第 6 章

Software Engineering: A Practitioner's Approach, Eighth Edition

软件工程的人员方面

要点浏览

概念：我们都在努力学习最新的编程语言、最好的新型设计方法、最流行的敏捷过程或最新发布的热门软件工具。但说到底，是人在开发计算机软件。因此，在软件工程中，人对一个项目的成功所起的作用和那些最新最好的技术是一样的。

人员：软件工程工作是由个人和团队完成的。在某些情况下，个人要承担大部分的责任，但大多数行业级软件要由团队完成。

重要性：只有团队的活力恰到好处，软件团队才能成功。软件工程师有时会给人不好相处的印象。但实际上，在要构建的产品中，团队里的工程师与同事及其他利益相关者进行良好合作是非常必要的。

步骤：首先，你要了解并不断积累一个成功的软件工程师应具备的个人特质。然后，你需要提升软件工程工作中应具备的综合心理素质，这样才能在进行项目时少走弯路，避免失误。而且，你要理解软件团队的结构和动态，因为基于团队的软件工程在行业中很常见。最后，你需要重视社交媒体、云端和其他协作工具的影响力。

工作产品：对人员、过程和最终产品有更好的洞察力。

质量保证措施：花时间去观察成功的软件工程师是如何工作的，并调整自己的方法来利用他们所展现的优点。

在《IEEE 软件》的一期特刊中，客邀编辑们 [deS09] 做出如下评论：

软件工程包括大量可以改善软件开发过程和最终产品的技术、工具和方法。技术不断进步并产生了很多鼓舞人心的成果。然而，软件不单是用适合的技术方案解决不适合的技术手段而创造出的一种产品。软件由人开发、被人使用并支持人与人之间的互动。因此，人的特质、行为和合作是实际的软件开发的重心。

在本章之后的章节中，我们将讨论构建成功的软件产品所需的“技术、工具和方法”。但在此之前，我们有必要知道，如果没有技术娴熟并积极参与的人员，那么软件项目是不太可能成功的。

关键概念

敏捷团队
云计算
合作开发环境 (CDE)
全球化团队
有凝聚力的团队
心理学
角色
社会媒体
团队属性
团队结构
团队毒性
特性
XP 团队

6.1 软件工程师的特质

你想成为软件工程师吗？很显然，你需要掌握技术，学习并运用那些理解问题所需的技能，设计有效的解决方案，构造软件并努力测试以达到质量最优。你需要管理变更，与利益相关者沟通，并在合适的情况下使用合适的工具。这些我们都会在本书后面章节中用大量篇幅进行讨论。

但有些事和上述的这些同样重要——就是人的方面，掌握这些方面会让你成为卓有成效的软件工程师。Erdogmus[Erd09] 指出软件工程师个人要想展现“非常专业的”行为，应具备七种特质。

引述　大多数优秀的程序员不是因为报酬或公众关注而做编程，而是因为兴趣。
Linus Torvalds

一个卓有成效的软件工程师有个人责任感。这会让软件工程师去努力实现对同伴、其他利益相关者和管理者的承诺。为获得成功的结果，他会在需要的时候不遗余力地做他需要做的事情。

一个卓有成效的软件工程师对一些人的需求有敏锐的意识，这些人包括同团队的成员、对现存问题的软件解决方法有需求的利益相关者、掌控整个项目并能找到解决方法的管理者。他会观察人们工作的环境，并根据环境和人本身来调整自己的行为。

提问　一个高效率的软件工程师需要具备哪些个人特质？

一个卓有成效的软件工程师是坦诚的。如果发现了有缺陷的设计，他会用诚实且有建设性的方式指出错误。即使被要求歪曲与进度、特点、性能、其他产品或项目特性有关的事实，他也会选择实事求是。

一个卓有成效的软件工程师会展现抗压能力。前面我们提到，软件工程工作经常处在混乱的边缘。压力（以及由此产生的混乱）来自很多方面——需求和优先级的变更、要求苛刻的利益相关者或同伴、不切实际或者令人难以忍受的管理者。但一个卓有成效的软件工程师可以在不影响业绩的情况下很好地处理这些压力。

一个卓有成效的软件工程师有高度的公平感。他会乐于与同事分享荣誉，努力避免利益冲突并且绝不破坏他人劳动成果。

一个卓有成效的软件工程师注重细节。这并不意味着追求完美，而是说他会利用产品和项目已有的概括性标准（如性能、成本、质量），在日常工作基础上仔细思考，进而做出技术性决策。

最后，一个卓有成效的软件工程师是务实的。他知道软件工程不是要恪守教条的宗教信仰，而是可根据当下情景需要进行调整的科学规则。

88

6.2　软件工程心理学

在一篇有关软件工程心理学的学术论文中，Bill Curtis 和 Diane Walz[Cur90] 针对软件开发提出了一种分层的行为模式（图 6-1）。在个人层面，软件工程心理学注重待解决的问题、解决问题所需的技能以及在模型外层建立的限制内解决问题的动机。在团队和项目层面，团队能动性成为主要因素。在这一层面，成功是由团队结构和社会因素决定的。团队交流、合作和协调同单个团队成员的技能同等重要。在外部层面，有组织的行为控制着公司的行为及其对商业环境的应对方式。

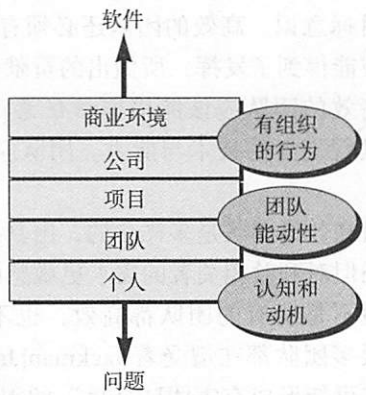

图 6-1　软件工程中的行为模式层（改自 [Cur90]）

在团队层面，Sawyer 和他的同事 [Saw08] 认为团队经常会制造虚拟的界线，这种界线会限制交流，因而会降低团队效率。他们提出了一组“跨界角色”，这能让软件团队成员在不同团队之间有效推进工作。下面这些角色要么是被特定指派的，要么是在工作中自然而然衍生出来的。

- 外联员——代表团队就时间和资源问题与外部的顾客谈判，并取得利益相关者的反馈。
- 侦察员——突破团队界线收集组织信息。“侦察活动包括：审视外部市场，寻求新技术，确定团队外界的相关活动，弄清潜在对手的
89 活动。”[Saw08]
- 守护员——保护团队工作产品和其他信息产品。
- 安检员——把控利益相关者和他人向团队传送的信息。
- 协调员——注重横跨团队及组织内部的交流（如与组织内部的专家团队讨论特定的设计问题）。

提问 软件团队中的成员都会担任哪些角色？

6.3 软件团队

Tom DeMarco 和 Tim Lister[DeM98] 在他们的经典著作《Peopleware》中讨论了软件团队的凝聚力：

在商业领域内，我们经常使用团队这个词，把任何一组被派到一起工作的人称为一个“团队”。但是很多这样的组合并不像是团队。这些组合中的人对成功没有共同的认识，或者是没有明确的团队精神。他们缺少的就是凝聚力。

提问 什么是“有凝聚力的”团队？

一个有凝聚力的团队中的人应该能强烈认识到整体比个体的简单相加更强大。

一旦团队凝聚起来，成功的可能性就会变大。整个团队会变得势不可当、坚不可摧……他们不需要传统的管理方式，当然也不需要外界的推动。他们本来就有动力。

DeMarco 和 Lister 认为有凝聚力的团队中的成员比其他人员生产力更高也更有动力。他们有共同的目标、共同的文化，而且在大多数情况下，一种“精英意识”会让他们变得独特。

现在还没有能打造具有凝聚力团队的万无一失的方法。但高效的软件团队总是会显现一些特征[⊖]。Miguel Carrasco[Car08] 认为高效的团队必须建立目标意识。比如，如果说团队成员认同团队的目标是开发可以转化产品类别的软件，并为此让公司一跃成为行业的领跑者，就可以说他们有很强的目标意识。高效的团队还必须有参与意识，让每个成员都能感受到自
90 己的技能得到了发挥，所做出的贡献是有价值的。

关键点 高效的软件团队是多样化的，是由具备目标意识、参与意识、信任意识和进步意识的人组成的。

高效的团队应该能培养信任意识。团队中的软件工程师应该相信同伴和其管理者的技术与能力。团队应该鼓励进步意识，定期审视软件工程方法并寻求改善途径。

最高效的团队是多样化的，由具备不同技能的人员组成。技术高超的工程师会与技术背景较弱但对利益相关者的需求更敏感的队员搭档。

但不是所有的团队都高效，也不是所有的团队都具有凝聚力。事实上，很多团队都在遭受着 Jackman[Jac98] 所说的“团队毒性”。她定义了五个“可能形成有害团队环境”的因素：（1）混乱的工作氛围；（2）会造成团队成员分裂的挫败；（3）“支离破碎或协调不当”的软件过程；（4）对软件团队中角色

提问 为什么团队会没有凝聚力？

⊖ Bruce Tuckman 发现成功的团队在取得成果的工程中会经历四个阶段（形成，争执，规范，执行）（http://www.realsoftware development.com/7-key-attributes-of-high-performance-software-development-teams/）。

的模糊定义；（5）“持续且重复性的失败”。

为避免混乱的工作环境，团队应获得工作所需的所有信息。一旦确定了主要目标，不到必要时刻就不轻易改变。为避免挫败，软件团队应该尽可能地对所做的决定负责。通过了解要完成的产品、完成工作的人员以及允许团队来选择过程模型，可以避免不当的软件过程（如不必要或过于繁重的任务，或者选择错误的工作产品）。团队自身要建立责任机制（技术评审[⊖]是完成这一事项的好方法），在团队成员出现失误时找出正确的方法。最后，避免陷入失败氛围的关键是建立以团队为基础的反馈和问题解决技巧。

引述 不是每个组合都是一个团队，不是每个团队都是有效率的。
Glenn Parker

除了 Jackman 提到的五个毒性，软件团队还经常遇到团队成员特质不同的问题。有些团队成员性格外向，有些性格内向。有的成员会直观地收集信息，从各种事实中提取概括性观点。而有的成员会有序地处理信息，从已知数据中搜集和整理详尽的细节。有的成员在进行逻辑性、有序的讨论之后做决定。而有的凭直觉，更愿意凭“感觉”做决定。有的成员需要组织任务的详细进度安排，以使他们能够完成项目的某些部分。而有的团队希望有更加自发性的环境，也可以存在未定论的问题。有的成员会在里程碑日期之前很久就把工作完成，以免到时候有压力，而有的人会从最后时刻的紧迫感中受到激励。本节总结了对人的差异的认识以及其他指导规则，这些有助于更好地构建具有凝聚力的团队。 91

6.4 团队结构

“最佳”团队结构取决于组织的管理风格、团队组成人员的数量以及他们的技术水平，还有整体的问题难度。Mantei[Man81] 提出了一些在策划软件工程团队结构时应考虑的项目因素：（1）需解决问题的难度；（2）基于代码行或功能点[⊖]的结果程序的“规模”；（3）团队成员合作的时间（团队寿命）；（4）问题可模块化的程度；（5）所建系统的质量和可靠性；（6）交付日期要求的严格程度；（7）项目所需的社会化（交流）程度。

提问 选择软件团队的结构时应考虑哪些因素？

Constantine[Con93] 针对软件工程团队提出了四种“组织模式”：

1. *封闭模式*组成的团队遵循传统的权力层级模式。这样的团队在建立与之前的成果十分相似的软件时能做得很好，但以封闭模式工作时创新性上相对较弱。
2. *随机模式*组成的团队是松散的，并依靠团队成员的个人自发性。在需要创新和技术性突破时，这类团队可以做得很优秀。但是很难完成“有秩序的操作”。
3. *开放模式*尝试组成一种团队，既具有封闭模式团队的可控性，还具有随机模式团队的创新性。成员们合作完成工作，并有丰富的交流和达成共识的决定，这些都是开放模式团队的特点。开放模式团队适合解决复杂的问题，但没有其他团队的效率高。
4. *同步模式*组成的团队有赖于问题的自然区分，不需要很多的交流就可以将成员组织起来共同解决问题。

提问 定义软件团队的结构时有哪些选择？

引述 如果想逐渐变好，就要具有竞争力；如果想指数级变好，就要合作。
作者不详

作为历史前鉴，最早的软件团队之一是被称为*主程序员团队*的封闭模

⊖ 有关技术评审的详细讨论见第 20 章。

⊖ 代码行（Lines of Code，LOC）和功能点可测量电脑程序规模，见第 33 章。

式团队。这种结构最早由 Harlan Mills 提出，由 Baker[Bak72] 建立。这类团队的核心包括：
92 一个高级工程师（主程序员），负责计划、协调并审查团队的所有技术活动；技术人员（通常2～5人），进行分析和开发活动；一名支持工程师，协助高级工程师的工作，为最小化项目持续性的损失，有时可代替高级工程师。主程序员可以得到以下人员的服务支持：一位或者多位专家（如通信专家、数据库设计者）、支持人员（如技术写作人员、文书人员）和一个软件管理员。

对于主程序员团队结构，Constantine 的随机模式 [Con93] 观点认为，对于具有创造独立性的团队，其工作方法最好是创新的无序。尽管完成软件工作需要有自由意识，但是软件工程组织的核心目标必须是将创新活力运用于创建高效能团队。

SafeHome 团队结构

[场景] Doug Miller 的办公室，优先致力于 SafeHome 软件项目。

[人物] Doug Miller（SafeHome 软件工程团队的管理者），Vinod Raman、Jamie Lazar 以及其他产品软件工程团队的成员。

[对话]

Doug：你们有机会看一下市场部准备的有关 SafeHome 的基础信息。

Vinod（点了点头，看着他的队友们）：是的，但是我们遇到了很多问题。

Doug：我们先不谈问题，我想探讨一下怎样构建团队，谁对什么问题负责……

Jamie：Doug，我对敏捷理念很感兴趣。我觉得我们可以成为一个自发组织的团队。

Vinod：同意。鉴于时间有限，不确定因素很多，而且我们都很有能力（笑），这似乎是很好的方法。

Doug：我没意见，而且你们也知道怎么做。

Jamie（微笑并像在背诵什么一样说）：我们只做策略上的决定，谁在什么时候做什么，但是我们的任务是按时推出产品。

Vinod：还要保证质量。

Doug：很对。但是记住有一些限制。市场限制了需要制造出来的软件增量——当然是在与我们商讨的基础上。

Jamie：然后呢？

6.5 敏捷团队

在过去的 10 年里，敏捷软件开发（第 5 章）被认为放大了问题，扰乱了软件项目工作。回顾一下，敏捷理念支持：客户满意且尽早的软件增量发布，小型的充满动力的项目团队，非正式方法，最少的软件工程工作产品以及整体开发的简化。

6.5.1 通用敏捷团队

小型的并充满动力的项目团队也可称为敏捷团队，他们具备前面章
93 节中所讨论的成功软件项目团队的很多特征（我们会在之后的章节谈到这些特征），并且能够避免产生问题的很多毒素。但是，敏捷理念强调个人（团队成员）通过团队合作可以加倍的能力，这是团队成功的关键因素。Cockburn 和 Highsmith[Coc01a] 在他们的文章中谈道：

关键点 敏捷团队是一个能自主计划和做出技术性决定的自发组织团队。

如果一个项目中的人员都足够优秀，那么他们可以借助任意过程来完成任务。如果他们不够优秀，那么也就没有什么过程能弥补他们的不足——即“人员胜过过程”。然而，缺乏

用户和决策支持也会扼杀一个项目——即“政策胜过人员”。如果不能得到足够的支持，优秀的人员也无法完成工作。

为了有效利用每个团队成员的能力，并完成项目工程过程中的高效合作，敏捷团队都是自组织的。一个自组织的团队不必保持单一的团队结构，而是综合运用6.2节中讨论的Constantine提出的随机、开放和同步模式。

引述 集体所有体现了一种观点：工作成果是属于整个（敏捷）团队而非组成团队的个人的。
Jim Highsmith

很多敏捷过程模型（如Scrum）给予敏捷团队重要的自主性，允许团队自主做出完成工作必需的项目管理和技术决定。计划被保持在最低程度，团队可以选择自己的方式（如过程、方法、工具），仅受商业要求和组织标准的限制。在项目进行过程中，团队自发地及时将重点放在有益于项目特定点的个人能力上。为完成这项工作，敏捷团队可能每天都开例会，协商和同步当天必须要完成的事情。

基于例会期间获得的信息，团队将调整其方法以完成工作增量。随着时间的积累，持续的自发组织和合作可以促使团队完成软件增量工作。

6.5.2 XP团队

建议 尽量保持简单，但需认识到持续的“重构”能获得可观的时间和资源。

作为极限编程（eXtreme Programming，XP）的一部分，Beck[Bec04a]定义了五项价值作为实施所有工作的基础——沟通、简单、反馈、勇气、尊重。任何一项价值都可以促成特定的XP活动、动作和任务。

为使敏捷团队和其他利益相关者达到有效的沟通（如建立所需的软件特性和功能），XP强调客户和开发者之间密切而非正式（口头）的合作，构建用作交流媒介的有效的隐喻[⊖]，以便交流重要概念、获得持续反馈并避免冗长的文档。 94

为了达到简单，敏捷团队在设计时只是考虑当下需要而非长远需求。其目的是创建简单的设计，可以容易地用代码实现。如果必须要改进设计，以后可以对代码进行重构[⊖]。

反馈来源于三种途径：所实现的软件本身、客户以及其他软件团队成员。通过设计和实行有效的测试策略（第22～26章），软件（通过测试结果）可以为敏捷团队提供反馈。团队可以利用单元测试作为其初始测试手段。每个类开发完成后，团队会根据其特定的功能开发单元测试来不断完善每个操作。向客户交付增量时，要用增量所实现的用户故事或者用例（第9章）进行验收测试。软件实现用例的输出、功能和行为的程度是一种形式的反馈。最终，在迭代计划中，新的需求会不断出现，团队可以就成本和进度的影响给客户提供快速的反馈。

引述 怎样在不费力的情况下构建优秀的软件？XP就是答案。
未名

Beck[Bec04a]认为严格地坚持特定XP实践是需要勇气的。换个更好的说法是要有原则。比如说，设计经常会面临要考虑长远需求的巨大压力。大多数软件团队都妥协了，并辩称“为明天做设计”可以在长期发展中节省时间和精力。XP团队必须有原则（勇气），要为今天做设计，要认识到长远需求可能会发生意想不到的变更，因此会带来设计和实现代码的大量返工。

要遵循每一项价值，XP团队极力主张其团队成员之间、其他利益相关者和团队成员之

⊖ 在XP中，隐喻是“任何人——客户、程序员和管理者——都能讲述系统如何工作的一个故事。”[Bec04a]。

⊖ 重构是软件工程师在不改变设计（或源代码）的外部功能或行为的情况下改进其内部结构。实质上，重构可以用来改善设计或实现代码的效率、可读性或性能。

间以及间接地对软件本身的尊重。在成功发布软件增量时，团队对 XP 过程的重视就会油然而生。

6.6 社交媒体的影响

邮件、短信或者视频会议在软件工程工作过程中无处不在。但是这些交流机制不过是现
95 代化替代品或面对面沟通机制的补充。社交媒体是多种多样的。

Begel[Beg10] 和他的同事在文章中讲到软件工程中社交媒体的发展和应用：

围绕软件开发的社会化过程……极大程度上取决于工程师的能力——找到有相似目标和互补技能的个人并将他们连接起来，使团队成员的交流和整个团队都表现得更加和谐，让他们在整个软件生命周期中进行合作和协调，并保证他们的产品在市场上能获得成功。

从某种意义上来说，这种“连接”和面对面的交流一样重要。团队规模越大，社交媒体的价值越大，如果团队在地域上是分离的，这种价值就更大了。

首先，社交网络是为软件项目设定的。软件团队可以通过网络从团队成员、利益相关者、技术人员、专家和其他商业人士那里收集经验，这些人已经受邀参与该网络（如果该网络是私有的）或任何兴趣团体（如果该网络是公有的）。而且无论出现什么状况、疑惑或难题，社交网络都可以做到这一点。社交媒体有很多不同的形式，每一种在软件工程工作中都有一席之地。

博客可用来发表一系列短文以描述系统的重要方面，或者用来发表针对尚处于开发中的软件特性或功能的看法。其重要性还可体现在“软件公司经常用博客非常有效地与他们的雇主及客户分享技术信息和观点，内外兼顾。”[Beg10]

> **引述** 内容很重要，对话同样重要。
>
> *John Munsell*

微博（如 Twitter）由软件工程网络成员使用，对关注他们的人发表短消息。因为这些文章是即时的，而且在各种移动平台上都能阅读，所以信息的分散达到了实时性。软件团队遇到问题时可以随时召开临时会议，出现难题时可以随时寻求专业帮助，也可以实时向利益相关者传达项目的某些方面。

在 targeted on-line 论坛上，参与者可以发布问题、观点、案例或任何其他相关信息。一个技术性的问题在发布之后的几分钟之内就可能得到多个“答复”。

社交网站（如 Facebook、LinkedIn）使软件开发者和相关技术人员达到分离化的连接。同一个社交网站上的“好友”可以找到具有相关应用领域或者要解决问题的知识或经验的好友的好友。以社交网络泛型为基础建立的专业性私有网络可以在组织内部使用。

大多数社交媒体都能组织有相似兴趣的用户形成“社区”。例如，一个由擅长实时嵌入
96 式系统的软件工程师组成的社区，可以为相关领域的个人或团队之间的联系提供有效的方式，以改善他们的工作。随着社区的发展，参与者会讨论技术趋势、应用场景、新型工具和其他软件工程知识。最后，网址收藏夹网站（比如 Delicious、Stumble、CiteULike）为软件工程师或软件团队提供平台，让他们可以为有类似想法的个人组成的社交媒体社区推荐网页资源。

需要着重强调的是，在软件工程工作中使用社交媒体时，不能忽视隐私和安全问题。软件工程师所做的大多数工作可能是他们的雇主专有的，因此将其公开是有害的。所以，一定要在社交媒体的优点和私有信息不受控制的公开之间做出权衡。

6.7 软件工程中云的应用

引述 他们不再称其为网络，而是称其为云计算。我不纠结名称，随便叫什么都可以。

Larry Ellison

建议 云是软件工程信息的强大知识库，但你必须要考虑第29章所讨论的变更控制问题。

云计算为我们提供了一种机制，以获取各种软件工程工作产品、人工制品以及与项目相关的信息。它在各处都能运行，并能消除很多软件项目对于设备依赖的限制；可以让软件团队成员进行独立于平台的、低风险的新型软件工具的实验，并提供对这些工具的反馈；可以提供新的分配方法和 β 软件测试；可以提供针对内容和配置管理的更先进的方法（第29章）。

鉴于云计算可以完成上述工作，它很有可能影响软件工程师们组织团队的方式、工作的方法、交流和连接的方式，以及管理软件项目的方式。无论团队成员使用何种平台、处于什么位置，都可以即时获取某个团队成员开发出的软件工程信息。

从根本上说，信息迅速扩散并极度扩展，这也改变了软件工程动态，并对软件工程中人的作用产生了深远的影响。

但是软件工程中的云计算不是没有风险的 [The13]。云分布在多个服务器，其构造和服务往往不受软件团队的控制。因此，云有很多缺点，且存在可靠性和安全性风险。云提供的服务越多，相对的软件开发环境就越复杂。这些服务是否能与其他供应商提供的服务相匹配？这体现了云服务在协同性上的风险。最后，如果云成为开发环境，其服务必须强调可用性和性能。而这些属性有时会与安全性、保密性和可靠性相冲突。

97

但是从人文的角度来看，与存在的风险相比，云为软件工程师提供了更多的好处。Dana Gardner[Gar09] 将其优点总结如下（带有警告）：

软件开发中任何涉及社交或合作的地方都很可能会用到云。项目管理、进度安排、任务列表、需求和缺陷管理作为核心团队功能会很好地进行自我调整，其中，沟通对于项目同步以及使团队所有成员——无论他们在哪里——处于同一场景非常重要。当然，需要严重警惕的是——如果你的公司是想设计产品中的嵌入式软件，那么不推荐使用云。想象一下得到苹果公司关于下一版 iPhone 的项目计划会怎样。

如 Gardner 所叙述的那样，云的关键优势之一是其增强了“软件开发的社会和协作方面”。在下一节中，你会更多地了解协作工具。

6.8 协作工具

Fillipo Lanubile 及其同事 [Lan10] 认为20个世纪的软件开发环境（SDE）已变成协作开发环境（Collaborative Development Environments，CDE）。[⊖]他们是这样论述的：

工具对于团队成员之间的协作是很有必要的，它们能实现简易化、自动化以及对整个开发过程的控制。全球化软件工程特别需要充足的工具支持，因为距离因素对交流的消极影响直接或间接地加重了协作和控制问题。

在 CDE 中用到的很多工具和本书第二、三、四部分提到的辅助软件工程活动的工具没有什么不同。但是，有价值的 CDE 会提供为改善协同工作特别设计的一系列服务 [Fok10]。这些服务包括：

- *命名空间*使项目团队可以用加强安全性和保密性的方式存储工作产品和其他信息，仅允许有权限的人访问。

⊖ 协作开发环境（CDE）的概念是由 Grady Booch[Boo02] 提出的。

- 进度表可以协调会议和其他项目事件。
- 模板可以使团队成员在创造工作产品时保持一致的外形和结构。
- 98 度量支持可以量化每个成员的贡献。
- 交流分析会跟踪整个团队的交流并分离出模式，应用于需要解决的问题或状况。
- 工件收集可以通过回答以下问题的方式组织工作产品和其他项目制品："是什么导致了某个特定的变更？可以跟哪个讨论过特定制品的人商讨有关变更？（团队）成员的个人工作是如何影响他人的工作的？"[Fok10]

提问　协作开发环境中通用的服务有哪些？

软件工具　协作开发环境

[目标] 随着软件开发不断全球化，软件团队需要的不仅仅是开发工具。他们需要的是一套服务，使得团队成员在本地和远程都能协作。

[机制] 此类工具和服务能使团队建立起协作工作的机制。CDE 可以实现 6.6 节所描述的多种或所有服务，同时也能为实现过程管理（第 4 章）提供本书中讨论的传统软件工程工具。

[代表性工具][⊖]

- GForge。一种包括项目和代码管理设施的协作环境（http://gforge.com/gf/）。
- OneDesk。为开发者和利益相关者提供创造和管理项目工作空间的协作环境（www.onedesk.com）。
- Rational Team Concert。一个深度的、协作生命周期管理系统（http://www-01.ibm.com/software/rational/products/rtc/）。

6.9　全球化团队

在软件领域，全球化不仅仅意味着货物和服务的跨国交流。在过去的几十年中，由位于不同国家的软件团队共同建造的主要软件产品越来越多。这些全球化软件开发（GSD）团队具备传统软件团队（6.4 节）所具有的很多特点，但是 GSD 团队还会面临其他特有的挑战，包括协调、合作、交流及专业决策。本章前面已经讨论了协调、合作和交流的方法。对所有软件团队来说，决策问题因以下四个因素而变得复杂 [Gar10]：

- 问题的复杂性。
- 与决策相关的不确定性和风险。
- 结果不确定法则（比如，工作相关的决策会对另外的项目目标产生意外的影响）。
- 99 对问题的不同看法才是导致不同结论的关键。

引述　越来越多的公司管理者在应对不同的文化。公司变得全球一体化，但团队却被分开并散布在全球。

Carlos Ghosn, Nisson

对于 GSD 团队，协调、合作和沟通方面的挑战对决策具有深远的影响。图 6-2 解释了 GSD 团队所面临的距离挑战的影响。距离使交流复杂化，但同时也强调对协调的需求。距离也产生了由文化差异导致的障碍和复杂性。障碍和复杂性会减弱交流（比如**信噪比**的降低）。在这种动态环境中固有的问题会导致项目变得不稳定。

⊖ 这里提到的工具只是此类工具的例子，并不代表本书支持采用这些工具。在大多数情况下，工具名称被各自的开发者注册为商标。

虽然没有能够彻底改正图 6-2 中各种关系的银弹，但运用有效的 CDE（6.6 节）可有助于减弱距离的影响。

6.10　小结

一个成功的软件工程师必须掌握技术。此外，还必须对自己承担的义务负责、清楚同伴的需求、诚实地评估产品和项目、能适应压力、公平地对待同伴，还要关注细节。

软件工程心理学包括：个人认知和激励，软件团队的群体动力及公司的组织行为。为了改善交流和合作，软件团队成员可以承担跨界角色。

一个成功的（“有凝聚力的”）软件团队比普通团队更多产、更有动力。为实现高效，软件团队必须有目标意识、参与意识、信任意识以及进步意识。此外，必须避免“毒性”——混乱和消极的工作环境、不适合的软件过程、软件团队中模糊的角色定义以及不断暴露的故障。

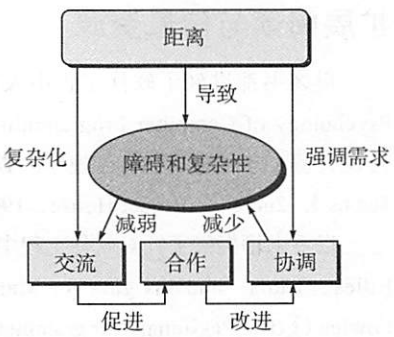

图 6-2　影响 GSD 团队的因素（改自 [Cas06]）

100

团队结构有很多种。有些团队是分层结构，另外一些团队则喜欢依赖于个人主动性的松散结构。敏捷团队采用敏捷理论，一般来说比大多数传统软件团队更自主。敏捷团队强调沟通、简单、反馈、勇气和尊重。

社交媒体逐渐成为很多软件项目中必需的部分。博客、微博、论坛和社交网络能使软件工程社区更有效地进行交流和协作。

云计算对于软件工程师组织团队的方式、工作的方式、交流和连接的方式以及管理软件项目的方式都具有潜在的影响。云能够加强软件开发的社交和协作方面，它的好处远远超过风险。

协作开发环境包含很多服务，可以增强软件团队的交流和协作。这些环境对全球软件开发尤其有益，因为地理上的分离会对软件工程的成功产生障碍。

习题与思考题

6.1　根据你对优秀软件开发者的观察，说出三个共同的个人特质。

6.2　怎样能做到“毫无保留地诚实”，同时又不被（其他人）视为有侮辱意图或有攻击性？

6.3　软件团队是如何构建“人工边界”来降低与其他人交流的能力的？

6.4　编写一个简要场景，描述 6.2 节中的每个“跨界角色”。

6.5　在 6.3 节中，我们提到目标意识、参与意识、信任意识和进步意识是高效软件团队的重要属性。在团队成立时，谁应该对建立这些属性负责？

6.6　对于团队的四种组织模式（6.4 节）：（1）保险公司的 IT 部门；（2）国防项目承包商的软件工程小组；（3）开发电脑游戏的软件小组；（4）大型软件公司。你认为哪种最有效？并阐明理由。

6.7　如果要选择敏捷团队区别于传统软件团队的一个属性，你会选哪个？

6.8　针对 6.6 节描述的软件工程工作社交媒体的形式，选出你认为最有效的并说明理由。

6.9　编写场景，使 SafeHome 团队成员可以在软件项目中利用一种或多种形式的社交媒体。

101

6.10　近来，云成为计算领域中的一个热门概念。运用有关为改善软件工程工作而特别设计的服务的具体例子，描述云是如何提升软件工程组织的价值的。

6.11 研究一下 6.8 节软件工具介绍中提到的某种 CDE 工具（或导师指定的一种工具），准备在班级中对工具的功能进行简要介绍。

6.12 参见图 6-2，距离为什么会使交流复杂化？距离为什么会强调对协调的需求？距离会导致哪些种类的障碍和复杂性？

扩展阅读与信息资源

很多书都讲到了软件工程中人的因素，但有两本书是公认的经典著作。Jerry Weinberg（《The Psychology of Computer Programming》，Silver Anniversary Edition，Dorset House，1998）首次提到构建计算机软件人员的心理学。Tom DeMarco 和 Tim Lister（《Peopleware: Productive Projects and Teams》，2nd ed.，Dorset House，1999）讨论了软件开发的主要挑战在于人而非技术。

以下书籍提供了针对软件工程中人的有用观察：Mantle 和 Lichty（《Managing the Unmanageable: Rules，Tools，and Insights for Managing Software People and Teams》，Addison-Wesley，2012），Fowler（《The Passionate Programmer》，Pragmatic Bookshelf，2009），McConnell（《Code Complete》，2nd ed.，Microsoft Press，2004），Brooks（《The Mythical Man-Month》，2nd ed.，Addison-Wesley，1999），Hunt 和 Tomas（《The Pragmatic Programmer》，Addison-Wesley，1999）。Tomayko 和 Hazzan（《Human Aspects of Software Engineering》，Charles River Media，2004）以 XP 为重点，讲述了软件工程中的心理学和社会学。

Rasmussen（《The Agile Samurai》，Pragmatic Bookshelf，2010）和 Davies（《Agile Coaching》，Pragmatic Bookshelf，2010）论述了敏捷开发中人的因素。有关敏捷团队的重要方面可参见 Adkins 的书（《Coaching Agile Teams》，Addison-Wesley，2010），还有 Derby、Larsen 和 Schwaber 的书（《Agile Retrospectives: Making Good Teams Great》，Pragmatic Bookshelf，2006）。

解决问题是一种独特的人类活动，这方面的著作有：Adair（《Decision Making and Problem Solving Strategies》，Kogan Page，2010），Roam（《Unfolding the Napkin》，Portfolio Trade，2009），Wananabe（《Problem Solving 101》，Portfolio Hardcover，2009）。

Tabaka（《Collaboration Explained》，Addison-Wesley，2006）提供了如何在软件团队中进行合作的指导。Rosen（《The Culture of Collaboration》，Red Ape Publishing，2009）、Hansen（《Collaboration》，Harvard Business School Press，2009） 和 Sawyer（《Group Genius: The Creative Power of Collaboration》，Basic Books，2007）提供了改善技术团队合作的策略和实践性指导。

以培养人员创新性为主题的书有：Gray、Brown 和 Macanufo（《Game Storming》。O-Reilly Media，2010），Duggan（《Strategic Intuition》，Columbia University Press，2007）和 Hohmann（《Innovation Games》，Addison-Wesley，2006）。

Ebert（《Global Software and IT: A Guide to Distributed Development，Projects，and Outsourcing》，Wiley-IEEE Computer Society Press，2011）描述了全球软件开发的整体情况。Mite 及其同事（《Agility Across Time and Space: Implementing Agile Methods in Global Software Projects》，Springer，2010）编写了有关全球开发中使用敏捷团队的论文集。

网上有讨论软件工程中人员方面的大量信息资源，软件过程相关的最新参考文献可在 SEPA 网站 www.mhhe.com/pressman 找到。