



***Software as Capital: An Economic Perspective
on Software Engineering* by Howard Baetjer,
Jr. Piscataway, NJ, The Institute of Electrical
and Electronics Engineers, Inc., 1998, 194 pages.
ISBN 0-8186-7779-1**

PETER LEWIN[†]

Howard Baetjer has written a book that will be of interest to a number of distinct audiences. First, this book will interest specialists in the computer field, especially those involved in software development. They will find here linkages from the technicalities of software engineering to the broader economic context. They will find important insights into the value dimensions of software components and perhaps they will be stimulated to think more deeply about the social institutions in which they work and how they may be improved. Second, this book will be of interest to curious lay readers interested in the history of the software industry and the evolution of software generally. Third, economists in general will find this book full of keen insights applicable to a number of subfields in their discipline. And finally, fourth, this book is an application of market process capital theory and is of particular interest to specialists in that field. It is primarily from this fourth perspective that I derived particular pleasure in reading the book.

Baetjer makes use of the theory of capital as developed in the Austrian tradition, by Böhm-Bawerk, Hayek and Lachmann. “What is true of software is true of capital goods in general” (11). And what is true for capital goods in general is true for software. In this way software is seen *both* as an analogy for capital in general and as part of capital in general. As an analogy it is very precise. Individual software products embody knowledge in the same way as all capital goods embody knowledge. As such they represent the fruits of roundabout methods of production. “Capital goods *are knowledge*, knowledge in the peculiar state of being embodied in such a form that it is ready-to-hand for use in production. The knowledge aspect of capital goods is the fundamental aspect. Any physical aspect is incidental” (10). So capital goods in general can be thought of as “embodied knowledge” that relate to each other much in the same way that individual software components relate.

In spinning out this analogy, Baetjer provides a very accessible and complete introduction to the market process theory of capital, notably that of Lachmann. Among the topics elucidated are capital heterogeneity (which in this context implies *knowledge heterogeneity*), complementarity, personal, tacit and intersubjective knowledge, the role of social institutions, the meaning of capital maintenance in a changing world, and the crucial relationship

[†]School of Management, University of Texas at Dallas, P.O. Box 830688, J051, Richardson, TX 75083-0688.

between innovation and capital accumulation. This last topic mentioned is particularly well elucidated by the software analogy since it is obvious to anyone that software grows by the continual upgrading of the capabilities it embodies. The traditional economic approach to capital accumulation is to separate it from the question of innovation. The software analogy (capital as knowledge) cautions against doing this and invites the realization that accumulation and innovation are inextricably linked. If time and knowledge belong together and capital accumulation proceeds over time, it must be expected that capital goods will come to embody new knowledge. Moreover, considering the evolution of software makes it clear that a “social learning process” is involved in which the disparate and specialized knowledge of many different individuals is combined to deliver the final products. The same is inescapably true of capital goods in general. “Because design is a process of bringing together and embodying productive knowledge in a handy, ready-to-use form, design is a learning process. Because that knowledge is of different kinds and widely dispersed among different people and institutions, design is a social learning process—it depends on the interaction of a number of people” (28).

The history of software development, outlined in chapter 2, is likewise a metaphorical window into the development of capital structures in general. An example is modular programming. “Modular programming... is a manifestation of division of knowledge in capital. In modules, different sets of knowledge are embodied in such a way that they can usefully be shared across time and space. . . .The modules must be *complementary* to one another in use. That means both that they must fit, and also that people who use them must be able to see without too much trouble just *how* they fit” (33). Similarly, “The evolution of programming languages gives us a classic instance of the way better tools enable a division of knowledge across time and space” (35). The evolution of software is as much about the evolution of *methods* of design and production as it is about the evolution of *products*.

In chapter 3 this dynamic perspective is extended to the question of designing new software. Production is seen as a learning process rather than a process of simple information transfer. Planning for software production is like planning a journey whose ultimate destination is unclear and where the correct route is not yet revealed. Once the journey has begun certain signposts along the way provide hints as to which road to take next, but there is no infallible roadmap. Certain techniques, like prototyping, automatic programming and particularly, object oriented programming are critically examined. The dynamic efficiency properties of object oriented programming is extolled and elaborated later in the book. In chapter 4 the focus is narrowed further to the question of designing evolvable software. Through all this Baetjer remains solidly and seamlessly within the capital-structure-theoretic framework. “Evolution is necessarily *coevolution* of the different elements of the system. In the capital structure, this means that which tools become useful and which become obsolete at any time is determined by what *other* tools happen to be developed also, and what technologies happen to be discovered” (92).

The dynamic perspective is crucially relevant to the question of optimal design. Baetjer appropriately warns against the “optimization trap.” Producing software optimally designed to perform today’s tasks will usually involve the inefficient sacrifice of ability to efficiently perform tomorrow’s task, where the latter is, as yet, unknown. “[T]his means that *optimization* of software for any task, *as defined at a particular moment*, should usually be

sacrificed for greater flexibility of design” (93). The application to capital goods in general should be obvious. Lachmann spoke about the necessity for “capital regrouping” (which Baetjer calls “capital recombination,”—perhaps more accurately) in the face of unexpected change. The capital stock contains “fossils” in the capital structure (Lachmann, 1986, p. 61) that represent failures to adapt. Thus Baetjer counsels, “In order to maintain evolvability of their designs, especially in our time of astonishingly rapid capital structure evolution, producers of capital goods must not commit themselves to a particular view of the future, but rather make their best estimate of a range of likely paths of capital structure evolution, and build into the design of their products the flexibility with which to cope with this range of paths” (96).

An important way to plan for evolvability is through modularity. In the software universe this implies object-oriented programming. Each software “object” “knows” how to accomplish some general task, or set of abstractly identified tasks. This facilitates programming by allowing programmers to rely on the accomplishments of other programmers. In the capital structure more generally, modularity is achieved through what Lachmann called a “division of capital.” Different capital goods (objects) embody the knowledge to accomplish (when combined with complementary goods) a variety of tasks. Should any one good fail, its effects can be isolated and easily remedied. Modularity facilitates diagnosis, adaptation, understanding (knowing “how” rather than “what”) and, therefore, dynamic accumulation. Knowledge economizing is achieved in the process of capital development as well as in the process of production as normally understood.

Chapter 5 offers an analysis of the limitations facing software development as presently constituted. Baetjer asserts that “by the standards of other industries, the software industry has astonishingly little specialization and division of knowledge” (119). The diagnosis of the cause of this malaise is a combination of blinkered thinking and inability to cheaply enforce property rights in software products. According to Baetjer the solution lies in effecting a paradigm shift away from selling products to selling usage. That is, consumers should receive the software for free and pay a fee based on intensity of use, for example in a manner similar to the way electric utilities meter and charge its customers. This would provide both greater incentives for development and very accurate feedback on product usage, which would greatly aid development planning. Of course, one must assume that the ability to monitor usage is more easily accomplished than monitoring unauthorized duplication. Recent developments along these lines seem to add some support to Baetjer’s conjecture.

The book ends with a comprehensive summary and two appendices. The first appendix is a survey and critique of the theory of economic growth in the light of the insights derived from a market process approach. The second appendix explicitly extends the discussion of software development and evolution to “hard tools,” that is to physical assets. Thinking about the capital structure as analogous to the software structure leads to new insights. For example, modularization and standardization are ways of providing knowledge that is useful in product development and in the development of tools for further development.

The book is well written and organized. Each chapter begins with two pertinent quotations, one from Tennyson’s *Locksley Hall*, that add to its interest.