文章编号:1671-4598(2017)05-0136-04 DOI:10.16526/j.cnki.11—4762/tp.2017.05.038 中图分类号:TP311.56 文献标识码:A

基于持续集成的软件度量

姜 文,刘立康
(西安电子科技大学 通信工程学院,西安 710071)

摘要:软件度量是针对软件开发项目、过程及产品进行数据定义、收集和分析的持续度量化过程;持续集成工具上的的构建工程每天自动完成从版本库更新代码、静态检查、编译、出包、自动化用例测试等任务,在进行集成构建的过程中可以为软件度量提供多种相关的度量数据;结合工作实践,叙述了基于持续集成的软件度量的原理;软件度量管理涉及到的角色;软件度量实现过程;叙述了基于持续集成的两种类型的软件度量指标的定义以及提取方法;最后详细叙述了在软件度量过程中遇到的几个典型案例;工作实践表明在软件的开发过程中做好软件度量工作有助于软件开发部门控制、预测、和改进软件产品的质量与软件开发过程;从而提高软件质量和软件开发效率,降低软件开发成本。

关键词:软件度量;持续集成;静态检查;软件版本;集成构建

Software Metrics Based on Continuous Integration

Jiang Wen, Liu Likang
(School of Telecommunication Engineering, Xidian University, Xi'an 710071, China)

Abstract: The software metrics involved software development project, process and prodution data definition, collection, and analysis, it is a continuous quantitative process. Tasks as updating the code from the repository, static checking, compile, package, test automation cases are automatically running by building project in continuous integration tool, during the process of building a variety of related metrics data for software metrics are provided. In combination with working practice, the principle of software metrics based on continuous integration are described; the roles involved in the management of the software metrics; the process of software metrics implemented. Described two types definition of software metrics and extraction methods based on the continuous integration. At last, several typical cases encountered in the process of software metrics is described in detail. Practice shows that in the software development process, to do a good job of software metrics helps to control, prediction, and improve the quality of the software product and software development process in software development department. Improves the efficiency of software development and software quality, reduces the cost of software development.

Keywords: software metrics; continuous integration; static checking; software version; integration building

0 引言

软件度量(software metrics)是对软件在开发过程中进行的数据持续性采集、分析量化的过程。在软件版本进行开发的过程中进行软件度量有助于尽早发现软件存在的问题,进行软件缺陷预测,从而保证软件产品的质量。

随着软件行业开发技术不断发展,软件持续集成与配置管理技术也不断深入发展,已经成为软件开发过程中非常重要的组成部分。通过运行持续集成的构建工程可以比较方便的来获取某些软件度量数据。将这些质量数据进行加工、分析之后汇总到由公司质量部门的网站上,方便产品的质量工程师与产品经理随时查看本产品的各项度量数据,从而提高产品软件源代码的质量和安全性。

本文以持续集成工具 ICP—CI 为例,叙述基于持续集成的软件度量的原理;软件度量管理和涉及到的角色;基于持续集成的软件度量实现过程;叙述了基于持续集成的两种类型的软件度量指标的定义以及提取方法;最后详细叙述了软件度量过程中的几个典型案例。

收稿日期:2016-12-03; **修回日期**:2017-01-05。

基金项目:国家部委基础科研计划:国防预研基金项目(A1120110007)。

作者简介:姜 文(1986-),女,陕西西安人,工程师,硕士研究生,CCF 会员(E200032324M),主要从事图像处理与分析,数据库应用和软件工程方向的研究。

1 基于持续集成的软件度量原理

持续集成工具搭建的构建工程可以每天通过制定定时任务来自动完成从版本库更新代码、静态检查、编译、出包、自动化用例测试等任务。在进行集成构建的过程中 ICP—CI 的主控服务器的 MySQL 数据库会将许多数据记录在数据库中,主要包括在 ICP—CI 上进行集成构建成功与失败的结果、静态检查的结果、编译的结果;以及静态检查步骤、编译步骤、出包步骤、自动化测试步骤的时间;自动化测试用例数目、成功的用例数、运行阻塞的用例数、运行失败的用例数;失败的步骤所在的位置以及各类代码质量检查工具的运行结果等。在集成构建的过程中可能发生大量的各种问题,这就为软件度量数据采集提供了大量的数据资料。

1.1 软件度量类型

1.1.1 软件源代码质量度量

软件源代码在静态测试和编译过程中出现问题的各种数据,可以用来度量软件代码质量,检查并反馈软件源代码的各种缺陷。

1.1.2 软件版本包的质量度量

集成构建中的出包步骤充分反映了各个软件模块在集成过程中的问题,出包的相关数据可以作为重要的模块集成度量指标。

1.1.3 功能点测试的度量

通过调用自动化测试用例包,可以初步测试大量的软件功

能点，这些数据可以提取有关功能点度量数据，出包的相关数据也反映了功能点开发的进展情况，可以作为软件功能开发进度的度量指标。

1.2　软件度量的数据采集

在 ICP－CI 工具的页面上搭建度量数据工程，将其他构建工程在集成构建过程中的各阶段所需要的时间以及构建结果数据分别记录到 ICP－CI 工具自带的 MySQL 数据库中。持续集成工程师可以登录 MySQL 数据库查看对已经完成构建工程的各种数据。

集成构建过程中采集度量数据需要根据构建类型标志位筛选出定时完整构建的数据，将这些数据上报到质量部门的汇总数据库中。产品版本持续集成工程执行定时完整构建，需要在“管理”页面上选择定时构建的具体时间，构建任务，构建类型一定要选择完整构建。完整构建是指包含构建工程的所有步骤，对于持续集成工程，包括代码更新、基本静态检查、编译、出包与自动化用例测试；对于源代码静态测试，包括代码更新与所有静态检查步骤。

通过持续集成可以大大提高软件度量数据的采集效率。获得软件开发过程中每一天的软件度量数据，通过针对这些度量数据进行质量监控，可以尽早发现软件的各种缺陷。

2　基于持续集成的软件度量管理和涉及到的角色

在基于持续集成的软件度量活动中，涉及到的角色有产品经理、公司质量部、软件开发组、软件测试组、持续集成工程师、产品质量工程师。软件度量管理如图 1 所示。

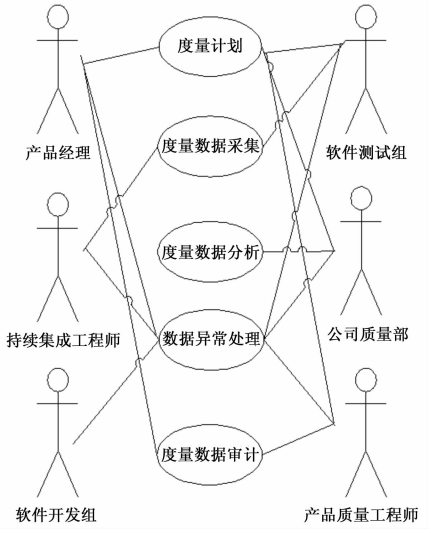

图 1　软件度量管理用例图

2.1　产品经理

制定软件产品度量计划，确定度量指标中的统计承诺值。度量计划提交给公司质量部门审查。关注度量数据的变化，指导和参与度量数据异常处理。

2.2　公司质量部

负责搭建与维护公司的质量网站，审查软件项目组的度量计划，对度量数据进行统计分析，指导项目组处理各种度量数据异常问题。

2.3　软件开发组

在产品的整个度量周期中关注开发组负责的软件模块的相关度量数据，当出现度量数据异常时，参与分析异常的度量数据并实施改进。

2.4　软件测试组

在产品启动度量采集之前，进行相关的自动化环境搭建，并配合持续集成工程提供相关的测试用例脚本。参与度量数据异常的分析和处理。

2.5　持续集成工程师

在产品启动度量采集之前，将软件产品版本信息提供给公司质量部门接口人，安排该版本在公司质量部进行数据呈现时的导航树配置。

负责持续集成度量数据与源代码质量度量数据的采集，采集的度量数据上报公司质量部；当出现产品度量数据异常时，参与度量数据异常情况的定位和处理；确保从 ICP－CI 主控服务器上采集的度量数据在度量周期内全部达标。

2.6　产品质量工程师

审计产品版本的质量计划，定期针对产品的度量数据进行审计，督促软件项目组及时处理度量数据异常问题，参与不达标度量数据的定位与解决，保证在度量周期内所有的度量数据达标。

3　基于持续集成的软件度量实现过程

3.1　度量数据采集环境的部署

度量采集工具的功能是从产品 ICP－CI 主控服务器的 MySQL 数据库中定时完整构建条件的度量数据，然后将数据上报到公司质量部的数据库中。

主控服务器（Master）安装 ICP－CI－Windows－Master 之后，为了采集度量数据需要将公司自研的度量采集工具部署到主控服务器上，具体部署过程如下：

1）登入 ICP－CI 的主控服务器，关闭 ICP－CI 相关与 MySQL 服务的相关进程；

2）度量数据采集工具解压之后为 ICP－CI.jar，将该文件部署到 Master 的 master/bin 目录下；

3）在启动 ICP－CI 工具的脚本 startup.bat 中加入启动度量数据采集工具的语句：CALL ICP _ CI.jar；

4）启动 MySQL 数据库后，再执行 startup.bat 启动 ICP－CI 工具的网页版页面。

3.2　基于持续集成的软件度量的分类

基于持续集成的软件度量可以分为两种类型：软件版本可用性度量与软件源代码静态测试度量。

3.2.1　软件版本可用性度量

软件版本可用性度量需要在产品迭代开发开始阶段进行软件度量，测试经理安排自动化测试工程师完成产品版本的自动化测试用例脚本的编写与自动化测试环境的搭建。持续集成工程师在 ICP－CI 工具上搭建产品版本的持续集成工程，在完成出包步骤后执行测试组提供的自动化测试用例脚本。

持续集成工程师在 ICP－CI 工具上搭建产品版本的代码质量工程。通过部署检查工具来提取版本可用性度量数据。软件版本可用性度量指标如表 1 所示：

（1）CI 构建成功率：（一个度量周期内 CI 构建成功次数）/（一个度量周期内 CI 构建总次数）；

（2）每日版本可用率：（一个度量周期内 CI 构建成功的天数）/（一个度量周期内进行 CI 构建总天数），工作日有一次

表 1 版本可用性度量指标

指标	达标	预警	过低
CI 构建成功率	＞＝80％	[65％,80％]	＜65％
版本每日可用率	＞＝90％	[80％,90％]	＜80％
构建恢复时长(单位:小时)	＜＝24	不涉及	＞24
有效构建天数比	＞＝80％	不涉及	＜80％

CI 构建成功可以确认该日为 CI 构建成功日；

(3) 构建恢复时长：一次 CI 构建失败到下一次 CI 构建成功的时间间隔；

(4) 有效构建天数比：(一个度量周期内有效 CI 构建成功的天数) / (一个度量周期内工作日数)，有效 CI 构建是指可以作为度量数据的构建。

度量周期通常可以选择七天、十五天、一个月。

软件产品版本可用性度量大多选择以月为单位，在度量周期中的度量指标达不到要求时，就会发邮件知会产品经理、质量工程师与持续集成工程师，需要分析定位产品版本可用性指标数据不达标的原因。通常有两类问题导致的版本可用性度量指标异常：1) 软件产品自身缺陷，2) 持续集成运行环境问题。

3.2.2 软件产品源代码静态测试度量

对于软件产品源代码静态测试数据的度量，由于各产品的情况不同，质量部门无法给出统一的度量指标来判定产品的源代码是否符合公司的标准，需要在产品迭代开发开始阶段，由产品经理根据产品特点，对于某些度量项的度量指标分别给出指标承诺值，并将这些承诺值提交到质量部门进行审核，质量部门审核通过之后，这些度量项的指标承诺值和度量数据一起上报质量部门。

持续集成工程师在 ICP－CI 工具上搭建产品版本的代码质量工程。通过集成各种检查工具来提取软件产品源代码静态测试数据度量。软件源代码静态测试数据的度量指标如表 2 中所示：

表 2 软件源代码静态测试度量指标

指标	达标	预警	过低
代码质量缺陷指数(QDI)	＜承诺值	－	＞承诺值
代码重复度	＜承诺值	－	＞承诺值
圈复杂度(全量代码)	＜承诺值	－	＞承诺值
圈复杂度(新增代码)	＜＝10	不涉及	＞10
PC－Lint/Findbugs	0	不涉及－	＞0
Coverity/Fortify	0	不涉及	＞0
危险函数	0	不涉及	＞0

1) 代码质量缺陷指数 (Qualiti Deficit Index)：是通过质量缺陷模型计算出来的，展示系统归一化和总的质量缺陷指数。QDI 取值的计算方法：

总的 QDI 是设计缺陷×权重的累加值，总的 QDI 和系统的规模大小有关。

归一化 QDI＝1000＊(总的 QDI/有效代码行数)。

通过将 Infusion 工具集成到 ICP－CI 工具中，完成对代码质量缺陷指数 (QDI) 的数据采集任务。

2) 代码重复度：重复的代码行数和总代码行数之比。通过将 Simian 工具集成到 ICP－CI 工具中，完成对代码重复度数据采集任务。

3) 代码圈复杂度 (全量代码)：，需要将 SourceMonitor 工具集成到 ICP－CI 工具中，完成对代码中所有的函数进行圈复杂度数据采集。

4) 代码圈复杂度 (新增代码)：统计新增代码的圈复杂度时，需要将 SourceMonitor 工具集成到 ICP－CI 工具中，完成对当前现有的代码与基线的代码进行比对，筛选出新增的函数，并完成针对新增函数的圈复杂度数据采集。

5) PC－Lint 检查：针对 C/C＋＋语言源代码的经典静态检查工具，可以通过将 PC－Lint 工具集成到 ICP－CI 工具中，完成对软件代码进行 PC－Lint 检查，被检查出来的缺陷可以分为错误、告警以及提示这三个级别。

6) FindBugs 检查：针对 Java 语言源代码的经典静态检查工具，可以通过将 FindBugs 工具集成到 ICP－CI 工具中，完成对软件代码进行 FindBug 检查，被检查出来的缺陷可以分为错误、告警以及提示这三个级别。

7) Coverity/Fortify 检查：重量型的软件源代码缺陷静态测试工具，需要通过插桩编译的方式在检查过程中编译生成中间文件，将中间文件上传到专门的分析平台上分析之后得到缺陷报告，缺陷根据问题级别可以分为高、中、低三个级别。Coverity 与 Fortify 工具可以对 C/C＋＋代码进行静态检查，也可以对 Java 进行静态检查。

8) 危险函数检查：针对 C/C＋＋语言源代码执行的安全测试，需要将源代码中的危险函数全部统计出来，可以通过将 Csec _ Check 工具集成到 ICP－CI 工具中，完成对软件代码进行危险函数统计。

3.3 软件产品版本交付使用后终止度量处理

软件产品版本的迭代开发与测试活动结束后，版本交付使用转为维护版本，产品版本研发阶段结束。产品经理知会公司质量部，该产品版本转为维护版本，该产品版本度量下线。由质量部门将目前处于度量中的产品版本转到度量下线的版本中。这个版本不会在度量页面上呈现新的度量数据，仅能从度量下线版本中查看该版本的历史数据。

4 典型案例

某大型软件公司有一个软、硬件结合的大型软件新项目，软件代码量为 2000 多万行，在该产品开发阶段，产品经理向公司质量部门提供该产品的产品名称与产品版本号，公司质量部门为该产品添加相关的度量页面，打开公司的软件度量数据网站可以看到该产品的信息。在开展软件度量的过程中处理了大量的各种问题，下面介绍几个典型的工作案例。

4.1 自动化测试用例执行失败的案例

持续集成工程师进行版本可用性度量时发现，在执行自动化测试用例脚本时部分用例会失败，排查持续集成环境没发现任何环境问题，联系自动化测试工程师定位发现，由于新的版本包中 lny 进程会发生异常复位的情况导致自动化测试用例脚本执行失败。联系软件开发工程师定位问题，发现是产品代码缺陷导致的，软件开发工程师修改源代码之后合入版本库，重新进行版本可用性度量，取得成功。

当月 20 个工作日，19 天共进行 CI 构建 19 次，其中 CI 构建成功 18 次，有效构建天数 18 天。该月的版本可用性度量数据如表 3 所示。

表 3 版本可用性度量数据实例

CI 构建成功率	版本每日可用率	构建恢复时长	有效构建天数比
94.73％	94.73％	＜24 小时	90％

4.2　构建环境出现异常状况的案例

持续集成工程师度量工程在执行编译时，会出现超时失败的情况，排查持续集成环境之后发现有一个编译环境的 ICP－CI 进程启动异常。重启这个编译环境之后，然后再启动 ICP－CI 相关进程，重新进行版本可用性度量，终于执行成功。

当月 22 个工作日，共进行 CI 构建 27 次，其中 CI 构建成功 22 次，有效构建天数 22 天。该月的版本可用性度量数据如表 4 中所示。

表 4　版本可用性度量数据实例

CI 构建成功率	版本每日可用率	构建恢复时长	有效构建天数
81.48%	100%	＜24 小时	100%

4.3　软件源代码问题导致度量数据异常的案例

进行源代码静态测试度量时，收到版本圈复杂度检查未达标的提示，发现圈复杂度（新增代码）的检查结果数据为 11.3 大于承诺值 10，数据不达标。持续集成工程师定位问题代码的模块之后联系软件开发组组长，最后确认是该模块源代码的函数圈复杂度超标。软件开发工程师修改源代码之后合入版本库，重新进行源代码静态测试数据度量，次日和度量周期内后续的圈复杂度（新增代码）度量数据为 9.8。其他的源代码静态测试度量数据没有出现任何异常。圈复杂度（新增代码）度量数据如表 5 中所示。

表 5　代码圈复杂度(新增代码)度量数据实例

度量指标	承诺值	实际值	改进值
圈复杂度(新增代码)	10	11.3	9.8

5　结束语

长期的工作实践表明基于持续集成的软件度量在软件开发过程中发挥了重要的作用。在集成构建过程中自动完成软件产品的度量数据提取，可以快速地向质量部门提交度量数据。对于采集到的数据质量部门进行相关的分析和处理，可以以表格形式显示到度量网站上，同时也给出相关度量数据的柱状图、折线图和饼图，以非常直观的形式显示度量数据的各种状态和发展趋势。工作实践表明在软件的开发过程中做好软件度量工作有助于软件开发部门控制、预测、和改进软件产品的质量或软件开发过程。软件度量为软件项目的质量管理提供了很好的保证。在很大程度上提高软件产品的质量和开发效率，降低软件开发的的成本。

参考文献：

[1]（英）Norman E. Fenton，（美）Shari Lawrence Pfleeger，杨海燕等译，软件度量（原书第二版）［M］. 北京：机械工业出版社，2004.

[2] 朱少民. 软件质量保证和管理［M］. 北京：清华大学出版社，2007.

[3] 于　波，姜　艳. 软件质量管理实践—软件缺陷预防、清除、管理实用方法［M］. 北京：电子工业出版社，2008.

[4] 赵宏远. 软件度量、过程度量与历史缺陷度量：工作量感知的缺陷预测能力比较［D］. 南京：南京大学，2013.

[5] Tibor Gyimothy，Rudolf Ferenc，Istvan Siket. Empirical validation of object－oriented metrics on open source software for fault prediction［J］. IEEE Transactions on Software Engineering，2005，31（10）：897－910.

[6] 计春雷. 全功能点方法和功能规模度量统一模型的研究与应用［D］. 上海：华东理工大学，2011.

[7] 姜　文，刘立康. 基于 SVN 的应用软件持续集成［J］. 计算机测量与控制，2016，24（3）：109－113.

[8] 姜　文，刘立康. 软件配置管理中的基线问题研究［J］. 计算机技术与发展，2016，26（6）：6－10.

[9] 姜　文，刘立康. 基于持续集成的 PC－Lint 静态检查［J］. 计算机技术与发展，2016，26（11）：31－36.

[10] 姜　文，刘立康. C＋＋ 与 Java 软件重量级静态检查［J］. 计算机技术与发展，2016，26（8）：17－23.

[11] N Fenton，P Krause，M Neil. Software Measurement：Uncertainty and Causal Modeling［J］. IEEE Software，2002，19（4）：116－122.

[12] Taek Lee，Jaechang Nam，Dongyun Han，et al. Developer Micro Interaction Metrics for Software Defect Prediction［J］. IEEE Transactions on Software Engineering，2016，42（11）：1015－1035.

[13] Lakshmi P，Latha Maheswari T. An Effective rank approach to software defect prediction using software metrics［A］. 2016 10th International Conference on Intelligent Systems and Control（ISCO）［C］. 2016：1－5.

[14] Wang Shuo，Yao Xin. Using Class Imbalance Learning for Software Defect Prediction［J］. IEEE Transactions on Reliability，2013，62（2）：434－443.

[15] Yang Xiaoxing，Tang Ke，Yao Xin. A Learning－to－Rank Approach to Software Defect Prediction［J］. IEEE Transactions on Reliability，2015，64（1）：234－246.

（上接第 108 页）

控制与常规的 PID 控制具有更快的调节速度，调节精度更高，提高了温度控制系统的动态品质和稳态精度。

参考文献：

[1] 李　超，高　鹏. 玻璃复合及组件技术［M］. 北京：化学工业出版社，2014.

[2] 唐　辉，张晓春，方瑞萍. 夹层玻璃的制造工艺及其生产和能源效率［J］. 玻璃与搪瓷，2014，42（3）：33－38.

[3] 刘志海. 夹层玻璃的发展现状及趋势［J］. 中国建材，2003，9：64－66.

[4] 首诺公司. PVB 在高级夹层玻璃中的有效加工［J］. 上海建材，2011，3：30－32.

[5] 孟兴蛟. 夹层玻璃关键生产工艺探讨［J］. 建材发展导向，2015，13（4）：42－45GB/T16159－1996.

[6] 孙　健. 基于神经单元 PID 的电阻炉智能温度控制系统［D］. 大连：大连理工大学，2008.

[7] 陶永华. 新型 PID 控制及其应用［M］. 北京：机械工业出版社，2002.

[8] 刘小河，管　萍，刘丽华. 自适应控制理论级应用［M］. 北京：科学出版社，2011.

[9] 刘小河. 非线性系统分析与控制引论［M］. 北京：国防工业出版社，1995.

[10] Andrievsky B，Fradkov A. Implicit model reference adaptive controller based on feedback Kalman－Yakubovich lemma［A］. The Proceedings of the IEEE Conference on Control Applications［C］. Scotland UK，1994. 1171－1174.