基于任务的分布式软件开发中软件集成的研究

Study of software integration in task - based distributed development

朱 东 (科学技术大学研究生院 (北京) 100039)
(西门子(中国)有限公司 北京 100102)
涂国防 (中国科学技术大学研究生院 (北京) 100039)

摘要:本文讨论了在多站点的分布式开发环境下基于任务的软件集成的方法、工作流程及同步方法等,提出了开发域及基于产品特性对软件集成等概念。

关键词:软件开发 软件集成 基于任务的软件开发 基于特性的软件集成 工作流程 软件配置管理 分布式配置管理 开发域 并行版本 冲突 合并 软件构建。

1 前言

软件集成是将所选择的软件部件装配在一起以提供最终软件应该提供的能力或特定的子集的过程。[1] 本文中讨论的是在软件开发过程中对软件变更的集成过程。软件集成的活动是一个长期持续的活动,伴随软件开发的大部分生命周期。在大型的软件开发项目如操作系统级的开发工作中,通常采用由低到高多个级别的集成过程,每个级别的集成有不同的周期、策略及相应的工具[2]。软件集成的方法对开发工作影响非常之大,高效的软件集成活动会极大地加快开发速度,增强软件机构的竞争力。本文针对一个基于任务的分布式开发的软件项目实例,就其软件集成实施的方法及改进等方面进行讨论。

2 项目简介

2.1 起源

某软件项目(以下称 A 项目)采用了基于构件的开发模式。该项目的开发工作分布在全球范围不同地域的多个内部及外部合作的软件机构的开发站点。北京的研发中心建有本地的开发库,负责其中多个构件的开发工作,和全球其他站点共同合作开发。

A 项目的配置管理系统主要使用了 Telelogic 公司的产品配置管理 Synergy(原名 Continuus)及其 DCM (Distributed Configuration Management)模块,该系统是一种基于任务的配置管理系统,适用于基于构件的软件开发模式。

2.2 配置管理结构

A 项目采用了多级的软件集成方案,为讨论方便,在此仅取其中的一部分,如图 1 所示。该部分采用了两层结构的主从式的分布式多站点软件开发:构件库 1~5 位于不同的地域,是进行构件开发的配置管理库,一个或多个构件可以在一个库中进行开发。构件库定期将本地变更上传至中心库 1,中心库 1 中对来自多个构件库及上一级数据库的变更进行集成,进行构建(Build)及测试,根据一定的标准形成基线,将这些基线上传到上一级数据库以进行更高级别的集成,同时将基线分发到各个构件库,以便每个构件库能够得到产品其他部分的最新变更,以此为新的基础进行新一轮的开发更新的过程。在此周期中,中心库起到了中心枢纽的作用,同一级别的构件库间不传递变更。

3 软件集成策略的讨论

3.1 开发域

针对目前越来越多的软件外包项目的开发,我们提出开发域(Development domain)的概念。开发域是指在一定范围的开发组织内部采用一致的配置管理工具、软件过程及组织结构,从而使得开发环境更加高效、灵活,能够更好的适应承接各种软件开发项目的要求。从开发环境的角度看,对于如图 1 的软件开发结构,对域外而言,构件库是不可见的,整个域通过中心

©《计算机系统应用》编辑部 http://www.c-s-a.org.cn

库对外接口。我们对配置管理过程进行改善的目的之一是在开发域的内部加强内聚性，对外减少各个开发域的耦合性。开发域的实际意义在于：大型的软件开发项目，通常采用了多层的软件集成架构，包括了一批内、外部的软件承包机构，要求全部的开发站点采用一致的配置管理架构及配置管理过程很难实现的，也是不必要的。进行承包的软件机构可以采用开发域的方式组织内部的配置管理结构，由一个对外的数据库（如图1的中心库1）作为接口，负责对外部的软件集成/更新，在域内部则可以使用原有的配置管理系统及过程。这样可以在与不同配置管理工具、架构及软件集成策略的软件机构合作开发时，最大限度地保持软件机构内部配置管理架构及过程的独立性。由于商用配置管理工具系统价格昂贵，并且转换配置管理工具及策略需要重新开发新的配置管理过程，对开发人员进行培训，需要花费大量的金钱及时间，因此，当与外部配置管理系统可以通过接口兼容时，通过这种方法可以保护对原有配置管理系统（包括工具及过程）的大量投资。此外，在我们的实践中，在保持开发域对外接口不变的前提下，需要对内部配置管理过程进行改善时，只需开发域内部相关部门对变更计划进行审查及批准，这一点使得变更的手续简化，时间大大缩短，提高了软件机构的效率和灵活性。

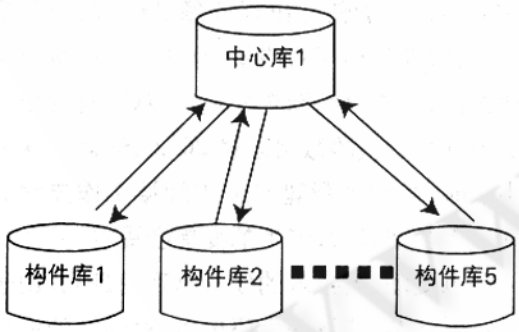

图 1

3.2 基于特性的软件集成

在基于构件的开发中，软件的开发是以构件为单位进行的，但从市场人员的角度看，一个产品是由若干特性组成的，软件的构件并不是市场所关心的。任务也有一定的局限性：

(1) 一个任务所包含的变更是有限的，通常将逻辑上直接相关的一组修改作为一个任务，其修改文件的数目及变更的内容不能太多，以便于问题的分解。因此一个特性的实现往往需要分解成若干任务来实现，开发人员需要人工维护相应的任务集，效率较低且易错。

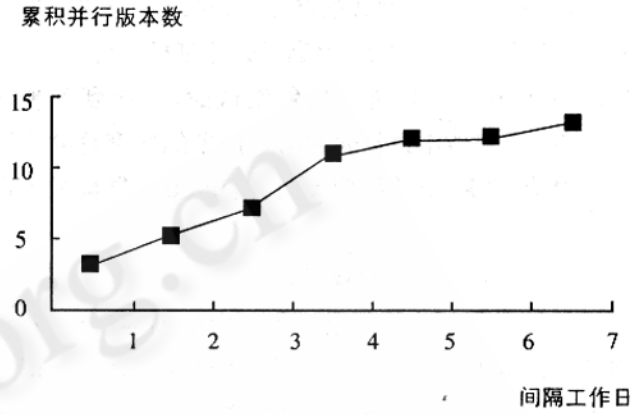

图 2

(2) 产品的一个特性的实现可能跨越多个不同的构件库，在这种情况下，基于任务进行集成时，在一个构件库中无法单独完成对特性的测试，需要来自其他构件库中的任务（称为外部任务）的协助，其操作较繁琐易错。针对这些情况，我们在 CM Synergy 的基础上，提出通过脚本程序以外包工具（Wrapper）的形式，在任务之上实现在配置管理中对特性的支持。特性可以包含一个或多个任务，是为实现某个产品特征的任务集。开发人员可以通过特性对其包含的任务进行管理。并且，通过外包工具可以进一步实现配置管理工具及变更管理工具之间对特性的对接。其优点是：

① 产品的市场人员与软件开发人员可以通过变更管理工具基于特性进行交流，加强了双方的共同语言；

② 在进行软件集成时，以特性为单位对与之相关的所有任务进行操作，效率较高且可避免其他不相干任务的影响；

③ 通过对任务及特性中定义相应的属性，可以在任务的捡入时，由配置管理系统的触发器（Trigger）对该任务进行检查：是否有有效的特性编号（Feature Number），同时检查该特性的任务单（Task List）中是否记录有当前的任务编号（Task Number），只有二者相符的任务才可以捡入，从而可避免无效任务被错误捡入；

④ 特性是产品在设计之初的规格书中就已确定的，因此在确定开发架构时，就可以根据特性对开发域

©《计算机系统应用》编辑部 http://www.c-s-a.org.cn

的构件进行分配，使属于同一特性的任务不会跨越开发域（在特定情况下仍可能跨越构件及构件库），当特性跨越构件库时，易于将相关的特性提升到中心库进行集成，避免了在构件库中对外部任务的依赖；

⑤ 在开发不同款式的产品时，可以将已完成的产品特性复用在新产品上，提高了软件复用的粒度。

3.3 在对并行开发进行软件集成时，会遇到各种冲突，在集成中最普遍并出现最多的冲突是并行版本的冲突，对并行版本合并（merge）是集成人员的主要工作之一

3.3.1 并行版本与软件架构的关系

我们发现完全由北京站点开发的构件项目中，出现的并行版本较少且解决并行版本的花费远比共同开发的构件项目小。原因是：北京的每个开发小组在进行合并时便于沟通，且容易避免不必要的并行版本。结论：在对同一构件的开发工作，应避免分配在不同地理位置的多个站点。

3.3.2 与集成周期的关系

理论上软件集成的间隔越短，每次集成时的冲突会越少。但过多的集成不仅需要付出更多的人力，并且易将不稳定的变更呈现给更大的范围。我们的经验是在如图 1 的结构中，开发人员约 50 人的构件库一级，每周集成两次是比较合适的。理由如下：

（1）软件集成的实质是对软件变更的汇总，集成时间间隔越长，产生的变更越多，集成的复杂度将越大。我们对某构件级项目（开发人员约 10 人）在代码开发的一段时期的的并行版本个数及集成间隔进行了统计，得出的趋势图见图 2 所示。一个典型的案例是在某构件项目中，由于多种原因导致两地的开发工作长期没能集成，最后进行集成时，产生了大量的并行版本及其他冲突，为解决这些冲突所花费的时间及人力资源远超过原来的估计；

（2）较多的集成可以减少下一次集成时的冲突。如在该构件库中，平均每周仅本地捡入的任务就约 90 个，因此，在每周 2 或周 3 进行一次集成并建立基线的过程，事实上减轻了周 5 集成的难度与负荷。

3.4 集成工作流程（workflow）比较

3.4.1 工作流程简介

在软件的集成中，可以采用直接工作流程或间接工作流程。直接工作流程是指对于开发人员捡入的任务不经处理即直接汇入公共的任务文件夹（task folder），使之对所有开发人员可见的过程。间接工作流程与直接工作流程相对，开发人员捡入的任务只对其本人可见。集成人员按照一定的标准，将所有已完成的任务放入特定的任务文件夹，使其仅对集成人员可见，通过检测并解决其可能存在的冲突，进行构建并通过测试后，再将其集成到公共任务文件夹，使之对所有开发人员可见。在间接工作流程中，集成人员负责对已捡入的任务进行甄别，保留“好的”任务，对“坏的”任务进行隔离处理，起到了防火墙的作用。由于集成人员的存在，使得公共工作空间总是保持稳定的状态。

3.4.2 普通单站点项目开发中的工作流程

我们的经验表明，直接工作流程通常只在较小的集成范围内使用。当开发人员的数目小于 10～15 人，且处于同一工作地点时，更适于采用直接工作流程。其特点是：

（1）当开发人员完成某个任务并进行完相应的测试将任务捡入后，同组其他的开发人员马上就可以通过更新的操作得到该任务；

（2）任务被捡入后自动进入公共任务文件夹，不需要设立集成人员的角色；

（3）由于工作流程简单，特别是不需要设立集成人员的角色，因此相应的培训工作较少；

（4）当捡入的任务出现问题的时候，由于项目的规模较小，软件架构相对简单，且在同一开发小组中人员沟通方便，较易对问题查找定位，从而表现出较高的效率。当开发人员的数目超过 20 人，通常分属两个以上的小组时，我们强烈建议采用间接工作流程。原因是当开发人员增多时，产生的变更会更多，同时出现问题的时候也更多，当捡入的任务出现问题后，由于软件结构相对更复杂，涉及的代码更多，对问题的调试难度也会增大；同时，当开发人员分属不同开发小组时，容易导致扯皮，使问题复杂化；且当代码无法编译成功时，会阻碍其他人员捡入新的任务，造成开发工作的停滞。集成人员的工作，可以维持开发环境的稳定，得到更大的好处和更高的效率。

3.4.3 分布式开发中的工作流程

在分布式的软件开发中，软件集成的对象不仅包括构件库内部捡入的本地任务，还包括外部相关站点发送的外部任务，工作流程的作用更加重要。在我们

©《计算机系统应用》编辑部 http://www.c-s-a.org.cn

的实践中，对两种工作流程进行了比较，体会如下：

如图1所示的开发域中，中心库的工作流程的选择非常重要。当中心库和构件库共同选用直接工作流程时，从软件配置管理的角度看，等于整个开发域中的所有开发人员在同一个公共的工作空间工作。在使用直接工作流程时，任务在域中传播的速度，取决于进行任务同步和集成的频率。在我们的项目中，捡入的新任务在24小时内即可对域内的所有用户可见。其好处是变更传递迅速，缺点是当某个任务有问题时，会影响到域内所有的开发人员。在考虑使用直接工作流程时，其适用的范围应包括整个开发域。如图1所示的开发域中，每个开发站点的平均开发人员人数约30人，则开发域中全部开发人员超过150人，在这样的域中全部使用直接工作流程的结果是灾难性的，任何一个捡入的坏的任务会影响到整个开发域。因此，在一个分布式配置管理系统中，在开发域一级使用直接工作流程是不适宜的，因为会导致未经检测的任务汇集，使得无法保持稳定的开发环境。对其改进的一种解决方案是：在中心库中使用间接工作流程，在小型的构件库中使用直接工作流程，这样，既保持了采用直接工作流程时较高的效率，同时通过中心库中集成人员的工作隔绝了"坏的"任务，保证了整个域中公共工作空间的稳定。

3.5 基于任务的配置管理系统在分布式环境中变更的同步

基于文件的配置管理系统在不同的站点对变更同步时，需要对每个站点中所有的文件的版本树原封不动地在其他站点进行复制。而在基于任务的配置管理系统中，由于软件的变更是以任务为单位的，因此，在不同的站点间传递变更时，不需要将全部的文件版本树进行复制，而只需传递和有效任务相关联的文件的相应版本即可。变更的同步分为变更的发送及接受两个方面。在接受站点，集成人员可以根据不同的规则，对不同性质的任务进行集成。这不仅可以大大减少网络流量，也使得集成人员可以有更大的灵活性。如图1的中心站点1每天可以将在本站点中经过测试的任务放在TTF(Tested Task Folder)中，将已捡入但未经测试的任务放在CTF(Completed Task Folder)中发送到构件库1～5中；构件库1中的集成人员接收后，可以将中心库1的TTF及CTF中的所有任务进行集成，也可以只将中心库1的TTF中的任务进行集成。

在传统的分布式开发中，软件变更的同步方式通常可以分为点对点方式或主从式。点对点的同步方式对变更传递速度更快，但对变更的检查、控制较少；主从式中在每一个中心库中对变更进行测试，然后再进一步发送，因而稳定性较好，但变更从一个构件库传递到另一个构件库的有着较长时间的延迟，在多级的软件集成结构中，任务传递的延迟会更加严重。在基于任务的软件集成中，一个构件库在对外部传递的任务进行集成时，我们可以根据任务的不同属性，对任务进行选择分类，然后进行发送。并将软件变更的传递方式与软件集成的方式分离，即使用点对点的方式进行变更的同步，主从式的方式进行软件集成。如在构件库1中，CTF及TTF以广播的方式分发到中心库1及构件库2～5中。这样，在所有的数据库中，构件库1分发的任务都已经存在，只不过未经集成因而对开发人员不可见。此时，如开发人员急需其中的某个任务时，可以手动将其拷贝到自己的私有任务文件夹中，通过更新得到需要的任务。而构件库的正常集成只包括中心库1的TTF但不包括其他构件库的任务文件夹。这样既可以迅速得到其他构件库完成并测试过的任务，同时又可以保持公共空间的稳定，兼得点对点结构和主从式结构二者的优点。

4 总结

基于任务的分布式的软件集成是目前比较先进的技术，适用于基于构件的软件开发模式，在今后可望替代基于文件的软件集成。它给软件集成人员提供了更大的灵活性，使得软件集成人员可以根据项目的实际情况定制相应的集成方案，以得到最佳效率。到目前为止，有关的集成方法和策略还处于早期的阶段，不够成熟，相信随着更多的实践和研究，会使之不断完善成为软件开发的有力工具。

参考文献

1 SEI.《能力成熟度模型(CMM)：软件过程改进指南》电子工业出版社－2001年。

2 Don Bollinger, Steve Varnau. HP－UX software integration. http://www.interex.org/pubcontent/enterprise/jan99/08varn/08varn.html.

©《计算机系统应用》编辑部 http://www.c-s-a.org.cn