中国科学: 信息科学 2014 年 第 44 卷 第 1 期: 1–19
www.scichina.com info.scichina.com
《中国科学》杂志社
SCIENCE CHINA PRESS
软件维护与演化技术专刊 · 论文

基于网络的可信软件大规模协同开发与演化

王怀民[①*], 尹刚[①], 谢冰[②], 刘旭东[③], 魏峻[④], 刘江宁[⑤]

① 国防科学技术大学计算机学院, 长沙 410073
② 北京大学信息科学技术学院软件研究所, 北京 100871
③ 北京航空航天大学计算机学院, 北京 100191
④ 中国科学院软件研究所, 北京 100080
⑤ 山东中创软件商用中间件股份有限公司, 济南 250014
* 通信作者. E-mail: whm_w@163.com

收稿日期: 2013–08–13; 接受日期: 2013–12–27
国家高技术研究发展计划 (批准号: 2007AA010301, 2012AA011201) 资助项目

摘要 随着网络时代的来临, 软件的开发模式、运行环境和提供方式发生了巨大变化. 互联网作为一种开放的协同工作环境, 其中孕育的大规模协同创作机理对软件开发和应用产生了深刻影响. 如何将其与工业化可信软件生产相结合以提高软件生产效率和质量, 是网络时代软件技术面临的新课题. 本文提出汇聚群体智慧的可信软件开发新方法 —— 群体化方法, 该方法的核心是 "群体协同、资源分享、运行监控、可信分析", 支持创新软件作品向可信软件产品转化, 支持软件的可信演化. 提出一种基于证据的可信软件概念模型, 将软件在开发阶段、分享阶段和应用阶段的基础数据作为软件的可信证据, 并据此建立了软件演化过程模型; 提出一种支持可信软件协同开发与演化的服务模型, 支持软件创作与生产深度协同的可信软件社会化生产、开放有序的可信软件资源分享, 以及基于海量数据分析的软件可信评估. 最后以 "十一五" 国家高技术研究发展计划重点项目 "高可信软件生产工具及集成环境" 为背景, 阐述了面向群体化方法的关键技术、开发环境和应用实践.

关键词 可信软件 群体协同 资源分享 运行监控 可信分析 软件演化

1 研究背景

提高软件生产效率和质量一直是软件领域的重大课题. 随着网络时代的来临, 软件的开发模式、运行环境和提供方式发生了巨大变化, 经典软件开发方法在提高软件生产效率和质量方面的局限性日益凸显, 软件可信问题更加尖锐. 传统的可信软件解决之道中隐含 2 个假设: 第一, 高质量的软件是设计和构造出来的, 本文称其为 "构造论"; 第二, 软件缺陷是人在设计和构造软件过程中的失误导致的, 本文称其为 "人为论". 因此, 传统的软件开发方法和技术更加关注在软件开发阶段排除人为失误对软件质量的影响, 产生了自动化和工程化两个具有里程碑意义的经典的软件开发方法 (见表 1).

自动化方法源起于 20 世纪 60 年代高级程序设计语言及其自动编译技术 [1,2]. 该方法的核心思想是建立高级形式化系统, 给出软件需求和设计的高层次形式化规约, 在此基础上, 通过自动变换由规约产生符合规约的可信目标代码. 自动化方法取得的成就是无容置疑的. 但是, 人们很快发现自动化

引用格式: 王怀民, 尹刚, 谢冰, 等. 基于网络的可信软件大规模协同开发与演化. 中国科学: 信息科学, 2014, 44: 1–19, doi: 10.1360/N112013-00128

表 1　软件开发方法的 3 个重要里程碑

Table 1　The three important milestones of software development methods

Time of emergence/rise	Software development methods	Typical technology and practices
1960s	Automation-based method	High-level language
1970s	Engineering-based method	Software engineering
1990s	Crowd-based method	Open source software

方法面临一系列不可逾越的理论极限 [3~5]. 进入网络时代, 大量成功的网络化软件系统越来越远离自动化方法, 而自动化方法在其重点关注的领域 (例如, 关键领域的嵌入式软件), 随着问题复杂性和软件规模的增大其理论研究与实践步履维艰.

工程化方法源自 20 世纪 70 年代初软件工程概念的提出与初步实践 [6]. 工程化方法的核心思想是将软件视为一类新型产品, 借鉴传统工业产品的产生模式, 建立严格的工程规范, 在软件生命期的各个阶段, 通过规范管理和辅助工具, 提高软件开发质量与效率, 最大限度地减少人为错误的机会, 或尽可能早地发现人为错误. 工程化方法在推进软件产业化进程中的贡献也是无容置疑的. 但是, 人们在实践中也很快发现, 软件不同于有形的工业产品, 突出的一点就是很难给出稳定清晰的需求定义 (这一特征在网络时代更加鲜明), 需求获取成为软件工程的瓶颈 [7], 而工程化方法要求在有限的成本和时间内发布符合预期的软件, 这种矛盾导致超预算或推迟发布的软件项目大量出现, 大量软件项目最终不得不被撤销, 失去继续发育的机会. 进一步, 工程化方法极大地影响人的创作灵感的发挥.

进入网络时代, 软件技术和应用的发展呈现出 “人本” 和 “演化” 的鲜明特征, 从而影响了我们对软件开发活动和可信软件生成方法的认识. 首先, 兴起于 20 世纪 90 年代的开源软件取得巨大成功, 其开发活动具有开放分享、频繁发布等特点, 能够有效汇聚软件各利益相关方的创意和贡献, 其自由松散的表象下隐藏着重要的软件开发新理论 [8], 对软件产业的发展和格局产生了深远的影响. 随后, 以 Web2.0 为标志的网络应用软件掀起了新一轮互联网革命, 此类系统使未受过专业训练的业余用户成为各类高品质内容的创造者, 在其他领域也大获成功 [9]. 这两类活动成功实践了大规模协同创作中的两个核心机理:

群体智慧. 以大众为基础的群体协同奇迹般的实现了工业界通过严格工程化手段才能完成 (甚至难以完成) 的开发任务, 专业开发者和业余爱好者通过将软件作品 (以及其他知识产品) 在网络平台上充分分享, 能够充分释放大规模群体的开发潜力, 群体创作成为需求获取和缺陷发现的有效模式.

可信演化. 软件作品 (以及其他知识产品) 的价值在网络环境中得到评估, 网络使得这种评估和反馈能够便捷迅速传播到开发群体, 促使作品朝着用户实际需求的方向快速演化为高质量产品.

这种以群体智慧和可信演化为核心的互联网大规模协同创作机理将对软件开发和应用产生深刻影响. 如何将其与工业化可信软件生产相结合, 提高可信软件的工业化生产效率和质量, 是网络时代软件技术值得高度关注的新课题. 网络时代的软件开发活动是软件创作与软件生产相互交织和迭代的活动. 软件开发技术应该突破线性的 “构造论” 和消极的 “人为论” 的思维定势, 将基于群体智慧的 “人本论” 和基于可信演化的 “演化论” 结合起来. 开源软件的成功实践可能给软件开发带来具有里程碑意义的新途径, 本文将其概括为群体化方法 (表 1). 该方法具有以下四个重要特征:

(1) 群体协同. 软件是客观事物的一种程序化表述, 是知识的固化、凝练和体现 [10], 其开发活动本质上是一种智力和知识密集型的群体协同活动 [11]. 网络为充分发掘开发群体智慧和创造力提供了理

想平台, 开发群体的组织结构和协同机制对软件开发效率和质量有重要影响. 软件开发技术既要关注面向软件产品的协同生产机制, 也要关注面向软件作品的协同创作机制, 以及二者的融合.

(2) 资源分享. 软件生命周期中由开发群体和用户群体产生的软件制品、软件工具和各类数据对后续软件开发活动具有重要的复用和参考价值. 软件的开发环境不仅需要支持对项目内部资源进行充分分享, 还要尽可能利用项目以外的海量软件资源, 让软件资源在 "阳光下" 获得用户认同, 并根据需求和环境变化动态调整用户对软件资源的主观认同, 以支持在大数据条件下的软件资源分享和复用.

(3) 运行监控. 软件在网络环境中往往具有更为复杂的运行结构 [7], 这对判断软件在应用阶段能否适应环境和需求变化提出了挑战. 在对软件系统运行影响最小的前提下, 如何有效获取软件运行数据, 通过分析和挖掘等手段对系统状态和运行故障进行评估和定位, 实现软件系统的动态调整和软件问题的及时反馈, 对软件在运行阶段的快速可信演化至关重要.

(4) 可信分析. 高质量的软件是不断演化而来的 [12], 用户预期的更新和验证将伴随软件的整个生命周期, 网络平台使软件的各利益相关方得以直接参与到软件持续演化过程中. 可信软件不仅需要具有足够的客观质量品质, 其软件质量属性还应当易于被用户分析和评估. 因此管理和分析软件生命周期的相关数据以支持软件的可信评估和快速演化, 是建立可信软件开发平台的重要基础.

群体化方法改变了我们对可信软件的认识, 以及对软件开发活动的认识. 网络时代使建立群体化的可信软件开发方法和技术体系成为可能. 我们提出并付诸实践的基于网络的可信软件大规模协同开发方法和技术体系, 旨在系统的将互联网大规模协同机理引入软件工业化生产活动, 建立以群体协同、资源分享、运行监控、可信分析为核心的可信软件开发服务环境, 实现软件创作与生产深度协同、软件资源的充分分享、软件协同过程的按需定制、软件资源的可信评估, 以支持可信软件的群体化协同开发和可信演化.

本文的内容组织如下. 第 2 节给出可信软件模型, 给出可信软件的内涵描述和生命周期的三个主要阶段, 据此给出可信软件的证据模型和演化模型; 第 3 节提出可信软件开发环境的核心服务模型, 详细分析旨在将互联网协同机制引入软件工业化生产的四类服务机制, 即群体协同服务、资源分享服务、运行监控服务和可信分析服务的核心机制和设计原则; 第 4 节总体介绍在国家高技术研究发展计划重点项目支持下开展的实践验证工作; 第 5 节是相关工作分析; 第 6 节总结全文并介绍未来工作.

2 可信软件模型

网络技术的发展深刻改变了软件的开发、运行和提供方式, 也使其内涵和形态发生了变化. 我们提出的可信软件模型是一种以证据为核心的软件模型, 是对传统软件模型的继承和发展.

2.1 概念模型

关于软件的基本内涵, 一种经典的描述是计算机系统中的程序和有关的文档, 其中程序是计算任务的处理对象和处理规则的描述, 文档是为了便于了解程序所需的资料说明 [2]. 在某些情况下, 软件的文档还包括应用阶段的问题和维护记录. 传统软件的生命周期主要划分为开发阶段和应用阶段, 软件的成熟版本完成后通过分发过程从开发阶段进入应用阶段, 应用阶段形成的用户意见将反馈给开发团队. 上述观点适于描述以计算机系统为中心的软件形态, 其变化过程也更适于工业化软件生产活动的分工, 本文将其称为经典软件模型, 如图 1 (a) 所示.

网络时代以来, 软件所处的环境已经由计算机系统为中心改变为网络为中心, 网络促成了开发方

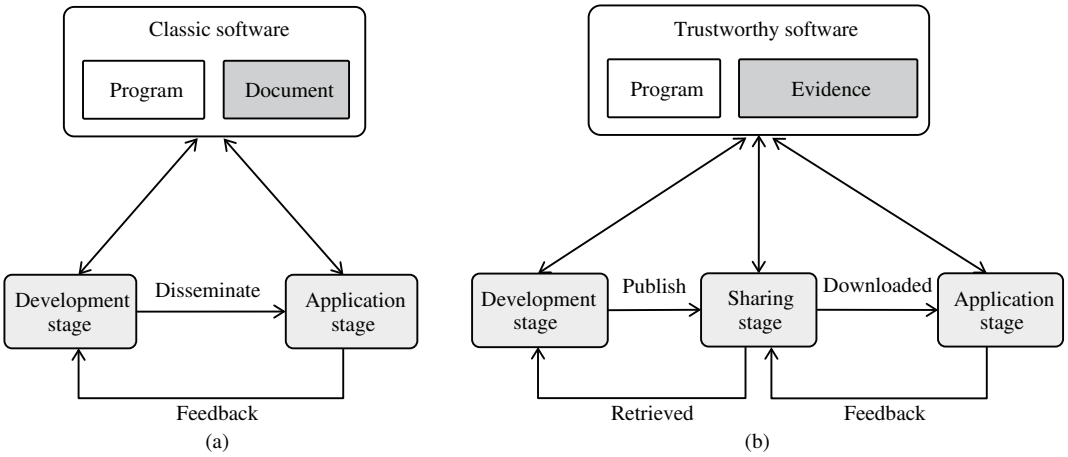

图 1　经典软件与网络时代可信软件的概念模型

Figure 1 The concept models of classic software and trustworthy software in Internet era. (a) Classic software model and its life-cycle; (b) trustworthy software model and its life-cycle

式和应用模式的多样化, 软件开发、分享和应用阶段产生的数据类型多样、内涵丰富, 而且得以实现海量存储和快速分享. 这些数据不仅能够对软件的功能和性能进行说明 (如各类文档), 还能用来分析和度量软件的各种性质 (如各类过程和制品数据). 为此, 本文在经典软件模型基础上建立一种可信软件模型, 如图 1 (b) 所示.

可信软件模型中, 可信软件由程序和证据组成. 其中证据是软件开发阶段、分享阶段和应用阶段产生的有助于度量和描述软件可信属性的数据, 如图 1 (b) 所示. 经典软件模型中的文档是证据的一种形式. 分享阶段是可信软件在其生命周期中集中形成和利用群体智慧的最重要的阶段. 软件的阶段性版本通过网络进入分享阶段 (通常是提交到某个软件分享和交易平台). 软件进入分享阶段后即可供互联网用户下载和使用、分析和评价, 这些数据都反馈到软件的分享阶段, 软件开发群体可以直接从中获取该软件的各种反馈.

基于该模型, 现实世界中的软件主要有三种管理状态: 软件项目、软件资源和软件实例. 软件项目是软件开发群体形成的软件制品和开发数据的集合; 软件资源是软件在分享阶段发布并汇集的程序及其说明文档、软件各利益相关方反馈数据的集合; 软件实例是在特定运行环境中程序的执行实体、状态数据和应用数据的集合.

2.2　可信证据模型

可信证据是能够直接或间接反映软件可信性的软件利益相关方在开发、分享和应用阶段产生的数据. 其中, 可信属性不仅包括软件的客观质量属性, 如可用性、可靠性、性能、正确性、保险性、安全性和私密性 [13], 也包括人们对软件的主观判断 [12], 在某些情况下还包括软件之间经过比较的这些质量属性的相对性度量. 可信软件在不同生命周期阶段会产生不同类型的可信证据.

可信软件的证据模型主要包括 3 类:

• 开发证据. 开发活动产生的相关数据, 包括反映软件制品、开发过程、开发群体等的各种属性的度量和描述. 例如, 对软件版本的代码质量、缺陷回归效率以及开发行为等的度量, 其中包括形式化工具和工程化工具产生的证据.

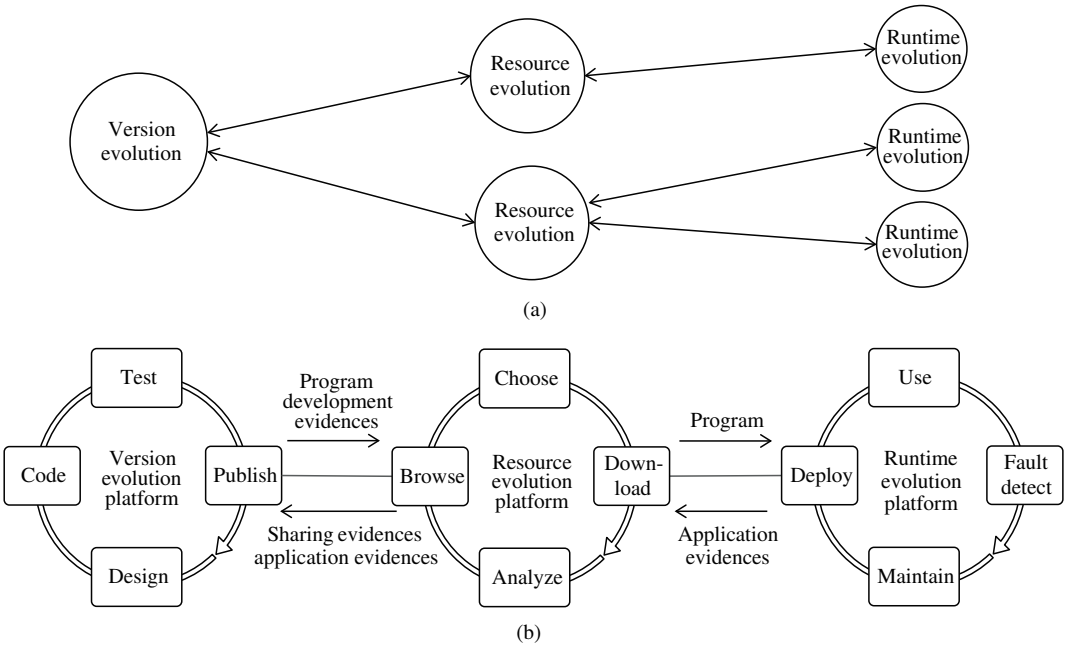

图 2 软件的三类演化循环及其交互关系

Figure 2 The three software evolution loops and their interconnections. (a) The relations between the three kinds of software evolution; (b) the internal actions in software evolution loops

• 分享证据. 分享活动产生的相关数据, 包括软件的下载量、关注度、活跃度以及各种排名等网络社区特有的度量和描述. 分享证据是一种能够对同类软件进行相互比较和排序的特殊证据.

• 应用证据. 应用活动产生的相关数据, 包括软件用户、测试机构等提交的关于软件质量和性能等反映软件功能和性能特征的度量和描述, 以及可检测的软件系统的运行状态和日志等.

分享证据是软件在网络分享平台上形成的一种特殊可信证据, 本质上反映了软件的社会属性, 具有主观性和流行性等特点, 但往往具有更实际和更直接的说服力.

从可信证据的角度看, 软件项目包含软件的所有开发证据, 软件实例包含软件所有应用证据, 软件资源则包含所有分享证据, 以及部分开发证据和部分应用证据.

2.3 软件演化模型

软件演化是软件不断修改以满足用户预期的过程, 是网络环境下大规模群体协同推动形成高质量软件的有效途径. 其一方面根据软件可信证据对软件进行修改和升级, 另一方面软件的演化活动也将产生新的可信证据. 基于可信软件的生命周期模型, 本文将可信软件的演化活动划分为版本演化、资源演化和运行演化, 分别对应于软件开发阶段、分享阶段和应用阶段. 给定某一特定软件, 一个演化的版本可能产生多个演化的资源, 一个演化的资源可能产生多个演化的运行实例, 如图 2 (a) 所示. 其中:

• 版本演化. 主要是软件代码在开发阶段不断改进的过程, 包括对新需求实施的设计、编码、测试, 以及软件的发布等, 不同软件可能采用不同的开发过程. 版本演化活动将产生开发证据, 同时也会利用资源演化和运行演化活动产生的可信证据调整开发任务, 如图 2 (b) 所示.

• 资源演化. 主要是由于三个阶段的软件可信证据的不断更新直接或间接导致的软件可信属性的改变. 基本的资源演化步骤包括程序的更新、可信证据及其相对性的更新、可信属性的计算等.

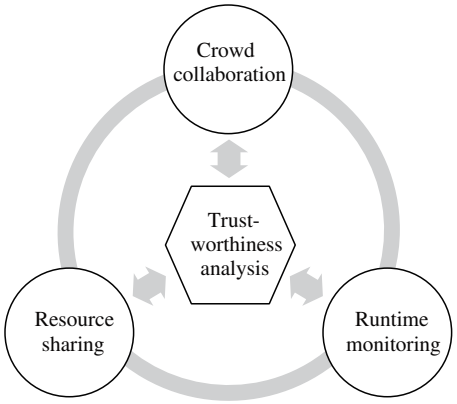

图 3　基于网络的可信软件开发与演化服务体系结构

Figure 3　The service architecture for development and evolution of network-based trustworthy software

• 运行演化. 主要是软件在运行时的演化活动, 包括软件更新、应用部署、系统维护、故障处理等, 不同类型和不同规模的系统在应用阶段的演化过程往往存在很大差异. 运行演化产生的应用反馈和运行记录将作为可信证据提供给资源演化活动, 如图 2 (b) 所示.

软件的不同演化活动存在复杂的相互影响和制约关系, 任何一种演化活动的低效都可能对软件的发展造成严重影响. 例如, 软件运行演化的实例越多, 其反馈给资源演化实例的可信证据就可能更加及时和充分, 其版本演化活动就更快的获得软件缺陷和潜在需求, 从而做出快速修正和完善. 版本演化则是其他演化活动的原始推动力, 不良的版本演化活动 (如管理不善的软件项目) 很可能导致软件的失败, 或者促使形成新的版本演化活动 (如开源环境下的软件项目的一个新分支可能比原始项目更成功). 此外, 资源演化则需要关注软件证据数据的汇聚、分享和分析, 并通过构建社区机制便于软件厂商和用户的参与.

在云计算技术的推动下, 越来越多的软件开发商将原本在客户端运行的单机版软件以服务的形式通过网络提供给用户, 从某种程度上实现上述三种演化活动的有机统一, 这有助于提高软件演化活动的整体效率和质量.

3　可信软件协同开发与演化服务模型

可信软件协同开发与演化服务是一种基于网络的基础软件, 支持软件的协同开发, 支持软件在三个阶段的可信证据的形成、采集、分享和利用, 加速软件的可信演化, 以提高可信软件的开发效率和软件质量. 该服务体系结构以可信证据为中心, 通过群体协同、资源分享、运行监控和可信分析四类核心服务实现 (如图 3 所示). 其中群体协同服务提供大规模群体创作、生产以及两类活动的融合与转化机制; 资源分享服务提供实体分享和证据分享等机制, 实现软件的快速分发和应用反馈; 运行监控服务提供软件实例运行时行为数据采集、汇聚和分析服务; 可信分析则能够对群体协同服务、资源分享服务和运行监控服务中获得的数据进行综合度量和关联分析, 为开发群体提供面向不同开发任务的综合分析机制.

该体系结构以软件演化过程中形成的海量可信证据为基础, 四个核心服务将输出不同类型的可信证据, 同时也利用可信证据实现各种度量和分析工具. 其中, 群体协同服务产生的开发过程数据是开

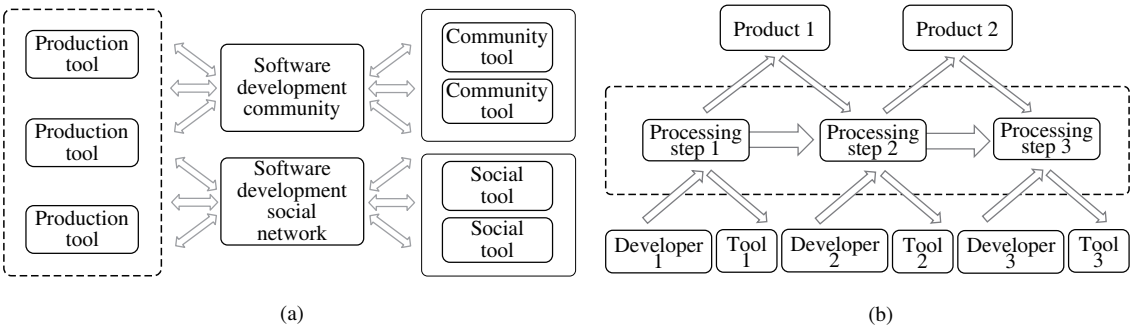

图 4 基于网络的群体协同服务的实现模型

Figure 4 The implementation model of network-based crowd collaboration service. (a) The two integration models for software creation and production; (b) the tool integration model for development progress organization

发证据的主要来源, 包括版本库、提交日志、缺陷库等开发过程数据和阶段性制品数据; 资源分享服务主要任务是建立资源分享社区, 积累相关开发证据和应用证据, 不断形成并更新分享证据; 运行监控服务是软件实例行为数据的主要来源, 对于不可预测应用环境, 软件实例的行为数据对于评估和改进软件运行时的可信性至关重要; 可信分析服务对各类证据进行综合度量、分析和评估, 为软件开发者和用户提供不同的度量和分析工具.

四类核心服务相互联系, 彼此通过开放接口提供服务. 例如, 群体协同服务为了更好的支持敏捷开发, 可以通过资源分享服务提供的接口为程序员推荐可能会用到的软件资源 (如构件、服务和源代码); 资源分享服务可以调用可信分析服务的接口以计算某个软件资源的可信度量, 群体协同服务则可以调用可信服务对当前的开发活动进行评估, 如代码质量、开发效率评估等; 运行监控服务则可以将识别出来的关键性系统故障发布到资源分享平台, 实现软件缺陷的直接发布.

3.1 群体协同服务

群体协同服务的基本出发点是支持软件创作和软件生产两者之间的融合与转化. 软件创作活动往往由创作群体的灵感和创意驱动, 其输出的作品 (如原型软件) 本质上是灵感的捕获和表达; 软件生产活动则是由预先定义的用户需求驱动, 以控制软件缺陷和降低成本为主要出发点. 两类活动的融合是指创作活动与生产活动之间的相互感知和集成, 两类活动的转化是对软件作品进行筛选、改进并集成到软件产品中, 是软件创新活动中不可缺少的中间环节 [14].

兴起于互联网的交流和分享工具对软件创作活动具有重要支撑作用, 如论坛、博客、维客、微博等. 此类工具能够使软件创意或作品在短时间内快速传播, 引起潜在用户与贡献者的关注和参与. 而面向软件生产的各种工具主要用于提高软件开发活动的自动化水平和过程管理水平, 以在项目预算和期限内完成产品开发, 包括桌面开发、版本管理、过程管理、缺陷管理等工具. 群体协同服务为可信软件大规模协同开发提供上述两类开发工具及其集成机制. 其中的难点是如何将两类工具有效结合以支持软件大规模群体协同开发, 这里提出 3 种参考集成模式:

• 社区模式. 社区 (community) 是一种支持具有共同兴趣的群体进行交流和分享的机制, 社区工具通常包括新闻组、论坛、邮件列表等. 社区模式是将社区工具和软件生产工具相结合, 围绕开发过程或软件制品形成开发社区. 例如, 开发者和用户可以围绕某个软件缺陷建立讨论区, 以开展各种交流和分享. 社区模式能够为软件开发提供交互性更强的创意分享与技术支持平台, 如图 4 (a) 所示.

• 社交网络模式. 社交网络 (social network) 是一种用于维护用户间社交关系的机制, 社交工具通过关注、好友、群组等机制建立社交网络, 用户动态会通过社交网络传播给其他相关用户. 软件开发者之间通常会形成较为复杂的社交网络, 如合作关系网络、缺陷抄送网络 [15] 等. 面向软件开发的社交网络能够维护开发活动中的社交关系, 实现开发行为的自动感知和互动, 如图 4 (a) 所示.

• 过程组织模式. 有效复用软件工程活动中行之有效的协同过程及工具能够显著提高生产效率, 该集成模式利用软件生产线实现开发过程的定制和复用. 软件生产线是一种基于网络的集成化、可扩展、协同化的新型软件开发环境 [16], 其按照给定的开发步骤将开发过程涉及的开发者、工具和制品等要素进行有序组织和定制, 能够为开发团队快速定制专用软件开发环境, 如图 4 (b) 所示.

3.2 资源分享服务

资源分享服务是可信软件资源演化的主要平台, 主要提供软件程序分享和可信证据分享, 需要解决的关键技术问题包括大规模软件资源的获取、存储、检索、可信分析等. 其中, 软件程序分享机制能够加速软件代码的分发和传播过程, 使软件尽快从版本演化阶段进入运行演化阶段, 使软件的特性和问题尽可能早的显露出来; 软件证据分享机制能够加快软件可信证据的传播、更新、积累, 提升软件开发团队对各种应用反馈的响应速度, 从而加速软件的可信演化. 这两类机制具体描述如下:

• 软件程序分享. 支持包括软件构件、软件服务等形态的软件程序 (或软件服务地址) 以及相关说明文档的发布、检索、访问和更新等. 对于软件服务, 软件实体数据还包括各服务实例的接口描述和访问地址等. 特别的, 对于开源软件, 软件程序还包括软件的版本库地址、源代码、编译和安装脚本等.

• 软件证据分享. 支持包括软件在开发、分享和应用三个阶段的可信证据的发布、检索访问和更新等. 对于软件服务, 其可信证据还包括服务实例的实时可用性和运行状态等; 对于开源软件, 其可信证据通常包括缺陷库、邮件列表、许可证、赞助商、开发活跃度等.

目前, 互联网中的软件资源以不同形式广泛分布在各类软件资源站点中. 以开源软件为例, 资源站点包括各种开源社区网站 (如 Linux, Apache 和 Eclipse 等)、开源软件托管社区 (如 SourceForge, GitHub, Freecode, RubyForge 等)、开源软件目录站点 (如 ohloh). 这些站点包含大量公众可访问的软件程序和不同类型的可信证据, 由于软件资源的各类数据标准尚未在互联网中广泛采用, 而且这些资源站点的资源组织模式也各有特点, 如何从开放软件资源站点中收集和筛选高质量的软件资源是资源分享服务面临的挑战.

此外, 互联网中分布着大量软件社区站点 (如 StackOverflow 和 CSDN 等), 其中包含大量软件的描述和评价信息, 是软件应用证据和分享证据的新的数据源. 因此, 软件资源分享服务一方面应支持软件资源的发布和管理, 另一方面应能快速吸纳互联网中的开源软件资源和可信证据 (如图 5 所示), 资源分享服务的具体支撑机制包括: (1) 资源封装. 支持软件资源中不同类型的数据有序组织和存储, 如 RAS 机制 [17] 等. (2) 资源发布. 支持用户向资源分享服务平台注册和提交各类软件实体及其可信证据. (3) 资源抓取. 能够从互联网的各类软件资源平台中收集、解析和整理软件资源数据, 为建立大规模软件资源分享服务提供自动化工具. (4) 资源组织与搜索. 支持软件资源的分类检索和搜索, 并根据可信属性对结果进行排序, 实现高效的海量软件资源检索和查询能力.

3.3 运行监控服务

运行监控服务是软件在应用阶段实现可信演化的基础支撑设施. 其基本思想是通过对软件实例运行状态的监测, 获取软件系统的行为和状态信息, 为可信分析服务提供原始的或经过过滤的运行日志

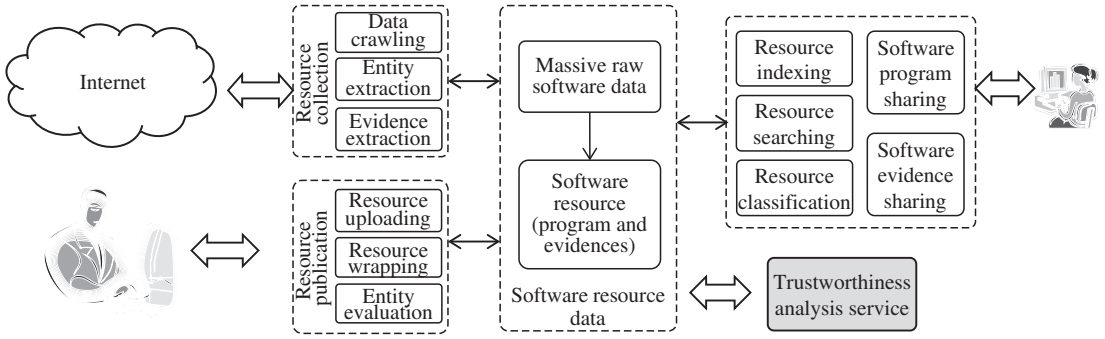

图 5 软件资源分享服务的核心机制

Figure 5 The key mechanisms of software resource sharing service

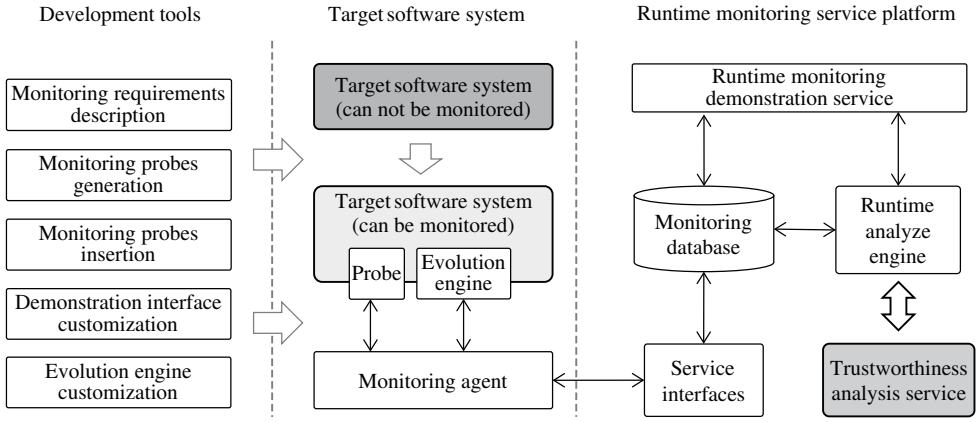

图 6 软件运行监控服务模型

Figure 6 The model of software runtime monitoring service

数据, 以支持软件系统的故障定位和动态调整, 使软件系统尽可能健康可信的运行.

运行监控服务的参考模型包含 3 方面内容, 如图 6 所示.

• 监控开发工具. 主要是将一个不能被监控的软件转化为一个可被监控的软件, 一种通用的解决思路是在目标软件系统中植入监控探针 (基于标准化运行平台的软件系统也可采用网络消息分析的手段进行监测). 该方法通过监控需求描述、探针生成、探针注入、界面定制等工具实现目标软件的监控改造. 此外, 监控开发工具还可以包括在软件系统中植入演化引擎以支持其动态调整, 但此类工具的实现需要得到目标软件系统运行平台的支持.

• 目标软件系统. 监控开发工具能够将目标软件系统从一个不可监控的软件转化为一个可被监控的软件. 其中, 监控探针将系统的相关运行状态发送给监控代理, 通过监控代理发送到运行监控服务平台. 对于自适应要求较高的软件系统, 可以通过演化引擎实现系统的动态调整: 演化引擎可以执行监控代理发来的演化指令, 也可以根据本地监控数据自行调整系统.

• 运行监控服务平台. 一种支持对多个目标软件系统进行监控的远程服务, 包括监控数据库、监控服务访问接口、运行分析引擎、运行监控展示服务等. 其中监控数据库负责存储原始监控数据和识别后的故障事件数据 (对于大型复杂软件系统, 该数据库需要具备流数据和海量数据处理能力); 运行分析引擎调用可信分析服务对监控数据进行分析和挖掘, 实现系统运行状态的评估、系统故障的诊断,

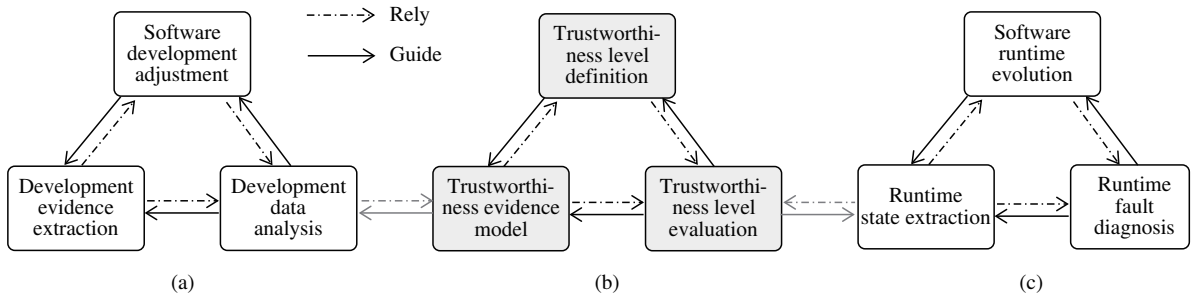

图 7　基于证据的软件数据可信分析服务模型

Figure 7 The model of evidence-based software data trustworthiness analysis service. (a) Development data analysis model of software project; (b) trustworthiness evaluation model of software resources; (c) runtime data analysis model of software instances

并更新监控数据库; 监控代理可以通过监控服务访问接口发送监控数据、获取故障信息或者系统动态调整指令等.

上述运行监控服务模型强调运行监控可以作为一种新型服务提供给网络中运行的软件系统, 这不仅有助于简化目标软件系统的应用逻辑, 也能够发挥监控服务的数据优势, 提升监控效果.

3.4　可信分析服务

可信分析服务是软件在开发阶段、分享阶段和应用阶段实现可信演化的支撑机制, 通过对大量软件证据数据的分析和挖掘实现对软件项目、软件资源和软件实例的度量和评估, 使软件的演化活动向正确的方向发展. 可信分析服务主要包括对开发证据、分享证据和应用证据的综合评估, 其参考模型包含 3 方面内容, 如图 7 所示.

• 开发数据分析. 根据提升软件项目开发效率和制品质量的实际需要, 对软件开发证据 (包括开发阶段产生的过程数据和阶段制品) 进行分析, 以对软件开发活动进行度量和评估的各种机制. 此类机制的核心模型如图 7 (a) 所示. 软件开发数据提取就是从软件项目任务开发环境中抽取与分析目标相关的开发证据, 该过程可能反过来对软件项目过程数据的组织方式提出新要求; 软件开发数据分析就是对提取到的证据进行分析和度量, 对软件开发状态和面临问题进行评估和定位, 为优化软件开发活动提供依据.

• 运行数据分析. 对软件在运行阶段产生的应用证据进行分析和挖掘, 以评估系统运行状态和诊断故障, 据此对软件系统进行动态调整和问题反馈, 实现软件的可信运行演化, 其核心模型如图 7 (c) 所示. 软件运行状态提取是从软件应用证据中抽取与分析目标相关的关键证据, 包括与应用逻辑无关的平台层证据和应用逻辑相关的业务层证据; 软件故障诊断主要是对证据进行分析以定位和诊断软件运行故障, 识别出来的故障既是运行演化依据, 也可作为应用证据发布到资源分享平台.

• 资源可信分级. 根据用户对软件资源的各种预期 (可信属性) 的不同关注点, 对软件三个阶段的证据进行自动或者人工分析和评估, 最终形成对软件资源关于某类可信属性 (或整体可信性) 的等级评定, 其核心模型如图 7 (b) 所示, 软件可信等级定义是根据用户预期建立的可信分级的需求模型, 可信证据模型则是相关证据集合, 是软件可信等级评定的依据. 可信等级评定是对软件实体进行可信等级评定方法和机制, 该机制通常与软件的应用领域密切相关.

在具体可信软件开发环境的构造中, 图 7 所示的软件项目数据分析模型和资源可信分级模型具有广泛的适用性. 其中, 软件开发数据可以是软件项目的阶段性软件制品 (如源代码或可执行的软件模

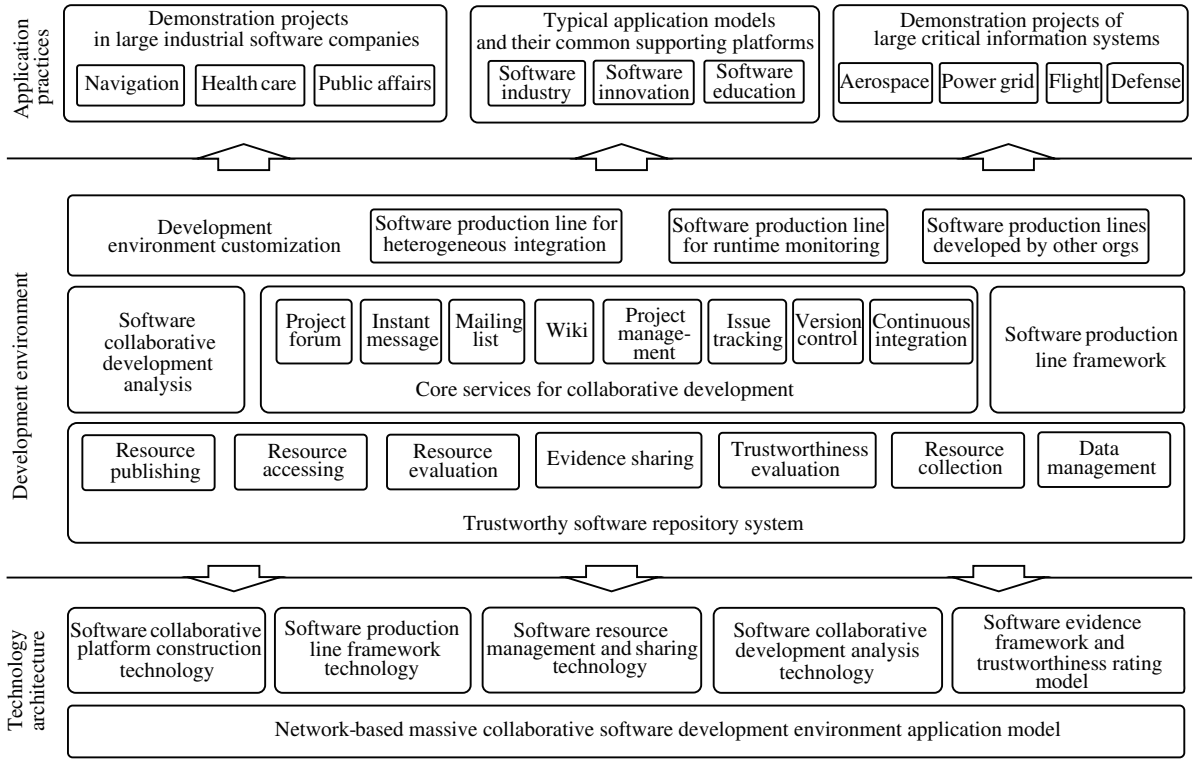

图 8 Trustie1.0 软件开发环境、技术体系和应用实践

Figure 8 The software development environment, technology system and application practices of Trustie1.0

块), 也可以是软件项目某个任务的过程日志 (如开发日志、缺陷列表和邮件列表等); 软件可信等级定义可以显式的给出软件可信级别及其内涵的描述, 也可以是按照某种可信属性给出的一种无监督的排序需求. 软件项目数据分析过程产生的度量和描述是软件开发阶段证据的重要依据.

4 可信软件大规模协同开发环境

基于可信协同软件开发与演化服务模型, 在国家 “十一五” 高技术研究发展计划重点项目 “高可信软件生产工具及集成环境” 支持下, 国防科学技术大学、中国科学院软件研究所、北京大学、北京航空航天大学、山东中创软件商用中间件股份有限公司联合研制出一种基于网络的可信软件大规模协同开发环境 ——“确实 1.0” (Trustie1.0).

Trustie1.0 在我国 “九五” 和 “十五” 期间形成的构件化软件开发技术、软件工具和平台的基础上, 自主研发了 60 余项覆盖软件需求、设计、组装、维护等主要软件生产活动的软件工具以及面向交流与分享的社区和社交工具, 研制出软件生产线框架系统和软件协同开发分析系统, 并研发了 4 个面向自动化软件生产的生产线系统. Trustie1.0 于 2008 年 6 月在长沙软件园完成部署并开始对外提供服务, 其技术和平台已在我国软件产业、人才培养和技术创新等领域形成了典型应用模式 [18], 得到了广泛的应用验证, 如图 8 所示.

4.1 可信软件协同开发服务

Trustie1.0 可信软件协同开发服务由软件创作工具和软件生产工具组成, 通过社区模式和社交网络模式实现了深度集成. 在此基础上, Trustie1.0 实现了协同开发分析系统以对开发行为进行统计和分析, 实现了软件生产线框架系统以支持协同过程的组织和开发环境的定制. 其关键技术包括:

• 社会化协同开发核心服务. 实现了软件创作和生产活动的深度融合, 将项目论坛、协同编辑、邮件列表、即时通讯等交流分享工具和项目管理、配置管理、缺陷管理、持续集成等工业化生产工具集成在统一的开发环境中 [19], 并通过社区机制和社交网络机制实现技术问题的交流和分享, 以及开发活动的感知和互动.

• 协同开发分析系统. 建立了海量软件开发数据的获取和存储平台, 在配置管理工具中集成了多种开发活动度量工具, 将制品证据和团队证据相结合实现了一种软件项目综合评估方法 [20]. 此外, 该系统集成了一种程序员开发能力度量工具, 能够对开发者的成长轨迹进行精确分析 [21].

• 软件生产线框架系统. 能够按照给定的开发步骤, 将软件开发过程涉及的开发者、开发工具和软件制品等三方面要素进行有序组织和定制, 形成相应的软件生产线, 为开发团队快速定制专用软件开发环境 [22]. 例如, 支持运行监控的软件生产线 [23] 是一种典型的专用开发环境.

截止 2013 年 6 月, Trustie1.0 可信软件协同开发服务积累了 600 多个开发项目和 700 多个竞赛项目的开发证据, 还从众多主流开源社区的高质量开源项目开发数据中提取出符合 Trustie1.0 证据规范 [24] 的开发证据. 这些开发证据可以支持软件协同开发分析系统对国际开源软件项目的实时度量, 也有助于支持各种软件工程实证研究.

4.2 可信软件资源分享服务

Trustie1.0 的可信软件资源库系统支持用户发布、检索、评估软件资源, 是一种能够将传统的 “封闭、静态、无序” 的软件构件存储模式转变为网络条件下 “开放、动态、有序” 的软件资源分享模式 [25] . 结合课题组提出的软件可信分级与评估模型 [26] , 将软件可信证据框架引入软件资源的信息体系 [27]、将集中式资源静态存储和开放式资源动态聚合技术进行有机结合, 实现了基于网络搜索的开放式软件资源获取、软件资源分享及其可信分级.

• 软件资源收集整理. 采用资源发布和资源搜索两种技术实现大规模软件资源的收集, 并基于 RAS 技术规范实现了支持软件资源全生命周期的证据封装和使用, 支持基于文本和标签的跨社区大规模软件资源分类机制 [28] 和特征分析 [29], 能够有效提升资源组织管理的效率. 截止 2012 年 6 月, Trustie1.0 已发布项目组自主研发的 170 余项软件资源, 从国际软件构件库、服务目录和开源社区收集筛选了超过 61000 项软件资源, 其中包括软件构件 11000 余项、软件服务 48000 余项.

• 软件资源可信评估. 主要包括根据软件资源结构化证据的自动评估机制和人工评估机制, 如面向服务可信演化的评估机制 [30,31]. 虽然软件可信分级与具体应用领域密切相关, 但是本文认为存在一个具有普遍意义的、能够反映基本软件特征的软件可信分级参考模型, 并且可针对不同应用领域的需要进行定制.

Trustie1.0 软件可信分级模型是一种以用户预期、应用实证、评估手段为主要分级维度的软件可信分级参考方法 [26], 参见表 2. 在某些特殊领域 (如航天领域), 软件在首次应用前即要达到较高的可信级别 (如第 4 级), 而达到第 3 级的前提是存在成功应用案例. 这就要求此类领域的可信分级规范针对领域特点对相关描述进行具体化, 如可将 “成功应用案例” 定义为 “在特定的仿真或模拟环境中得实验成功”.

表 2 Trustie1.0 软件可信分级模型

Table 2 The software trustworthiness classification model of Trustie1.0

Level	Name	Description
0	Unknown	The lowest trustworthiness level. It means no software trustworthiness evidence is found, it cannot be determined whether the target software satisfies the expectations of users over the trustworthiness of the same software category.
1	Available	The software entity can be accessed and work as described by the software provider. It implies that the target software satisfies the basic expectations of users over the functional attributes of the same software category.
2	Verified	On the basis of the available level, software provider declares the software trustworthiness attributes by some documents. The declaration can be verified through domain-specific software evaluation mechanisms. It indicates that the target software satisfies the general expectations of users over the trustworthiness attributes of the same software category, and these user-expected trustworthiness attributes are verified.
3	Applied	On the basis of the verified level, the software has been applied in related domains and has verifiable cases of successful applications. It implies that the software satisfies the general expectations of users over the trustworthiness attributes of the same software category, and these user-expected trustworthiness attributes have been verified by practical applications.
4	Assessed	On the basis of the applied level, trustworthiness of the software should pass the assessment conducted by some authoritative agencies according to specific trustworthiness evaluation standards. This indicates that the software satisfies relatively higher expectations of users over the trustworthiness attributes of the same software category, and these user-expected trustworthiness attributes are confirmed by the authorities.
5	Proved	The highest trustworthiness level. It means on the basis of the assessed level, the user-submitted software trustworthiness attributes can be formally proved.

5 相关工作

在软件技术发展史上, 软件开发技术及支撑环境一直扮演着推动软件技术创新和产业发展的重要角色, 对于提高软件的开发效率和质量具有重要意义. 网络技术的兴起和广泛应用为软件开发环境带来了新的发展机遇, 软件开发环境从传统的以计算机为中心且主要关注个人开发效率的软件开发工具、集成开发环境、扩展开发环境, 逐步发展为以网络为中心的关注群体开发效率和制品质量的协同开发环境.

在软件开发方法和开发环境体系结构层面, 国内外学者从多个角度研究了网络对软件开发技术的影响. 我国学者较早认识到互联网对软件形态和开发活动产生的深刻影响, 率先系统的提出了网构软件 (Internetware) 的新型软件模型, 形成了一套以体系结构为中心的网构软件技术体系和开发方法 [10]. 对进一步发展网络环境下可信软件的概念模型、演化过程以及支撑环境构造具有重要指导意义. Grady 等人 [32] 在分析 SourceForge 等开源项目托管平台的基础上给出了软件协同开发环境的定义和基本特征, 并提出了协同开发环境的概念体系结构. Rick 等人 [33] 则以开源软件和社区服务系统为原型构建一种适于软件众包的大都市模型 (metropolis model), 提出了一种区分核心人员、外围人员和大众的层次化软件开发体系结构, 强调了开放、混搭、非正式软件需求、持续演化、动态资源、行为涌现等原则. Tapscott 等人 [9] 通过大量案例分析指出, 随着互联网的不断发展和 Web 2.0 的日益流

行, 包括软件开发在内的诸多商业模式皆越来越倚重以对等、开放、分享和全球行动为核心的大众协作 (mass collaboration) 的思想与机制. Herbsleb[34] 提出了全球化软件工程的概念, 并分析了全球化软件开发在软件体系结构设计、需求获取、软件开发环境与工具支持方面的新要求, 在此基础上提出值得深入的应对技术与研究方向. Howe[35] 全面阐述了在软件开发等业务中众包模式的形成、现状与未来发展, 认为开源开发运动、协作工具的发展与自组织社区是形成众包环境的关键, 对构建完整的软件开发生态环境具有启发意义.

在软件协同开发工具和技术方面, 近年来出现了大量面向分布式和社会化软件开发技术和行为分析的研究工作. Mockus 等人 [36] 通过对著名开源软件 Apache 服务器的开发数据进行分析, 认为将开源社区和工业界的最有效的开发工具结合起来, 可能有助于提高软件开发效率, 该研究在软件工程领域产生了重要影响. Crowston 等人 [37] 分析了 120 个来自 SourceForge 的项目团队及其缺陷跟踪系统, 揭示了开源软件项目中的程序员交互模式, 并发现大型开源开发团队中的交互模式往往更为分散. Sarma 等人 [38] 提出了一种称为 Tesseract 的开发活动中的 “社会－技术” 依赖关系的可视化浏览器, 旨在深化对分布式开发活动中的 “社会－技术” 关联的感知与理解. Dabbish 等人 [39] 对 GitHub 这一项目托管与社会开发网站进行了研究, 通过对其用户进行访问调查, 分析了 GitHub 透明、开放的社会化开发模式对于素未谋面的用户群体进行协作开发的支持: 主要包括用户之间开发特长、个人目标、影响力的感知. Posnett 等人 [40] 分析了分布式开发中的程序员对软件制品的关注、拥有的问题, 提出一种类似生态学中 “捕食者－猎物” 食物链的程序员贡献网络模型, 并通过实证研究发现程序员的关注点分散情况与软件缺陷的数量存在联系. Bird 等人 [41] 通过对 Windows Vista 开发过程的研究, 分析对比分布式开发和集中式开发模块对应的发布后缺陷数据, 说明地理位置的差异并不会对软件缺陷产生很大的影响, 组织结构的差异相对于地理位置更能对软件质量产生影响. 目前, 协同开发技术的研究更侧重实证分析, 据此对现有协同开发工具的改进将是今后发展的趋势.

在软件资源及其可信评估方面, 随着开源软件的快速发展, 开源托管网站已成为互联网中规模最大的软件资源库, 对开源软件的度量评估也成为研究热点. 目前国际主要开源社区如 SourceForge、GitHub 等积累了海量的软件资源. 2009 年 Mockus[42] 对全球大型开源软件资源库的资源进行了统计分析, 其中 SourceForge 中的开源项目总数超过 12 万, GitHub 中的 Git 仓库数目超过 13 万; 2013 年这个数字分别为 47 万 (SourceForge, 2013.06) 和 400 万 [43], 分别增长了 4 倍和 30 倍. 随着开源软件的大规模应用, 研究人员开始关注开源软件的可信性度量问题. 事实上, 软件可信性在本质上可理解传统软件质量概念在互联网时代的延伸 [10], 如何科学地评价软件质量一直是软件工程领域面临的挑战性问题之一 [44]. 经过 40 多年的发展, 以度量为基础的软件度量学 (software metric) 成为了软件工程领域的一个重要研究方向, 使得软件的质量评估技术逐渐趋向专业化、标准化和工程化, 并产生了很多具有重要影响力的软件质量模型 (如文献 [45,46] 等). 在此基础上, 人们从支持软件项目的社区情况入手, 提出了包括产品功能、社区支持、服务以及应用等质量属性, 形成了 Navica[47]、OpenBRR[1] 和 SQO-OSS[48] 等质量模型. 其中 OpenBRR 模型是相对成熟的开源质量评估模型, 旨在使整个开源社区 (包括企业用户和开发者) 以标准和开放的方式来对开放源代码软件进行评级, 促进开源软件的评估和应用, 其质量属性包括功能性、支持度、服务、应用范围以及开发过程等, 其评估过程需要首先指定一组的评估属性的关注程度, 进而通过打分和加权方法给出整体分值. 目前, 对互联网中开放软件资源的利用和评估仍属于前沿性课题.

在工业界, 软件开发环境与网络协同工具的结合是一个重要发展趋势. CollabNet 是较早有意识

1) OpenBRR, Business Readiness Rating for Open Source, http://www.openbrr.org.

的将开源开发方法集成到软件开发环境中的软件厂商, 其近年来推出的 TeamForge 将配置管理、持续集成、问题跟踪、项目管理、实验室管理 (lab management) 和协同工具集成到 Web 应用生命周期管理平台, 以支持分布式协同开发和高质量软件交付, 是目前功能相对丰富的协同开发环境 [49]. 软件集成开发环境 Visual Studio[50] 是在桌面开发环境基础上发展起来的企业级商用软件开发环境, 其近年来新增的 TFS (team foundation server) 等产品提供团队协同机制和报告机制, 提供版本控制、迭代跟踪、任务面板等分布式开发工具. IBM Rational Jazz[51] 采用一种开放和透明的商用协同开发平台, 允许客户参与持续反馈循环, 其 2008 年推出的团队协同工具、需求创作工具、质量管理工具能够为开发阶段的可信软件生产提供综合支撑. IBM 曾在其商用软件开发环境中引入一种开源市场机制 (IBM's internal open source bazaar, IIOSB)[52], 是将软件创作和生产相结合的有益探索.

6 结论

传统的软件开发技术以 "构造论" 和 "人为论" 为基本假设, 形成了以形式化途径和工程化途径为代表的理论与技术体系, 是计算机时代构造可信软件的技术基础. 网络时代以来, 以互联网为平台的软件开发模式和应用形态给了我们重要启示, 促使我们从 "人本论" 和 "演化论" 出发, 将群体协同、资源分享、运行监控和可信分析四类核心机制引入软件工业化生产, 提出基于群体智慧的可信软件群体化开发方法. 这是在网络环境下研究新一代可信软件构造途径的重要尝试. 可信软件开发环境的效能的释放依赖于软件证据数据的积累和完善, 以及利用可信证据数据和软件工具的有效集成, 实现更大规模和更深层次的群体协同、资源分享、运行监控和可信分析.

从总体上看, 互联网对软件的开发技术、运行形态和提供方式都产生了前所未有的影响, 并催生出一种根植于网络的 "在网络环境中构造、运行在网络上、通过网络提供服务" 的新型软件形态, 我国学者将其形象的称为 "网构化软件"[9]. 在 "十二五" 国家高技术研究发展计划的支持下, 我们正在开展大型网构化软件的可信生产、服务和运行支撑技术研究, 构建 Trustie2.0. Trustie2.0 将采用数据中心和云计算等技术提升软件可信证据管理和分析能力, 提升协同开发服务和资源分享服务的集成度和易用性, 形成完善的服务环境运营机制, 将软件运行演化活动的支持机制与开发环境有效集成, 实现对软件开发、运行和演化的全面有效的支撑.

致谢 在此感谢国家高技术研究发展计划信息领域专家、项目总体组和责任专家的支持与指导, 感谢项目组成员在核心技术攻关和应用实践方面做出的贡献.

参考文献

1 Chen H W, Liu C L, Tan Q P, et al. Programming Language Compiler Theory. 3rd ed. Beijing: National Defense Industry Press, 2000 [陈火旺, 刘春林, 谭庆平, 等. 程序设计语言编译原理. 第 3 版. 北京: 国防工业出版社, 2000]

2 Xu J F. System Programming Language. Beijing: Science Press, 1983 [徐家福. 系统程序设计语言. 北京: 科学出版社, 1983]

3 Turing A M. On computable numbers, with an application to the Entscheidungsproblem. Proc London Math Soc, 1936, 42: 230–265

4 Gödel K, Meltzer B, Schlegel R. On formally undecidable propositions of principia mathematica and related systems. Phys Today, 1964, 17: 92

5 Lehman M M, Belady L A. Program Evolution – Processes of Software Change. London: Academic Press, 1985

6 Naur P, Randell B, eds. Software Engineering: Report of a Conference Sponsored by the NATO Science Committee. Garmisch: NATO, 1968

7 Li D Y. Software engineering in Internet age. China Comput Federation Lett, 2009, 35: 7–12 [李德毅. 网络时代的软件工程. 中国计算机学会通讯, 2009, 35: 7–12]

8 Raymond E S. The Cathedral & the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary. Sebastopol: O'Reilly, 1999

9 Tapscott D, Williams A D. Wikinomics: How Mass Collaboration Changes Everything. New York: Penguin Group, 2006

10 Yang F Q, Lu J, Mei H. Internetware technology system: an architecture-centric approach. Sci China Ser E-Inf Sci, 2008, 38: 818–828 [杨芙清, 吕建, 梅宏. 网构软件技术体系: 一种以体系结构为中心的途径. 中国科学 E 辑: 信息科学, 2008, 38: 818–828]

11 Strubing J. Designing the working process: what programmers do besides programming. In: Gilmore D J, Winder R L, Detienne F, eds. User-Centered Requirements for Software Engineering Environments. Berlin: Springer, 1994. 81–90

12 Wang H M, Yin G. Software credible evolution of the Internet age. China Comput Federation Lett, 2010, 48: 28–36 [王怀民, 尹刚. 网络时代的软件可信演化. 中国计算机学会通讯, 2010, 48: 28–36]

13 Hasselbring W, Reussner R. Toward trustworthy software systems. IEEE Comput, 2006, 39: 91–92

14 Wang H M. Build collaborative sharing production environment of trusted software. China Comput Federation Lett, 2009, 36: 56–61 [王怀民, 构建协同共享的可信软件生产环境, 中国计算机学会通讯, 2009, 36: 56–61]

15 Serrano Zanetti M, Scholtes I, Tessone C J, et al. Categorizing bugs with social networks: a case study on four open source software communities. In: Proceedings of the 2013 International Conference on Software Engineering. Piscataway: IEEE, 2013. 1032–1041

16 Dou W S, Wang W, Gao C S, et al. Collaborative software development environment and its construction method. J Front Comput Sci Technol, 2011, 5: 624–632 [窦文生, 王伟, 高楚舒, 等. 面向协作的软件开发环境及其构造方法. 计算机科学与探索, 2011, 5: 624–632]

17 Trustie Team. Trustie Software Resource Management Specification. TRUSTIE-SRMC V2.0. 2011

18 Trustie Team. Trustie Network-Based Trustworthy Software Production Processes and Environments. TRUSTIE-PE V2.0. 2011

19 Trustie Team. Trustie Collaborative Development Environment Reference Specification. TRUSTIE-FORGE V2.0. 2011

20 Yuan L, Wang H M, Yin G, et al. Mining and analyzing behavioral characteristic of developers in open source software. Chin J Comput, 2010, 33: 1909–1918 [袁霖, 王怀民, 尹刚, 等. 开源环境下开发人员行为特征挖掘与分析. 计算机学报, 2010, 33: 1909–1918

21 Zhou M H, Mockus A. Developer fluency: achieving true mastery in software projects. In: Proceedings of the 18th ACM SIGSOFT International Symposium on Foundations of Software Engineering. New York: ACM, 2010. 137–146

22 Trustie Team. Trustie Software Production Line Integration Specification. TRUSTIE-SPL V3.1. 2011

23 Liu D H, Guo C G, Wang H M, et al. Monitoring enabled distributed software construction method. J Softw, 2011, 22: 2610–2624 [刘东红, 郭长国, 王怀民, 等. 监控使能的分布式软件系统构造方法. 软件学报, 2011, 22: 2610–2624]

24 Trustie Team. Trustie Software Trustworthiness Evidence Framework Specification. TRUSTIE-STE V2.0. 2011

25 Zhao J F, Xie B, Wang Y S, et al. TSRR: A software resource repository for trustworthiness resource management and reuse. In: Proceedings of 22nd International Conference on Software Engineering and Knowledge Engineering, Redwood, 2010. 752–756

26 Trustie Team. Trustie Software Trustworthiness Classification Specification. TRUSTIE-STC V2.0. 2011

27 Cai S B, Zou Y Z, Shao L S, et al. Framework supporting software assets evaluation on trustworthiness. J Softw, 2010, 21: 359–372 [蔡斯博, 邹艳珍, 邵凌霜, 等. 一种支持软件资源可信评估的框架. 软件学报, 2010, 21: 359–372]

28 Wang T, Wang H M, Yin G, et al. Mining Software Profile across Multiple Repositories for Hierarchical Categorization. In: Proceedings of 29th IEEE International Conference on Software Maintenance. Piscataway: IEEE, 2013. 240–249

29 Yu Y, Wang H M, Yin G, et al. HESA: the construction and evaluation of hierarchical software feature repository. In: Proceedings of the 25th International Conference on Software Engineering and Knowledge Engineering, Boston, 2013. 624–631

30 Zeng J, Sun H L, Liu X D, et al. Dynamic evolution mechanism for trustworthy software based on service composition. J Softw, 2010, 21: 261–276 [曾晋, 孙海龙, 刘旭东, 等. 基于服务组合的可信软件动态演化机制. 软件学报, 2010, 21: 261–276]
31 Shao L S, Zhao J F, Xie T, et al. User-perceived service availability: a metric and an estimation approach. In: Proceedings of IEEE International Conference on Web Services. Piscataway: IEEE, 2009. 647–654
32 Booch G, Brown A W. Collaborative development environments. Adv Comput, 2003, 59: 1–27
33 Kazman R, Chen H M. The metropolis model a new logic for development of crowdsourced systems. Commun ACM, 2009, 52: 76–84
34 Herbsleb J D. Global software engineering: the future of socio-technical coordination. In: Proceedings of 2007 Future of Software Engineering. Piscataway: IEEE, 2007. 188–198
35 Howe J. Crowdsourcing: Why the Power of the Crowd Is Driving the Future of Business. 1st ed. New York: Crown Publishing Group, 2008
36 Mockus A, Fielding R T, Herbsleb J. A case study of open source software development: the Apache server. In: Proceedings of the 2000 International Conference on Software Engineering. Piscataway: IEEE, 2000. 263–272
37 Crowston K, Howison J. The social structure of free and open source software development. First Monday, 2005, 10
38 Sarma A, Maccherone L, Wagstrom P, et al. Tesseract: interactive visual exploration of socio-technical relationships in software development. In: Proceedings of IEEE 31st International Conference on Software Engineering. Piscataway: IEEE, 2009. 23–33
39 Dabbish L, Stuart C, Tsay J, et al. Social coding in GitHub: transparency and collaboration in an open software repository. In: Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work. New York: ACM, 2012. 1277–1286
40 Posnett D, D'Souza R, Devanbu P, et al. Dual ecological measures of focus in software development. In: Proceedings of the 2013 International Conference on Software Engineering. Piscataway: IEEE, 2013. 452–461
41 Bird C, Nagappan N, Devanbu P, et al. Does distributed development affect software quality? An empirical case study of Windows Vista. Commun ACM, 2009, 52: 85–93
42 Mockus A. Amassing and indexing a large sample of version control systems: towards the census of public source code history. In: Proceedings of 6th IEEE International Working Conference on Mining Software Repositories. Piscataway: IEEE, 2009. 11–20
43 Begel A, Bosch J, Storey M A. Social networking meets software development: perspectives from GitHub, MSDN, Stack Exchange, and TopCoder. IEEE Softw, 2013, 30: 52–66
44 Liu K, Shan Z G, Wang J, et al. Overview on major research plan of trustworthy software. Bull National Natural Science Foundation of China, 2008, 22: 145–151 [刘克, 单志广, 王戟, 等. "可信软件基础研究" 重大研究计划综述. 中国科学基金, 2008, 22: 145–151]
45 Boehm B W, Brown J R, Kaspar H, et al. Characteristics of Software Quality. TRW Software Series TRW-SS-73-09. 1973
46 McCall J A, Richards P K, Walters G F. Factors in Software Quality. RADC TR-77-369. 1977
47 Golden B. Succeeding with Open Source. Boston: Addison-Wesley, 2005
48 Samoladas I, Gousios G, Spinellis D, et al. The SQO-OSS quality model: measurement based open source software evaluation. In: Proceedings of IFIP 20th World Computer Congress, Working Group 2.3 on Open Source Software, Milano, 2008. 237–248
49 Schubert L. From revolution to evolution – the case for agile ALM. CollabNet, 2012
50 Microsoft Corporation. Microsoft Visual Studio 11 Beta Product Guide, 2012
51 Barnett L. The IBM rational jazz strategy for collaborative application lifecycle management. EZ Insight, 2008
52 Capek P G, Frank S P, Gerdt S, et al. A history of IBM's open-source involvement and strategy. IBM Syst J, 2005, 44: 249–257

Research on network-based large-scale collaborative development and evolution of trustworthy software

WANG HuaiMin[1*], YIN Gang[1], XIE Bing[2], LIU XuDong[3], WEI Jun[4] & LIU JiangNing[5]

1 *National Laboratory for Parallel and Distributed Processing, School of Computer, National University of Defense Technology, Changsha* 410073*, China;*
2 *School of Electronic Engineering and Computer Science, Peking University, Beijing* 100871*, China;*
3 *School of Computer Science and Engineering, Beihang University, Beijing* 100191*, China;*
4 *Institute of Software, Chinese Academy of Sciences, Beijing* 100080*, China;*
5 *CVIC SE Business Middleware Co., Ltd., Jinan* 250014*, China*
*E-mail: whm_w@163.com

Abstract With the coming of the Internet era, great changes have taken place in the development models, runtime environments and providing styles of software. Being an open and collaborative working environment, Internet gives birth to mechanisms for successful crowd-based and collaborative creation, which have had profound impacts on development and application of software. How to combine these mechanisms with industrial approaches for trustworthy software production, and to improve the development efficiency and quality of software, are new challenging issues of software technologies in Internet era. This paper proposes a new approach for trustworthy software development based on the crowd wisdom, namely Crowd-based Method, which centers on crowd collaboration, resource sharing, runtime monitoring and trustworthiness analysis, to support the transformation from creative software works to mature software products, and supports the evolution of software. Firstly, an evidence-based concept model of trustworthy software is proposed, which classifies the data generated at the stages of software development, sharing and application as three kinds of trustworthiness evidence of software, and constructs a new process model for software evolution. Secondly, a service model for collaborative development and evolution of trustworthy software is proposed, which supports: (1) the socialized software development deeply combining the software creation and production; (2) the open and ranked trustworthy software resource sharing; and (3) the evaluation of software trustworthiness based on mass data analysis. Finally, with the background of the highly trustworthy software production tools and integrated environment (Trustie), this paper expounds the key technologies, development environments and application practices for the proposed Crowd-based Method.

Keywords trustworthy software, crowd collaboration, resource sharing, runtime monitoring, trustworthiness analysis, software evolution

WANG HuaiMin was born in 1962. He received the Ph.D. degree in computer science from the National University of Defense Technology, Changsha, in 1992. Currently, he is a professor at National University of Defense Technology. His research interest includes distributed computing, internet computing and software engineering. He is a CCF Fellow.

YIN Gang was born in 1975. He received the Ph.D. degree in computer science from the National University of Defense Technology, Changsha, in 2006. Currently, he is an associate professor at the National University of Defense Technology. His research interest includes distributed computing, software engineering and data mining. He is a member of CCF.

XIE Bing was born in 1970. He received the Ph.D. degree in computer science from the National University of defense Technology, Changsha, in 1998. Currently, he is a professor at Peking University. His research interest includes software engineering and formal methods. He is a senior member of CCF.

LIU XuDong was born in 1965. He received the Ph.D. degree in computer software from the Beihang University, Beijing, in 2007. Currently, he is a senior researcher at School of Computer Science and Engineering of Beihang University. His research interest includes network software development methodology, trusted software technology, middleware technology, and information technology standards.