Institutions as Abstraction Boundaries:
Negotiated Categories and the Self-Reorganization of the Market Order

Bill Tulloh, Pricing Institute, btulloh@pricinginstitute.org
Mark S. Miller, HP Labs, markm@caplet.com

To appear in: Economics, Philosophy, & Information Technology: The Intellectual Contributions of Don Lavoie, September 20, 2002, George Mason University, Fairfax, VA.

Abstract: A central claim of the modern Austrian school is that a competitive market order can solve the knowledge problem while a centrally planned economy cannot. While Austrians such as Hayek have focused on the essential role of abstract rules and the coordinating role of prices, they have largely neglected the familiar, day-to-day institutions of store fronts, standardized contracts, and specific markets towards which people orient their actions. These secondary institutions, as Lachmann calls them, are examples of what software developers call abstraction boundaries. Abstraction boundaries both categorize knowledge into productive divisions and coordinate plans through time. We apply the software concepts of abstraction and modularity to better understand how these institutions promote both coherence and change. We argue that the drawing and redrawing of abstraction boundaries is a neglected aspect of the market process.

* The authors would like to thank Darius Bacon, Howard Baetjer, Jack High, Ken Kahn, Thomas McQuade, Nick Szabo, and Lauren Williams for helpful comments.

I. Introduction

The exploration of the knowledge problem – the ability of an economic system to generate and sustain a complex structure of production based on an extensive division of knowledge – formed the central theme of Don Lavoie's research. His critique of central planning and investigations into the nature of knowledge led him to emphasize the indispensable role of the market process in solving the knowledge problem.[1] Only the market process can harness the diverse knowledge of millions of people into an evolving pattern of cooperative relationships that serves an ever increasing variety of individual purposes.

According to the critique of central planning developed by Mises, Hayek, Lavoie and others, central planners lack the knowledge necessary to successfully plan production and calculate economic tradeoffs. Central planners do not have access to the tacit and contextual knowledge that individuals use in planning their activities, nor do they have access to competitive price and profit signals that are required for calculating tradeoffs among the myriad uses of scarce means. The effective use of knowledge, Lavoie argues, requires a decentralized solution – a solution provided by the market process.

For the market process to make effective use of knowledge, it must solve two problems: 1) it must divide knowledge to capture gains from specialization, and 2) it must coordinate the separate plans of

[1]Lavoie (1985b) provides an historical overview of the calculation debate which focused on the limits of comprehensive planning. Lavoie (1985a) extends this critique to non-comprehensive planning. Lavoie (1986), drawing on the work of Michael Polanyi, emphasizes the role of inarticulate knowledge. His subsequent investigations into the nature of the knowledge problem, influenced especially by the hermeneutic philosopher Hans Georg Gadamer, led him to focus on the interplay of articulate and inarticulate knowledge. See for example, Lavoie (1990b, 1991, and 2003). Vaughn (1994) provides an account of the importance of the calculation debate for the development of the modern Austrian school.

market participants to secure the gains from cooperation. Gains from specialization arise when "people can use other people's knowledge to their own advantage without themselves acquiring it" (Vaughn, 1999:133). Specialization economizes on knowledge production by reusing proven solutions; specialization increases knowledge production by encouraging the discovery of new solutions – "knowledge grows through division" (Loasby, 1999:50). Each increase in specialization, however, "necessitates additional coordination somewhere in the social structure" (High, 1986:117). Specialized activities must be combined with complementary activities, and individual plans based on time and place specific knowledge must be coordinated with those of others (Hayek 1948a, 1948b). The market process must do more than just coordinate plans based on dispersed knowledge that exists at a point in time; it must also adapt these plans to new knowledge constantly being created in the pursuit of further gains from specialization and cooperation.

The market process solves the twin problems of the division and coordination of knowledge through abstraction. Abstractions, by selectively hiding information, enable market participants to make use of a complex network of specialized knowledge without needing to acquire the knowledge themselves. As one computer scientist (Turbak, 2002:4) describes it:

> Abstraction is ubiquitous in the modern world and we depend on it for functioning in our day-to-day lives. We are able to use a wide array of machines and devices (e.g. cars, telephones, stereos, computers) without having to understand the details of how they work. Supermarkets, department stores, and utilities are purveyors of abstractions; for the most part, we do not need or want to know how a loaf of bread is baked, how a piece of clothing is made, or how our electricity, water, and gas are produced.

To facilitate cooperation, we need more than abstract mental categories. We need to make the abstractions meaningful to others so that they can use them for their own purposes. Programmers call the process of carving up knowledge into meaningful units creating abstraction boundaries. An abstraction boundary captures a specialized solution to a type of problem and packages it for reuse. The abstraction boundary hides the details of how the solution is implemented from potential users, while providing them with the information they need to apply the solution to their particular problem. Programmers use abstraction to divide programs into modules that can be combined with other modules through well-defined interfaces (the boundaries) to create complex behavior.

Abstraction boundaries enhance productivity by bundling reusable solutions to common problems. As Cunningham and Beck (1989:19) explain, "The abstractor … comprehends a problem space, isolates a portion of that space for which a general solution is possible, and provides a reusable solution." The reusable solution embodies a resolved decision: "the decision to use an abstraction replaces the decisions resolved by the abstraction. Abstractions increase productivity when the former decisions are more easily resolved than the latter" (1989:17). The abstraction becomes useful to others when the cost of reusing the abstraction is less than the cost of re-creating it.[2]

Abstraction boundaries secure gains from specialization by providing reusable solutions. They enhance cooperation by reducing the cost of using the solutions. Abstraction boundaries limit what clients need to know about solution providers, and what solution providers need to know about their clients. The boundary defines what the solution is, but not how it is provided, nor why it is used. Clients, by orienting their plans to the boundary, can ignore implementation details which are likely to change.[3] Providers, meanwhile, are free to change how they provide the solution so long as they continue to provide the type of solution defined by the boundary. Abstraction boundaries enable both solution providers and their clients to coordinate and adapt their particular plans based on the abstract plans embodied in the boundary.

[2] Demsetz (1988:159) describes a similar process occurring in the market: "Because it is uneconomical to educate persons in one industry in the detailed knowledge used in another, recourse is had to developing or encapsulating this knowledge into products or services … A production process reaches the stage of yielding a salable product when downstream users can work with, or consume, the 'product' without themselves being knowledgeable about its production."

[3] The separation of interface from implementation is known in software engineering as information hiding. Information hiding is a key principle for good modular design. See Parnas (1972).

By borrowing concepts from software engineering, we are drawing on work, done during the late 1980s and early 1990s, by Don Lavoie, the current authors, and others. The Agorics Project explored the relationships between object-oriented concepts from software engineering and the market institutions of property and contract.[4] The recognition that software developers faced an analogous knowledge problem to that of markets and that they had independently evolved solutions that resemble market solutions suggested that this would be a fruitful line of research to increase our understanding of both systems. We argue that the commonality of solutions is no mere coincidence, but a consequence of underlying principles for organizing complex adaptive systems. The concepts of abstraction and modularity that underlie the object-oriented approach to organizing a complex software system also underlie the market process approach to organizing a complex structure of production.[5]

We argue that the Austrian vision of the market process can be enhanced by recognizing the role that abstraction boundaries play in dividing knowledge and coordinating plans. Hayek's account of an abstract order based on abstract rules provides a framework for understanding abstraction boundaries, but fails to acknowledge their role. What Lachmann (1971) calls secondary institutions – the day-to-day institutions towards which people orient their actions – fills the gap, capturing what we mean by abstraction boundaries. Abstraction boundaries also shed light on how markets deal with the problem of institutional coherence and change (Lachmann 1971, 1978, 1991). By facilitating orderly change, abstraction boundaries coordinate plans through time. Following Lavoie, we argue that abstraction boundaries emerge as part of the market process; they are best viewed as negotiated categories resulting from the dynamic process of exchange and dialogue.

II. Hayek and the Primacy of the Abstract

Mind as a System of Classification

Friedrich Hayek, in particular, emphasizes the fundamental importance of abstraction. Abstractions, he (1973:29) explains, "are a means to cope with the complexity of the concrete which our mind is not capable of fully mastering." What he calls "the primacy of the abstract" (1978b), is deeply woven into his vision of the market order, visible in many diverse strands of his thought. The primacy of the abstract stems from his work on the mind, begun while he was his twenties and later elaborated and presented in *The Sensory Order* (1952b). His theory of the mind informs numerous aspects of his work including the limits to understanding complex phenomena, his concept of spontaneous order, and the importance of abstract rules of conduct. While Hayek paints a complex picture of the multi-faceted role of abstraction, he does not seem to recognize the role of abstraction boundaries. Abstraction boundaries, we argue, emerge as a natural complement to his discussion of abstract orders based on abstract rules.

Hayek presents the mind as a system of classification. We do not first perceive the world in its detailed particulars and then build up more abstract representations. To the contrary, our ability to perceive particulars depends on pre-existing abstractions which we use to classify our various sense perceptions. Perception must be viewed as an act of classification; we perceive only certain abstract properties of objects, not concrete details. As Heinrich Klüver (Hayek, 1952b:xviii) explains, "What we perceive are never unique properties of individual objects, but always only properties which the objects have in common

[4] The name comes from the "agoric systems" papers (Miller and Drexler 1989a, 1989b, and Drexler and Miller 1989) which applied Hayekian insights to coordinating software systems. The discovery of these papers by Tulloh inspired Lavoie to launch the Agorics Project as an interdisciplinary research effort between economists and software engineers. The Agorics Project consisted of Lavoie and a group of his graduate students, including Bill Tulloh, Howard Baetjer and Kevin Lacobie, in collaboration with a number of software engineers, including Phil Salin, Mark S. Miller, Dean Tribble, and Eric Drexler. See Lavoie (1990a) and Lavoie, Baetjer, Tulloh (1990) for an overview of the Agorics Project.

[5] See Lavoie, Baetjer, Tulloh (1991a, 1991b), Lavoie, Baetjer, Tulloh and Langlois (1993), Cox (1996), and Baetjer (1998) for explorations of the relationship between object oriented programming and market processes. See Baldwin and Clark (1997), Langlois (2002), and Garud, Kumaraswamy, and Langlois (2002) for recent and complementary discussions on the role of modularity in markets.

with other objects. Perception is thus always an interpretation, the placing of something into one of several classes of objects."

As Hayek makes clear, the classifying of objects does not proceed based on a simple one-to-one mapping of observed objects into mental categories. It is rather the "product of superimposition of many 'classifications' of the events perceived according to their significance in many respects (1978b:36)." Each classification triggers a response in terms of a kind of action; we respond to a certain class of events with a disposition to a certain kind of action. It is only through the superimposition of many classifications and dispositions that a particular action is specified: "both the specification of a particular experienced event and the specification of a particular response to it are the result of a superimposition of many such dispositions" (Hayek, 1978b:40).

Specification by superimposition, Hayek (1978b:48) claims, is the "best description of the mechanism for the operation of … the primacy of the abstract, because each of its causal determinants decides only one of the attributes of the resulting action." The specification of action through the superimposition of multiple dispositions enables the individual to make use of knowledge of typical aspects of a situation while adapting the response to the unique aspects. The various dispositions serve as adaptations to the typical features of the environment. By superimposing many dispositions, we are able to generate unique responses to novel situations.[6]

The Market as a System of Classification

Hayek, especially in his later work, presents a vision of the market order that is remarkably similar to his account of the sensory order. Following through on this analogy suggests treating the market as a dynamic system of classification. Both mind and market are examples of what Butos and McQuade (1999) describe as knowledge-generating orders – orders where the knowledge generated consists of classifications meaningful in the context of that order. They (1999:29) explain:

> If orders are characterized as knowledge-generating objects, then knowledge needs to be defined with reference to the order in which it was generated, since it is the set of classification categories particular to the order in question. In this view, therefore, it is strictly incorrect to say that the market gathers up 'divided' individual knowledge and makes this information available to many others - if what is meant is that the problem the market solves is simply the accumulation of separate pieces of knowledge or information into one, generally accessible bundle. Instead, the market takes, as input, knowledge in the individual sense – individual knowledge with respect to potentially useful goods and services – and classifies this, producing a totally different kind of knowledge. This is a vital thing for human survival only because the classifications produced by the market can be perceived by individuals and interpreted by them as enhancements to their individual knowledge which can be used to increase their adaptive ability and further their pursuit of happiness.

Hayek's description of the market as a system of classification is perhaps most visible in the shift in his characterization of the price system. In his 1945 paper, "The Use of Knowledge in Society," he makes the deservedly celebrated argument that the price system acts as a telecommunication system: "It is more than a metaphor to describe a price system as … a system of telecommunication" (1948b:87). Prices help coordinate the diverse plans of individuals by conveying to them the knowledge needed to coordinate their plans. The price system, Hayek (1948b:86) tells us, is "a mechanism for communicating information," which is conveyed in an "abbreviated form, by a kind of a signal."

By 1988, Hayek no longer presents the price system as a "telecommunication system." This "more than a metaphor" comparison he made in 1945 has been replaced. Instead, he (1988:15) writes that, "the values or prices formed by interaction in markets prove to be further superimposed means of classifying kinds of actions according to the significance they have for an order of which the individual is merely one element in a whole which he never made." What is noteworthy is not that these two descriptions, made more than fifty years apart, should differ, but that the latter description uses the language of the primacy of the abstract. By referring to prices as "further superimposed means of classifying," Hayek is emphasizing

[6] Holland et al. (1989) provide a similar account of the mind as a classification process.

that the discovery process of the market performs a cognitive function similar to that performed by the mind.

For Hayek, the knowledge problem consists of both a classification problem – "what are goods", and a calculation problem – "how scarce or valuable they are" (1978a:181). Market participants classify what are goods based on application of their own particular knowledge and preferences guided by the abstract rules of property and contract. Monetary calculation – through prices and profit – superimposes a further ordering of goods according to their relative value in competing uses. A specific price embodies the abstract result of the process of classification in a form usable by market participants. Market participants do not need to know the reasons why something is more or less valued, only that it is.

Hayek's description of prices as "further superimposed means of classifying" raises the question: superimposed over what? It suggests that the classification process of the market takes place at many levels in addition to price. Hayek, for example, (1988:15) points out the important role of language in classifying different kinds of objects, and the important role of custom, morality, and law in classifying different kinds of actions. Classifications capture general knowledge applicable to certain kinds of circumstances. Market participants can make use of these classifications in pursuit of their own particular aims. As Hayek (1988:15) argues, "Ordering in the sense of classifying objects and events is a way of actively rearranging them to produce desired results."

We classify objects, Hayek tells us, according to the purposes they serve. We recognize objects not based on their particular physical characteristics, but by their intended purpose. As he explains (1952a:44), "People behave in the same manner toward things, not because these things are identical in a physical sense, but because they have learned to classify them as belonging to the same group, because they can put them to the same use or expect from them what to the people concerned is an equivalent effect." Hayek (1952a:44-45) illustrates this with the example of a tool:

> This is best shown by an example for which we can take almost any object of human action. Take the concept of a "tool" or an "instrument," or of any particular tool such as a hammer or a barometer. It is easily seen that these concepts cannot be interpreted to refer to "objective facts," that is, to things irrespective of what people think about them.... If the reader will attempt a definition he will soon find that he cannot give one without using some term such as "suitable for" or "intended for" or some other expression referring to the use for which it is designed by somebody. And a definition which is to comprise all instances of the class will not contain any reference to its substance, or shape, or other physical attribute. An ordinary hammer and a steamhammer, or an aneroid barometer and a mercury barometer, have nothing in common except for the purpose for which men think they can be used.

By classifying tools and other objects by their purposes, we are not only abstracting from the physical details of a particular tool, but also from the particular use to which it is put. We classify an object by what type of purpose it serves, not by a particular purpose. Tools are types of solutions to recurring kinds of problems. They can be viewed as multipurpose adaptations to general circumstances. Abstract rules, Hayek argues, can be understood as a kind of multipurpose tool. Abstract rules enable us to learn from the experience of others "about effects expected from different types of actions" (Hayek, 1988:15). He (1976:21) explains:

> Like all general purpose tools, rules serve because they have become adapted to the solution of recurring problem situations and thereby help to make the members of the society in which they prevail more effective in the pursuit of their aims. Like a knife or a hammer they have been shaped not with a particular purpose or view but because in this form rather than some other form they have proved serviceable in a great variety of situations. They have not been constructed to meet foreseen particular needs but have been selected in a process of evolution. The knowledge which has given them their shape is not knowledge of particular future effects but knowledge of the recurrence of certain problem situations or tasks ….

Classifications provide knowledge that others can use. They abstract from detailed particular circumstances and purposes to provide reusable knowledge of general circumstances that can be applied to multiple purposes. They enable individuals to economize on knowledge by substituting abstract knowledge of the class for specific knowledge of the instances. Hayek's work suggests that these classifications exist at multiple layers and adjust continuously to changing circumstances and accumulated experience. This multi-layered view of the knowledge-generating properties of the market order stands in contrast however, to the stark distinction Hayek seems to draw between the roles that abstract rules and concrete purposes play in creating that order.

Between Abstract Rules and Concrete Purposes

Hayek argues that the emergence of "the extended order of cooperation"(1988) rests on the gradual change from a society organized around the common pursuit of a limited number of concrete purposes to a society organized around adherence to a common set of abstract rules of conduct that coordinate a countless number of diverse purposes.[7] Rules of conduct are abstract in the sense that they abstract from the concrete purposes and circumstances. They abstract from particular circumstances of time and place to capture certain general circumstances that have proven useful for the pursuit of many diverse plans. Abstract rules are purpose independent: they serve no particular purpose, but improve the chances for success of a large number of different purposes.

Actions based on abstract rules generate an abstract order. Abstract orders are formed spontaneously, not by deliberate arrangement. They are based on abstract relations serving multiple purposes, not concrete relations serving a single purpose. By serving multiple purposes, the complexity of abstract orders is "not limited to what a human mind can master" (1973:38). The abstract order exists independently of any particular member; the order persists through the abstract relations that define it. As Hayek explains, abstract orders:

> consist of a system of abstract relations between elements which are also defined only by abstract properties…The significance of the abstract character of such orders rests on the fact that they may persist while all the particular elements they comprise, and even the number of such elements, change. All that is necessary to preserve such an abstract order is that a certain structure of relationships be maintained, or that elements of a certain kind (but variable in number) continue to be related in a certain manner. (1973:39)

Abstract rules enhance the ability of individuals to predict the actions of others. They increase predictability by delimiting protected domains of actions. Domains of action provide a place where individuals can plan based on their particular knowledge. They provide a space where individuals can make use of their resources free from interference by others. Abstract rules must do more, however, than just prevent plan interference by delimiting protected domains of actions. They must provide us with a way to connect these separate domains into cooperative relationships. Hayek (1973:99) explains:

> What is required if the separate actions of the individuals are to result in an overall order is that they not only do not unnecessarily interfere with one another, but also that in those respects in which the success of the action of the individuals depends on some matching action by others, there will be at least a good chance that this correspondence will actually occur. But all rules can achieve in this respect is to make it easier for people … to form that match; abstract rules cannot actually secure that this will always happen.

While Hayek highlights the importance of matching complementary actions, he provides us with little guidance on how the matching process takes place. His distinction between purpose-independent rules and purposeful action suggests that coordination occurs at the level of concrete purposes and plans. The attempt to match concrete plans directly with other concrete plans, however, leads to plan failure as circumstances

[7] Hayek (1988:12) explains the nature of abstract rules: "What are chiefly responsible for having generated this extraordinary order, and the existence of mankind in its present size and structure, are those rules of human conduct that gradually evolved (especially those dealing with several property, honesty, contract, exchange, trade, competition, gain, and privacy). These rules are handed down by tradition, teaching, and imitation, rather than by instinct, and largely consist of prohibitions ('shall not's') that designate adjustable domains for individual decisions."

change. In the face of change, people must constantly adapt their plans in unforeseeable ways. Concrete plans are highly brittle in the face of such changes. All plans, however, need not share the same level of concreteness; plans may differ in their degree of specificity and in their reference to particulars. Abstract plans, that encompass a range of specific purposes, are likely to be more robust in the face of changing circumstances. By capturing the abstract aspects of many individual plans and embodying them in market institutions, market participants can enhance their ability to coordinate their plans through time. The matching process occurs not through direct matching of concrete plans but through matching of abstract plans embodied in institutions.

The problem with Hayek's treatment of abstract rules, one might say, is that it is too abstract. He does not explain how these abstract rules are specified into concrete actions that coordinate plans. While he highlights the importance of judge-made law in delimiting domains of action, he fails to provide an account of how market participants through their everyday actions convert abstract rules into institutions that facilitate the matching of complementary activities. The boundaries defined by abstract rules must do more than separate activities where they conflict. They must also combine activities where they complement. Hayek's account leaves a large gap in how this occurs; it leaves a gap between abstract rules and concrete purposes. What is missing is the role that market institutions play in filling the gap by classifying useful knowledge and facilitating the matching process required for its reuse.[8]

III. Lachmann and Abstraction Boundaries

Secondary Institutions as Abstraction Boundaries

Ludwig Lachmann, in his 1971 book, *The Legacy of Max Weber*, discusses the role that market institutions play in coordinating plans. Like Hayek, he highlights the fundamental role played by property, contract, and the rule of law in shaping the market order. Unlike Hayek, he moves beyond abstract rules to discuss the specific institutions of the market order, what he calls secondary institutions. Secondary institutions "gradually evolve as a result of market processes and other forms of spontaneous individual action" (1971:81). They are the familiar institutions of day-to-day life: the restaurants, the banks, the post offices, the stock exchanges, etc.[9]

These concrete institutions embody abstract plans. They fill the gap between purpose-independent abstract rules and specific concrete purposes. A plan, Lachmann tell us, (1971:33) "is but a generalization of purpose;" it abstracts the multiple purposes, means, and obstacles pursued by an individual into a comprehensive framework for action. Similarly, a secondary institution is but a generalization of plan, it abstracts from the multiple plans of various individuals to provide a "common signpost" for coordination. As Lachmann (1971:50) explains, "If the plan is a mental scheme in which the conditions of action are coordinated, we may regard institutions as it were, as orientation schemes of the second order, to which planners orient their plans as actors orient their actions to a plan."

Secondary institutions serve as means of orientation that help coordinate action in society. By orienting our plans towards these institutions, we make use of a vast amount of knowledge without the need to acquire it in detail. As Lachmann (1971:50) explains:

> Whether we post a letter, wait for a train, or draw a cheque, our action is in each case orientated towards a complex network of human action of which we know enough to make it serve our ends, though we may know next to nothing about the internal working order of these institutions. We know of course that such an internal working-order exists, but in our everyday life take no interest whatever in its details. We know very well that the Post Office works according to a general plan,

[8] Vaughn (1999) recognizes a similar gap in Hayek's account of the market order.

[9] Lachmann (1971) seems to be groping for the right terminology: he distinguishes first between external and internal institutions, only to later switch to a distinction between fundamental and secondary institutions. Our concern here is not with the adequacy of his terminology, but with the similarity between fundamental institutions and abstract rules, and the similarity between secondary institutions and abstraction boundaries. See Horwitz (1998) for a criticism of Lachmann's distinction between external and internal institutions.

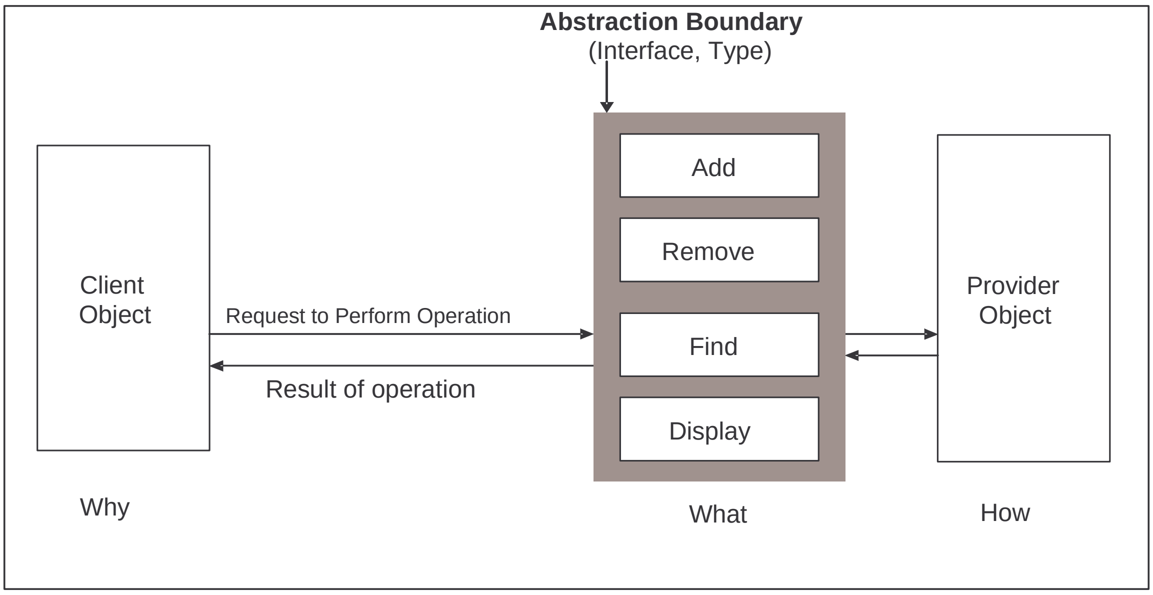

Figure 1: Abstraction Boundary for Objects (adapted from Carrano & Prichard, 2001)

> but such knowledge as we have about it is usually quite irrelevant to the achievement of our purpose in posting a letter. Only a few aspects of this general plan, perhaps the times of collection and delivery of mail, need be of concern to us.

By drawing our attention to the aspects of an institution with which one interacts, Lachmann is presenting what software designers refer to as an abstraction boundary – the interface between two actors. The abstraction boundary captures the abstract plan common to the various actors. The actors coordinate their actions to the boundary, not to the detailed plans of the other actors. As Lachmann notes, the postal customers need concern themselves with only "a few aspects of this general plan," for example, "the times of collection and delivery." These few aspects form the abstraction boundary. The post office communicates these aspects to the postal customers, who rely on these details in forming their own particular plans.

The abstraction boundary defines a division of knowledge. Instead of each of us having to deliver our own letters and packages, we can rely on the specialized services of a post office. We no longer deliver our own mail because some entrepreneur, forgotten to history, recognized a profitable division of labor and created an abstraction boundary.[10] The entrepreneur 1) recognized an abstraction – the commonalities in the separate plans of people delivering their own letters, 2) drew a boundary around the abstraction defining the service of mail delivery, and 3) encapsulated the implementation of these services behind the abstraction boundary while convincing others that it would serve their purposes to use this new service.[11] The post

[10] We oversimplify to make a point, but this can be misleading. By presenting the creation of the abstraction boundary as the result of a single entrepreneurial act, we are ignoring the interaction between providers, customers, and other interests that shape the evolution of most abstraction boundaries. This is particularly true of the gradual development of the postal system into its current familiar shape. As one commentator notes, "their very familiarity … and the consequent unthinking acceptance of them make it important to emphasize that until recently postal systems lacked many of these features. Although the basic need for a system to exchange written communications has been felt by all human societies and has been met in many ways, the evolution of varied postal systems adopted by different societies through the centuries into the basically similar pattern of today's state monopoly service has been a long and difficult process." (Encyclopedia Britannica, 2003)

[11] Liskov (2002:xvii) identifies three steps involved in creating an abstraction boundary: 1) *abstraction* - the programmer abstracts a common solution to a recurring problem, 2) *specification* - the programmer defines a boundary that captures the meaning of the abstraction in terms of the services it provides potential

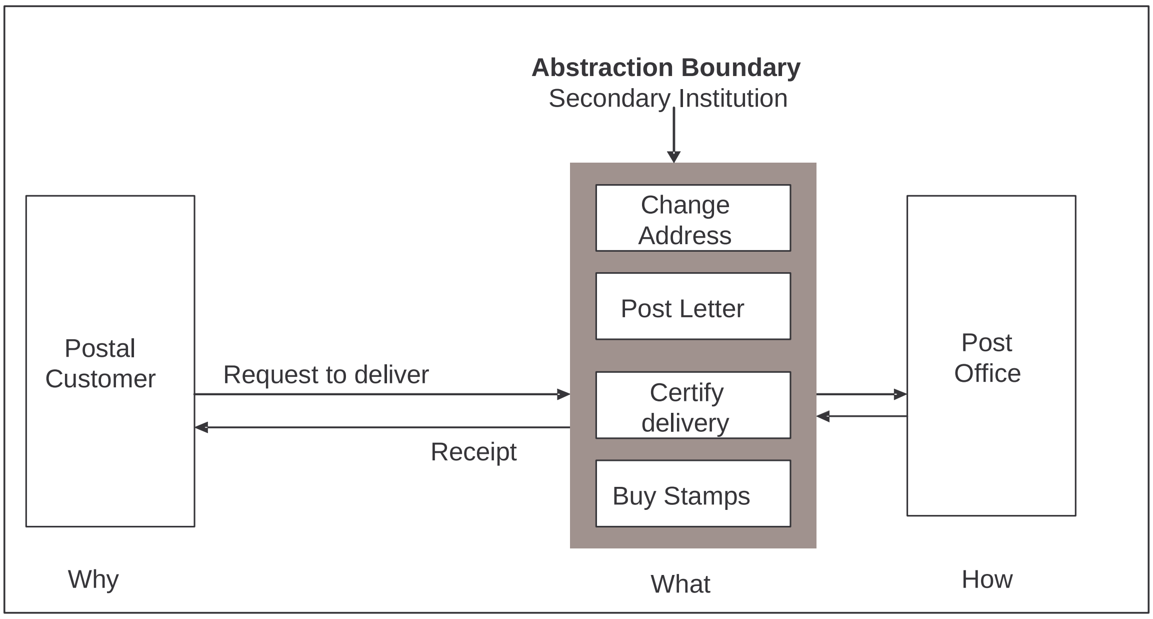

Figure 2. Abstraction Boundary for Post Office.

office, by reusing its specialized knowledge across many instances of mail delivery, is able to capture economies of specialization. Postal customers benefit when the cost of their using a post office to deliver a letter is less than the cost of their delivering it themselves.

Institutions as abstraction boundaries are not the same as organizations. An abstraction boundary is the interface that lies between organizations (or people), such as that between postal customers and the postal system.[12] The post office as an organization differs from post office as an abstraction boundary. The post office as an abstraction boundary represents a particular kind of provider-client relationship. The post office as an organization implements the services defined by the abstraction boundary. The boundary defines how the post office as an organization relates to other elements of the more comprehensive order of which it is part. A concrete post office instantiates the abstract interface and implements the delivery services. It is part of a larger postal system that encompasses additional entities such as delivery vehicles, warehouses, sorting machines, and tracking facilities. From the point of view of the postal customer, however, the implementation detail is hidden behind the abstraction boundary of the post office.

An abstraction boundary separates a program into distinct objects; it forms the interface between the client making a request and the provider fulfilling the request (see figure 1). It describes what the object does – the services it provides – by providing a vocabulary for the interaction between an object and its clients.[13] It also defines what type of object a particular object is: objects of the same type share the same interface. The abstraction boundary embodies and reifies what it means to be an object of a particular type.

---

users, and 3) *encapsulation* – the programmer hides the details of how the services are to be provided behind the abstraction boundary.

[12] The focus in on what happens between entities, not the entities themselves. Secondary institutions, Lachmann (1971:81) tells us, “accumulate in the interstices.” Similarly, object-oriented pioneer Alan Kay (1998) writes, “The Japanese have a small word – ma – for ‘that which is in between’ – perhaps the nearest English equivalent is ‘interstices.’ The key in making great and growable systems is much more to design how its modules communicate rather than what their internal properties and behaviors should be.”

[13] Wirfs-Brock and McKean (2002:12) explain: “An interface describes the vocabulary used in the dialog between an object and its customers: ‘Shine my shoes. Give me my shoes. That’ll be five bucks please. Here is your receipt.’ The interface advertises the services and explains how to ask for them.”

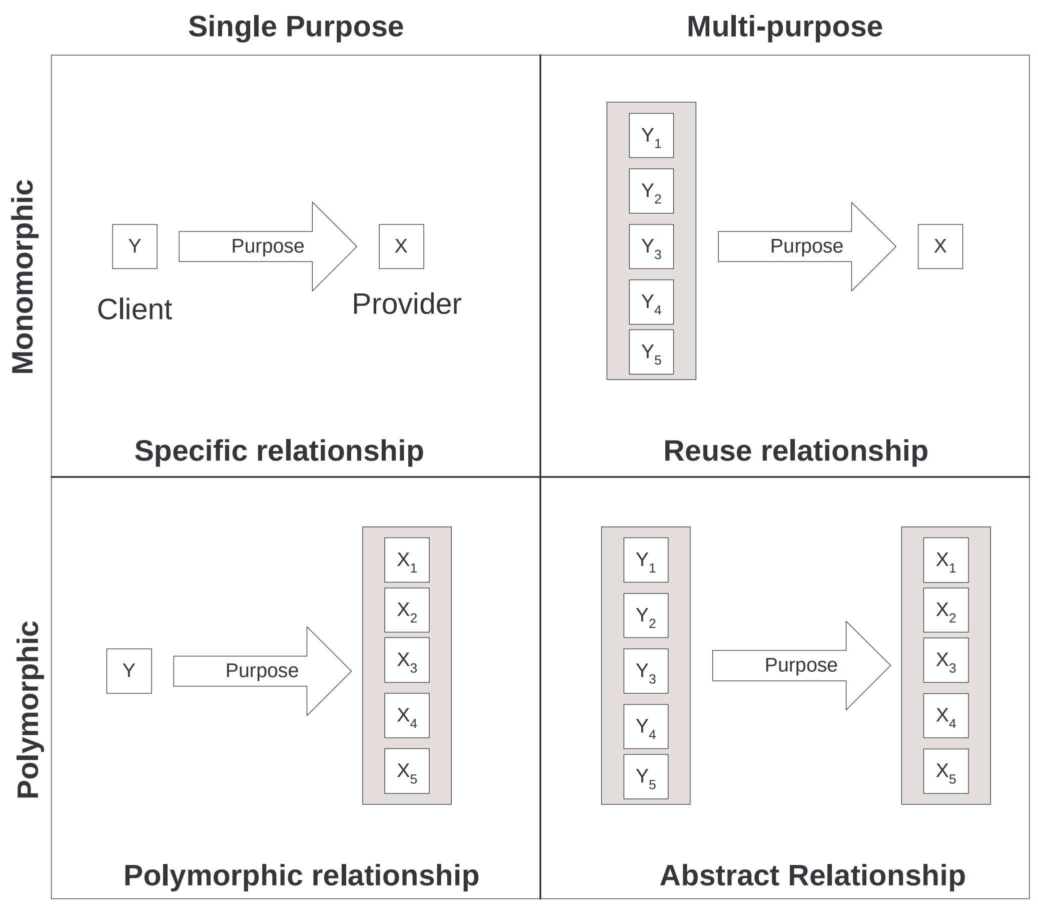

Figure 3. From Specific to Abstract Relationships

In so doing, it defines the kinds of behavior that can be expected from objects of that type; while also distinguishing this type of object from other types of objects.[14]

Lachmann's example of posting a letter follows the same pattern (see figure 2). The abstraction boundary of the post office separates the relationship into distinct actors; it forms the interface between a postal customer requesting delivery of a letter and a specific post office that fulfills that request. The boundary describes the vocabulary used in the dialogue between a post office and its customers: post my letter, change my address, certify delivery, etc. The abstraction boundary also defines what it means for a post office to be an instance of the type post office. The abstraction boundary represents the concept of what it means to be a post office. It defines what services can be expected from a post office, while distinguishing this type of service from other types of services.

Abstract Plans Serving Abstract Purposes

The abstraction boundary enables what software designers refer to as separation of concerns; it separates:

a) the *why* of mailing a letter (the concern of the postal customer) from,
b) *what* it means to mail a letter (the mutual concern of the parties) from,

[14] Type and interface are used by programmers as synonyms for abstraction boundary. They can be seen as emphasizing two related notions of abstraction: interface refers to what an object *does* abstracted from how it does it; type refers to what an object *is* – what kind of object it is – abstracted from concrete instances of that type. The two are related because we classify objects based on what services they provide; we classify what an object is, by what the object does.

c) *how* a letter gets delivered (the concern of the postal system).

The post office does not need to know that Alice wants to invite Bob to a birthday party, or that Carol wants to send her monthly payment to the power company. The boundary abstracts from the details of the customers' plans to focus on the common attributes of transporting a sealed envelope to a specified address. Similarly, the postal customer does not need to know how the commitment to deliver a letter will be implemented. They do not need to know that the mail will be delivered using a tractor-trailer truck if the address is within 500 miles, or delivered by airplane if the address is more than 500 miles away. Both parties, however, need to know the "what," i.e., the specific steps needed to post the letter. This is the boundary.

By abstracting on the one side from the specific purpose of a request (the why), and on the other side from the specific way the requested action is implemented (the how), the abstraction boundary enables a large degree of flexibility on both sides. The post office can accommodate a large range of specific reasons for mailing a letter, including ones not yet known. At the same time, the post office is free to try to discover better ways of delivering letters. The entities obtain this flexibility because of the stability of the boundary. The abstraction boundary by capturing the mutual concerns of both parties (the what) while abstracting from their particular purposes, is able to remain relatively stable in the face of change. The stability and familiarity of the category of a post office makes it easy, for example, for someone visiting a new town to locate a new instance of the type post office and mail a post card home even without any prior knowledge of that particular post office.

The creation of an abstraction boundary converts a specific relationship into an abstract relationship (see figure 3). A specific relationship directly links a particular solution (a how) to a particular purpose (a why). An abstract relationship links a type of solution (multiple hows) to a type of purpose (multiple whys) through an abstraction boundary (a common what). A relationship is abstract when the common services defined in the boundary can be used for multiple purposes, and provided by multiple providers. The abstraction boundary limits what the parties need to know about each other; it allows the provider to treat as common a diverse set of clients, and the client to treat as substitutes a diverse set of providers.

An abstract relationship combines both *reusability* and *polymorphism*. Reusability occurs when a single solution can be reused for multiple purposes. Polymorphism occurs when multiple providers can serve a single purpose. Reusability generalizes across the purposes of the various clients (the particular whys) to focus on the aspects they share in common. Polymorphism generalizes across the implementations of the various providers (the particular hows) to focus on the common services that they provide. Polymorphism enables clients to substitute one provider for another so long as they provide the same type of service; it shields the clients not only from how the services are provided, but also from who provides the service.

Lachmann's post office example highlights the advantages of moving from specific to abstract relationships. Organizing around specific relationships is neither economical nor adaptable. We would not expect to see mail delivery organized around specific relationships on a large scale: a post office specializing in delivering Alice's birthday cards to her grandmother, or a post office specializing in delivering Bob's payment to the gas company. By abstracting from the various purposes, a provider can reuse the common mail delivery solution for multiple purposes. By abstracting from the particular implementations, clients can substitute different providers of those services, based on their more exact requirements. As Wirfs-Brock and McKean (2002:3) explain, "Objects that play the same role can be interchanged. For example, there are several providers that can deliver letters and packages: DHL, FedEx, UPS, Post, Airborne. They all have the same purpose, if not the same way of carrying out their business. You choose from among them according to the requirements that you have for delivery. Is it one-day, book rate, valuable, heavy, flammable? You pick the mail carrier that meets your requirements." An abstract relationship by abstracting on both sides of the boundary brings the benefits of both economies of reuse and adaptability.

Abstraction boundaries also play an important cognitive role by classifying various activities into types of activities, and by making these classifications accessible to others. The boundary takes an abstraction (the recognition of the common attributes of a recurring pattern of action) and makes it concrete. It reifies the abstraction by articulating it in the design of the interface – in its contracts, in its store designs, in its customer-facing procedures, and in the design of its products. The abstraction boundary is the invention of the possibility that two entities can interact in a particular way; without the boundary the abstraction is too intangible to be useful.

In the market context, abstraction boundaries often take the form of transaction boundaries. The abstraction boundary increases the potential for gains from trade by reducing the costs for people seeking a particular kind of trade. An abstraction boundary increases the chances that a particular kind of gains from trade will happen by limiting the types of transactions that can occur through the boundary. There are all sorts of potential gains from trade that may exist between a particular postal customer and a particular postal worker. For example, the postal customer may be in the market for a used car, and the postal worker may be looking to sell his old one. However, this kind of transaction is not permitted through the boundary of the post office. The post office enables the realization of some of the potential gains from trade between postal customers and the postal workers, while ignoring a vast number of other potential gains from trade. By doing so, it increases the chances that many more postal customers and post offices will realize the gains from posting a letter. The abstraction boundary serves as a meeting place for people seeking a particular kind of gain from trade; it provides a familiar source to which people seeking a type of transaction can orient their plans.[15]

Institutional Coherence and Change

Lachmann (1971:51) raises the issue of the coherence and flexibility of the institutional order: "If institutions are to serve us as firm points of orientation," he writes, "their position in the social firmament must be fixed. On the other hand, it is hardly possible to imagine that banks, railways, and other institutions are totally exempt from change." How are we to allow institutions to change yet still fulfill their role as orientation points for coordinating plans? How can we adapt abstraction boundaries to support plan coordination through time? Or as he (1971:90) puts it, "How is the need for coherence and permanence reconciled with that for flexibility in the real world?"

Lachmann (1971:90-91) identifies four "devices" that serve to balance flexibility and coherence: 1) relatively immutable fundamental institutions; 2) frequently mutable secondary institutions; 3) widening of existing institutions; and 4) the prohibition of certain kinds of change that threaten the stability of the overall order. The framework of abstraction boundaries can help us make sense of Lachmann's "devices".

Lachmann's first device is relatively stable fundamental institutions. Fundamental institutions represent what Hayek refers to as abstract rules of conduct (e.g. rules of property and contract). These abstract rules provide the fundamental framework for the creation of secondary institutions that serve a wide range of different purposes. They define the rules by which market participants can divide knowledge, and the rules by which they can combine their separate actions with those of other market participants. The stability of the framework enables flexibility in creating different types of secondary institutions.

In software, the abstract rules or "constitutional constraints" are typically provided and enforced by the operating system or programming language. A programming language sets the rules that determine what is to be considered an object and how those objects are allowed to interact. Object-oriented programming languages enforce a minimal number of abstract rules (e.g. rules of encapsulation and message passing), which enable all computational entities to be treated as first class objects. By providing abstract rules rather than just collections of abstractions, object-oriented languages enable programmers to create new abstractions to meet their particular situation. Programmers are able to create a vast number of different types of objects that can easily interact with other types of objects, including types not known to the language designers. Programmers can respond to new problems by creating new types of solutions.[16]

Lachmann's second device, the need for relatively mutable secondary institutions, is more problematic. The real issue for plan coordination is not the mutability of the secondary institutions per se, but what kinds of mutability they enable. Lachmann (1979:253) seems to recognize this when he writes, "One is tempted to think of the institutional order as an array of hinges: the institutions within each hinge can move a good

[15] Abstraction boundaries serve as focal points for a type of transaction. (Schelling, 1960)

[16] Anderson and Hill (1988:210) write, "a constitution that protects property rights is one that makes no attempt at predicting the future… Instead of attempting to specify solutions to presently unknown problems, it puts in place an institutional framework that allows individuals to take actions in response to changes in the economic and social environment."

deal, if within limits, but the hinges themselves cannot." Abstraction boundaries play the role of the hinges, enabling flexibility on both sides of the boundary. Relatively stable abstraction boundaries permit a wide range of changes in purposes and implementations through the separation of concerns. Changes on either side of the boundary will not upset plans so long as the interface does not change. Relatively stable boundaries also support greater flexibility through polymorphism. Polymorphism opens up possibilities for competition in the provision of the services specified in the boundary. It enables what Hayek (1945b:83) describes as, "The continuous flow of goods and services." It allows for, "constant and deliberate adjustments, by new dispositions made every day in light of circumstances not known the day before, by B stepping in at once when A fails to deliver."

Lachmann's third device – the "widening of existing institutions" – can be illustrated by the concept of subtyping. A subtype extends the scope of an existing abstraction boundary by adding additional behaviors. Since a subtype must fulfill all of the obligations defined by its supertype, it can be substituted in place of its supertype without upsetting the expectations of existing clients for that type. This enables what Lachmann calls the widening of institutions.[17] As Lachmann (1971:91) puts it, the widening of institutions "takes the form of meeting a situation requiring change not by the creation of a new institution, nor by replacing an old by a new, but by "widening" an existing institution in such a way that it can serve new interests without upsetting the plans which have thus far made use of it."

Lachmann's fourth device is less a device than a warning that some kinds of change may lead to a breakdown of the institutional order despite built in sources of institutional flexibility. By orienting their plans to abstraction boundaries, market participants gain greater reliability of their expectations, while still permitting a broad scope for change. Changes to the rules that respect boundaries will tend to support plan coordination; changes to rules that break boundaries will tend to upset plans.[18] Yet in a changing world, existing abstraction boundaries may turn out to be ill-adapted. Balancing the need to protect expectations based on existing boundaries with the need to adapt those boundaries to meet unforeseen circumstances becomes an important challenge for both programmers and society.

IV. Programmers and Plan Coordination

Objects as Property

As Lachmann's post office example suggests, programming can be usefully analyzed in terms of the Hayekian notion of plan coordination. Programs perform actions: plans are specified by the programmer at design-time; plans are executed by the computer at run-time.[19] In order to cope with the intellectual complexity of large programs, programmers divide a program into separate modules, each embodying its own plan of action. The challenge facing programmers can therefore be seen as one of plan coordination; programmers must coordinate the plans of the various modules to ensure the harmonious working of the overall system.

[17] Lachmann's widening is sometimes referred to as narrowing by software engineers. This apparent contradiction results from different perspectives. For software engineers, the subtype is narrower than the supertype because it is less general: a smaller set of providers provide the more specific functionality of the subtype. For Lachmann the subtype is wider since the new services that are provided by the subtype serve a broader range of client purposes.

[18] Lachmann (1971:70, 1978:252) argues that this is especially true for intertemporal plans such as long-term loan contracts. But as Lachmann (1971:70) argues, "it is impossible to separate the legal provisions governing loans from the rest of the legal order. Every concrete business transaction involves such a large number of legal rules that it would be impossible to enumerate them all and, hence, separate them from the rest of the legal order."

[19] Run-time is what occurs when a program is run on a specific computer. Design-time is what occurs when a programmer designs and codes the program. A programmer writes source code (readable by the programmer) that embodies the design of the program. The source code is compiled into machine-readable code that is capable of being run on a particular type of computer. The user executes this code when running the program.

Most software bugs result from plan interference. Breakdown occurs because of coordination failure between parts of the program. Actions taken by one module destroy the assumptions used by another module in planning its actions. Hidden dependencies between modules render assumptions unstable and plans mutually incompatible. For example with procedural programming, the plans of two subroutines may conflict because they unknowingly share the same data. Subroutine A performs a calculation and stores the resulting data for future use. Meanwhile, subroutine B performs its operation overwriting A's data. A retrieves the data for further processing, not realizing that it has been modified by B – leading to unexpected results and potential system failure.

The history of software development can be seen as a continuing development of techniques – embodied in programming languages and development methods – to cope with the challenges of plan coordination through time.[20] Centrally planned approaches to software development broke down in the face of increasing complexity, forcing programmers to introduce a division of labor into their programs. Attempts to divide labor according to the principles of procedural programming enabled programmers to build more complex programs, but were subject to massive plan failure when program requirements changed. Procedural programming, which divided the tasks into subroutines while keeping program data in common, could not easily adapt to changing circumstances. Object-oriented programming improved on procedural programming by dividing both the tasks and the data; it combined the data with the procedures that act on it. By eliminating the dependencies caused by shared data, it provided greater flexibility in adapting to change.

Object-oriented programming divides a program into separate objects that are each instances of a particular type. An object combines a set of related behaviors (its methods) with its private data (its state). The object's type is defined by its set of methods (its interface). An object's state – its particular circumstances of time and place – is said to be encapsulated because only its methods have the ability to access or modify that state. Other objects cannot directly access the internal state of another object; they must send a request to the object asking it to perform a particular action.[21]

The move from procedural programming to object-oriented programming can be seen as a move from concrete plans to abstract plans. Objects do not coordinate their actions with the specific plans of other objects; they coordinate their actions to the abstract aspects of those plans. An object's interface abstracts from a specific plan to a kind of plan. Object-oriented programming, or more generally the move to encapsulation, message passing, and polymorphism, is essentially trying to move from plan coordination to pattern coordination, where what is meant by patterns is abstracted plans. It does this by re-discovering the virtues of property rights and contract.[22]

Hayek's explanation of the primary virtue of property rights for organizing large scale economic activity parallels the rationale for encapsulation in object-oriented systems: to provide a domain (an object's encapsulation boundary) in which an agent (the object) can execute plans (the object's methods)

---

[20] Software design provides a language for thinking about plan coordination. It provides a set of concepts for reasoning about the problem of dividing and recombining knowledge. This language is much richer than we are able to show here. It also exists in two important senses: 1) as a set of concepts for thinking about these distinctions, and 2) as programming language constructs for implementing these distinctions. As an engineering discipline, software design concepts must pass the empirical test of building better programs. For good overviews of these distinctions see Meyer (1998), Gamma et al (1995), and Liskov (2001). E (Miller et al, 2000, www.erights.org) is an open source programming language that takes a rights-based approach to problems of plan coordination in programming.

[21] In addition to types and objects, object-oriented programming has the concept of class. A class is a template for creating objects; it implements an object's methods, but not its state. For simplicity, we ignore this distinction in this paper.

[22] The similarity between object and market concepts has been widely recognized by software designers (Miller and Drexler 1988b, Cox 1996, Meyer 1997). For example, Bertrand Meyer (1997:127) writes, "like an economist of the most passionate supply-side, invisible-hand, let-the-free-market decide school, we are interested in individual agents not so much for what they are internally as for what they have to offer to each other ... The economic analogy will indeed accompany us throughout this presentation; the agents – the software modules – are called suppliers and clients; the protocols will be called contracts, and much of object-oriented design is indeed *Design by Contract*."

that use resources (an object's private state), where the proper functioning of these plans depends on these resources not being used simultaneously by conflicting plans. By dividing up the resources of society (the state of a computational system) into separately owned chunks (private object states), we enable a massive number of plans to make use of a massive number of resources without needing to resolve a massive number of conflicting assumptions.

Collaboration through Message Passing

Similarly, the explanation of the primary virtue of freedom of contract in coordinating large scale economic activity parallels the rationale for message passing in object-oriented systems: to enable cooperation between actors (interaction between objects) regardless of their status through a system of offers and acceptances (message passing) that ensures respect for property rights (an object's encapsulation boundary) by requiring both freedom to contract – the actors may consent to particular interactions, and freedom from contract - the consent of the actor is required to perform an action. The resulting contract between two agents (the interface) defines what actions they can rely on and what actions they must perform.

Message passing distributes the responsibility for performing the work of an application across many objects. Each object is responsible for performing services or knowing information useful to others.[23] Objects must collaborate to fulfill their responsibilities. They collaborate by sending messages requesting other objects to perform an action. The object receiving the message is responsible for determining how to respond to the message. The object's interface defines the set of messages to which it will respond. The interface represents a contract between the object making requests and the object providing the requested service.[24] As Wirfs-Brock and Wilkerson (1989:72) explain, "the ways in which the client can interact with the server are described by a contract: a list of the requests that can be made of the server by the client. Both must fulfill the contract: the client by making only those requests it specifies, and the server by responding only to those requests."

Object-oriented programming reduces plan interference by encapsulating the computational state of a system into separate objects. It increases plan coordination by combining objects through message passing. Message passing increases the adaptability of the system by basing the interaction between specific objects on the contract – the abstract relationship between types of objects. Collaboration between concrete plans occurs through the coordination of abstract plans.

The run-time composition of objects by message passing stands in contrast to the design-time organization of abstraction boundaries into a type hierarchy. The type hierarchy defines a static relationship between types, organizing them from more general supertypes to more specific subtypes. The run-time interaction of objects consists of the dynamically formed relationships among specific objects.[25] Programmers can enhance the ability of the program to adapt to unforeseen change by "programming to an interface and not an implementation" (Gamma et al, 1995:18). By programming to an interface, the programmer orients the behavior of an object to the abstract plans of other objects, not to the concrete plans of specific objects. Objects with different concrete plans can be substituted for one another at run-time without requiring the programmer to rewrite and recompile the program. Market participants, similarly, gain adaptability by *planning* to an interface and not an implementation. Programmers and market participants benefit from orienting their plans to stable boundaries, not frequently changing concrete plans.

Programmers create the abstraction boundaries of the program at design-time. The major creative act of programming is to look at some large set of computational activity and to discover boundaries in it – to find distinctions that enable programmers to see commonalities or patterns. The distinction reveals

[23] Wirfs-Brock and Wilkerson (1989) and Wirfs-Brock and McKean (2002) present the responsibility-driven design approach to object-oriented design.

[24] Contract in this sense is another name for abstraction boundary, a standardized contract for a type of interaction. Meyer (1998) provides a narrower definition of object contracts which emphasizes the terms and conditions (assertions) required for an object to respond correctly to a particular request.

[25] Gamma et al. (1995:22) caution, "Trying to understand one from the other is like trying to understand the dynamism of living ecosystems from the static taxonomy of plants and animals and vice versa."

commonalities – among providers on one side and clients on the other – that were previously unnoticed. Once the commonality or pattern is recognized, programmers attempt to find a way to package it, so that they can embody the usefulness of that part of the computational space into a module that other programmers can use.

The perspective of programming as plan coordination also sheds light on the role of the entrepreneur. It suggests that the role of the entrepreneur is to create abstraction boundaries. Programming to abstraction boundaries, as we have seen, adds flexibility at run-time. The boundaries, however, remained fixed; interaction occurs between objects with fixed boundaries. The abstract plans are fixed by the programmer in advance; they do not change during run-time.

Several software designers have proposed moving to market-oriented programming as a way to increase run-time flexibility.[26] Market-oriented programming introduces price signals into a program. Price signals allow objects to dynamically trade-off the scarce computational resources used in carrying out their plans. Objects are alert to opportunities presented by changes in price, and thus able to dynamically shift from one plan to another. All of the adaptation of plans, however, must still take place within the divisions created by the programmer; the system still does not have creativity in the sense of being able to create new divisions. New abstractions are created outside of the system. It is the programmers, the human beings, who create new abstraction boundaries; the evenly rotating carrying out of plans is something we get the computer to do.[27]

V. Lavoie and the Re-categorization of the Market Order

Negotiated Categories

By recognizing the role of the software designer in creating abstraction boundaries, we highlight an important difference between abstraction boundaries in software and those in markets. In markets, there is no programmer standing outside the economy ready to change the boundaries; there is no exogenous source of change. Changes to abstraction boundaries occur as part of the ongoing market process. Market participants are engaged in the process of drawing and redrawing abstraction boundaries. Abstraction boundaries emerge as negotiated categories that help orient plans.

The software design process is highly collaborative; software designers engage in a process of dialogue – with each other and with prospective users of the system – in order to discover abstraction boundaries.[28] The importance of dialogue in creating abstraction boundaries should not be surprising. As Hayek (1989:15) tells us, “We learn to classify through language, with which we not merely label known kinds of objects but specify what we are to regard as objects or events of the same or different kinds.”

Don Lavoie (1990b, 2003), in particular, has emphasized the creative role of dialogue in markets. He draws on the work of Gadamer (1989) to emphasize the emergence of new meaning from the spontaneous ordering process of dialogue – a process writ large in the spontaneous ordering process of the market. Lavoie (2003:12) explains, “Markets are an extension of language, and they work through concrete acts of articulation, whether oral or written or electronic. You could say that markets simply add whole new kinds of texts, prices, profits, contracts, advertising, etc., to the conversation.” He (2003:16) goes on to urge that, “Among the institutional forms that Austrian economists ought to pay attention to are the articulation processes and the resulting texts through which human actors communicate their meanings.”

[26] See for example, Malone et al. (1988), Miller and Drexler (1988b), and Wellman (1996).

[27] Our distinction bears a resemblance to the one that Buchanan and Vanberg (2002) make between creative and reactive choice. Creative choice creates alternatives from which other individuals choose; reactive choice is choice among the alternatives presented. Abstraction boundaries define what alternatives are available. New alternatives are created by changing the abstraction boundaries. Objects at run-time can react to the alternatives presented to them by the existing abstraction boundaries, but they normally cannot change the boundaries to create new alternatives. Evolutionary programming techniques can be seen as attempts to build creative choice into programs. See Baum (1999) for a market-oriented approach to evolutionary programming inspired by Hayek.

[28] See Baetjer (1999) for a description of the role of dialogue in software development.

We suggest that we ought to pay attention to these institutional forms because they represent the process by which abstraction boundaries evolve and the means by which they are formed. Market articulations are attempts to create shared categories of meaning that enable us to coordinate our actions. Abstraction boundaries, in order to serve as orientation points, must be articulated; they must be made concrete.[29] Articulation in dialogue – by words and by texts – creates a space where shared meanings can form. Articulation in markets – by contracts, by product designs, and by other marketing efforts – creates a space where shared categories can form. These shared categories serve as points of orientation that help coordinate plans. As Peter Lewin (1999:34) writes, "Expectations and plans are, for the most part, fulfilled because of the existence in the social world of shared categories and standards that facilitate the synchronization and coordination of activities."

Lewin illustrates this point by examining the emergence of new products and new categories of products. These categories emerge from attempts to articulate the purpose of new products – how they are similar, how they are different, and how they relate to other products. He (1999:39) writes:

> We cope with the complexity in the world by converging on institutions. Thus once the arrival of new products, made possible by the development of a new technology, has been digested, new categories of classifications tend to be developed, into which these products are grouped. These categories emerge spontaneously out of individual attempts to communicate the attributes of the new products. A good example is the products of the computer industry. A whole range of products exist, whose workings remain a mystery to the vast majority of people, but whose purposes needed to be explained. Laptops evolved into notebooks, micro computers into desktops…. All these shorthands provide the increasingly informed public with a way to tailor their expectations when choosing between products. They enhance predictability by enhancing the interpretability of information. But these relatively stable predictable elements change with time and its no accident that conscious innovation involving product differentiation is often referred to using the phrase 'category killer.'

Shared categories, the abstraction boundaries of the market, emerge from the interaction of entrepreneurs and their potential customers. They emerge from the attempt by entrepreneurs to define what services their products can provide; and their customers' attempt to express what services they desire. Entrepreneurs orient their actions to existing categories: they mimic current categories by imitation; they modify them by differentiation; and they try to create new ones through innovation. Successful categories, those that persist, reflect not just the action of entrepreneurs, but also the responses of potential customers. Potential customers express their requirements through various means, through negotiation, through novel uses of products, through responses to market research, and ultimately through their purchase or failure to purchase the entrepreneur's offering. The resulting categories become embodied in the design of products, in the design of secondary institutions, and in the grouping of product and institutional types into more general categories that help orient people's actions. Shared categories emerge from the market process of dialogue and exchange.[30]

Lavoie (2003:6) argues that "entrepreneurship should be conceptualized as a social process of mutual orientation" – a process that "make[s] it easier for people who see the world differently from one another, to nevertheless make sense of each other, to orient their actions to one another" (2003:8). "Our ability to coordinate with one another," he (2003:2) argues, "may depend on our capabilities for mutual orientation, to see not the things themselves, but to see how one another are seeing things." The focus on mutual

[29] Articulations, while concrete utterances, are also necessarily abstract. Lavoie (1985a:60) explains, "nothing can be completely articulated, since the very process of articulation involves the abstraction from some real features of the entity or process that is being described."

[30] Lavoie (2003:15) writes, "Entrepreneurship is not an individual act of perception … it is an interpretive and communicative process involving efforts of different persons to communicate meaning about the future to one another. It is perhaps misleading to even present this coordinating process in terms of the activity of 'the entrepreneur,' the single individual who is more or less alert to opportunities. What we have here is a coordination process that is taking place among persons who are oriented towards the future."

orientation suggests that we shift our attention to how market participants articulate and share meanings, and how they negotiate shared categories that orient their actions.

In recasting entrepreneurship as mutual orientation, Lavoie generalizes the account of the entrepreneur provided by Lachmann.[31] Lachmann (1956) presents the function of the entrepreneur as one of combining heterogeneous capital goods into a specific production plan which must fit into a complex capital structure. Lachmann emphasizes both the need to specify a concrete use, and the need to fit the particular use into a complex web of complementary capital goods.[32] As we have seen, abstraction boundaries are the key to flexibly combining complementary activities. The process of "specifying" and "fitting" requires both the discovery of an abstraction and the creation of a boundary that allows it to connect to complementary activities. Entrepreneurs must look beyond existing categories to specify new solutions, yet at the same time fit the new solution into a pattern of complementary activities. The entrepreneur oriented to the existing categories reads a situation anew, identifies a problem or opportunity, specifies a solution, and fits the solution into a complementary web of activity so that the plans of others can be easily reoriented to the new categories.

Self-Reorganization of the Market Order

Entrepreneurs and marketers look beyond existing boundaries to identify commonalities currently hidden behind those boundaries. By creating new distinctions (new abstractions), entrepreneurs are identifying both what should be treated as similar and what should be treated as different. They re-categorize existing shared categories in order to capture overlooked commonalities, or to separate overlooked differences. Entrepreneurs create value by creating new distinctions: where two similar things are being treated as different, there are gains to be had from removing the differences; where two different things are being treated as similar; there are gains to be had from removing the similarities.

Israel Kirzner's (1992) theory of entrepreneurship as arbitrage provides an example of one way that entrepreneurs can capture gains from removing differences. The entrepreneur notices that two similar goods are being treated as if they were different; two instances of the same type of good are selling for two different prices. The alert entrepreneur, by buying low and selling high, can profit from removing this difference. The entrepreneur helps re-categorize the undervalued good by bringing it to the attention of those who value it more.

Arbitrage, however, is not the only source of potential gains from noticing commonalities; recognizing commonalities can also lead to economies of scale and scope. Langlois (1999) argues that economies of scale and scope result primarily from the reuse of knowledge.[33] Scale economies arise from abstracting out the common aspects of a solution and reusing them across multiple instances. He gives the example of multiplex cinemas: cinema owners gain economies of scale by reusing the services of ticket windows, lobbies, restrooms, snack bars, and management across multiple individual theaters. The common features of stand-alone cinemas were abstracted out, and packaged for reuse by the theatres in the multiplex.[34]

Gains can also be had by recognizing differences where others see similarities. Fred Smith (2002), for example, when starting Federal Express (now FedEx) recognized that shippers and recipients of small packages differed in the value they placed on the timeliness and reliability of delivery. He recognized that

[31] See especially Lavoie (1991 and 2003). See also Spinoza, Flores and Dreyfus (1997).

[32] Lavoie (2003:8) explains, "Lachmann's work *Capital and Its Structure* depicts the entrepreneur as involved in trying to mesh her plans with those of the ultimate consumers, by in some sense 'fitting' into an evolving structure of capital. This idea of fit is what Lachmann called 'complementarity.' The process of economic production involves the continuous adaptation of heterogeneous parts of the capital structure to one another. It is in this context that he sees the essential role of the entrepreneur as one of figuring out how to specify particular ways of using heterogeneous capital goods that in principle can be used in a multiplicity of alternative ways. Lachmann identifies the key role of the entrepreneur in terms of deciding how to initially select (or later adapt) particular patterns of use of capital goods."

[33] As Langlois (1999) points out, there is a blurry line between economies of scale and scope: scale economies result from the reuse of knowledge across many identical units; scope economies result from the reuse of knowledge across many similar units.

[34] Third-party ticket-providers show that some boundaries can also support polymorphism.

customers who value speed and reliability (for example those who need spare parts to keep expensive capital equipment running) would be willing to pay for guaranteed overnight delivery. Standard approaches to package delivery abstracted from these differences treating them all the same. By looking past the existing boundary, Smith was able to discern differences in why people wanted package delivery services. By identifying a new commonality among these differences (desire for guaranteed overnight deliver), he was able to create a new abstraction boundary defining this new type of service.[35]

Just recognizing the new abstraction, however, was not enough. Federal Express had to create new capital combinations, and fit them into an evolving capital structure. Airplanes were purchased and flight routes were organized into a then novel hub-and-spoke configuration. Air delivery was combined with delivery trucks to provide door-to-door service. New interfaces that defined this new type of service were created and marketed to both shippers and receivers of packages. The creation of the new abstraction boundary of overnight delivery required a reconfiguration of the capital structure involved in providing that service. The change in abstraction boundary generated a new structure of cooperating objects involved in providing guaranteed overnight delivery. By hiding these changes from the customers, the abstraction boundary made it easy for customers to adapt their plans.

As the FedEx example suggests, abstractions boundaries, by virtue of being abstract, provide a source for endogenous change. Abstractions, by emphasizing commonalities and suppressing differences, create opportunities for changing the boundaries. When the differences being hidden are unimportant, there is no incentive to change the boundary; when the differences being hidden start to make a difference, there is an incentive for the boundary to change. An abstraction necessarily hides the diversity of the various instances of the particular type. As long as those differences can be ignored for the purpose at hand, the abstraction boundary will be dynamically maintained by the system. When certain types of purposes and characteristics being hidden behind the abstraction become identifiably important, the abstraction boundaries will need to adapt to accommodate them.

The pervasiveness of the creation of new abstraction boundaries is illustrated by the role of marketing: marketers look across existing boundaries to better understand the *whys* of their customers and potential customers. It is marketing's job to identify commonalities that may be hidden behind existing boundaries, and to identify new ways of servicing these commonalities. Marketing theorist, Wroe Alderson (1957), argues that marketing involves trying to match segments of heterogeneous supply to segments of heterogeneous demand. Marketers try to create offerings that provide value to a particular market segment. The market segment identifies a group that shares certain requirements in common despite their underlying diversity. Marketers design new products and services that serve these common needs, while at the same time trying to identify opportunities for further segmentation and differentiation.

Marketers, however, must go beyond the creation of offerings matched to the needs of particular market segments. They must also invest in creating secondary institutions that reduce the costs of transactions. Secondary institutions reduce the transaction costs associated with a type of transaction. Streit and Wegner (1992), also drawing on Lachmann, calls these “the sunk costs of transactions.” They emphasize that the costs incurred to create an interface for a particular type of transaction are sunk costs from the perspective of a particular transaction occurring through the interface. “The costs of making economic agents familiar with institutions become the sunk costs of transactions as soon as the corresponding knowledge is used repeatedly. What remains as a source of current transaction costs,” they write (1992:138), “is the task of adapting or supplementing the institutional framework to make it fit individual transactions.”

[35] This is an example of subtyping. The type post office can be extended to meet more specific purposes. A subtype of post office can be created that includes the ability to post a letter with guaranteed overnight delivery. New instances of this subtype can be substituted to provide these new enhanced services. FedEx makes use of the postal customer’s familiarity with the same general interface (delivering a sealed envelope to a specified address) to offer the innovation of overnight delivery, adding more specific services without upsetting those with more general plans.

From this perspective, the essential role of marketing is making markets – both creating a type of product or service, and creating secondary institutions to facilitate transactions of that type.[36] Market institutions provide the customers "with readily-usable knowledge about how to make particular classes of transactions" (Loasby, 2000:306). They are attempts to lower the information costs (Dahlman, 1979; Streit and Wegner, 1992) associated with a type of transaction. The particular shape of market abstraction boundaries will be influenced by the nature of those information costs– the cost of discovering whom to transact with, of informing others of exchange opportunities, of comparing options, of negotiating terms of trade, and of policing and monitoring compliance. Streit and Wegner (1992:137) give the example of negotiation costs: "quite a number of internal [secondary] institutions have evolved precisely to reduce this highly uncertain cost component, e.g. standardized contracts, and general conditions of sale." Marketing efforts are investments in creating secondary institutions that generate informational externalities designed to lower the current costs of transactions.[37] Secondary institutions by reducing these costs for a class of transactions remove the need for individuals to incur these costs for each specific transaction.

Alderson (1957) makes a similar point with his distinction between fully negotiated transactions and routine transactions. Fully negotiated transactions are unique or non-recurring transactions, which involve bilateral negotiations between the parties. Routine transactions are recurring transactions of a particular type: "transaction routines apply to classes of transactions that are judged to be essentially similar in their purposes and in the means of carrying them out" (Alderson, 1957:297). With fully negotiated transactions all transaction costs are current; with routine transactions many of the transaction costs are sunk. Investments in secondary institutions reduce the costs of routine transactions; previous marketing activities and institutions replace the need to negotiate every feature of each transaction

Abstraction boundaries bring the benefits of separation of concerns, but they also provide an incentive for providers to look across the boundary for unmet needs, and an incentive for customers to express needs that are not being met. The potential for gains from reconfiguring the boundaries always exists. The abstraction boundaries are necessary to coordinate actions and to match segments of supply to segments of demand. Yet the essential diversity being hidden behind the abstraction provides an incentive for the discovery of new commonalities, and the creation of new abstraction boundaries to service them. The particular form of market abstraction boundaries will be shaped by the nature of the solution as well as the transaction costs associated with that type of solution. By ignoring the role of entrepreneurs and marketers in creating new products and new secondary institutions, economists do themselves a disservice. They miss an important source of endogenous changes to abstractions and their boundaries.

VI. Conclusion

We are embedded in a world of abstraction boundaries; their very familiarity renders them largely invisible. We shop at instances of the type grocery store, we mail letters at instances of the type post office, and we live in instances of the type house purchased through instances of standardized mortgage contracts which are in turn bundled and sold on instances of the type secondary mortgage market. While we always interact with concrete instances, it is the abstract boundary that enables us to coordinate our plans and adapt to changing circumstances.

---

[36] Marketers emphasize the importance of managing the 4 P's: price, product, place and promotion. Promotion often gets most of the attention in marketing; while price usually gets most of the attention in economics. We would suggest that the other two P's – the design of new products and the design of places (the secondary institutions) – are of fundamental importance.

[37] Brian Loasby (2000:300) provides a similar perspective on the importance of investing in institutions that make markets: "What makes a market for a good is its potential for enabling large numbers of transactions to be arranged and carried out at lower costs than would be incurred if that market did not exist.... 'Large numbers' are important because the institutions of a market provide an oriented set of complementary immaterial capital goods which reduces the direct cost of individual transactions, just as an oriented configuration of complementary physical capital goods reduces the direct costs of manufacturing a single product; how much it is worth investing in capital goods for either purpose depends upon the potential investor's expectation of volume."

Abstraction boundaries represent abstract plans that fill the gap between abstract rules and concrete purposes. The abstraction boundary enforces a separation of concerns: it separates particular implementations of a solution from particular uses of the solutions. Abstraction boundaries convert specific relationships into abstract relationships: they enable the reuse of solutions by multiple clients, and the provision of solutions by multiple providers. By orienting their actions to the abstract plan defined in the boundary, the parties on both sides of the boundaries can more easily coordinate and adapt their particular plans. Abstraction boundaries enhance coordination by making it easier to separate plans where they conflict, and combine them where they complement. By abstracting from the particulars on both sides, boundaries provide a point of stability in the face of change. This abstractness however also provides an incentive for people to look beyond existing boundaries and create new ones that may better serve needs hidden by the boundaries.

Secondary institutions, understood as abstraction boundaries, play an important but overlooked role in the market process. They are the chief means by which the market classifies knowledge into reusable solutions, and by which the market reduces the costs of reusing the solutions. They enable market participants to make use of a wide range of knowledge without the need to acquire it themselves. As such, they are indispensable for understanding how the market process solves the knowledge problem. Abstraction boundaries are the negotiated categories that emerge from what Lavoie calls the dialogue of the market. From the boundaries of the firm (Langlois 2002) to the emergence of new markets (Loasby 2000), new capital goods (Baetjer 1998), new products (Demsetz 1988) and new product categories (Lewin 1999), the definition and redefinition of abstraction boundaries permeates the market process.

References

Alderson, Wroe (1957) *Marketing Behavior and Executive Action*, Homewood: Richard D. Irwin

Anderson, Terry L. and Peter J. Hill (1988) “Constitutional Constraints, Entrepreneurship, and the Evolution of Property Rights” in James G. Gwartney and Richard E. Wagner eds. *Public Choice and Constitutional Economics*, Greenwich, CT: JAI Press. 207-227.

Baetjer, Howard (1998) Software as Capital: An Economic Perspective on Software Engineering, Los Alamitos, CA: IEEE Computer Society.

Baldwin C.Y. and Clark, K.B. (1997) “Managing in an age of modularity,” *Harvard Business Review*, Sep.-Oct: 84-93.

Baum, Eric B. (1999) “Toward a Model of Intelligence as an Economy of Agents,” *Machine Learning*, 35:2, 155-185.

Buchanan, James M. and Viktor J. Vanberg (2002) “Constitutional Implications of Radical Subjectivism” *Review of Austrian Economics*, 15:2/3

Butos, William N. and Thomas J. McQuade, (2002) “Mind, Market, and Institutions: the Knowledge Problem in Hayek’s Thought” in *Hayek as Political Economist*, eds. J. Birner, P. Garrouste, and T. Aimar, New York: London. 113-33.

Carrano, Frank M. and Janet Prichard (2001) *Data Abstraction and Problem Solving with Java: Walls and Mirrors*, Reading, MA: Addison-Wesley.

Cox, Brad (1996) Superdistribution: Objects as Property on the Electronic Frontier, Reading MA: Addison-Wesley.

Cunningham, Ward and Kent Beck (1989) “Constructing Abstractions for Object-Oriented Applications” *Journal of Object-Oriented Programming*, July/August.

Dahlman, Carl J. (1979) “The Problem of Externality,” *The Journal of Law and Economics*, 22:1, April.

Demsetz, Harold (1988) “Theory of the Firm Revisited” in *Ownership, Control and the Firm: The Organization of Economic Activity Volume 1*, Cambridge: Basil Blackwell.

Drexler, Eric and Mark S. Miller (1988) “Incentive Engineering for Computational Resource Management” in B. A. Huberman, (ed.) *The Ecology of Computation*, Amsterdam: North-Holland.

Encyclopedia Britannica (2003) "Postal System." *Encyclopedia Britannica Online*. 31 March, 2003, http://search.eb.com/eb/article?eu=115192.

Gamma, Eric, Richard Helm, Ralph Johnson, and John Vlissides (1995*) Design Patterns: Elements of Reusable Object-Oriented Software*, Boston: Addison-Wesley.

Gadamer, Hans Georg (1989) *Truth and Method*, second revised edition, New York: Crossroad Publishing.

Garud, Raghu, Arun Kumaraswamy and Richard N. Langlois (2002) *Managing in the Modular Age: Architectures, Networks, and Organizations*, New York: Blackwell Publishers.

Hayek, Friedrich A. (1948a [1937]) “Economics and Knowledge” reprinted in F. A. Hayek, *Individualism and Economic Order.* Chicago: University of Chicago Press

(1948b [1945]) “Use of Knowledge in Society”, reprinted in F. A. Hayek, *Individualism and Economic Order.* Chicago: University of Chicago Press.

(1952a) *The Counter-Revolution of Science: Studies in the Abuse of Reason*, Glencoe, Illinois: The Free Press.
(1952b*) The Sensory Order: An Inquiry into the Foundations of Theoretical Psychology*. London: Routledge & Keegan Paul Limited.
(1973) *Law, Legislation and Liberty, Vol. I: Rules and Order*. Chicago: University of Chicago Press.
(1976) *Law, Legislation and Liberty, Vol. II: The Mirage of Social Justice*. Chicago: University of Chicago Press.

(1978a) “Competition as a Discovery Procedure,” *New Studies in Philosophy, Politics, Economics and the History of Ideas*. Chicago: University of Chicago Press.
(1978b) “The Primacy of the Abstract,” *New Studies in Philosophy, Politics, Economics and the History of Ideas.* Chicago: University of Chicago Press.
(1979) *Law, Legislation and Liberty, Vol. III: The Political Order of a Free People.* Chicago: University of Chicago Press.
(1989) *The Fatal Conceit: The Errors of Socialism. The Collected Works of F. A. Hayek, Vol. 1*. Ed. by W. W. Bartley. Chicago: University of Chicago Press.
High, Jack (1986) “Equilibration and Disequilibration in Market Processes” in Israel Kirzner, ed. *Subjectivism, Intelligibility and Economic Understanding: Essays in Honor of Ludwig M. Lachmann on his Eightieth Birthday*, New York: New York University Press.
Holland, John H., Keith J. Holyoak, Richard E. Nisbet, Paul R. Thagard, (1989) *Induction: Processes of Inference, Learning, and Discovery*, Cambridge: The MIT Press.
Horwitz, Steven (1998) “Hierarchical Metaphors in Austrian Institutionalism: A Friendly Subjectivist Caveat”, in Roger Koppl and Gary Mongiovi, eds., *Subjectivism and Economic Analysis: Essays in Memory of Ludwig M. Lachmann*.
Kay, Alan (1998) “Re: prototypes vs. classes was: Re: Sun's HotSpot,” message to Squeak-dev, 10/10/1998, http://lists.squeakfoundation.org/pipermail/squeak-dev/1998-October/017025.html
Kirzner, Israel M. (1992) The Meaning of the Market Process: Essays in the Development of Modern Austrian Economics, New York: Routledge.
Lachmann, Ludwig, (1956) *Capital and Its Structure*, Kansas City: Sheed Andrews and McNeel Inc.
(1971) *The Legacy of Max Weber*, Berkley, CA: Glendessary Press.
(1979) “The Flow of Legislation and the Permanence of the Legal Order” *Ordo,* reprinted in Don Lavoie (ed.) (1994) *Expectations and the Meaning of Institutions: Essays by Ludwig Lachmann*, New York: Routledge.
(1991) “Austrian Economics: A Hermeneutic Approach” in Don Lavoie, ed. *Economics and Hermeneutics*, London: Routledge, reprinted in, in Don Lavoie (ed.) (1994) *Expectations and the Meaning of Institutions: Essays by Ludwig Lachmann*, New York: Routledge.
Langlois, Richard N. (1999) “Scale, Scope, and the Reuse of Knowledge,” in Sheila C. Dow and Peter E. Earl, eds., *Economic Organization and Economic Knowledge: Essays in Honour of Brian J. Loasby*. Cheltenham: Edward Elgar, 239-254.
(2002) “Modularity in Technology and Organization” *Journal of Economic Behavior and Organization* 49 (1), 19-37, September
Lavoie, Don, (1985a) *National Economic Planning: What is Left?* Cambridge: Ballinger.
(1985b) Rivalry and Central Planning: The Socialist Calculation Debate Reconsidered, Cambridge: Cambridge University Press.
(1986) “The Market as a Procedure for the Discovery and Conveyance of Inarticulate Knowledge,” *Comparative Economic Studies*, 28, Spring.
(1990a) Prefatory Note: The Origins of “The Agorics Project” *Market Process*, v8, Spring.
(1990b) “Understanding Differently: Hermeneutics and the Spontaneous Order of Communicative Processes” in Bruce Caldwell, ed. *Carl Menger and his Legacy in Economics*, History of Political Economy, annual supplement to volume 22.
(1991) “The Discovery and Interpretation of Profit Opportunities: Culture and the Kirznerian Entrepreneur.” In Brigitte Berger, ed. *The Culture of Entrepreneurship*. San Francisco: Institute for Contemporary Studies.
(2003) “Subjectivism, Entrepreneurship, and the Convergence of Groupware and Hypertext,” in Birner, Jack and Pierre Garrouste, eds., *Markets, Information, and Communication: An Austrian Perspective on the Internet Economy*, London: Routledge
Lavoie, Don, Howard Baetjer and Bill Tulloh, (1990) “High Tech Hayekians: Some Possible Research Topics in the Economics of Computation” *Market Process,* v.8, Spring.
(1991a.) “Coping with Complexity: OOPS and the Economist’s Critique of Central Planning.” *Hotline on Object-Oriented Technology,* 3: 1, Nov, 6-8.
(1991b) “Order in Complex Systems: Object-Oriented Programming and the Economists’ Critique of Central Planning” unpublished manuscript

Lavoie, Don, Howard Baetjer, Bill Tulloh, and Richard Langlois (1993) *Component Software: A Market Perspective on the Coming Revolution in Software*, Patricia Seybold Group Special Research Report, Strategic Technologies Series, April.

Lewin, Peter (1999) Capital in Disequilibrium: The Role of Capital in a Changing World, London: Routledge.

Liskov, Barbara, with John Guttag (2001) Program Development in Java: Abstraction, Specification, and Object-Oriented Design, Boston: Addison-Wesley

Loasby, Brian (1999) Knowledge, Institutions and Evolution in Economics, London: Routledge.

(2000) "Market Institutions and Economic Evolution" *Journal of Evolutionary Economics*, 10:3, 297-309.

Malone, Thomas W., Richard E. Fikes, Kenneth R. Grant, and Michael T. Howard (1988) "Enterprise: A Market-like Task Scheduler for Distributed Computing Environments" in B. A. Huberman, ed. *The Ecology of Computation*, Amsterdam: North-Holland.

Meyer, Bertrand (1997) *Object-Oriented Software Construction, 2nd Edition*, Prentice Hall, Upper Saddle River, NJ

Miller, Mark S. and Eric Drexler (1988a) "Comparative Ecology: A Computational Perspective" in B. A. Huberman, (ed.) *The Ecology of Computation*, Amsterdam: North-Holland.

(1988b) "Markets and Computation: Agoric Open Systems" in B. A. Huberman, (ed.) *The Ecology of Computation*, Amsterdam: North-Holland.

Miller, Mark S., Chip Morningstar, and Bill Frantz (2000) "Capability-based Financial Instruments" *Proceedings of Financial Cryptography 2000*

Parnas, David L. (1972) "On the Criteria to Be Used in Decomposing Systems into Modules," *Communications of the ACM* 15(December): 1053-58.

Schelling, Thomas C. (1960) *The Strategy of Conflict*, Cambridge, MA: Harvard University Press.

Smith, Fred (2002) "How I Delivered the Goods" *Fortune Small Business*, October.

Spinoza, Charles, Fernando Flores, and Hubert L. Dreyfus (1997) Disclosing New Worlds: Entrepreneurship, Democratic Action, and the Cultivation of Solidarity, Cambridge MA: MIT Press.

Streit, Manfred and Peter Wegner (1992) "Information, Transaction, and Catallaxy. Reflections on some Key Concepts of Evolutionary Market Theory"; in: Witt, U. (Ed.), *Explaining Process and Change. Approaches of Evolutionary Economics*, Ann Arbor 1992, S. 125 -149.

Turbak, Franklyn (2002) "Big Ideas", course notes from *CS111: Introduction to Programming and Problem Solving*, Spring 2002. On the web at http://cs.wellesley.edu.

Vaughn, Karen (1994) *Austrian Economics in America: The Migration of a Tradition*, Cambridge: Cambridge University Press.

(1999) "Hayek's Implicit Economics: Rules and the Problem of Order", *Review of Austrian Economics*, 11.

Wellman, Michael P. (1996) "Market-oriented Programming: Some Early Lessons," in S. Clearwater (ed.), *Market-Based Control: A Paradigm for Distributed Resource Allocation*. World Scientific.

Wirfs-Brock, Rebecca and Alan McKean (2002) *Object Design: Roles, Responsibilities and Collaborations*, Boston: Addison-Wesley.

Wirfs-Brock, Rebecca and Brian Wilkerson (1989) "Object-Oriented Design: A Responsibility-Driven Approach" *OOPSLA '89 Conference Proceedings*, 71-75.