# 一种基于Internet的分布式软件生产线框架

王晓斌 郭长国 王怀民
(国防科技大学计算机学院 长沙 410073)

**摘 要** 提出了一种Internet环境下的分布式软件生产线框架DSPLF(Distributed Software Production Line Framework)。该框架详细描述了分布式软件生产线的体系结构,阐述了分布式软件生产线体系结构中各组成模块及其相互关系,在此基础上给出了基于该框架创建、运行分布式软件生产线的方法和详细流程,并实现了一条基于该框架的监控软件生产线。

**关键词** Internet,分布式软件生产线,框架
**中图法分类号** TP311 **文献标识码** A

## Internet-based Distributed Software Production Line Framework

WANG Xiao-bin GUO Chang-guo WANG Huai-min
(Department of Computer Science, National University of Defense Technology, Changsha 410073, China)

**Abstract** This paper presented DSPLF (Distributed Software Production Line Framework), a framework for developing distributed software production lines on Internet. DSPLF proposed an architecture for distributed software production line, including the composing components and their relationships, based on this architecture, methods and detail processes for constructing and running distributed software production lines were given, also with a distributed software production line project for monitoring, which was developed based on this framework.

**Keywords** Internet, Distributed software production Line, Framework

## 1 概述

随着软件规模不断扩大,软件复杂度越来越高,如何有效地保证软件质量以及提高软件开发效率,成为软件界普遍关注的重点。通过工程化来实现软件大规模生产是人们为解决这一问题而提出的重要思想,基于软件产品线和软件生产线进行软件复用是其中的重要方法。随着Internet的发展,软件开发出现了分布式、协同化的趋势。在Internet环境下,软件开发者摆脱了地域限制,可以随时随地进行交流和协同。如何将软件复用、软件产品线和软件生产线的思想和方法运用于Internet环境下的软件开发,对实现分布式、协同化的大规模软件生产具有重要的意义。为此,本文提出了一种Internet环境下的分布式软件生产线框架,该框架描述了如何复用Internet上的共享软件工具,如何将它们组建成分布式软件生产线以及如何在组建的分布式软件生产线上进行分布式、协同化的软件生产。

## 2 相关研究工作

### 2.1 软件复用

软件复用是指重复使用"为了复用目的而设计的软件"的过程[1]。软件复用旨在软件开发过程中充分利用已有的开发成果,避免重复性工作,从而提高软件开发效率。同时,复用高质量的开发成果能够保证软件的质量,避免重新开发可能带来的错误[2]。软件复用的思想最初来源于子程序,虽然当时的出发点是为了节约昂贵的内存资源,但众多子程序库的开发确实节约了不少的人力资源。面向对象技术更加推动了软件复用的实践。目前,基于构件[3,4]的软件复用得到了成功应用,大量具有一定标准的软件构件的开发使软件开发逐渐走上了工业化、产业化的道路。

### 2.2 软件产品线和软件生产线

近年来,人们逐渐认识到,要提高软件开发效率,提高软件产品质量,必须改变手工作坊式的开发方法,采取工程化的开发方法和工业化的生产技术[5]。借鉴工业界的生产线概念,人们提出了软件产品线和软件生产线的方法。

软件产品线首先由卡内基梅隆大学软件工程研究院(CMU/SEI)提出,它的定义如下:软件产品线(Software Product Line)是多个软件密集系统组成的集合,这些系统共享一个公共的、可管理的特征集,这个特征集能满足选定的市场或任务领域的特定需求。这些系统遵循一个预描述的方式,在公共的核心资产基础上开发[6]。软件产品线的核心资产包括代码、算法、子程序、模块、架构、各种开发文档等,通过复用核心资产可以快速组装出新的软件产品,以适应市场对该类产品需求的变化。软件产品线强调对核心资产和产品组装的严格管理,只有成熟的管理机制才能保证产品开发的效

到稿日期:2009-05-14 返修日期:2009-08-20 本文受863国家重点基金项目(2007AA010301)资助。
王晓斌(1985-),男,硕士生,主要研究方向为分布式计算等,E-mail:wxb1107@hotmail.com;郭长国(1973-),男,副研究员,CCF会员,主要研究方向为分布式计算、可信计算等;王怀民(1962-),男,教授,CCF会员,主要研究方向为分布式计算、可信计算等。

率和产品的质量。软件产品线发展到现在，取得了很多成功的应用，典型的代表是诺基亚手机操作系统产品线[6]。

基于软件产品线，人们又提出了软件生产线(Software Production Line)的概念。国内早在“七五”期间就由北大提出了“青鸟”软件生产线[7]。北大青鸟软件生产线由3部分构成：构件生产、构件库管理和构件复用，强调以构件为基础，以复用为手段，采取工程化的开发方法和工业化的生产技术，取得了许多成功的应用，在国内软件行业具有较大的影响力。

继北大青鸟软件生产线之后，国家863计划重点课题“可信的国家软件资源共享与协同生产环境”聚焦协同环境下的软件生产，并给出了软件生产线的定义：软件生产线是按照一定的软件生产方法，将若干软件生产工具和构件有序组织起来的软件开发环境，生产线能够较为完整地提供成套的软件开发支撑。

结合上述研究，我们提出了一种Internet环境下的分布式软件生产线框架，本框架描述了分布式软件生产线的体系结构，给出了基于本框架创建、运行分布式软件生产线的方法和详细流程。本框架是在Internet环境下进行分布式、协同化软件生产的一种有效实现形式。

## 3 分布式软件生产线体系结构

分布式软件生产线框架的体系结构如图1所示，该体系结构分为3层：

①工具库层。工具库是可重用的软件生产工具的存储库。

②服务层。服务层是软件生产过程中进行协同和管理所需的服务集合。

③集成框架层。集成框架是生产线工具的运行平台以及工具之间协同的代理。

这3层构成了一个C/S风格的软件生产环境，其中第1层和第2层是Server层，第3层是Client层。操作生产线的用户有两类，一类是生产线的创建者，一类是生产线上的工作者。

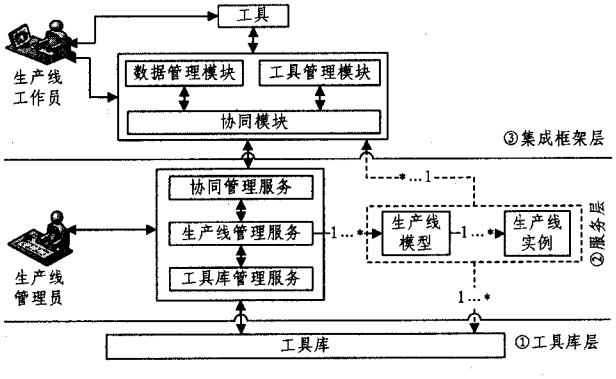


图1 分布式软件生产线框架体系结构

我们用三元组 $F=\langle TD,S,IF\rangle$ 来表示整个框架。其中，$TD=\{t_1,t_2,\cdots,t_n\}$ 是工具库中所有工具的集合。$S=\{S_{tdm},S_{plm},S_c\}$ 是服务的集合，$S_{tdm}$ 是工具库管理服务，$S_{plm}$ 是生产线管理服务，$S_c$ 是协同服务。$IF=\{f_1,f_2,\cdots,f_n\}$ 是分布在Internet上的集成框架的集合。

### 3.1 生产线工具

在分布式软件生产线框架中，生产线工具是指那些可被多种软件生产线重用，用来组建不同功能软件生产线的共享的可重用软件工具。就生产线工具的工作形式而言，生产线工具接收一定的输入数据，在生产线工作员的操作下手动或自动实现对输入数据的处理，产生相应的输出数据，可以表示为 $t=\langle I,O,P_t\rangle$，$I\wedge P_t\rightarrow O$。其中 $I=\{i_1,i_2,\cdots,i_n\}$ 为输入数据集合，$n\geqslant 0$。$O=\{o_1,o_2,\cdots,o_m\}$ 为输出数据集合，$m\geqslant 1$。$P_t$ 是工具 $t$ 对输入数据 $I$ 的一系列处理操作，其中可能包括生产线工作员的人机交互操作。输入不同的数据或对相同数据采取不同的处理都会产生不同的输出数据。

在分布式软件生产线中，为了实现生产线的整体功能，工具之间在逻辑上需要交换输入或者输出数据，然而工具本身只实现了特定的业务功能，根据输入产生相应的输出。工具之间的协同还需要集成框架和协同服务的支持，如图2所示。

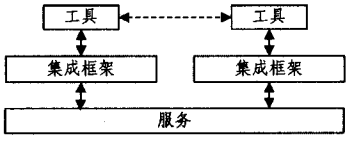


图2 工具协同示例

### 3.2 软件生产线

软件生产线由工具按照一定的流程组建而成。根据生产线的需要，不同功能的工具位于流程中的不同位置，并在生产线的统一管理下完成各自的工作，生产线通过管理、协同工具之间的工作来实现整体功能。

以元组 $M=\langle T,R\rangle$ 来表示软件生产线，$T=\{t_1,t_2,\cdots,t_n\}$ 是组成该生产线的工具集合，$t_1,t_2,\cdots,t_n\in TD$；$R=\{r_1,r_2,\cdots,r_n\}$ 表示生产线工具之间的交互关系，其中 $r_i=\langle F_i,B_i\rangle$，$F_i,B_i\subset T$，$F_i=\{t_{i1},t_{i2},\cdots,t_{im}\}$ 表示工具 $t$ 的前驱工具集合，$B_i=\{t_{i1},t_{i2},\cdots,t_{in}\}$ 表示工具 $t$ 的后继工具集合。

图3示例了一个简单生产线，其中 $T=\{t_1,t_2,t_3,t_4,t_5\}$；$R=\{r_1,r_2,r_3,r_4,r_5\}$，$r_1=\langle\emptyset,\{t_2,t_4\}\rangle$，$r_2=\langle\{t_1\},\{t_3\}\rangle$，$r_3=\langle\{t_2\},\{t_5\}\rangle$，$r_4=\langle\{t_1\},\{t_5\}\rangle$，$r_5=\langle\{t_3,t_4\},\emptyset\rangle$。

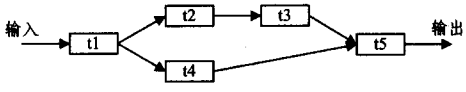


图3 软件生产线示例

### 3.3 软件生产线划分

根据工具的功能和生产线工作员的特点，可以将一个或多个生产线工具部署在同一个集成框架下运行，为此需要对生产线进行划分。用 $P_m$ 表示对生产线 $M$ 的划分，$P_m=\{P_1,P_2,\cdots,P_n\}$，$P_i\subset T$，$P_i\cap P_j=\emptyset$，$P_1\cup P_2\cup\cdots\cup P_n=T$，$1\leqslant i<j\leqslant n$。

图4是对应于图3的生产线划分，$P_m=\{P_1,P_2,P_3,P_4\}$，$P_1=\{t_1\}$，$P_2=\{t_2,t_3\}$，$P_3=\{t_4\}$，$P_4=\{t_5\}$。

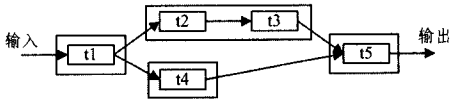


图4 软件生产线划分示例

### 3.4 生产线模型和生产线实例

生产线模型是对软件生产线的一个抽象表示，形成于生产线的创建阶段，反映了生产线的配置信息，包括组成生产线的工具、生产线的流程以及生产线的划分等。

生产线实例在生产线运行阶段被创建。对于不同的输入

数据，生产线创建不同的生产线实例来处理，拥有独立的工作空间来保存该实例运行过程中的中间数据和运行状态信息。

### 3.5 生产线服务

服务层为分布式软件生产线的创建和管理提供支持。生产线服务包括生产线管理服务 $S_{plm}$、工具库管理服务 $S_{tdm}$ 和协同服务 $S_c$。

(1)生产线管理服务是生产线的管理、控制中心，生产线管理员通过该服务来创建、配置、启动生产线以及查询生产线的状态信息。

(2)工具库管理服务是工具库的管理中心，包括工具的上传、下载以及多种方式的检索。

(3)协同服务是生产线协同运行的基础，包括与集成框架之间的消息和数据传递，协同服务辅助生产线管理服务来实现对生产线运行的协同管理。

### 3.6 生产线集成框架

集成框架是工具的容器，管理工具的运行和运行时的协同。集成框架主要由工具管理模块、数据管理模块和协同模块 3 部分构成。

(1)工具管理模块。对生产线工具进行管理，包括工具的下载、安装以及工具的生命周期管理。

(2)数据管理模块。负责对生产线工具的输入输出数据进行管理，包括将输入数据传递给工具以及从工具接收相应的输出数据。

(3)协同模块。实现了集成框架的协同功能，包括与服务器之间的交互和相关消息、数据的传递。

相对于服务的集中部署，集成框架分布在 Internet 上，在服务器的统一协同管理下，协助生产线工作员操作生产线工具来完成相应的任务。

## 4 分布式软件生产线的创建和运行流程

在上述分布式软件生产线框架的基础上，本节给出基于该框架创建、运行分布式软件生产线的方法和详细流程。

图 5 为软件生产线的工作流程图，分为创建生产线和运行生产线两部分。其中，创建生产线包括第 1,2,3,4,5 步，运行生产线包括第 6,7,8,9 步。此外，步骤 1,2,3,6,7 主要在 Server 端完成，步骤 4,5,8,9 主要在 Client 端完成。

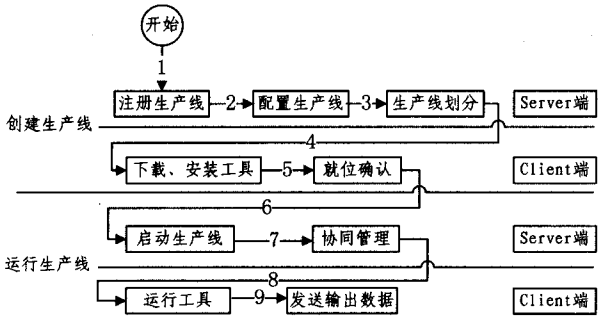


图 5 软件生产线流程图

### 4.1 创建流程

生产线的创建包括注册生产线、配置生产线、生产线划分、下载、安装工具和就位确认 5 个步骤，其中前 3 步由生产线管理员调用相应的服务来完成，后两步由生产线工作员通过集成框架完成。

(1)注册生产线。生产线管理员向服务器申请注册一个新的生产线项目，系统为该生产线项目分配一个独立的工作空间。

(2)配置生产线。生产线管理员调用相关服务选取生产线项目所需的工具，根据工具的不同功能制定生产线流程，明确工具之间的交互关系。系统自动将上述生产线的配置信息保存到相应的生产线模型。

(3)生产线划分。生产线管理员对生产线进行划分，为每一个划分指派一名生产线工作员，并将生产线划分信息保存到生产线模型。

(4)下载、安装工具。生产线工作员通过集成框架与服务器进行交互，查找自己所属的生产线项目和对应的划分，然后将属于该划分的生产线工具从服务器下载到本地，并安装到集成框架，使工具能够在集成框架下正常工作。

(5)就位确认。生产线工作员向服务器进行就位确认，表明自己对应的生产线划分准备就绪。服务器接收到就位确认信息后，更新生产线模型中相应划分的状态信息。

### 4.2 运行流程

分布式软件生产线创建完成后，即可运行。对不同的初始输入数据，系统创建不同的生产线实例运行。生产线的运行采用数据驱动的方式，并基于数据进行协同。运行过程中，生产线自动检测每个工具所需的输入数据信息，一旦工具的输入数据在生产线实例中准备就绪，生产线就将数据发送到该工具。该工具经过一定的处理后将产生的输出数据传回生产线实例，启动下一个工具。当生产线流程中最后一个工具产生输出数据后，该生产线实例运行结束。一条生产线上可以并行运行不同的生产线实例。具体的实现步骤如下。

(6)启动生产线。当所有的生产线划分就位确认后，生产线即开始工作。管理员首先将初始数据(即第一个工具所需的输入数据)传递给生产线，系统为该数据新建一个生产线实例，用以保存该生产线实例运行过程中的中间数据和运行状态信息。

(7)协同管理。经过步骤 6，生产线检测到第一个工具的输入数据准备就绪，随即通知工具所在的集成框架接收该数据，并启动工具开始工作。

(8)运行工具。集成框架接收到输入数据后，将该数据传递给工具并启动工具进行处理，直到产生相应的输出数据。

(9)发送输出数据。工具产生相应的输出数据后，由集成框架发送到对应的生产线实例，该输出数据用于后继工具的输入数据。

生产线实例接收到工具的输出数据后，判断是否为最后一个工具的输出。如果是，则该生产线实例运行结束；否则，重复步骤 7,8,9，将数据发送到后继工具继续运行。

## 5 分布式软件生产线的实现与应用

在以上研究的基础上，我们开发了一条分布式的监控软件生产线。服务层以 OSGi[8] 服务来实现，集成框架和生产线工具以 Eclipse 插件[9] 来实现。监控软件生产线的功能由 4 个生产线工具来完成：源代码分析工具、监控需求建模工具、监控探针生成工具和监控探针注入工具。监控软件生产线运用面向方面编程技术(Aspect-Oriented Programming，AOP)[10,11]，通过将监控代码作为方面自动注入到软件系统，

(下转第 131 页)

〈/log〉

日志中包含了下步操作节点信息，但系统实际并不依据这个信息来调用下步操作节点，这个信息更多的作用是作为验证，因为系统是根据合成服务的定义生成的合成 Web 服务操作。从合成服务执行的角度看，合成 Web 服务操作是合成服务执行的各个阶段的异步消息入口点。操作本身包含了在合成服务执行中的位置信息，系统能够根据被调用的 Web 服务操作确定合成服务内部的下一步动作。

### 5.2 基于执行文件的异常处理器和事务适配器

系统中提供了执行异常处理机制，依靠执行阶段的日志进行异常处理。当执行出现异常时，首先由执行监控器写入执行日志，异常处理器解析日志中 XML 文档片段〈exception〉〈/exception〉的部分进行处理。在一个 Web 服务环境下对事务的参与者没有公共的事务语义、事务上下文表达和协调协议。在一个合成服务中，参与者来自不同的服务提供者，可能来自异构的平台，还有各自的商业规则，而且是完全自治的独立实体。传统的事务的 ACID 特性不能完全适用于 Web 服务，传统的事务模型不能完全适用于 Web 服务合成。分布的 Web 服务的事务管理要求 Web 服务的事务支持和协调。引入事务适配器[10]的目的是为了把合成逻辑和事务补偿逻辑相分离，把流程中的事务要求封装在事务适配器中。本系统中事务适配器依靠执行日志中的 XML 文档片段〈transaction〉〈/transaction〉的部分，根据流程中定义执行相应事务处理，进行补偿或恢复操作。

**结束语** 本文基于 Web 服务合成生命周期的定义、合成、执行 3 个阶段，提出了较为完善、有效的 Web 服务合成系统架构，综合考虑了 Web 服务合成的各个方面以及各个阶段所涉及到的问题，如合成定义语言、动态服务绑定、服务的异构解决、流程执行监控、异常处理、事务处理等，为 Web 服务合成的进一步研究提供了较好的参考架构。

## 参 考 文 献


[1] Casati F, Krishnamoorthy S I. Adaptive and Dynamic Service Composition in eFlow[EB/OL]. Software Technology Laboratory HP Laboratories

[2] Sheng Q Z, Benatallah B, Dumas M. SELF-SERV: A Platform for Rapid Composition of Web Services in a Peer-to-Peer Environment [EB/OL]. University of New South Wales Marlon Dumas Queensland University of Technology

[3] Pires P F, Benevides R F M, Mattoso M. WebTransact: A Framework for Specifying and Coordinating Reliable Web Service Compositions[EB/OL]. http://www.cos.ufrj.br/~pires/WebTransact.html. 2002

[4] Thatte S. XLANG: Web Services for Business Process Design [EB/OL]. http://www.gotdot.net.com/team/xml_wsspecs/xlang-c/default.htm

[5] Leymann F. WSFL: Web Service Flow Language 1.0[EB/OL]. http://www-3.ibm.com/software/solutions/Webservices/pdf/WSFL.pdf

[6] Arkin A. Business Process Modeling Language [EB/OL]. http://www.bpmi.org/BPML

[7] Curbera F, Goland Y, Klei J, et al. BPEL4WS: Business Process Execution Language for Web Services Version1.0 [EB/OL]. http://www-900.ibm.com/developerEorks/cn/Webservices/ws-bpel_spec/index.shtml

[8] 杨丹，申德荣，等. 一种支持 Web 服务合成的模型定义语言-e_SPDL[J]. 计算机集成制造系统-CIMS，2003，9(10)：932-936

[9] 张蓉，申德荣，等. 基于本体的 Web 服务查找和合成技术研究[J]. 计算机集成制造系统-CIMS，2003，9(10)：921-925

[10] Schafer M, Nejdl W. An Environment for Flexible Advanced Compensation of Web Service Transactions[J]. ACM Transactions on the Web，2008，2(2)


(上接第 127 页)

将原本不具备监控能力的软件改造为可被监控的软件系统，使得运行时可以监视软件的内部运行状态，从而提高其可靠性和安全性。

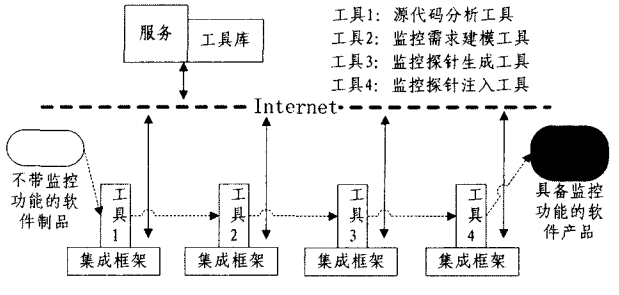


图 6 分布式监控软件生产线示意图

图 6 为分布式监控软件生产线的示意图，该生产线的功能由分布在 Internet 上的 4 个工具来实现，并由服务器进行统一的协同管理，有效地实现了分布式、协同化的监控软件生产。

**结束语** 本文提出了一种 Internet 环境下的分布式软件生产线框架，详细阐述了该框架的体系结构，给出了基于此框架创建、运行分布式软件生产线的方法和详细流程，并介绍了基于此框架开发的一个分布式监控软件生产线。实践证明，本框架能够有效支持 Internet 环境下分布式、协同化的软件生产，具有一定的理论及应用价值。

## 参 考 文 献


[1] Tracz W. Confessions of a Used Program Salesman-Institutionalizing Software Reuse[M]. New York, NY: Addison-Wesley Publishing Co., April 1995

[2] Yang F Q, Mei H, Li K Q. Software reuse and software component technology[J]. Acta Electronica Sinica, 1999, 27(2): 68-75

[3] Szyperski C. Component Software: Beyond Object-Oriented Programming(2nd ed)[M]. Boston: Addison-Wesley Professional, 2002

[4] Veryard R. Component-based business, plug and play[M]. London: Springer, 2001

[5] Yang F Q. Thinking on the development of software engineering technology[J]. Journal of Software, 2005, 16(1): 1-7

[6] Northrop L M, Clements P C, Bachmann F, et al. A Framework for Software Product Line Practice[EB/OL]. Version 5.0. 2007. http://www.sei.cmu.edu/productlines/framework.html

[7] Yang F Q, Mei H, Li K Q, et al. The summary of JB III supporting components reuse[J]. Computer Science, 1999, 26(5): 50-55

[8] http://www.osgi.org/About/Technology

[9] http://www.eclipse.org

[10] Kiczales G, Lamping J, Mendhekar A, et al. Aspect-Oriented Programming[C]// Proceedings of the European Conference on Object-Oriented Programming. 1997, 1241: 220-242

[11] A look at aspect-oriented programming[OL]. http://www.ibm.com/developerworks/rational/library/2782.html