Increased Productivity Through Reuse: An Economist's Perspective

Don Lavoie, Howard Baetjer, and William Tulloh
Center for the Study of Market Processes
George Mason University
Fairfax, VA 22030

agorics@gmuvax.gmu.edu

Abstract

From the economic viewpoint, increased software reuse seems to depend on these factors:

- continuing development and use of the modularization techniques of encapsulation and information hiding in order to save on maintenance costs
- division of programmer labor into increasingly specialized areas, accompanied by area-specific tools, frameworks, and reusable components
- investment in a healthy reuse environment, including market-based management structures that address the communication, discovery, and incentives problems, and the increased use of component libraries and/or markets
- property rights regimes that appropriately balance producer and user interests.

1 Introduction

Widespread reuse of software components has long been advocated as a key to increased productivity and better quality software. In addition, markets in reusable components have been seen by some as an important element in realizing these improvements [McIl 68, Broo 87, Cox 90]. While much progress has been made in addressing the relevant technical factors, it has become increasingly apparent that organizational, cultural, institutional, and economic factors must also be addressed in order to reap the gains from wide-scale reuse.

Among the important technical advances have been developments in abstraction mechanisms, modularization techniques and object oriented technologies. These technologies promise to play an important role in helping to realize the benefits of reuse by enabling alternative ways of packaging software through modular construction. Software developers will be able to take advantage of economies generated by the increased division of labor and capital, while maintaining the flexibility

needed to adapt to changing requirements and demands. Some of the benefits taht are seen as deriving from reuse include lower development costs, reduced time of development, greater predictability of budget and schedule, and lower maintenance costs.

The Agorics Project, a team of researchers at the Center for the Study of Market Processes at George Mason University, is now engaging in a comprehensive study of the software components industry. Our study will examine the interaction of these technological advances with the institutional and economic factors that will affect the development of a vigorous software components industry. In particular we will investigate the possibility of markets in reusable components.

2 Modular Construction and Object-Oriented Technologies

Many different visions of markets of software components have been proposed, based on a wide variety of abstraction mechanisms and resulting definitions of components. Our own investigation is taking a rather inclusive view of the possible scale and nature of these components, but we see object-oriented methods as an indispensable part of the move to reuse. The more modular methods of software development that result from the adoption of object-oriented technologies promise to provide a dramatic increase in the reuse of components and frameworks.

A key aspect of object-oriented programming in encouraging reuse is the support of encapsulation. Encapsulation of domain abstractions allows for greater reuse of others' experience. Information hiding through encapsulation also eases the task of integrating different components. Mark S. Miller and K. Eric Drexler have liked encapsulation in object-oriented programming to the discovery by programmers of property rights over data [Mill 88]. They write, "encapsulation in software serves the same crucial function as property rights in human affairs: it establishes protected spheres in which entities can plan the use of their resources free of interference from unpredictable external influences." This modularization of data and action helps provide greater flexibility in the reuse of software components.

One great virtue of property rights in markets is the ability to rearrange them in response to changing circumstances and in order to meet new demands. Similarly, object-oriented methods promise better maintainability and evolvability through encapsulation and information hiding. The modular nature of object-oriented development encourages reuse by facilitating alternative packaging of functionality across different applications, allowing a wider variety of user requirements to be met. In addition, modular construction allows greater scope for reuse in sequential versions of an application by facilitating quicker response to changing user requirements through time.

3 The Structure of Production of Software

Reuse in software involves a shift to a more capital-intensive software development process [Wegn 84]. The accumulation of capital is the accumulation of knowledge and experience. Capital goods such as components, frameworks and designs embody domain knowledge, while tools, environments and methodologies embody knowledge that aids in the solution of recurrent software development problems. Effective reuse must build on prior experience and adapt this knowledge to constantly evolving

needs. To achieve large gains in productivity and quality from reuse will require reuse across a variety of developers and organizations. New software communities [Gibb 90] must emerge that enables large-scale cooperative reuse.

Traditionally economists have seen increased productivity as resulting from the increased complexity of the structure of production – the extension of the division of labor and greater use of capital. Software development, however, is still based largely on a model of craft production.

Craft production has certain advantages; most important among them is its flexibility in adapting to rapid change and in customizing the product to meet specific needs. Human capital, the skills and experience of programmers, represents the main source of reuse, as well as the main source of creativity and flexibility. But a reliance on specialized human capital can be risky: wide diversity in skills, high programmer turnover, and the expense of training and education, all limit the effectiveness of this form of reuse. More importantly, however, this form of production does not scale up well in its ability to cope with more and more complex software projects.

A common solution to this problem in other industries has involved a shift from a complex production process to a more complex structure of production. Rather than build in a single complex stage with multipurpose human capital, the idea is to build a complex series of simpler stages with more specialized human capital. The increasing division of labor is aided by the division of tasks into easily manageable units, and by the standardization among resulting static advantages of the division of labor: 1. increased mechanization – the substitution of capital goods for human capital 2. reduction in the amount of human capital (skill) required for each task due to the increased routinization of tasks, and 3. economies of scale in manufacturing.

But software is not amenable to traditional scale economies of mass production. Software development requires greater flexibility than is offered by the efficiency gains in standardized manufacturing processes. In software, development, not manufacturing, is of the essence. Moreover, attempts to benefit from an increased division of labor are often overwhelmed by coordination and communication problems [Broo 75].

What is needed is the ability to maintain the flexibility and creativity of the craft approach, while reaping the gains in productivity due to a greater division of labor and a greater use of capital, thus enabling more complex software projects to be undertaken. Such an approach which some have termed "lean production" as contrasted with craft and mass production, must focus on the dynamic advantages of division of labor and capital [Woma 90]. These advantages come from the more effective use of knowledge and enhanced learning that accompanies such coorperative efforts. The advantages can be found in: 1. the increased rate of discovery of product and process improvements 2. the move from human capital to knowledge embodied in capital goods seen not as a dumbing down of production, but as a freeing up of human capital for application on the margin of improvement, and 3. economies of scope, not of scale, in development from the sharing of resources and experience across multiple projects [Cusa 91].

4 Investment in Reuse

To realize these benefits requires new organizational and institutional responses to the increased division of labor and capital. Changes in the technological structure of production must be accompanied by changes in the institutional structure of production. A variety of property right regimes,

contractual relationships, and management structures must be explored to solve problems that arise with the communication and incentive that coordinate different stages of the production. The forms that will emerge depend on tradeoffs between various institutional and technological solutions to these coordination and communication problems.

One important change that arises is the need to ensure an adequate return on the investment required in developing and maintaining the capital asstes necessary for wide-scale reuse. This problem emerges whether reusable components are produced within or across firms. In fact, the problems within firms can often be usefully analyzed as analogous to problems in the market.

Investment in reusable software requires that software be managed as a valuable asset, and that the necessary organizational support be given to ensure a chance at an adequate return on the investment. This requires a long term commitment by management, including investment in the complimentary assets (infrastructure), and changes in evaluation criteria and reward structures. One of the key problems facing the firm is how to provide appropriate incentives for both the creation and use of reusable components, and assuring adequate communication of what components are available.

Much work on reuse has dealt with the most appropriate repository for reusable software components. Advances in search techniques and classification schemes provide an important area of research in improving access to components. Technological advances can help reduce the transaction costs of search: such techniques as browsers, hypertext navigation, classification methods, software agents, and good old librarians can all play an important role.

In addition to such improvements in library technology, increasing attention is being paid to the organizational incentives and communication flows needed to maintain a flow of reusable components into the library and a flow of reusable components to projects when and where they are needed. Repositories by their nature are not well suited to providing useful incentives. The problem arises of how to allocate the costs of building and maintaining the library among the various projects that use components from the libraries. Software, of course, is not subject to wear and tear, but it is subject to economic obsolescence. Continual investment is required in the maintenance of value to user. Rewards for useful contributions to the library and rewards for making use of the library need to be a fundamental aspect of project development.

Libraries as a shared resource serving many masters can be viewed as what economists call a commons. The problem is the opposite of the usual story of the commons; the problem is not the danger of overgrazing a fixed supply of pasture, but of "undergrazing" reusable components due to the lack of incentives and information regarding what makes for good reusable components, and when it is appropriate to use them. Care must be taken in setting up the appropriate rewards for contributions, and in reducing the costs of accessing and learning about the contents of the libraries on the part of users.

Electronic markets also offer a potential solution to the problem of affordable access to a wide variety of reusable components. Libraries are essential passive; they wait for users to come search them. Electronic markets, in contrast, reflect the active matching of buyers and sellers. Component vendors have an economic incentive to produce useful components and to reduce the transaction costs to users of gaining knowledge of and access to components. We can imagine the emergence of component brokers who try to match available components to developers' needs.

5 Property Rights and Markets in Reusable Software

Many barriers must be addressed before vigorous markets in reusable components will emerge. Problems of importance include the limited extent of the market due to lack of standards, language incompatibilities, and domain specificity, and the need to establish certification, testing and reputation mechanisms. The biggest challenge facing the emergence of markets, however, is establishing appropriate property rights regimes which can balance producer and user interests. Entrepreneurs must discover combinations of attributes and functionality that users value (and means of delivery which keep low the transaction costs to the user) and price them so as to earn profits. The challenge to establishing markets in software components lies in finding the institutional and technological solutions which allow producers to be adequately rewarded, and ensure users receive quality software in a timely and affordable manner.

The ease with which software can be copied signals the difficulty of defining property rights in software. Component pricing policy, and the very nature of what is bought and sold, may well be shaped by tradeoffs between legal and technological means of protecting intellectual property. Current developments in patent and copyright law may determine what types of reusable components may be bought and sold. In addition to intellectual property law, liability issues in regard to reusable components are also an important issue that must be addressed, but as recent Nobel Laureate Ronald Coase has argued, this too is at root of a question of ownership [Coas 60].

We need not depend entirely on legal means; as experience from other industries has shown us, there exists a wide array of possible means of appropriating sufficient return on the investment in intellectual property. In addition to legal solutions, there are a variety of technological and marketing options that have worked in other industries. The solutions most appropriate for different kinds of granularities of software components should emerge as markets evolve.

References

[McIl 68] McIlroy, M. D. (1968) "Mass Produced Software Components" in P. Nauer and B. Randall (eds.) *Software Engineering*, NATO Science Committee, October.

[Broo 87] Brooks, Frederick (1987) "No Silver Bullet: Essence and Accidents of Software Engineering" *Computer*, April.

[Cox 90] Cox, Brad (1990) "Planning the Software Industrial Revolution" *IEEE Software*, November.

[Mill 88] Miller, Mark S. and K. Eric Drexler (1988) "Markets and Computation: Agoric Open Systems" in B. Huberman (ed.) *The Ecology of Computation*, North-Holland.

[Wegn 84] Wegner, Peter (1984) "Capital-Incentive Software Technology" *IEEE Software*, 1:3, July. See also Ludwig M. Lachmann (1956) Capital and Its Strucutre Kansas City: Sheed Andrews and McMeel.

[Gibb 90] Gibbs, Simon, Dennis Tsichritzis, Eduardo Casais, Oscar Nierstrasz, and Xavier Pintado (1990) "Class Management for Software Communities" *Communications of the ACM*, September.

[Broo 75] Brooks, Frederick P., Jr. (1975) *The Mythical Man-Month: Essays on Software Engineering*, Addison Wesley Publishing Co.

[Woma 90] Womack, James P., Jones, David T., Ross, Daniel, and Carpenter, D.S. (1990) *The Machine That Changed the World* New York: Maxell MacMillan International.

[Cusa 91] Cusamano, Michael A. (1991) *Japan's Software Factories: A Challenge to U.S. Management*, Oxford university Press, pp. 427-28.

[Coas 60] Coase, Ronald (1960) "The Problem of Social Cost" *Journal of Law and Economics*, October.

6 About the Authors

Don Lavoie is Associate Professor of Economics at George Mason University. He is a leading critic of centrally planned economics and co-director (with Mark S. Miller) of the Agorics Project, a research effort sponsored by the Center for the Study of Market Processes and aimed at investigating issues that overlap between computer science and economics. Howard Baetjer and William Tulloh are graduate students in the Economics Department at George Mason University and members of the Agorics Project.