미국의 소프트웨어 연구 전략

변 석우

미국 National Science and Technology Council의 Working Group 중의 하나인 IT^2 (Information Technology for Twenty First Century)는 2000년부터 향후 약 5년간에 걸쳐 미국이 정보 기술 분야에서 실행할 연구 내용들과 예산 안을 작성하여 미국 대통령에게 제출하였다. (원 제목은 Information Technology For Twenty-First Century: A Bold Investment In America's Future 이다.) 이 안은 1999년 6월에 작성된 것으로, 본 고에서는 그 중에서 우리에게 관심을 줄 수 있는 소프트웨어 분야에 대한 연구 내용을 발췌하여 소개한다.

미국 정부가 정보 기술 분야에 대한 투자를 이 권고 안의 내용대로 이행할 지에 대해서는 확신할 수 없으나, 이 보고서의 작성 시기 및 수행 시기 등을 고려할 때 그럴 가능성이 높다고 보여진다. 그러나 그런 가능성 여부를 떠나서, 제안되는 내용 그 자체가 매우 타당하고 설득력이 있으므로 우리도 한번 고려해 볼 만하다고 생각한다.

이 계획안 내에서 분류는 Software로 되어 있지만, 이 내용들을 성취하기 위해서는 proof and verification, analysis, OOP and declarative programming, domain-specific programming, script programming, proof-carrying codes, concurrent programming 등의 프로그래밍 언어 기술의 도움이 필수적이라는 것을 쉽게 느낄 수 있을 것이다.

Fundamental Information Technology Research and Development for Software

Research and development in this area will investigate long-term, high risk issues confronting computer science and engineering. This IT^2 component has four focal points: software, human computer interfaces and information management, scalable information infrastructure, and high end computing. Research and development in these areas will encourage a diversified, long term strategy to make computing and information systems easier to use, more reliable and secure, more effective, and more productive.

NSF will lead this multi-agency effort, and DoD will make significant contributions in software and scalable information infrastructure. DOE, NASA, NIH, and NOAA will also fund activities in this area. Each agency will encourage participation through its normal mechanisms. Selection of research proposals will be coordinated through cross agency review panels, and agency program managers will help ensure that the proposals do not unnecessarily duplicate other efforts. The intent is to ensure a range of complementary research efforts including single and multiple principal-investigator (PI) research, multiple-PI/multiple-field collaborations, intramural research in institutes and Federal laboratories, and joint industry-government-academia experiments or proofs-of-concept.

Details of each agency's plans, including research programs, funding mechanisms, and milestones, are described in the Agency Specifics at the end of this section.

From the desktop computer to the phone system to the stock market, our economy and society have become increasingly dependent on software. The PITAC concluded that not only is the demand for software exceeding our ability to produce it, but that the software produced today is fragile, unreliable, and difficult to design, test, maintain, and upgrade. The small software failures that shut down large parts of the Nation's phone systems and the "Year 2000" (Y2K) problem are but two significant examples of what can go wrong.

Software — unlike bridges, airplanes, and nuclear power plants — is constructed without the benefit of standard practice by technologists who lack thorough training in the engineering sciences of reliability, maintainability, and cost-effectiveness.

The research and development proposed in this area will significantly improve the concepts, techniques, and tools to engineer the software infrastructure, and will help educate software developers in well-founded and more productive engineering methods to develop techniques and tools to create productive, reliable, and useful software.

- *Software engineering:* Currently, we do not understand how to design and test complex software systems with millions of lines of code in the same way that we can verify that a bridge is safe. Fundamental software research will increase software productivity, make software more reliable and easier to maintain, and automatically discover errors. These efforts will reduce the cost and time penalties currently required to make systems safe and reliable. Progress on safety and reliability is important in critical systems such as the telecommunications network, medical devices, the electric power grid, and the air traffic control system.
- *End-user programming:* One way to address our shortage of programmers is empower end-users through the creation of domain-specific development tools — the spreadsheet being perhaps the most common example. Just as telephone companies, when faced with a scalability barrier with switchboard operators, solved the problem by putting dialing technology in the hands of the end-users, so can software developers push portions of development out to end-users through various kinds of "smartware," domain-specific and user-friendly templates for filling a variety of needs. Developing tools and techniques to make end-user programming more widespread will require advances in intelligent templates, domain-specific languages, and programming-by-example.
- ***Component-based software development:*** Today, most programs are written from scratch, one line of

code at a time. The software industry lacks the equivalent of the interchangeable parts used in manufacturing. Component-based research will make it easier to find the right software component, to accurately predict the behavior of a software system assembled from smaller components, and to support an electronic marketplace in software components.

- ***Active software****:* Active software participates in its own development and deployment. We see the first steps towards active software with "applets" that can be downloaded from the Internet. Research in active software will lead to software that can update itself, monitor its progress toward a particular goal, discover a new capability needed for the task at hand, and safely and securely download additional software needed to perform that task.
- ***Autonomous software****:* Research in this area will lead to more intelligent software and robots, such as unmanned vehicles that keep our troops from harm's way; intelligent agents or "knowbots" that search the Internet on our behalf; robots and knowbots that plan, react appropriately to unpredicted changes, and cooperate with humans and other robots; cars that can drive themselves and automatically avoid collisions; and robots that can explore planets or places on Earth that are unsafe for human travel.