本文作者简介：

比雅尼•斯特劳斯特鲁普 (Bjarne Stroustrup) 是C++的设计者和最初实现者。同时也是《C++程序设计语言》、《C++语言的设计和演化》、《程序设计原理与实践》和其他诸多著作的作者。斯特劳斯特鲁普博士是德州农工大学工程学院计算机科学首席教授。他是美国国家工程院会员、AT&T会员、IEEE会员和ACM会员。他的研究领域包括分布式系统、原型设计、编程技术、软件开发工具和编程语言。他积极参与了ANSI/ISO C++的标准化工作。

翻译：龚小聪，石葆光（华中科技大学本科生， Dian团队队员）

What Should We Teach New Software Developers? Why?

我们应该教给未来的软件开发者什么？原因是什么？

---为了更好的迎合工业界的需求，我们得从根本上改变计算机科学教育。

在系统研发中，处于中心地位的应当是计算机科学知识。否则，我们在研发的过程中通常只会依靠经验法则，最后做出来的系统扩展性差、可靠度低、成本高。因此，我们需要改变计算机科学教育，来提高软件研发的水平。

问题所在

计算机科学教育和工业界需求之间，往往有很大的差距。设想以下场景：

著名的计算机教授很自豪地说："我们教的不是编程，而是计算机科学。"

企业经理："他们在编程方面简直就是白痴。"

很多情况下，他们都是对的，并且还不只是表面上的正确。学术界的工作不仅仅就是培养普通的程序员；工业界也不仅仅需要"面面俱到的高水平的思想者"和"科学家"。

另一名计算机教授说："我从不写代码。"

另一名企业经理说："我们不雇佣计算机专业的毕业生。教物理系毕业生编程，比教计算机系毕业生物理简单多了。"

双方各执一词，但太过理想化，并且都有误解。教授不能教那些自己不去实践（很多时候甚至就是从未实践过）也没有理解的东西；而对于企业经理来说，只有在软件质量需求异常低，以至于连物理系的学生（还有一些没有受过训练的计算机系的学生）都能应付这样的情况下，他的话才是对的。当然，有些物理系的学生十分刻苦努力，在物理学之外又同时掌握了计算机科学，他们不包括在内——这样具备双重技能的人，在我看来才是最理想的人才。

计算机教授（谈论一个学生）："他在企业找到了一份工作。"

另一个计算机教授："太可惜了，他是那么有前途的一个学生。"

这样的脱节带来了很多问题，解决起来也很麻烦。

工业界希望计算机系的毕业生去写软件（至少在他们的职业初期是这样的）。这种软件通常是某个长线代码库的一部分，并且会被用到需要高可靠度的系统上去，例如嵌入式系统和分布式系统。但是，很多毕业生除了有业余编程爱好外，没有真真实实地接受过软件开发的教育或培训。更甚的是，很多学生只是把编程当做完成家庭作业的小把戏，很少在系统测试、系统维护、文档化和代码可读性等方面上有所重视。另外，学生们不能把在一门课中学到的知识和另一门课联系起来。这样一来，我们经常看到学生在算法、数据结构和软件工程的考试中取得很高的分数，但是他们在操作系统实验课上却全然不顾数据结构、算法和软件架构，只去照搬现成的方法，实验结果一团糟。

对于很多人来说，"编程"已然成为胡乱修改代码、调用他人库函数这两种行为的结合。对背后的原理，他们几乎一无所知。"系统维护"和"代码质量"的概念他们理解甚少，甚至无视。在工业界，很多人抱怨找不到能够理解"系统"和"软件构建"的毕业生。这种抱怨反映出了现实的问题。

我的电脑还没死机呢

大家常抱怨软件质量。不过在过去的十年里，很多软件已经大有改善。不幸的是，这种改善是代价昂贵的。这个代价体现在人的精力耗费和计算机资源消耗上。我们学会了怎样从不可靠的

部分组建出相对可靠的整体，方法就是不断地添加运行时检查，并构建大量的测试代码。有时连代码结构本身都被改变了，而且还不是被改得更好。软件层次过多、依赖关系错综复杂，让一个无论多么有能力的人都难以理解整个系统，这不是个好征兆：或许在将来，我们不能理解自己的软件系统，也不能测试系统中的关键部分。

当然，有一些系统架构师，他们顶着压力反对继续构建规模膨胀、难以理解的系统。当飞机没有坠毁、电话还能用、邮件也按时到达的时候，我们得感谢他们。他们是值得表扬的，他们的努力使得软件形成了一套成熟可靠的法则、工具和技术。可惜，他们只是少数。大多数情况下，膨胀性的软件构建方式支配着人们的印象和思维。

类似的也有一些教育工作者，他们反对理论和工业实践分离。他们同样也值得赞扬和支持。事实上，我所知道的每一所教育机构都有一些项目，它们旨在为学生提供实践机会；一些教授们也投入了大量精力，想在某些项目中取得成功。但从更广的角度看，我觉得没那么好——实践项目或者实习固然是好的开始，但在教会学生全面方法这一点上，它们并不能取代平衡的课程体系。比起“计算机科学”，更加注重“软件工程”或者“IT”这样的东西，这或许就反映出了教学侧重点上的变化。然而对于解决问题来说，这样的变化可能还是换汤不换药。

我对“工业界”和“学术界”的描述有些讽刺意味。但我相信，稍有经验的人都会意识到这其中是反映着现实问题的。我是一个工业界的研究人员和管理人员（我在AT&T贝尔实验室工作了24年，其中7年是部门经理）；在学术界我也工作了6年（在一个工科学校的计算机系任教）。我经常出差，每年跟来自数十家公司（大多数是美国的公司）的技术层和管理层的人员都有很认真的探讨。我发现了大学培养的学生不能适应工业界需要，这对计算机系和计算机产业来说，都是个严重的威胁。

学术界和工业界的脱节

那么我们该怎么做呢？工业界倾向于雇佣那些受过新技术和新工具培训的“开发者”们，而学术界最大的目标在于培养出更多的教授。我们必须调整这些目标来取得更大的进步。即将进入工业界的毕业生们，要更好的理解软件开发；工业界也要研究出更好的机制，以便自己能吸纳新想法、新工具和新技术。让一个优秀的开发者进入到一个企业中，结果这个企业旨在避免中等水平程序员犯技术错误，这样是没有好处的，因为这会限制他们尝试任何更新、更好的东西。

接下来谈谈程序规模的问题。许多工业系统包含数百万行代码。然而一个甚至连1000行代码的程序都没写过的学生，也可以从计算机专业毕业。主流的工业项目都涉及很多人，而在校的项目和实践注重个人的努力，这样就阻碍了团队精神的培养。许多组织意识到这一点后，开始专注于简化工具、技巧、语言及作业流程，以尽量减少对开发者技能的依赖。这是对天赋和精力的浪费，让每个人的作用都被削减到不能再削减的地步。

工业界既依靠“实践证明正确”的工具和技术，也沉迷于梦想中的“灵丹”、“革命性突破”，“杀手级应用”，等等。他们希望技术水平低、可被随时替换的开发者，被少数牛到不屑于关注代码质量的“梦想家”所引导。这会导致他们在基础工具（例如编程语言和操作系统）的选择上极度保守。这将导致众多的专有代码和互不兼容的基础设施被开发出来：他们会需要一些超越底层的东西，这样开发者才能开发应用程序。尽管基础工具之间存在共性，但是平台的供应商限制了开发人员。奖励制度需要同时适合于企业的长远计划以及短期的成果。由此产生的费用，以及新项目的失败率，都是惊人的。

面对工业界的既有现实和其他类似的阻碍，学术界却闭关自守，做自己最擅长的东西：一小群志同道合的人在一起，研究可以被隔离观察的现象，建立了坚实的理论基础，为理想化的问题构造出完美的解决方案。那些用来处理规模庞大、风格古老代码的专有工具，不适合这个模型。像工业界一样，学术界也应有适合自己的激励制度。这一切可以很好的提高学术研究的质量。因此学术界的成果和工业界的需求如同方枘圆凿一样不合拍，为此工业界需要为培训和基础设施的开发付出很大代价。

总有人建议：“只要企业能支付给开发者足够高的工资，问题就解决了。”这种方法可能会有所

帮助，但是为同样的工作支付更多工资的作用不会太大。一个可行的方案是，企业雇佣更优秀的开发者。把软件开发当作生产流水线，流水线上是水平一般并且可以被替换的开发者，这一想法从根本上就是有问题的，这会带来资源浪费。它把最具竞争力的人们赶出这一行业，并且让学生们不愿进入。为了打破这种恶性循环，学术界必须培养出更多的具备相关技能的毕业生，而工业界必须采用能够利用这些技能的工具、技术和工艺。

关于专业化的设想

“计算机科学”是一个糟糕的、有误导性的术语。计算机科学并不主要是关于计算机，也不主要是一门科学。相反，它涉及的是计算机的使用，以及与计算（算法和定量思维）有关的思维方式和工作方式。它结合了各个方面，有科学、数学和工程，并且通常通过计算机结合在一起。对于几乎所有计算机系的人来说，它是“纯计算机科学”，与实际的应用程序是脱节的。

一个计算机科学专业的人，和其他领域（如医学或物理学）的人，在开发一个应用程序时，怎么区分他们呢？答案一定是“对计算机科学核心的掌握”，那么这个“核心”是什么呢？它包含许多在计算机系内开设的课程——算法、数据结构、计算机体系结构、编程规范、数学（主要是定量推理）和系统（如操作系统和数据库）。为了整合知识、获得解决较难问题的灵感，每个学生必须完成若干需要团队合作的项目（你可以称之为初级软件工程）。理论和实践之间需要一个平衡，这是很基本的原则。计算机科学不仅仅是原则和理论，也不仅仅是累代码。

和“计算”这整个领域相比，这个核心明显地更面向计算机。因此，没有计算机领域专业知识（例如图形学、网络、软件架构、人机交互和安全）的人，就不能被称为计算机科学家。然而，这些仍然不够。计算机科学的实践本质上是应用的和跨学科的，因此，每一位计算机系的毕业生，都应该具备一些其他领域（例如，物理，医学工程，历史，会计，法国文学）的知识，并且能够达到和辅修学位一样的水平。

有经验的教育者们会发现：“这根本不可能，几乎没有学生能在四年内掌握那些。”他们是对的，需要做一些妥协。我的建议是，计算机科学系的第一个学位应该是硕士学位，并且整个课程体系都是按照以培养硕士为目的来设计的，而不是本科阶段加上一两年的硕士阶段。立志于学术研究的学生则在硕士学位之后，继续攻读博士学位。

很多教授会反对：“我没有时间写程序！”然而，我认为，那些教育准备成为软件开发专业人士的学生的教授们，要抽出时间来写程序，他们所在的教育机构也应该想办法鼓励他们写程序。计算机科学的终极目标是研发出更好的系统。让一个没见过病人的医生，来教学生怎么做手术，或是一个从没碰过键盘的钢琴教师教学生弹钢琴，你觉得靠谱吗？计算机系的教育，必须要让学生站在比书本学习更高的层面，使他们能够掌握所学知识在一个完整系统里的实际应用，以及对代码中美学价值的鉴赏能力。

我使用了“专业人士”这样一个词，这是一个具有众多意义和暗示的词。在医学和工程等领域，这意味着认证。认证是一个非常棘手并且感性的话题。然而，我们的文明有赖于软件。如果任何人都可以根据个人的喜好，或者公司的决策，来修改代码的重要部分，那这样是合理的么？如果现在是，那50年内仍然是合理的么？那些数百万人使用的软件，如果没有质量保障，合理么？真正的问题是，通过认证实现的专业认可，依赖大量的知识、工具和技术。一个具有认证资格的工程师，可以保证一栋楼房用合适的材料和技术建造起来。然而，计算机系学生应当具备哪些素质，现在还没有一个被普遍接受的清单，我也不知道该如何做。如今，我甚至不知道如何挑选出一群人来设计一个认证考试。

工业界怎么做才能缩小与学术界之间的差距？比起谈论学术，描绘“工业界”和“工业需求”要困难得多。毕竟，学术界有一个相当标准的结构和比较规范的方法，以实现其目标。工业界则更为多样化：有大有小，有盈利也有非盈利，在构建系统时使用的方法有一般的也有高端的，等等。因此，我对于消除这一脱节还没有一点可行的方法。然而，我已经发现这个脱节带来的问题：很多明明十分依赖于计算的企业，技术水平却低得可怕。

工业界的经理：“专业知识和技术的引入，对企业生存至关重要。”

没有“机构记忆(Institutional Memory)”以及一套能够挖掘和招募有天赋之人的体制，任何组织都不能成功。增强企业界与对软件有兴趣的学术界的合作，可能让双方都更有成效。联合研究，以及对超越课程学习的终生学习的强调，可能会在合作中扮演重要角色。

结论

我们必须做得更好，否则我们的设备就会继续出毛病、膨胀、耗费资源。最后，有些部分会以一些不可预知的和灾难性的方式停止工作（想想网络路由、网上银行、电子投票以及电网控制）。我们必须让学术界和工业界都做出改变，使得他们之间的脱节能够被减轻。我的建议是计算机系的教育要基于专业核心部分，并且加上一些专业方向和偏应用的知识，教育的目标是学生做出的软件能够取得认证，至少是一部分能取得。这可能需要工业界和学术界的长期合作才能实现。

原文地址：http://cacm.acm.org/magazines/2010/1/55760-what-should-we-teach-new-software-developers-why/fulltext