The Economics of Software and the Importance of Human Capital

By Richard R. Nelson and Paul M. Romer
Challenge

Although economists have long appreciated the centrality of technical advance in the process of economic growth, a complete understanding of the key processes, investments, and actors that combine to produce it has not come easily. Indeed, these processes are very complex and variegated. Economists broadly understand that the advance of technology is closely associated with advances in knowledge. It also is clear that new knowledge must be embodied in practices, techniques, and designs before it can affect an economic activity. Beyond this, different economic analyses focus on or stress different things.

Some discussions stress the "public good" aspects of technology, seeing new technology as ultimately available to all users. Others treat technology as largely a "private good," possessed by the company or person that creates it. Many economists have studied research and development as the key source of new technology. Those that have focused on R&D done by private, for-profit business firms naturally assumed that the technology created through corporate R&D is, to some extent at least, a private good. By contrast, economists who have stressed the "public good" aspects of technology have focused on government investments in R&D, "spillovers" from private R&D, or both. (These spillovers are another manifestation of the divergence between the public and private returns noted above.) Still others argue that a single-minded emphasis on organized R&D as the source of technical advance sees the sources too narrowly. They point to evidence that learning-by-doing and learning-by-using are important parts of the processes whereby new technologies are developed and refined.

Another matter on which economists have been of different minds is whether technical advance and economic growth fueled by technical advance can adequately be captured in the mathematical models of economic equilibrium that economists developed to describe a static world. Joseph Schumpeter and economists proposing "evolutionary" theories of growth have stressed that disequilibrium is an essential aspect of the process. By contrast, recent theories that descend from neoclassical models presume that the essential aspects of technical advance and economic growth can be captured by extending the static equilibrium models.

While we do not want to underplay the important open questions about how economists ought to understand technical advance, a workable consensus for policy analysis seems to be emerging from these divergent perspectives. Technology needs to be understood as a collection of many different kinds of goods. These goods can have the attributes of public goods and private goods in varying proportions. Some are financed primarily by public support for R&D, others by private R&D. Both business firms and universities are involved in various aspects of the process. Other parts of technology are produced primarily through learning-by-doing and learning-by-using, both of which can interact powerfully with research and development. There are aspects of the process that are quite well treated by equilibrium theories, with their emphasis on foresight, stationariness, and restoring forces. Still other aspects are better suited to the evolutionary models, with their emphasis on unpredictability and the limits of rational calculation.

One way to summarize this emerging view is to focus on three types of durable inputs in production. We will take our imagery and language from the ongoing digital revolution and refer to these three different types of inputs as hardware, software, and wetware. Hardware includes all the nonhuman

objects used in production - both capital goods such as equipment and structures and natural resources such as land and raw materials. Wetware, the things that are stored in the "wet" computer of the human brain, includes both the human capital that mainstream economists have studied and the tacit knowledge that evolutionary theorists, cognitive scientists, and philosophers have emphasized. By contrast, software represents knowledge or information that can be stored in a form that exists outside of the brain. Whether it is text on paper, data on a computer disk, images on film, drawings on a blueprint, music on tape - yen thoughts expressed in human speech - software has the unique feature that it can be copied, communicated, and reused.

The role of software, hardware, and wetware can be discerned in a wide variety of economic activities. Together they can produce new software, as when a writer uses her skills, word processing software, and a personal computer to write a book. They can produce new hardware, for example, when an engineer uses special software and hardware to produce the photographic mask that is used to lay down the lines in a semiconductor chip. When an aircraft simulator and training software are used to teach pilots new skills, they produce new wetware.

These three types of inputs can be discerned in activities that are far removed from digital computing. In the construction of the new city of Suzhou in Mainland China, the government of Singapore says that its primary responsibility is to supply the software needed to run the city. The hardware is the physical infrastructure - roads, sewers, and buildings, etc. - that will be designed according to the software. The wetware initially will be the minds of experts from Singapore, but eventually will be supplied by Chinese officials who will be trained in Singapore to staff the legal, administrative, and regulatory bureaucracies. The software comprises all the routines and operating procedures that have been developed in Singapore, examples of which range from the procedures for designing a road, to those for ensuring that police officers do not accept bribes, to instructions on how to run an efficient taxi service.

Traditional models of growth describe output as a function of physical capital, human capital, and the catch-all category, "technology." The alternative proposed here has the advantage of explicitly distinguishing wetware (i.e., human capital) from software. This is an essential first step in a careful analysis of the intangibles used in economic activity. The next step is to identify the reasons why software differs from both hardware and wetware.

Economists identify two key attributes that distinguish different types of economic goods: rivalry and excludability. A good is rival if it can be used by only one user at a time. This awkward terminology stems from the observation that two people will be rivals for such a good. They cannot both use it at the same time. A piece of computer hardware is a rival good. So, arguably, are the skills of an experienced computer user. However, the bit string that encodes the operating-system software for the computer is a nonrival good. Everyone can use it at the same time because it can be copied indefinitely at essentially zero cost. Nonrivalry is what makes software unique.

Although it is physically possible for a nonrival good to be used by many people, this does not mean that others are permitted to use it without the consent of the owner. This is where excludability, the second property, comes in. A good is said to be excludable if the owner has the power to exclude others from using it. Hardware is excludable. To keep others from using a piece of hardware, the owner need only maintain physical possession of it. Our legal system supports each of us in our efforts to do this.

It is more difficult to make software excludable because possession of a piece of software is not sufficient to keep others from using it. Someone may have surreptitiously copied it. The feasible

alternatives for establishing some degree of control are to rely on intellectual property rights established by the legal system or to keep the software, or at least some crucial part of it, secret.

Our legal system assigns intellectual property rights to some kinds of software but not others. For example, basic mathematical formulas cannot be patented or copyrighted. At least at the present time, there is no way for the scientists who develop algorithms for solving linear programming problems to get intellectual property rights on the mathematical insight behind their creation. On the other hand, the code for a computer program, the text of a novel, or the tune and lyrics of a song are examples of software that is excludable, at least to some degree.

The two-way classification of goods according to excludability and rivalry creates four idealized types of goods. Private goods and public goods are the names given to two of these four types. Private goods are both excludable and rival. Public goods are both nonexcludable and nonrival. The mathematical principles used to solve linear programming problems are public goods. Because they are software, they are nonrival; it is physically possible to copy the algorithms out of a book. Because the law lets anyone copy and use them, they are nonexcludable.

In addition to private goods and public goods, there are two other types of goods that have no generally accepted labels but are important for policy analysis. The first are goods that are rival but not excludable. The proverbial example is a common pasture. Only one person's livestock can eat the grass in any square foot of pasture, so pastureland is a rival good for purposes of grazing. If the legal and institutional arrangements in force give everyone unlimited access to the pasture, it is also a nonexcludable good. Frequent allusions to "the tragedy of the commons" illustrate one of the basic results of economic theory: Free choice in the presence of rival, nonexcludable goods leads to waste and inefficiency.

The fourth category, and one of central importance to the study of technical advance, is of nonrival goods that are excludable, at least potentially. We stress the term "potentially" here because society often has a choice about the matter. It can establish and enforce strong property rights, in which case market incentives induce the production of such goods. Alternatively, it can deny such property rights. Then if the goods are to be provided, support through government funding, private collaborative effort, or philanthropy is needed. Many of the most important issues of public policy regarding technical advances are associated with this latter choice. For rivalrous goods, establishing and enforcing strong property rights is generally a good policy (although there are exceptional cases.) But for nonrivalrous goods, the matter is much less clear.

By and large, society has chosen to give property rights to the kind of software commonly called "technology" and to deny property rights but provide public support for the development of the software commonly referred to as "science." Establishing property rights on software enables the holder of those rights to restrict access to a nonrival good. When such restriction is applied - for example, by charging a license fee - some potential users for whom access would be valuable but not worth the fee will choose to forego use, even though the real cost of their using it is zero. So putting a "price" on software imposes a social cost - positive-value uses that are locked out - and in general the more valuable the software is to large numbers of users, the higher will be the cost. To cite just one example that influences the choices of working scientists, there are experiments that could be carried out using PCR (polymerase chain reaction) technology that would be done if the scientists involved could use this technology at the cost of materials involved. Some of these are not being done because the high price charged by the current patent holder makes this research prohibitively expensive.

Note that this is very different from what is entailed in establishing property rights on rival goods.

Only one user can make use of a rival good at any one time. So property rights, or options to sell them, encourage the rival good to be used by those to whom it is most valuable.

Our legal system tries to take account of the ambiguous character of property rights on software. We give patents for some discoveries, but they are limited in scope and expire after a specific period of time. For rival goods this would be a terrible policy. Imagine the consequences if the titles to all pieces of land lapsed after seventeen years. For some nonrival goods, such as works of literature or music, we grant copyright protection that lasts much longer than patent protection. This can be rationalized by the argument that costs from monopoly control of these goods creates relatively little economic inefficiency. For other goods, such as scientific discoveries and mathematical formulas, the law gives no protection at all. This presumably reflects a judgment that the cost of monopoly power over these goods is too high and that we are better off relying on such nonmarket mechanisms as philanthropic giving and government support to finance and motivate the production of these types of software.

One important distinction between different types of software is the difference in the amount and variety of additional work that needs to be done before that software makes an actual contribution that consumers would be willing to pay for. Property rights on software that is directly employed by final consumers can lead to high prices - consider the high prices on some pharmaceuticals - and cut out use by some parties who would value use, but will not or cannot pay the price. For software such as this, however, that is close to final use, it is possible for users to make reasonably well founded benefit-price calculations.

It is quite otherwise with software whose major use is to facilitate the development of subsequent software. Any market for software, such as mathematical algorithms and scientific discoveries far removed from the final consumer, would risk being grossly inefficient. Over time, many producers have to intervene, making improvements and refining the basic idea, before such software can be finally embodied in a technique, practice, or design that produces value and is sold to a final consumer. Economic theory tells us that the presence of monopoly power at many stages in this long and unpredictable chain of production can be very bad for efficiency.

In the worst case, property rights that are too strong could preempt the development of entire areas of new software. In the computer software industry, people capture this dilemma by asking the rhetorical question, "What if someone had been able to patent the blinking cursor?" The point applies equally well to many other important discoveries in the history of the industry - the notion of a high-level language and a compiler, the iterative loop, the conditional branch point, or a spreadsheet-like display of columns and rows. Extremely strong property rights on these kinds of software could have significantly slowed innovation in computer software and kept many types of existing applications from being developed.

In the production of computer software, basic software concepts are not granted strong property rights. Software applications, the kind of software sold in shrink-wrapped boxes in computer stores, is protected. This suggests a simple dichotomy between concepts and final applications that mirrors the distinction noted in the beginning between the search for basic concepts by a Niels Bohr and the search for practical applications by a Thomas Edison. As the work of Pasteur would lead us to expect, this dichotomy hides important ambiguities that arise in practice. At the extremes, the distinction between concepts and applications is clear, but in the middle ground there is no sharp dividing line. Courts are forces to decide either that software for overlapping windows or specific key sequences should be treated as essential parts of an application that are entitled to patent or copyright protections, or that they are basic concepts that are not given legal protection. In the realm of

software, there are many shades of gray. The simple dichotomy nevertheless serves as a useful framework for guiding the economic and policy analysis of science and technology, for science is concerned with basic concepts, and technology is ultimately all about applications.

This article is excerpted from a longer article entitled ***Science, Economic Growth, and Public Policy****, which appears in the March-April 1996 issue of* ***Challenge****. It is also part of a forthcoming book by the authors.*