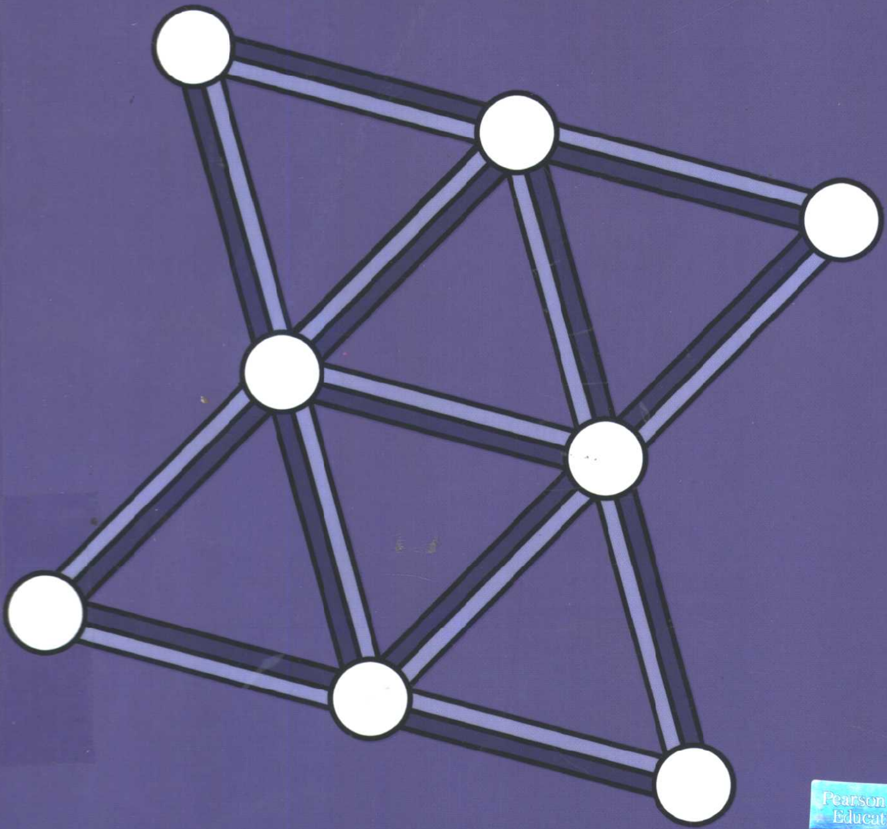
SEI软件工程丛书 · 影印版
卡内基 · 梅隆大学软件工程研究所推荐 · 软件学院教材

PEARSON
Addison Wesley

基于构架的软件项目管理

Architecture-Centric Software Project Management:
A Practical Guide

[美] 丹尼尔 · J · 鲍里斯 [Daniel J. Paulish] 著



Pearson Education

清华大学出版社

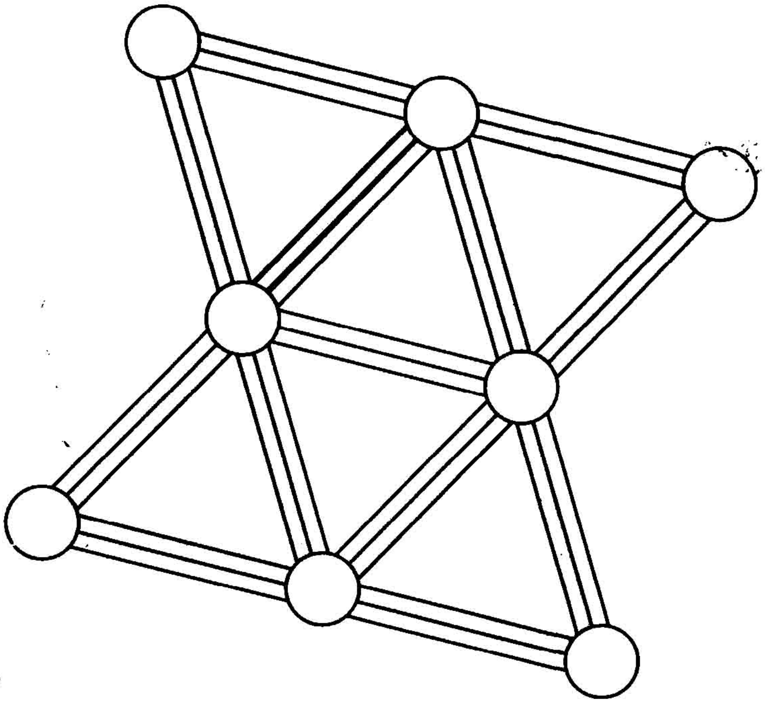
SEI 软件工程丛书 · 影印版
卡内基 · 梅隆大学软件工程研究所推荐 · 软件学院教材

PEARSON
Addison Wesley

基于构架的软件项目管理

Architecture-Centric Software Project Management: A Practical Guide

[美] 丹尼尔 · J · 鲍里斯 [Daniel J. Paulish] 著



清华大学出版社
北京

内 容 简 介

基于构架的软件项目设计（ACSPP）是一种重要的设计软件项目的软件开发方法，该方法可以准时、按预算、高效地完成任务。本书说明了如何利用软件构架来设计进度表、生成评估表、决定作用域和管理开发团队。本书涉及高效软件管理的基础——计划、管理、实现和度量。作者给出了大量经过实践检验的宝贵例子，还以实际的案例来说明本书的策略、方法和技术。

本书可作为软件学院及大学计算机等专业相关课程的教材，也可以作为软件公司各级管理和开发人员参考。

图书在版编目（CIP）数据
基于构架的软件项目管理 Architecture-Centric Software Project Management / （美）鲍里斯著. — 影印版.—北京：清华大学出版社，2003
（SEI 软件工程丛书）
ISBN 7-302-06350-8
Ⅰ.基... Ⅱ. 鲍... Ⅲ. 软件开发—项目管理—英文 Ⅳ. TP311.52
中国版本图书馆 CIP 数据核字（2003）第 010095 号

出 版 者：清华大学出版社（北京清华大学学研大厦，邮编 100084）
http://www.tup.com.cn
http://www.tup.tsinghua.edu.cn
责任编辑：尤晓东
印 刷 者：北京广益印刷有限公司
发 行 者：新华书店总店北京发行所
开　　本：787×960 1/16　印张：20.5　彩页：1
版　　次：2003 年 3 月第 1 版　2003 年 3 月第 1 次印刷
书　　号：ISBN 7-302-06350-8/TP・4792
印　　数：0001～3000
定　　价：39.00 元

出 版 说 明

1984年，美国国防部出资在卡内基·梅隆大学设立软件工程研究所(Software Engineering Institute，简称SEI)。SEI于1986年开始研究软件过程能力成熟度模型（Capability Maturity Model，CMM），1991年正式推出了CMM 1.0版，1993年推出CMM 1.1版。此后，SEI还完成了能力成熟度模型集成（Capability Maturity Model Integration，简称CMMI）。目前，CMM 2.0版已经推出。

CMM自问世以来备受关注，在一些发达国家和地区得到了广泛应用，成为衡量软件公司软件开发管理水平的重要参考因素，并成为软件过程改进的事实标准。CMM目前代表着软件发展的一种思路，一种提高软件过程能力的途径。它为软件行业的发展提供了一个良好的框架，是软件过程能力提高的有用工具。

SEI十几年的研究过程和成果，都浓缩在由SEI参与研究工作的资深专家亲自撰写的SEI软件工程丛书（SEI Series In Software Engineering）中。

为增强我国软件企业的竞争力，提高国产软件的水平，经清华大学出版社和三联四方工作室共同策划，全面引进了这套丛书，分批影印和翻译出版，这套丛书采取开放式出版，不断改进，不断出版，旨在满足国内软件界人士学习原版软件工程高级教程的愿望。

清华大学出版社

2002年8月

总　　序

周伯生

美国卡内基•梅隆大学软件工程研究所（CMU/SEI）是美国联邦政府资助构建的研究单位，由美国国防部主管。他们确认，为了保证软件开发工作的成功，由软件开发人员、软件采办人员和软件用户组成的集成化团队必须具有必要的软件工程知识和技能，以保证能按时向用户交付正确的软件。所谓“正确的”就是指在功能、性能和成本几个方面都能满足用户要求且无缺陷；所谓“无缺陷”就是指在编码后对软件系统进行了彻底的穷举测试修复了所有的缺陷，或保证所编写的代码本身不存在缺陷。

CMU/SEI 为了达到这个目的，提出了创造、应用和推广的战略。这里的“创造”是指与软件工程研究社团一起，共同创造新的实践或改进原有的实践，而不墨守成规。这里的“应用”是指与一线开发人员共同工作，以应用、改进和确认这些新的或改进的实践，强调理论联系实际。这里的“推广”是指与整个社团一起，共同鼓励和支持这些经过验证和确认的、新的或改进的实践在世界范围内的应用，通过实践进行进一步的检验和提高。如此循环，往复无穷。

他们把所获得的成就归纳为两个主要领域。一个是倡导软件工程管理的实践，使软件组织在采办、构建和改进软件系统时，具有预测的能力与控制质量、进度、成本、开发周期和生产效率的能力。另一个是改进软件工程技术的实践，使软件工程师具有分析、预测和控制软件系统属性的能力，其中包括在采办、构建和改进软件系统时，能进行恰当的权衡，作出正确的判断和决策。CMU/SEI 通过出版软件工程丛书，总结他们的研究成果和实践经验，是推广这两个领域经验的重大举措。

SEI 软件工程丛书由 CMU/SEI 和 Addison-Wesley 公司共同组织出版，共分 4 个部分：计算机和网络安全（已出版了 2 本著作），工程实践（已出版了 8 本著作），过程改进和过程管理（已出版了 11 本著作），团队软件过程和个体软件过程（已出版了 3 本著作）。前两者属于软件

工程技术实践，后两者属于软件工程管理实践。目前这 4 个部分共出版了 24 本著作，以向软件工程实践人员和学生方便地提供最新的软件工程信息。这些著作凝聚了全世界软件工程界上百位开拓者和成千上万实践者的创造性劳动，蕴含了大量的宝贵经验和沉痛教训，很值得我们学习。

清华大学出版社邀请我和郑人杰教授共同组织 SEI 软件工程译丛编委会。清华社计划首先影印 6 本著作，翻译出版 15 本著作。据我所知，在 Addison-Wesley 公司出版的 SEI 软件工程丛书中，人民邮电出版社已经翻译出版了《个体软件过程》和《团队软件过程》，还拟影印出版《个体软件过程》和《软件工程规范》；电子工业出版社已经翻译出版了《净室软件工程的技术与过程》、《能力成熟度模型 CMM 1.1 指南》、《能力成熟度模型集成 CMMI》和《软件项目管理》；北京航空航天大学出版社已经翻译出版了《统计过程控制》。这些出版社共计影印 2 本著作，翻译出版 7 本著作。这样，可以预期我国在今年年底共可影印 8 本著作，翻译出版 22 本著作。各个出版社的有远见的辛勤劳动，为我们创造了“引进、消化、吸收、创新”的机遇。我们应该结合各自的实践，认真学习国外的先进经验，以大大提高我国软件工程的理论和实践水平。

在这套丛书中，特别值得一提的是，在过程工程领域被誉为软件过程之父的 Humphrey 先生所撰写的《软件过程管理》、《技术人员管理》、《软件工程规范》、《个体软件过程》、《团队软件过程》和《软件制胜之道》等 6 本著作，将于今年年内全部翻译出版，其中《软件过程管理》、《技术人员管理》、《软件工程规范》、《个体软件过程》和《软件制胜之道》等 5 本著作亦已经或将于今年年内影印出版。

《软件过程管理》是软件过程领域的开创性著作，是为软件公司经理和软件项目经理撰写的。用这本书提出的原理来指导软件开发，可以有效地按照预定进度得到高质量的软件，同时还可了解如何持续进行过程改进。美国 CMU/SEI 按照这本书提出的原理开发了能力成熟度模型，在国际上得到绝大多数国家的认可和广泛采用，是改进软件过程能力的有力武器。在信息技术迅速发展和企业激烈竞争的今天，能否持续改进过程往往决定企业的命运。

作为一个软件经理，在改进组织的能力之前，首先必须明确绝大多数软件问题是由管理不善所引起的。因此，要改进组织的性能，首先需

要改进自己的管理模式。同时还要认识到软件开发是一项智力劳动，需要拥有掌握高技能和忘我工作的技术人员。因此，有效的软件管理需要充分注意技术人员的管理。

《技术人员管理》这本著作就是为达到这个目的而撰写的。高质量的技术工作要求没有差错，这就要求人们高度专心和高度献身。因此要求人们对他所从事的工作不仅具有高度的责任感，而且具有浓厚的兴趣和高度的热忱。在当前知识经济群龙相争的今天，一个能激励人们进行创造性工作的领导群体，是众多竞争因素中最重要的因素。本书提供了大量的实用指南，可用来有效地改进工程人员、经理和组织的性能。

Humphrey 先生还认为这本书特别适合于在我国工作的软件经理。我国是一个人口大国，拥有大量能干的知识分子，而且信息领域的劳动力价格比国际市场的价格要低，因此吸引了许多国家到我国来投资。但若不提高人员的素质，不在产品质量和进度方面也狠下功夫，就不能在这方面持续保持优势。

《软件工程规范》是为编程人员撰写的。它精辟地阐述了个体软件过程（PSP）的基本原理，详尽地描述了人们如何来控制自己的工作，如何与管理方协商各项安排。在软件工程界，这本著作被誉为是软件工程由定性进入定量的标志。目前在世界范围内，有成千上万的软件工程技术人员正在接受有关 PSP 的培训，以便正确地遵循 PSP 的实践、开发和管理工作计划，在他们承诺的进度范围内，交付高质量的产品。

《软件制胜之道》这本著作描述了团队软件过程的基本原理，详尽地阐述了在软件组织中如何应用 PSP 和 TSP 的原理以及它所能带来的效益。此外，虽然 CMM 同样适用于小型组织，但在其他著作中都没有描述如何应用 CMM 于个体或小型团队，这本书填补了这个空白。应该指出，如果一个组织正在按照 CMM 改进过程，则 PSP 和 TSP 是和 CMM 完全相容的。如果一个组织还没有按照 CMM 改进过程，则有关 PSP 和 TSP 的训练，可以为未来的 CMM 实践奠定坚实的基础。

在软件工程技术实践方面目前共出版了 10 本著作，其中《用商业组件构建系统》、《软件构架实践》和《软件构架评估——方法和案例研究》等 3 本著作详尽地阐述了软件构架的构建、实践和评估。鉴于是否有一个稳定的软件构架，对软件的质量和成本影响很大，因此如何获得一个良好的构架就成为当今软件界研究的重点。我相信这几本著作的出

版，将对我国软件构架领域的研究与实践有重要的参考价值。此外，众所周知，计算机与网络的安全问题对信息系统的可靠使用关系极大，《CERT 安全指南——系统与网络安全实践》的出版将会对我国在这一领域的研究和实践起积极的促进作用。《风险管理——软件系统开发方法》、《软件采办管理——开放系统和 COTS 产品》、《项目管理原理》、《软件产品线——实践和模式》和《系统工程：基于信息的设计方法》等 5 本著作，分别从风险管理、软件采办、项目管理、软件产品线以及信息系统设计方法等几个方面阐述了大型、复杂软件系统的开发问题，是有关发展软件产业的重要领域，很值得我国软件产业界借鉴。

目前我们所处的时代是信息化时代，是人类进入能够综合利用物质、能量和信息三种资源的时代。千百年来以传统的物质产品的生产、流通、消费为基本特征的物质型经济，将逐步进入以信息产品的生产、流通、利用和消费为基本特征的知识型经济。在这个历史任务中，建造和广泛应用各类计算机应用系统是其公共特征。计算机软件是计算机应用系统的灵魂，没有先进的软件产业，不可能有先进的信息产业，从而也不可能建成现代化的知识型经济。

我们应该看到，在软件领域中我国在总体上离世界先进水平还有相当大的差距。但是，我们不能跟随他国的脚印，走他人的老路。我们应该抓住机遇，直接针对未来的目标，在软件工程技术和软件工程管理两个方面，注意研究 SEI 软件工程丛书中倡导的原理和方法，联系实际，认真实践，并充分利用我国丰富优秀的人力资源和尊重教育的优良传统，大力培养各个层次的高质量的软件工程人员，使其具有开发各类大型、复杂软件系统的能力。我衷心地预祝清华大学出版社影印和翻译出版这套丛书，在把我国建设成为一个真正现代化的软件产业大国的历史任务中起到推波助澜的作用，并请读者在阅读这些译著时，对这套丛书的选题、译文和编排等方面都提出批评和建议。

周伯生
于北京
2002 年 8 月 18 日

Architecture-Centric Software Project Management

Architecture-Centric Software Project Management

A Practical Guide

Daniel J. Paulish

Addison-Wesley

Boston • San Francisco • New York • Toronto • Montreal
London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

To my family,
Ellen, Terry, and Nick,
for their love and support

Foreword

A FUNDAMENTAL assumption of my work in the area of software architecture is that the architecture is *the* crucial artifact in the development of software systems. The architecture is the basis for achieving business goals and the basis for achieving software quality attributes. This has led to my interest in the design of software architecture for quality and the evaluation of how well an architecture does achieve its quality goals.

Now, however, I see that this is a technology-centric perspective. If the architecture is central to the achievement of the business goals for which the system is being produced, then the architecture must become central to the project manager as well as to the architect. In this book, Dan Paulish explores what it means for an architecture to be central to the project manager. I knew that the architecture is the basis for the work breakdown structure and for work assignments, but the project manager must do much more than just assign teams to work on portions of a development and monitor their progress.

Since, as Dan says, "schedule and effort estimates produced in the absence of a high-level architecture have minimal value," then the project manager must occupy higher management with other business until the high-level architecture can be produced. The project manager must simultaneously ensure that the high-level architecture is produced within a specified time frame. Dan gives techniques and rules of thumb for accomplishing this.

His rule of thumb about estimation—that 40% of the development time of a project is for design (up to 3 months for high-level design and the remainder for low level), 20% is for coding, and 40% is for testing—should speak very strongly to the research community. What techniques can be developed to leverage the high-level design so that the need for low-level design and testing are reduced? Why are we spending so much research time on coding techniques?

Once a realistic schedule has been achieved, the project manager must manage expectations based on the schedule. The schedule is used to motivate the development team, since they own it, and to give other elements of the organization a basis on which to plan. Building a vertical slice through the architecture enables adding other functionality in increments and enables the adjustment of functionality to meet releases.

Dan points out that just as the architecture embodies trade-offs between various qualities, so too the schedule embodies trade-offs between delivery dates, quality, and functionality. He advocates making clear to the development team what the priorities are among these three and using schedule pressures to keep from overemphasizing quality and functionality. That is, with incremental deliveries in relatively short increments (eight weeks) it is possible to make marketing prioritize features and to choose those features that can be implemented within an increment with acceptable quality.

The schedule depends on the architecture, and the incremental delivery depends on the schedule. This type of control is an essential feature of architecture-based development.

Just as the techniques for designing an architecture borrow heavily from existing design techniques and the techniques for evaluating an architecture borrow heavily from existing review techniques, so, too, the techniques for managing an architecture-centric project are not divorced from existing management techniques. Making schedules explicit, getting buy-in from stakeholders, setting realistic expectations, being sensitive to the foibles of employees, and keeping cool in the midst of tempests are all hallmarks of a good manager in any environment. Still, it is useful to see them articulated.

I was an evaluator for one of the case studies Dan mentions (DPS2000), and it is interesting for me as an evaluator to see what happened to this system. As a consultant, I am always seeing a slice of a development effort, and it is uncommon to find out the end of the story. It is like reading the middle of a novel, with someone to tell you how it started but with no one to tell you how it ended. Although, as with some novels, this is often enough, with others I want to know how the story turned out. The DPS2000 system was one of those where the plot line was interesting and the characters were well developed, and so it is enjoyable to find out the end.

I also resonated with Dan's description of managing international development teams. The time differences among various locations of the team are often the least of the problems. Holidays, vacations, and different attitudes toward work are also problems that must be overcome to successfully build a team with international membership.

In summary, if architecture is to be the centerpiece of a development effort, then the project manager must treat it as the centerpiece, must use it to design schedules, generate estimates, manage people. Identifying the various aspects of project management and discussing them in a lively and personal fashion make this a book I enjoyed reading and one from which I learned a great deal.

Len Bass

Preface

As computer hardware provides more functionality at a lower cost, the need for new applications software is exploding. The World Wide Web is providing more information to more people at an ever-faster rate. Software products must be developed more quickly, with increased functionality, performance, and quality. The pressure on the software engineers who are developing new products and maintaining existing products is increasing.

This book provides some support to the software project managers who are attempting to juggle the demands of meeting their schedule while delivering features with good quality. My experience with observing and participating in many software development projects indicates that good design and project management skills go a long way in achieving successful projects. What is very clear is that it is unlikely that projects will be successful when the software architecture is not well designed or project management skills are missing. I have observed the connections between good software architecture and good project management on many projects, and I hope that some of the tips provided will result in better products.

Motivation

As an industry, we have not been very successful in managing successful software projects. Successful projects are those that meet their planned development schedule, provide the functionality promised, and deliver good-quality software. From the 1995 Standish Group CHAOS report, their research on software projects reported that 16% of projects were completed successfully, 31% were cancelled outright, and 53% were substantially over budget and schedule and delivered less functionality than specified. By 1998, more projects were successful, with 26% completed successfully, 28% cancelled, and 46% over budget and schedule with less functionality [Johnson 1999]. Thus, things are improving, but we still have a terrible track record in our industry for successfully completing software development projects.

Background

I gained the experience for writing this book while managing software design and development projects at Siemens. As part of the Siemens Software Architecture R&D Program, a large number of Siemens projects have been investigated in order to capture how Siemens software architects design software systems. The knowledge gained from this research has been embodied in the four-views architecture design approach described in the *Applied Software Architecture* book written by Christine Hofmeister, Rod Nord, and Dilip Soni [2000]. As the four-views approach was being developed, we had opportunities to participate as architecture design team members for new products being designed in various Siemens businesses. In some cases, we were also asked to plan and manage these new product developments and implement subsystems or components of the architecture. Thus, our project planning and management methods were developed in parallel with the four-views design approach.

A concrete example of this correlation between architecture design methods and project planning methods is the **architecture-centered software project planning (ACSPP)** approach described in Chapter 2. We used this approach to develop cost and schedule estimates for the development projects, based on the software architecture. Since we were heavily involved with participating in software architecture design teams, we began to believe in the advantages to be gained when project planning was done in parallel with design. We were also called into Siemens companies as reviewers, from time to time, and we consistently observed warning flags for projects that either were not planned well or were missing a software architecture that could be easily communicated to the reviewers or the development team.