1
A small stand-alone Cash Register needs to be designed that reads barcodes of products that a Customer has selected.
REQ2
When a product has been identified, its name and price are displayed on a display
REQ3
If the barcode cannot be read automatically then the message "Unknown produce" will be displayed and the barcode can be entered via the Cashier's keyboard.
REQ4
When all the selected products have read a ticket is generated
Draw
Autoshapes
Page 1 Sec 1 1/1 At 4.7cm Ln 5 Col 3 English (U.K.)
802
804

图. 8

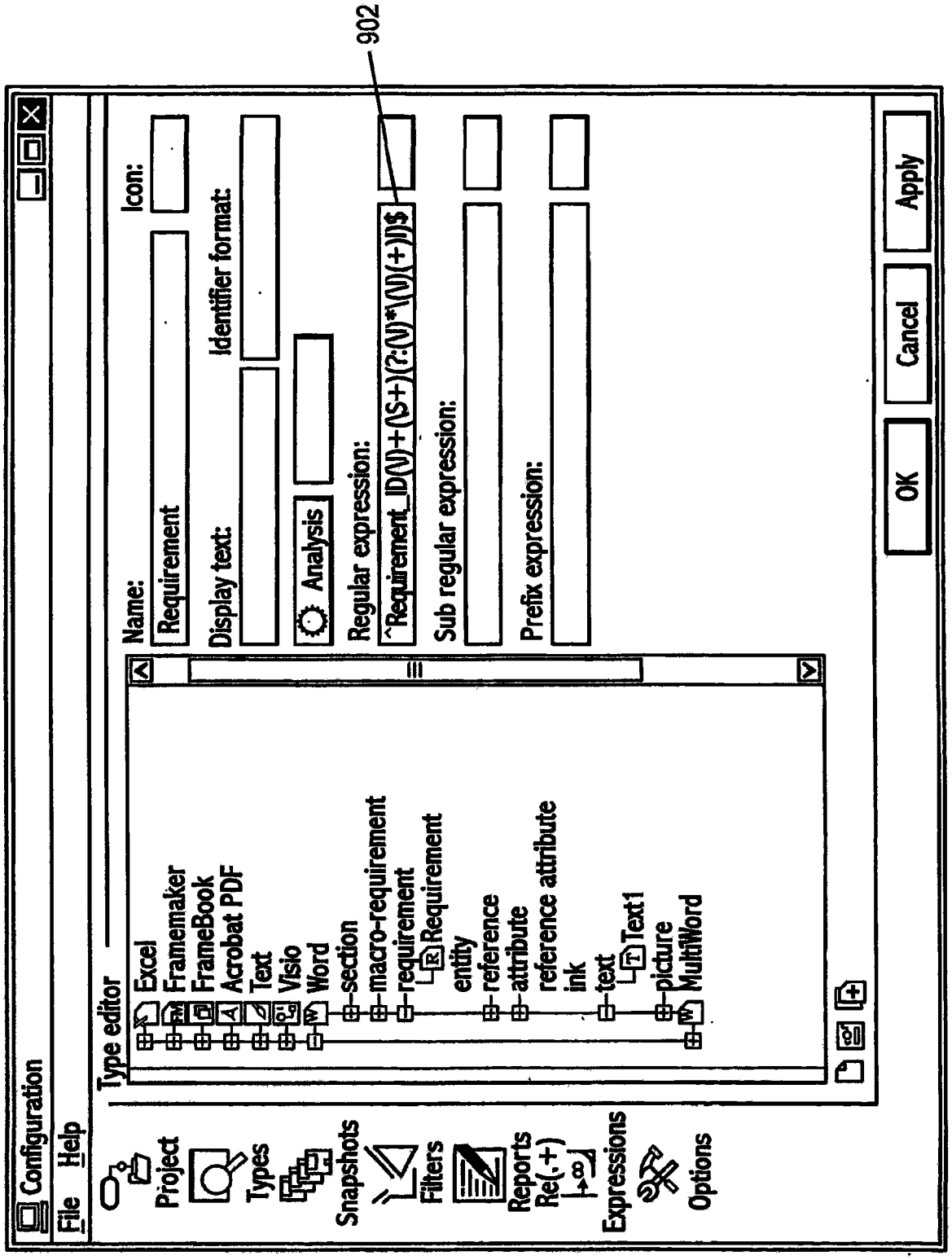

902
Configuration
File Help
Project
Types
Snapshots
Filters
Reports
Re(.+)
Expressions
Options
Type editor
Excel
Framemaker
FrameBook
Acrobat PDF
Text
Visio
Word
section
macro-requirement
requirement
Requirement
entity
reference
attribute
reference attribute
ink
text
Text1
picture
MultiWord
Name:
Requirement
Icon:
Display text:
Identifier format:
Analysis
Regular expression:
^Requirement_ID(\V)+(\S+)(?:(\V)*\(\V)(+))\$
Sub regular expression:
Prefix expression:
OK
Cancel
Apply

图 9

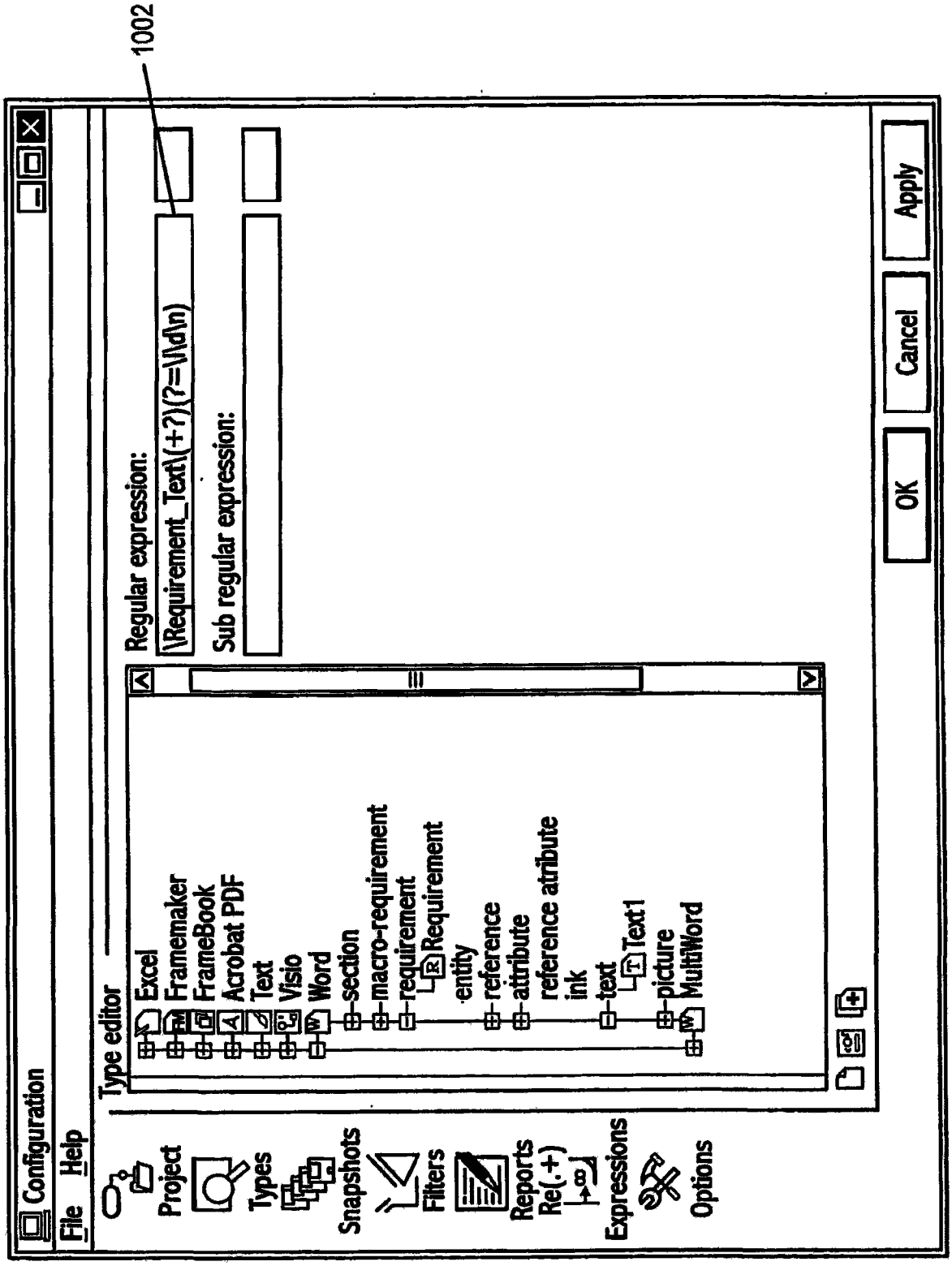

1002
Configuration
File Help
Project
Types
Snapshots
Filters
Reports
Re(.+)
Expressions
Options
Type editor
Excel
Framemaker
FrameBook
Acrobat PDF
Text
Visio
Word
section
macro-requirement
requirement
Requirement
entity
reference
attribute
reference atribute
ink
text
Text1
picture
MultiWord
Regular expression:
\Requirement_Text\(+?)(?=\\d\n)
Sub regular expression:
OK
Cancel
Apply

图. 10

图 . 11

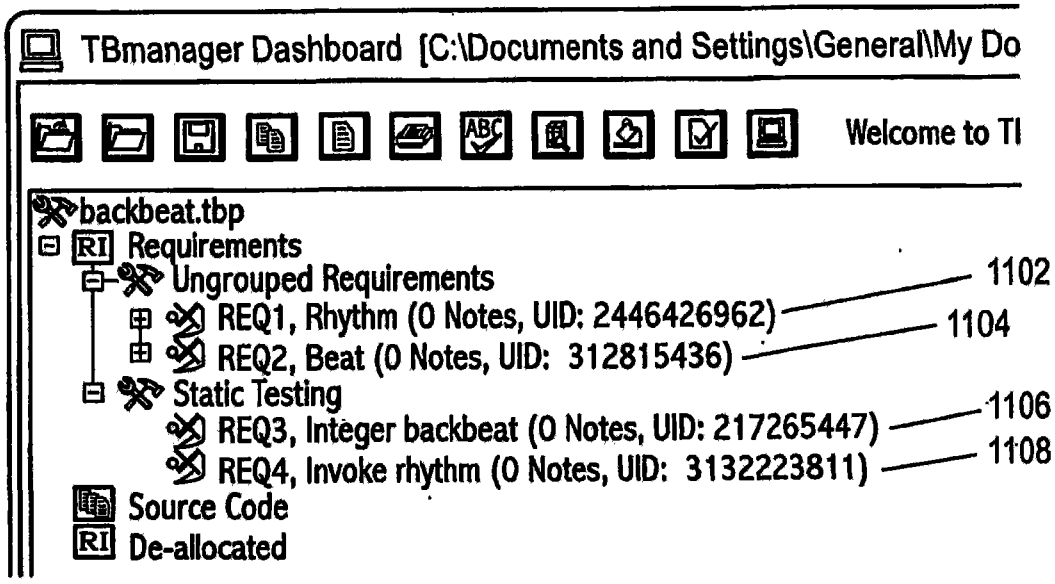

TBmanager Dashboard [C:\Documents and Settings\General\My Do
Welcome to TI
backbeat.tbp
RI Requirements
Ungrouped Requirements
REQ1, Rhythm (0 Notes, UID: 2446426962)
REQ2, Beat (0 Notes, UID: 312815436)
Static Testing
REQ3, Integer backbeat (0 Notes, UID: 217265447)
REQ4, Invoke rhythm (0 Notes, UID: 3132223811)
Source Code
RI De-allocated
1102
1104
1106
1108

图. 12

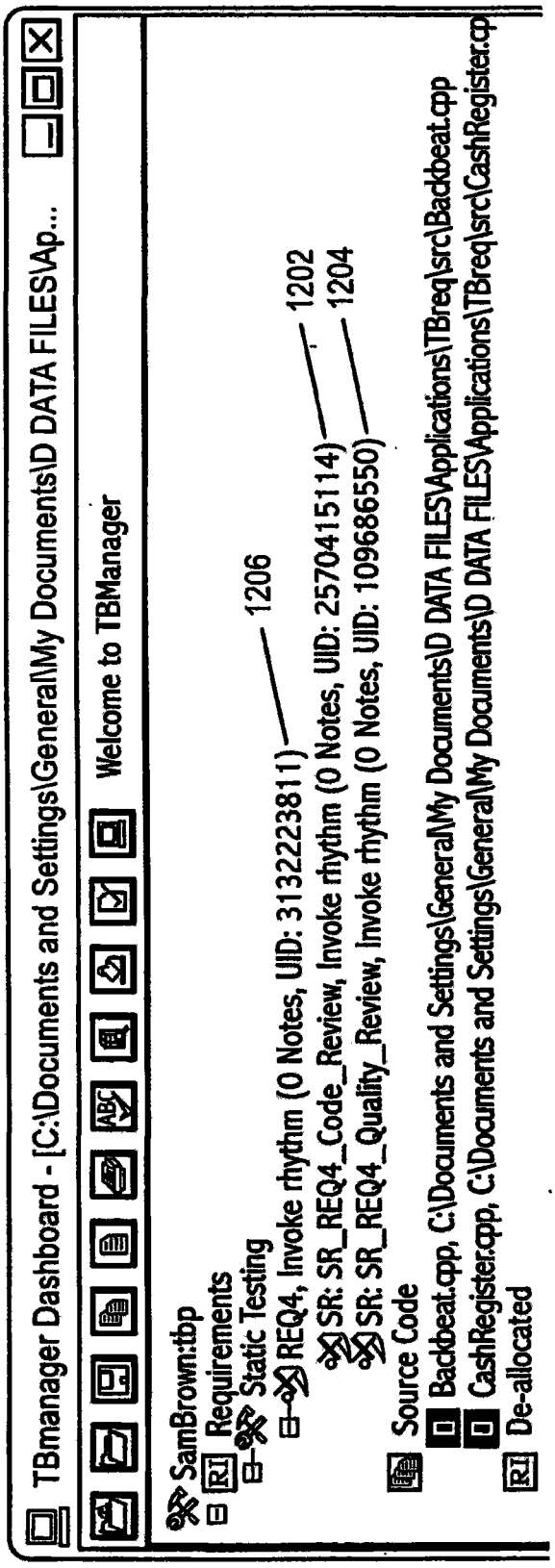
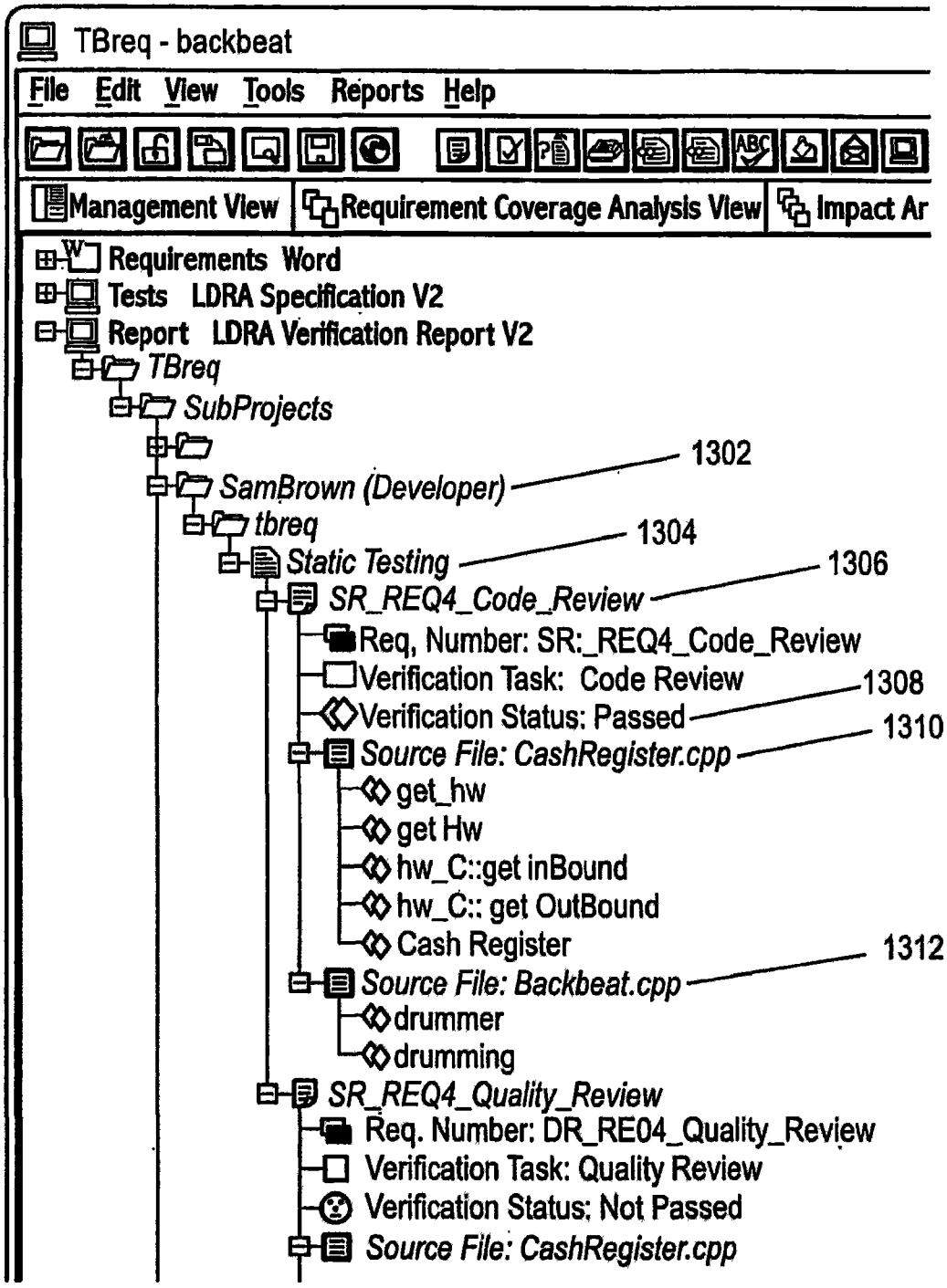

TBmanager Dashboard - [C:\Documents and Settings\General\My Documents\D DATA FILES\Ap...
Welcome to TBManager
SamBrown:tbp
Requirements
Static Testing
REQ4, Invoke rhythm (0 Notes, UID: 3132223811)
SR: SR_REQ4_Code_Review, Invoke rhythm (0 Notes, UID: 2570415114)
SR: SR_REQ4_Quality_Review, Invoke rhythm (0 Notes, UID: 109686550)
Source Code
Backbeat.cpp, C:\Documents and Settings\General\My Documents\D DATA FILES\Applications\TBreq\src\Backbeat.cpp
CashRegister.cpp, C:\Documents and Settings\General\My Documents\D DATA FILES\Applications\TBreq\src\CashRegister.cp
De-allocated
1206
1202
1204

图 .13


TBreq - backbeat
File Edit View Tools Reports Help
Management View
Requirement Coverage Analysis View
Impact Ar
Requirements Word
Tests LDRA Specification V2
Report LDRA Verification Report V2
TBreq
SubProjects
SamBrown (Developer)
tbreq
Static Testing
SR_REQ4_Code_Review
Req, Number: SR:_REQ4_Code_Review
Verification Task: Code Review
Verification Status: Passed
Source File: CashRegister.cpp
get_hw
get Hw
hw_C::get inBound
hw_C:: get OutBound
Cash Register
Source File: Backbeat.cpp
drummer
drumming
SR_REQ4_Quality_Review
Req. Number: DR_RE04_Quality_Review
Verification Task: Quality Review
Verification Status: Not Passed
Source File: CashRegister.cpp
1302
1304
1306
1308
1310
1312