Software Cooperatives:
Infrastructure in the Internet Era*

Terry Bollinger†
October 11, 2003‡

How widely can you distribute software for a dollar? A precise answer to this simple question helps explain the growing importance of cooperatively developed software [1] packages such as Apache and the Linux kernel.

Anyone aware of the benefits of natual gas as an energy source will be shocked if they drive near certain remote Texas oil fields at night. There they can see orange flames shooting up tens of meters into the air, the result of the burning each month of thousands of cubic meters of natural gas. Given global demand for natural gas, the question is obvious: Why is this valuable resource literally going up in smoke?

Stranded Resources

Natural gas from remote oil fields is known in the oil industry as *stranded gas*. It is burned for a simple reason: such gas is so voluminous that there is no easy way to get it out of an oil field without losing money. The first pane of Figure 1 shows stranded resources as production centers whose range of cost-effective transportation is severely limited.

Transportation and Synergy

What if the natural gas transportation problem could be solved? The implications are many, but the focus here is on how low-cost transportation might affect innovation. The rest of Figure 1 addresses innovation in terms of *synergy*, or how results that are greater than the sum of the individual parts arise. Unleashing the flow of natural gas creates opportunities for innovators to create new, dependent products such as gas furnaces (see Figure 1, Synergy – Step 1). Since gas furnaces are also resources from the perspective of cold customers, they too can be distributed, enabling still more products that are dependent on the availability of furnaces [2] (Steps 2 & 3).

---

† **The views expressed in this paper are those of the author only, and do not represent the official positions of either his employer or his employer's customers.**

‡ **Release History**:
October 11, 2003: Version 1.0 released.
October 14, 2003: Version 1.1.02 released. Disclaimer added, typos fixed, minor content updates made.
June 27, 2004: Version 1.1.03 (this version) released. BLT License added.

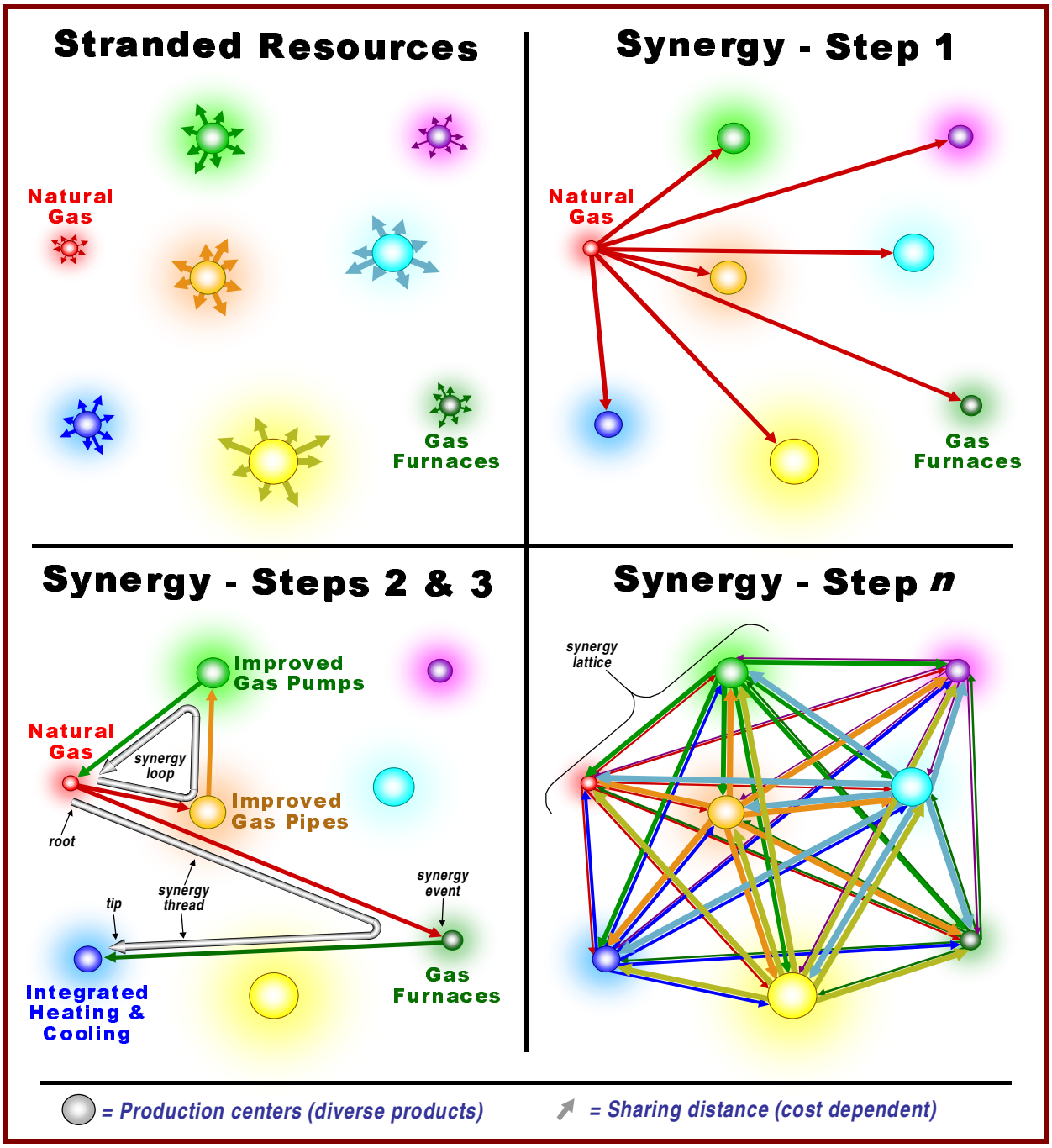


Figure 1. Synergy benefits arise stepwise from low-cost production and transportation. *Synergy events* occur when distributions of earlier products enable new products. *Synergy loops* return benefits to innovators. If such returning opportunity benefits exceed opportunity costs, synergy loops have the potential to influence innovator decisions on whether to distribute products.

A *synergy thread* traces exactly one of these domino-like cause-and-effect paths between the *root* and *tip* of the thread, with *synergy events* (new products) marking each step of the thread. Threads grow by adding synergy events at their tips. Since each thread traces only one path, threads split along their lengths if more than one product is added. (A separate construct, the *synergy tree*, can be used to map all the threads originating from one synergy event.) *Synergy loops* are a special case in which the thread tip returns to its root. Synergy loops are interesting because they provide return benefits to the original innovator, and so have the potential to influence product release decisions by that innovator. [3]

In Figure 1, the innovators are packed tightly together. In a real economy, such innovators would more typically be scattered randomly among tens or hundreds of thousands of *users*, participants who rely on innovations but do not contribute directly to the synergy process. The resulting matrix of a large number of users, a sprinkling of innovators, low-cost transportation, and minimal restrictions on product distribution is better known as a *free-market economy*. [4,5]

Given this link to economies, synergy threads provide an interesting analytical technique for exploring the relationship between innovation and free-market economies — that is, to building an *endogenous* theory of technical innovation. [6] Stranded resources are an example of how the absence of synergy threads correlates strongly with an economy in stagnation. A digital watch, on the other hand, is possible only through the intertwining of thousands of synergy threads with diverse roots in electronics, software, packaging, marketing, and many other fields. The ease with which such examples can be found and analyzed leads to a central assumption of this paper, which is that ability of an economy to innovate is strongly correlated to the richness and depth of its *palette*, or total set, of synergy threads. At the very least, the ability of an economy to innovate is strongly reflected by the health of its palette of synergy threads. A civilization that loses or caps its synergy threads is in decline, since its ability to innovate will shrink along with its synergy threads. [7] More importantly, it is argued in this paper that the synergy thread palette of an economy is dependent on a number of readily observable market features such as size, configuration, production costs, transportation costs, and transaction costs.

Figure 1 began this analysis by pointing out the consequences of high transportation costs on synergy. However, what if transportation costs cannot be reduced? Can synergy still be saved? The answer is yes. Figure 2 shows how *Coase condensation* can be used to relocate resources into a smaller region where synergy is still possible. Ronald H. Coase originally developed this concept by analyzing *transactions*, or the costs of finding, negotiating, and enforcing sales contracts with customers. [3] Coase proposed that firms separate out of a free-market economy to provide economic regions where transaction costs are low enough to permit innovation to occur. Coase's concept can be represented graphically in *transaction space*, in which transaction costs increase the farther away two points are located. The creation of a firm is equivalent to the relocation, or *condensation*, of selected resources into a much smaller region in transaction space. Within this smaller region, traversal costs are low enough for synergy to exist, albeit in a lessened form. This lessening is due both to the smaller number of participants and the cost penalties of moving those resources farther away from other useful resources in the same space.

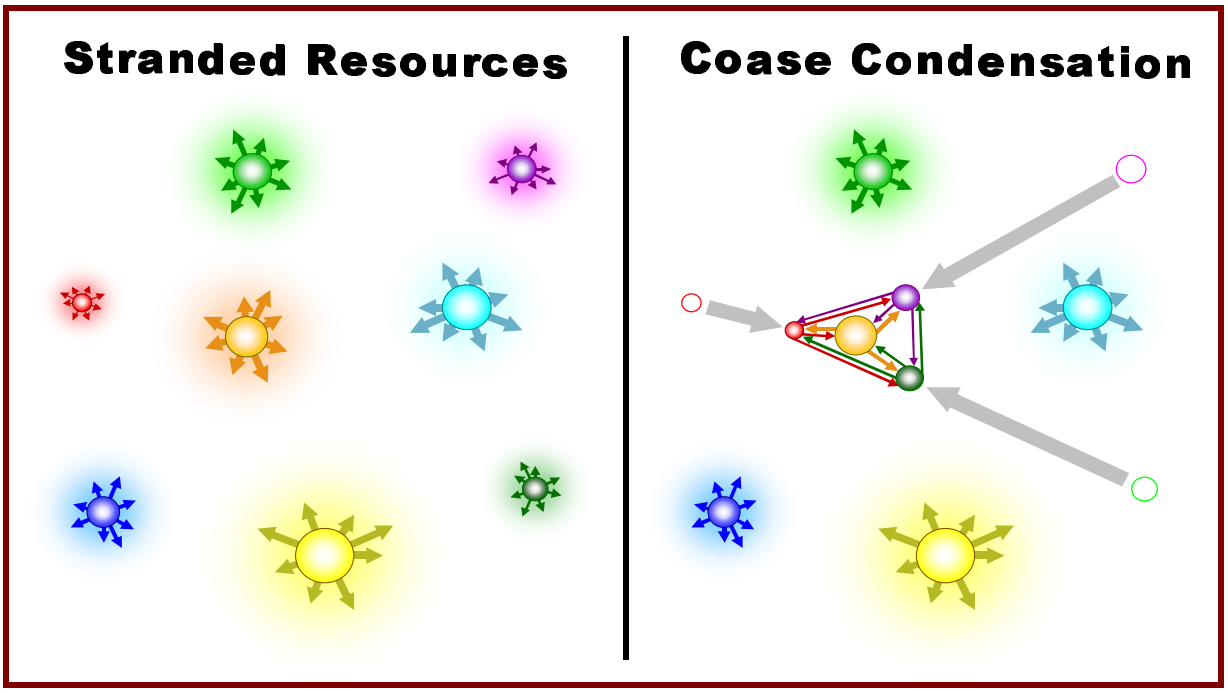


Figure 2. *Coase condensation* occurs if high-value resources are stranded and the cost of traversing the space cannot easily be reduced. Resources are relocated closer together to make synergy possible, at the cost of moving them farther from other resources. In Coase's original *transaction* space, *firms* were the condensations. Coase condensation in geographical space results in cities, corporate campuses, and other geographically localized economies.

Coase's condensation concept is not limited to transaction space. It can be generalized to any well-defined cost space — that is, any spatial representation of resources in which the cost of relocation is proportional to distance. One need only look at a map to see examples of Coase condensation in geographical space; they are called cities. Geographical Coase condensation is a plausible solution to the problem of stranded natural gas, and would mean clustering users around the gas source. This form of Coase condensation does not usually occur for natural gas because it would simultaneously move users too far away from other resources of higher value, such water, food, and entertainment. A resource with sufficient economic attraction to overcome such opposing cost tensions is gold; the resulting condensations are called gold-rush towns.

Maximally Synergistic Economies

Although Coase condensation makes synergy possible in an economy of otherwise stranded resources, it also unavoidably reduces the total synergy possible in a free-market economy. This can be seen visually by comparing the synergy paths possible before condensation (Figure 1, Step *n*) to those possible after condensation (Figure 2, panel 2). Because Coase condensation necessarily isolates some subset of the total resources of the economy, a large number of possible

synergy paths become inaccessible. One obvious consequence of this inaccessibility is the duplication seen in firms in the same industry. Such duplication is necessary if each firm is to have sufficient synergy to compete with other Coase condensations in the same market.

This observation presents an intriguing question: Are there any circumstances under which a free-market economy can exhibit maximal synergy — that is, synergy that fully engages all the resources of an economy, without the need for Coase condensation?

The first step in determining if maximally synergistic economies are possible is to disentangle synergy from market factors that might otherwise ignore, mask, or subsume its effects. The goal is to let the "invisible hand" [8] of free markets operate not just on products, but on synergy itself. The synergy loop in Figure 1, Steps 2 & 3, provides a focal point. Cost/benefit feedback loops are a defining feature of free-market economies, since they provide the mechanism by which economy-wide feedback can return to individuals and subtly guide their investment decisions towards actions that ultimately benefit the entire economy. [3]

Synergy Ranges

The first step in understanding synergy loops is to develop a more precise definition of the cost space in which they exist. In addition to transportation costs, a full accounting must include the *marginal production cost*, [9] or cost of producing each additional unit that is sent out.

Figure 3 shows the result obtained by combining transportation and marginal production costs into a single metric, the *synergy range*, that defines how widely a producer can distribute a product for a fixed amount of money. The complexity of the mathematical definition of synergy range belies its conceptual simplicity, which maps closely to the intuitive meaning of the arrows shown on stranded resources in Figure 1. The portrayal in Figure 3 of synergy range as the volume of a geometric figure provides a more intuitive feel for how transportation and marginal production costs are related to synergy range.

Notably, as the synergy range increases, transportation costs increase to the third power, while marginal production costs increase only to the second power (Figure 3, equation for consumption of available funds F). Consequently, market-wide synergy success tends to be dominated by transportation. Or stated conversely: Transportation infrastructure really is a good investment, because it helps increase market-wide synergy. [10]

It is also worth noting in the Figure 3 available funds equation that raising the density of users d within a region adds cost at only a linear rate, while increasing R involves cubic increases in cost as transportation begins to dominate over long distances. This provides some insights into the underlying incentives for Coase condensation in real space. Even when transportation costs are low, packing synergy sources more tightly in real space often provides significantly lower costs than the alternative strategy of increasing the synergy radius to include more space.

None of these results are particularly promising in terms of the earlier question of whether a maximally synergistic economy is possible in the real world. Even without looking at other constraints, the synergy range definition implies that Coase condensation is a deeply entrenched feature of free-market economies, since it has roots in the basic physics of living in a three-dimensional world in which encompassing larger spaces unavoidably incurs larger cost penalties.

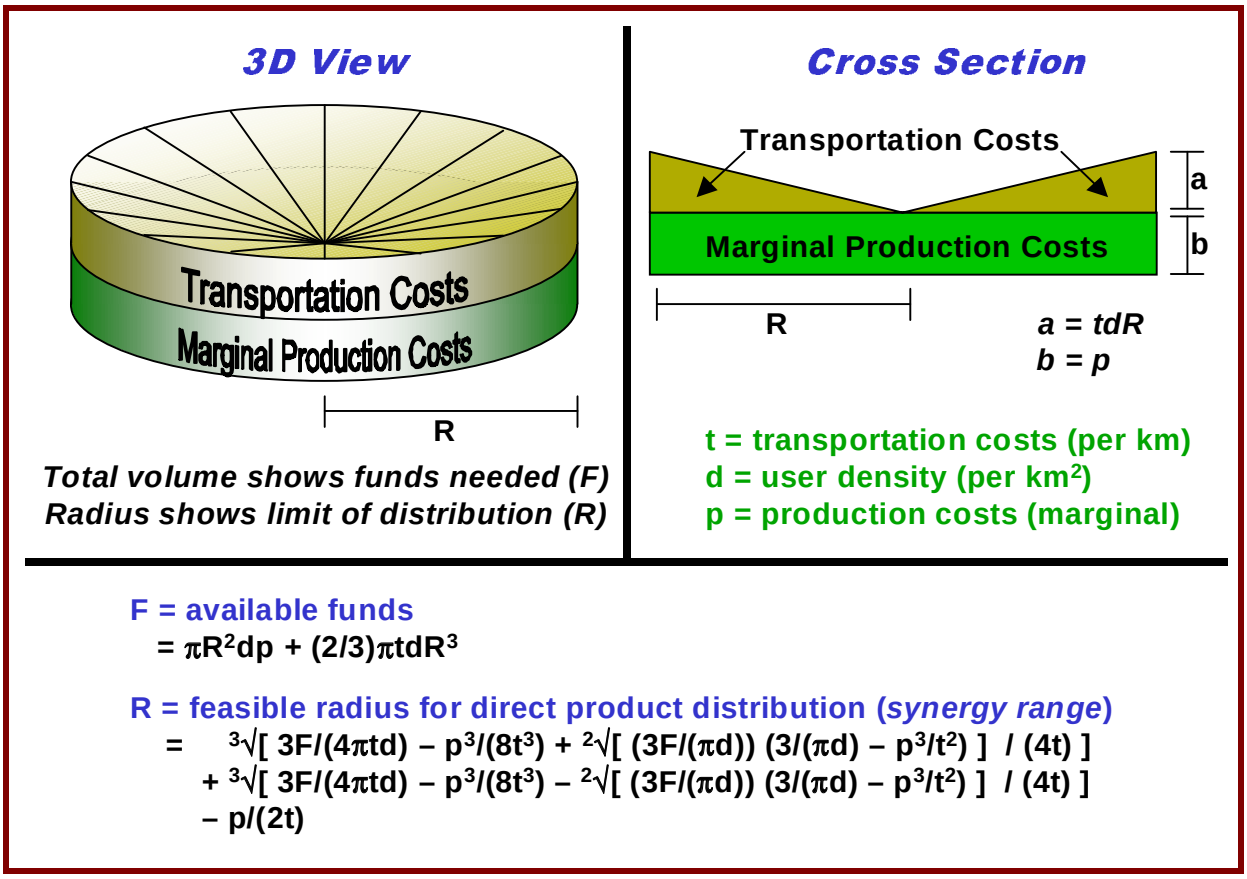


Figure 3. The *synergy range R* of a producer is the geographical distance over which she can distribute a product using funds *F*. If transportation costs dominate, *R* is roughly proportional to the cubic root of *F*. If production costs dominates, *R* is proportional to the square root of *F*.

Market Constraints

With synergy range as a tool, it is now possible to analyze synergy loops more closely and look for ways in which they can be made more relevant. The Figure 1 synergy loop example aptly demonstrates why most such loops have little impact on producer decisions: They are too slow and unreliable (stochastic) to be relevant. It would, after all, be a rare oil producer indeed who would pay for gas distribution infrastructure based solely on the hope that users of the gas might someday invent more efficient gas pipes and pumps that would create a net savings for the producer. It is this mismatch between slow-moving, stochastic synergy processes and rapid-moving, goal-focused investment strategies that allows *exogenous* theories of innovation to provide reasonable approximations of many markets behaviors.[10] Without a resolution of this mismatch, feedback from synergy loops will simply be ignored, and the odds of finding or designing a maximally synergistic economy become even less promising.

The problem of making synergy loops relevant can be broken down into the five constraints listed in Figure 4. The first two constraints deal with the stochastic nature of synergy loops. If randomness is the problem, then averaging over large numbers of users is the only obvious solution that does not rely on Coase condensation. Constraint #1 (a very large market) can be interpreted as a global market. Constraint #2 (a market-wide synergy range) is a restatement of the maximal synergy challenge, since anything less than this will lead to Coase condensations. Combined with Constraint #1, it implies a need for *global* synergy ranges.

Constraints for Creating a Maximally Synergetic Market

1. **The market must be very large.**
2. **The synergy range for most products must encompass the entire market.**
3. **The average size of an innovation event must be very small (e.g., staff-hours).**
4. **Once accomplished, innovations must be redistributed rapidly.**
5. **A large number of innovators in the market must be persuaded to participate.**

Figure 4. Constraints for creating a maximally synergistic economy.

Synergy loops are also very slow, since it may take years or even decades for the benefits represented by the tip of the synergy thread to return to the root producer. This leads to Constraints #3 (small task size) and #4 (rapid redistribution), both of which address the need to speed up synergy loops to time scales relevant to ordinary human decision making.

As difficult as the first four constraints are, the last one may well be the most challenging. The reason is that an ability to distribute products globally by meeting the first four constraints does not guarantee that innovators will actually use such products. A well-defined participation incentive is also needed, one that focuses enough attention on earlier products to ensure their use in later-generation innovations. Coase condensations do not have this problem, since within the their more limited regions they can apply goal-oriented incentive strategies such as salaries and management-based tasking. We will return to the incentivization problem for creating maximally synergistic economies after addressing whether systems exist that meet the first four constraints.

Software Synergy Ranges

Software is an obvious starting point in the search for systems that meet the five constraints. Contrary to what one might initially expect, software products are fully subject to the effects of Coase condensation, despite being composed of non-physical information. The reason is that for most of its short history as a marketable innovation, software has been packaged and transported using physical media. The use of such media makes software subject to the same rules as other mass-produced technology items. Before the mid-1980s, sending a software product to another user generally required mailing or otherwise transporting a tape or card deck. Both the marginal production costs and transportation costs for such media were typically in the range of several dollars.[11] Consequently, synergy ranges were in the range of a few hundred meters or less. In other words, programmers could only share software with a few co-workers and close friends.

The mid-1980s are significant because they mark the point where the Internet became a strong enough global presence to allow programmers to move away from depending mainly on physical media to transport software. [12,13,14,15,16] As shown in Figure 5, this move away from limitations imposed by physical media resulted in synergy ranges not easily matched by physical products.

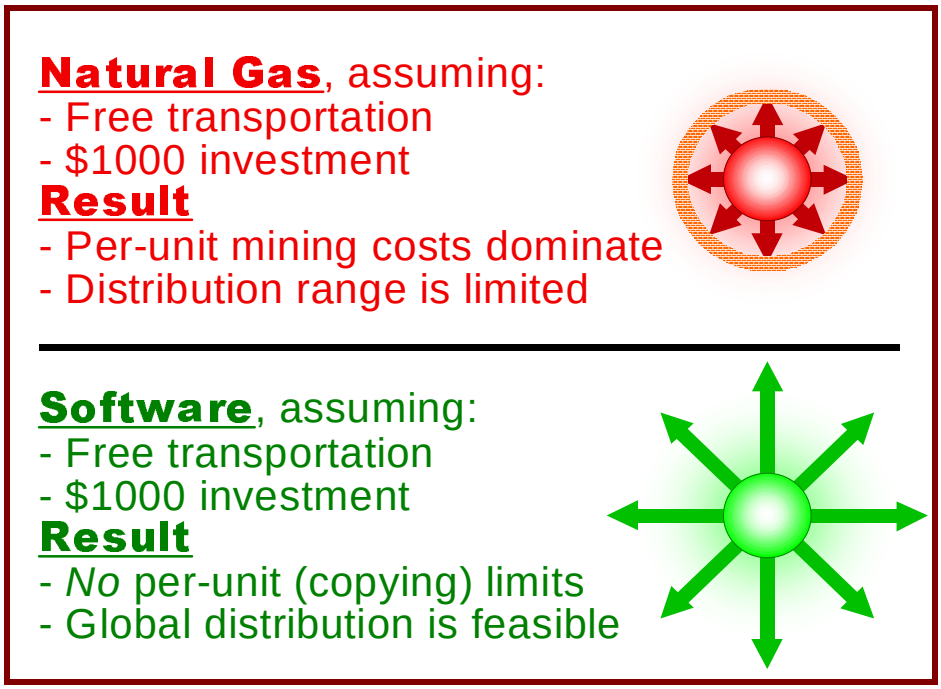


Figure 5. Synergy ranges differ significantly for physical and software products. Even if transportation is free for a physical product, per-unit marginal production costs still limit the synergy range. For software, marginal production costs approach zero when transportation is free, resulting in synergy ranges that are global in scope.

The Internet was a unique event, but also one about which much has already been written and speculated. [2,4,7,11,16] Programmers went from stranded resources to having easy global access within just a few years. Changes of this magnitude and rapidity are rare in history, especially for changes have significant (albeit unclear) global economic implications. What is relevant here is that the global synergy ranges provided by the Internet for software do surprisingly well at meeting maximal synergy Constraints #1 (global market) and #2 (global synergy range).

Software Synergy

If the premise that the size of the synergy range is critical to the growth of synergy threads, then one would expect Internet-mediated increase in synergy ranges to have resulted in a wealth of new synergy threads. [17] Figure 6 shows one such Internet-mediated synergy thread. The root

of this particular thread starts with development of an editing program (GNU Emacs) and ends with contracts to build some of the largest supercomputers [11] in the world. The stochastic, unpredictable aspect of synergy is notable in the prominent presence of Linux Torvalds' Linux kernel, [18] which the creators of the GNU utilities neither planned for nor anticipated. [2]

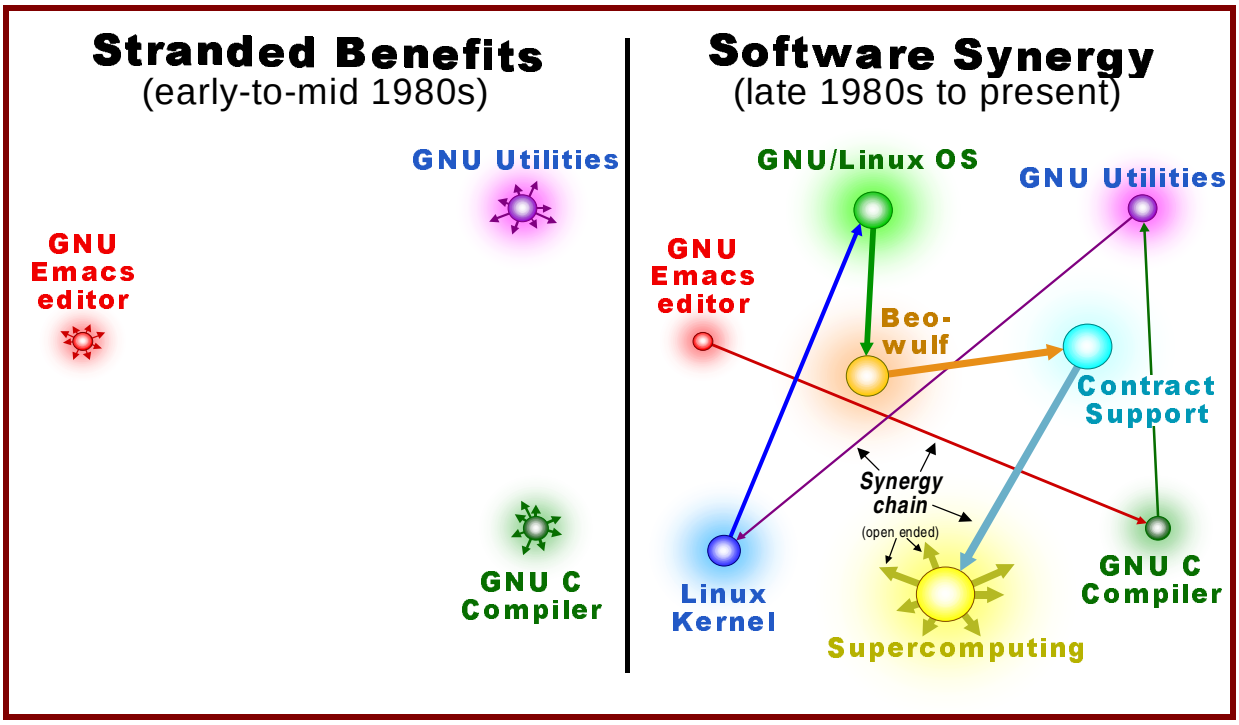


Figure 6. Example of an Internet-mediated synergy thread. Each producer depends on earlier products in this and other threads to create a new product at a viable cost. An *open ended* synergy thread is one that remains available for further growth of new synergy threads.

Closer examination of threads such as the one in Figure 6 provide insights into Constraints #3 (small task size) and #4 (rapid redistribution), both of which must be met to speed up synergy loops to human time scales. By exchanging mostly small to very small changes and additions to earlier work, Internet-mediated software threads allow programmers quite literally to see synergy threads return to them within days or even hours of sending out their particular innovations. Thus, for example, programmers who develop fixes or improvements to software tools often see both integrated final products and further improvements returned to them within days.

The Money Mystery

There is a profound difficulty with this analysis. If software such as Apache and the Linux operating system kernel are the result of the same synergy found in a free-market economy, why can they be downloaded for no charge? Isn't the profit motive also an integral component of all free-market economies, and thus a prerequisite for making free-market synergy work? [4,14,15,16]

This is an aspect of Constraint #5, the need to persuade innovators to participate. Without cash incentives, how can individuals around the globe be persuaded that providing their products *at no charge* to other innovators around the globe is worthwhile?

This question cannot be answered with a simple formula or data observation. Fortunately, economic precedents to this kind of behavior already exist within free market economies in the form of *infrastructure cooperatives*. Examples include rural electric cooperatives that in the early 1900s helped bring electricity to rural farmers in the U.S. and other parts of the world. [1] Figure 7 shows a simple example of how a natural gas cooperative could in principle make access to stranded natural gas economical. (Note: Any real-world solution to stranded natural gas would have to address the problem that gas sources change over time as older wells run dry.)

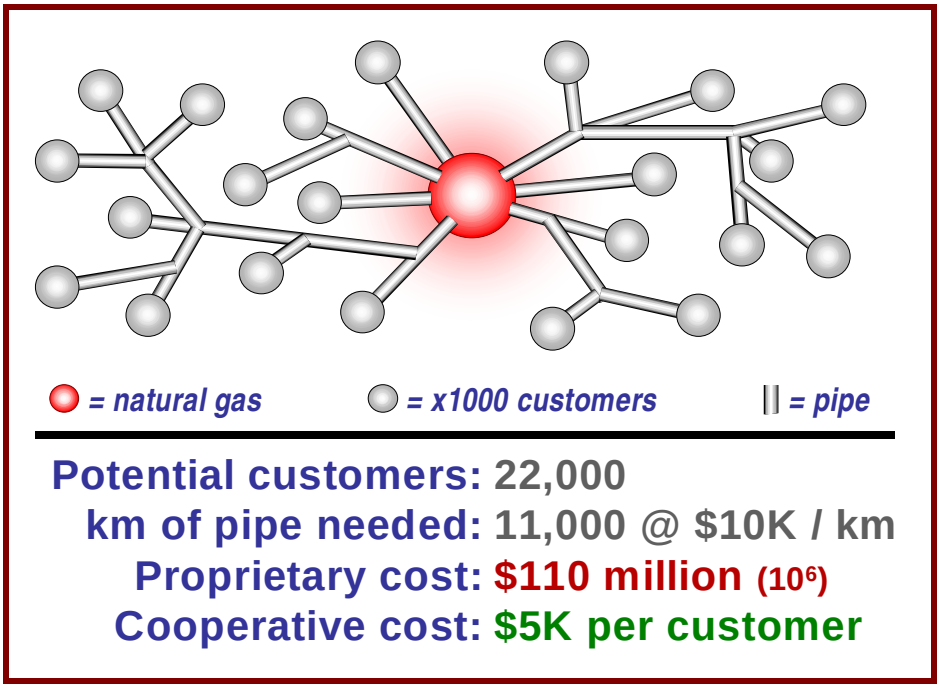


Figure 7. A *natural gas cooperative* lowers investment barriers by distributing pipeline costs across customers. Cooperatives become economically viable when: (1) an abundant stranded resource exists, (2) the number of potential customers is large, and (3) the necessary infrastructure for distributing the resource is too costly or has too slow of a return to attract conventional external investors.

The idea behind an infrastructure cooperative is that if the cost of the infrastructure needed to access a resource is too high to attract outside investors, why not divide up the investment problem among the intended users of that infrastructure? The greater the number of interested users is, and the stronger their interest in obtaining the resource, the more attractive a cooperative

approach becomes for creating large infrastructure. In the rural electric cooperatives of the early 1900s, direct profit from the resource was not the motive for joining. Farmers, for example, joined rural electric cooperatives to gain access to all the equipment and lifestyle innovations made possible by access to electricity, and not to sell electricity to other users.

Capability Spaces

While cooperatives clearly provide an example of why groups of people might contribute to creating infrastructure in the absence of a direct profit incentive, it is difficult to see how the idea applies to software. After all, where are the large geographical distances to be spanned, and what are the materials used to span them? Another difference is that farmers do not build electrical grids, only fund them. This contrasts to the hands-on programmer roles seen earlier in Figure 6.

The answer can be found by trading spaces. The lower panel of Figure 8 maps creation of software infrastructure into *capability space*, which is can be understood as a very large, all-encompassing "layers of abstraction" diagram turned sideways. The computing and network hardware that generate raw computational power reside at the left end of this directed space, and useful results reside at the right end. Between these two ends lie all the intermediate software capabilities needed to transform raw computing power into useful, goal-oriented results.

Distances in capability space are measured by how long it takes a programmer working alone to create the software that bridges two points. This distance can be expressed in units of staff years, but to provide a better feel for the human-scale implications of capability space distances, it helps to define a unit with a geographical interpretation that aptly expresses the magnitude of the effort required. This unit is the *magellan (Mg)*. A magellan can be interpreted either literally as four staff years, or figuratively as the distance a programmer would cover if she walked as she programmed for those same four staff years. At a typical pace, one Mg works out to be about 40,000 kilometers, or once around the circumference of the earth. Only the most dedicated full-time programmers manage to travel 10 Mg in their lifetimes, and most cover less than 5 Mg before their careers take them down other paths.

Because so much distance has already been covered for us, it is easy to underestimate the immense size of capability space. In the earliest days of computing, capability space was at most a few magellans from end to end, since computers had limited capabilities and software tended to focus on straightforward number crunching. By the early 2000s, however, capability space had expanded immensely. Its endpoints shifted to encompass the much larger distances between complex and powerful networked computers, and the more diverse and sensory-oriented needs of users. Consequently, the total length of capability space as of the early 2000s is probably in the range of tens of thousands of magellans. Including parallel and branching paths, the total travel that has occurred in capability space may be in the range of millions of magellans.

The need for a large shared infrastructure, combined with an intense interest in the resource (computing power) to which it provides access, are the main incentive ingredients for creating an infrastructure cooperative. As with other infrastructure cooperatives, the primary goal of such a *software cooperative* is not to sell infrastructure (software) or the resource that is being accessed (computing power), but to bring the benefits of the resource to the members of the cooperative.

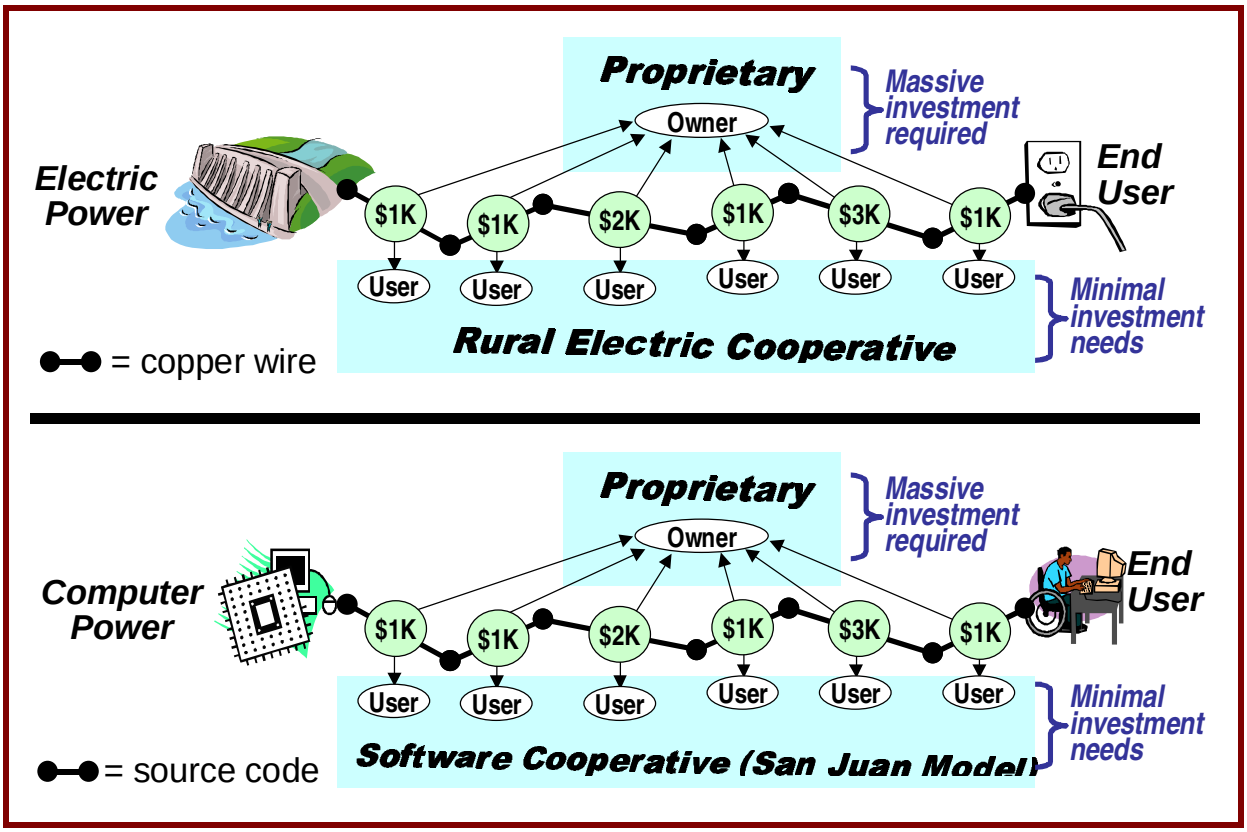


Figure 8. *Software cooperatives* create infrastructure not in real space, but in *capability space*. Abundant raw computing power must traverse this space to reach a global base of users, but the infrastructure is too large for any one user to fund. Before the mid-1980s, software synergy ranges were very low, making Coase condensation and use of proprietary investment the only viable approach to creating large infrastructure in capability space. After the mid-1980s, global synergy ranges made cooperative approaches viable by creating global user communities.

The Free Market Connection

The structure of capability space also helps answer the earlier question of how the synergy of a free-market economy can operate in a not-for-profit cooperative. In an electrical cooperative, all of the spanning elements (e.g., copper wire) are the same, so little innovation is needed to create the infrastructure. Capability space, however, is a directed space in which infrastructure must begin at the source (computing power) and extend outward like a vast pier into an unknown ocean of possibilities. The construction process is cooperative[2] because everyone depends on the work that was done before. At the point of extension, however, it is fiercely competitive[2] as members vie to bring a rich new set of capabilities home for themselves and for the cooperative.

Software cooperatives thus are examples of *free-market cooperatives*, which are asymmetrical cooperatives in which infrastructure must be added incrementally from a shared starting point. The leading edge of infrastructure construction exhibits free-market synergy, while the earlier segments are stabilized by the need for shared use by all members of the cooperative. Rural electric cooperatives and most real-space infrastructure cooperatives are *planned cooperatives* that permit parallel creation of infrastructure, and so require little or no innovation from the members of the cooperative. This distinction allows farmers to participate by funding only.

Synthestructure

It is useful to name the major parts of the infrastructure created by a free-market cooperative. The *synthestructure* is the shared infrastructure created by earlier cooperative work. It is the pier that everyone must share if anyone is to fish. Synthestructure can be abstracted as a tightly woven bundle of synergy threads that has been elaborated over time by the cooperative. *Deep synthestructure* refers to regions farther in the past, and *shallow synthestructure* to regions close to the present-day *leading edge* at which free-market synergy occurs.

Synthestructure enables new growth, but it also constrains growth paths by making the option of creating new, possibly superior synthestructure less attractive. Synthestructure also tends to be self-stabilizing as it grows larger, since the cost of changing it increases both as the number of participants increases (since all must use it) and as the changes move deeper (since deeper changes perturb or unravel larger sections of later synthestructure). The result is an incentive for innovators to focus change and innovation on the leading edge of the synthestructure, and to accept without change some range of non-optimal but adequate (that is, sufficing) features of the shared synthestructure. It is easy to find examples of such trade-offs in cultural and linguistic synthestructures around the world. In Western culture, the merging centuries ago of the Roman alphabet with Indian numerals (via Arabia) created confusing overlaps such as O/0 and l/1 that are far from optimal for conveying information. However, this event is so deeply embedded in the synthestructure of the West that changing it would cause cost perturbations far larger than the relatively minor benefits. Thus, the non-optimal thread is accepted as a given and Western cultural innovation continues onto with more immediate issues.

Proprietary Software

There is a problem with the analysis of free-market cooperatives as they apply to software. If software cooperatives provide such an effective means for creating infrastructure, why has proprietary software infrastructure dominated the 1980s, 1990s, and early 2000s? [16]

The answer is simple: Before the arrival of the Internet, proprietary development was the only economically viable way to create large, complex infrastructure (Figure 9). Low synergy ranges blocked the formation of software cooperatives and strongly favored Coase condensation as a way to collect sufficient synergy and resources to create large-scale infrastructure in capability space. Proprietary developers solved the stranding problem by using investment to cover early costs, with the condition that the early costs would later be recouped through sales of the resulting software. [16] To ensure the ability to sell the software later, proprietary projects required

the use of *investment walls*[2] that behaved like one-way mirrors, allowing developers see out while preventing future customers from seeing in.

An unfortunate and unintended side effect of using investment walls is that they cap synergy threads. Since others cannot see into the proprietary process, all innovation must come from the development group that resides within the wall. This can be a serious problem if a company is competing with unrestricted synergy threads, since even the smartest and best-trained employees cannot compete indefinitely with rates of problem solving possible when a global audience is looking at the same problem.[2] Consequently, for problems in which there is a strong and global interest in creating shared solutions, the proprietary model is likely to have difficulty competing with the synergy capabilities of a global software community.

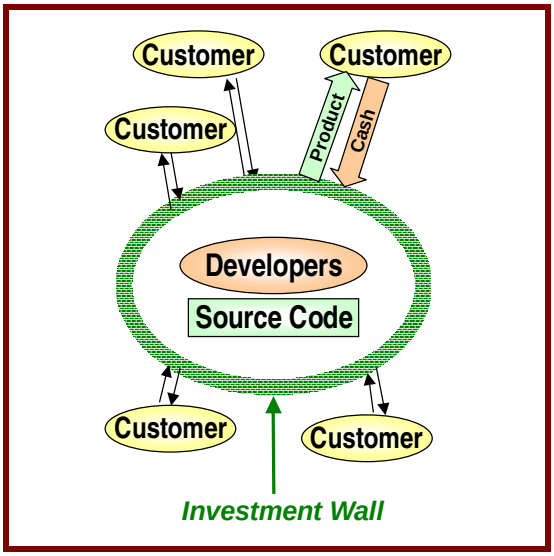


Figure 9. The proprietary model of software development overcomes low synergy range and overhead costs by using capital investments that will be recouped through sales of the final product. The *investment wall* is a necessary component of this strategy, but it limits synergy.

A review of the statistical nature of synergy loops shows that this relative advantage does not apply to all software development problems. In particular, if an unsolved problem is *irreducible* in the sense that it cannot easily be reduced to a sequence of smaller steps, the chances of timely resolution by a global audience drop dramatically. Irreducible problems thus remain strong candidates for proprietary development, which can gather and apply specialized resources to their resolution. However, this advantage can be frittered away if proprietary projects fail to take full advantage of available external infrastructure. Internal resources must instead be kept focused on resolving the irreducible problems.

Maximal Synergy and Software Cooperatives

Software cooperatives do appear to meet the incentive requirements of Constraint #5. When combined with the Internet-mediated resolutions of Constraints #1 through #4, this means that a maximally synergistic free-market economy not only can exist, but has existed since the creation of the first software cooperative in the mid to late 1980s. The relationship of this cooperative to free-market economics has been obscured by its reliance on a barter-style, synergy-loop-based investment strategy. However, even its name, *free software*, reflects its close relationship to free market economics, since in both phrases the word "free" refers to an unfettered ability to exchange goods, not giveaways.[19] Richard Stallman, who is both the philosophical and practical founder of this economy, notes that in his early days he made a living by selling copies of his

"free" software.[20] Both by statement and action Stallman thus has demonstrated his intent for the word "free" to mean unfettered distribution of goods, not giveaways. However, due to widespread confusion about the meaning of the phrase, Chris Peterson coined "open source" in 1998 as an alternative name for this economy.[21] Eric Raymond and others promoted the new phrase with great success,[22] and it is now the phrase used most often describe Stallman's maximally synergistic software economy.

Types of Cooperatives

If software cooperatives are likely to play a significant role in software infrastructure, how should they be used?[2,11,15,23,24] The first step is to realize that cooperatives differ significantly in their policies on membership, casual use, and innovation ownership. These policies can be found in licenses attached to the infrastructure software, although the interpretation of such licenses can be complex. The most important differences are those of innovation ownership. These affect not only the internal dynamics of a cooperative, but also how they interact with other organizations.

Membership Policies

Membership policies decide who can join a cooperative. C*losed policies* restrict membership so severely that the cooperative looks like proprietary development. At the other extreme are *open policies*, which allow essentially anyone to join. Because open policies maximize the synergy benefits possible with high synergy ranges,[7] they tend to be the most widely used.[23] Between these extremes are two groups: *theme policies* that apply arbitrary (non-development-related) synergy restrictions, and *community policies* that provide free access within a large but bounded group of users. Community policies are less attractive to developers because changes in jobs or locations may deny them access to their own contributions. However, with a sufficiently large community, a good level of synergy is possible. Community cooperatives work best when there are very few people outside of the community who would take an interest in the software infrastructure being developed. Even in these cases, however, community policies can diminish synergy by capping threads that could be extended by external participants and cooperatives.

Casual Use Policies

All of the major software cooperatives have extraordinarily generous *casual use* policies that permit unlimited, "no strings attached," operational (that is, non-infrastructure-extending) use of their software. This generosity is possible due to the very high synergy range of software cooperatives, and is not economically feasible for physical (e.g., electrical) cooperatives. Casual use policies are at the root of the common misconception that software cooperatives are pure giveaways. Anyone holding this misconception can correct it vigorously by calling the author of a cooperative license and asking if it would be OK to ignore the license and begin selling the software as if it they had written it themselves.

Casual use policies are more accurately understood as the software cooperative equivalents of enlistment advertising. By building up a large base of pure users, they allow a cooperative to expand and replenish its base of innovators who perform the real work of the cooperative. The result is one of those rare cases where maximal generosity coincides with maximal self-interest.

Innovation Ownership Policies

Since free-market cooperatives are all about sharing innovation, how they handle innovation ownership is one of their defining characteristics. Figure 10 shows the four main possibilities.

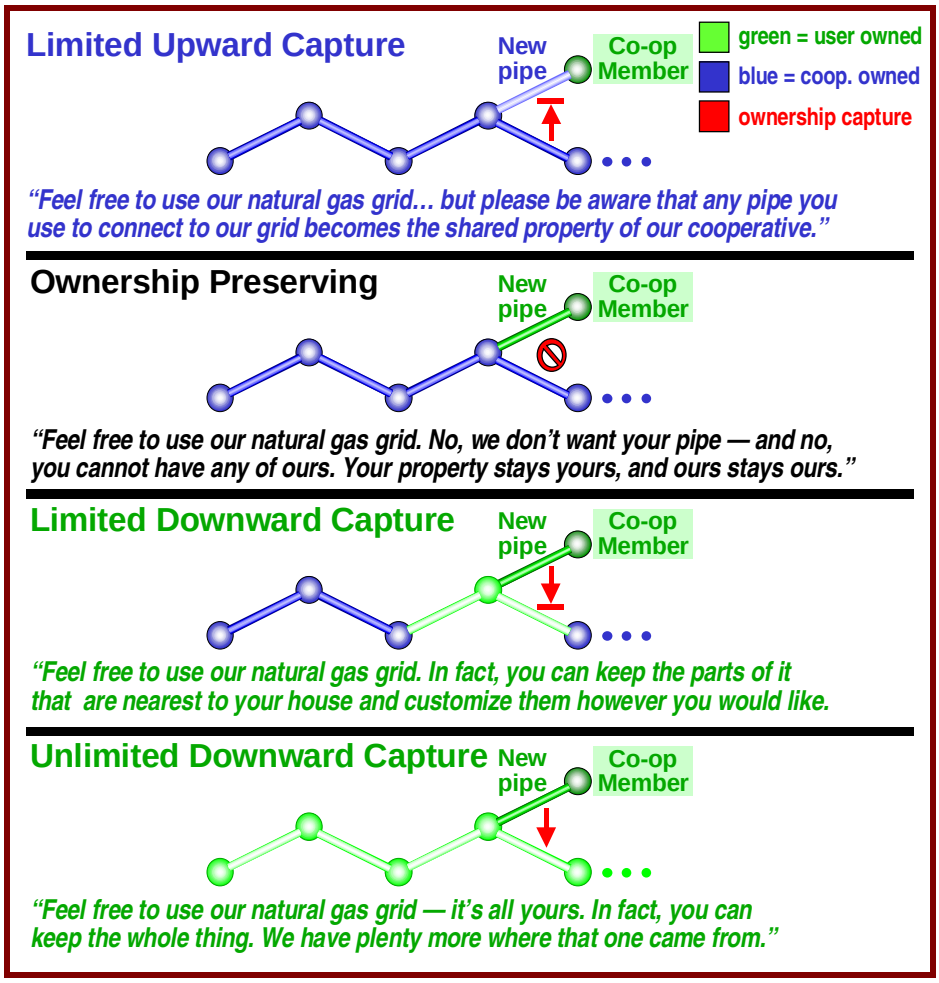


Figure 10. This gas pipe analogy shows the four major innovation ownership rules possible for software cooperatives. The last strategy of unlimited downward capture is unrealistic for physical cooperatives.

Limited Upward Capture. Limited upward capture requires that innovators donate infrastructure back to the cooperative. To prevent spoofing (e.g., donating useless pass-through modules), the donations typically must be complete applications. Because such donations accelerate growth, this type of cooperative tends to expand faster, [2,23] which in turn encourages more participation.

Ownership Preserving. Ownership preserving policies maintain existing ownerships; neither side acquires innovations from the other. Infrastructure using these rules can serve as a fence to isolate software developed under other sharing rules, and to help stabilize shared interfaces.

Limited Downward Capture. Limited downward capture allows proprietary software to obtain ownership of the upper layers of cooperatively developed software, generally as an enticement to begin proprietary work within a standardized framework (e.g., in a protocol such as TCP/IP).

Unlimited Downward Capture. This policy allows members to copy (clone) and take possession of an entire cooperative infrastructure. It is possible only with extremely high synergy ranges, and so makes no sense for physical cooperatives. Such cloning occurs less often than one might expect, since it has the unfortunate side effect of capping synergy threads, which in turn makes the clones increasingly expensive to maintain over time.

A Taxonomy of Software Cooperatives

The ranges of membership and innovation ownership policies in software cooperatives can be combined to create a taxonomy of cooperatives and related development methods (Figure 11).

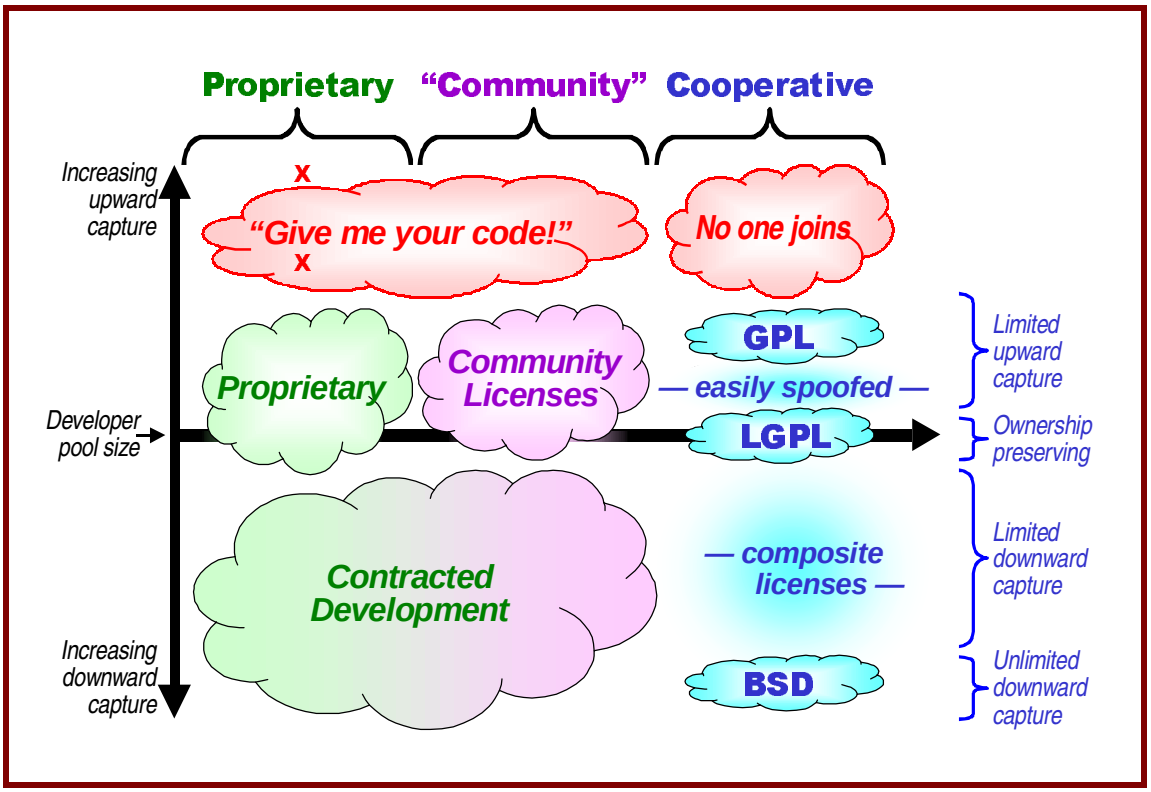


Figure 11. A taxonomy of cooperative and proprietary software development models. The dominant cooperative licenses are shown for three of the four cooperative niches.[2]

The simple layered architecture [11,2] in Figure 12 gives one example of how the different types of cooperatives and proprietary development can be combined. In general, the more software resembles shared infrastructure, the more likely it is to benefit from the participation and stabilization effects of cooperative development. Placing difficult problems in the more isolated upper regions of the architecture allows them to be resolved without damaging infrastructure.

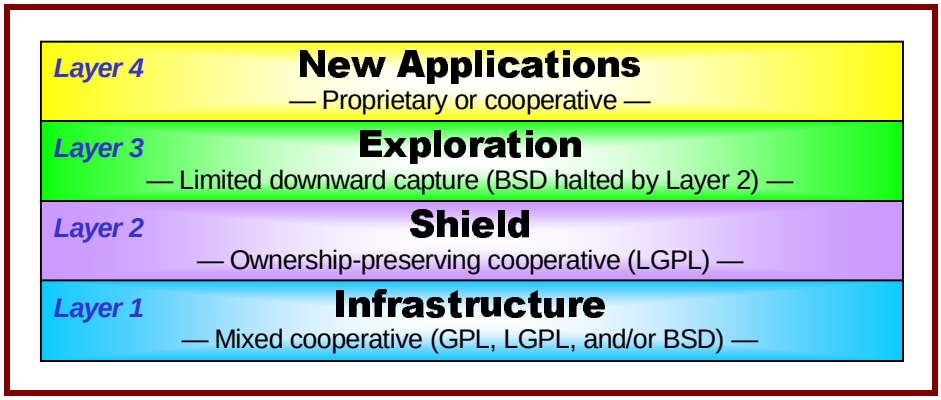


Figure 12. A generic layered architecture for applying the specific advantages of proprietary and cooperative development models. The goal of this architecture is to maximize the different benefits of the models while minimizing points of conflict.

Careful readers may have noticed an earlier example in which the synthestructure concept was used to describe alphabets. In that example, no mention was made of the "power source" feature of the electrical and software cooperatives. Was this "alphabetic synergy" description just a careless analogy, or an intentional extension of the synthestructure concept?

A full answer is beyond the scope of this paper, but the extension was intentional. Energy and processing are not the only high-value resources accessible via synthestructure. Memory is also a resource, and the synthestructure itself is a form of data storage. When synthestructure grows sufficiently rich and diverse, it becomes an immensely valuable resource in its own right, one that enables participants to share thoughts, ideas, insights, understandings, and critical information about resources in the physical world. The deep synthestructure of a culture provides a history of lessons learned, and summarizes insights that allow a culture to respond effectively to the world around it. At its leading edge, such a synthestructure makes provides a shared context of communication that makes innovation and industrial activity possible.

This paper therefore closes with a few examples of how synergy threads, Coase condensation, free-market cooperatives, maximmally synergistic economies, and synthestructure concepts can

be used to analyze diverse systems in which memory, rather than energy or processing power, is the resource to which the synthestructure provides access.

Scientific Publication

The early scientific publication process may hold first honors for a system that approximates a maximally synergistic economy. Its body of widely shared papers is its synthestructure, its system of references is an explicit instantiation of synergy threads, and its leading edge research is an example of free-market competition laying the foundations for still more infrastructure. As the scope of science has expanded over the centuries, Coase condensation within subject space (think Dewey Decimal) has increasingly become an issue, as the cost to human participants of keeping up with information outside of a limited topic area has increased vastly. The resulting condensations can limit synergy activity to highly specialized topics. The Internet has greatly accelerated the pace of this economy by increasing accessibility, in particular through the CiteSeer [25] online repository of papers. CiteSeer provides a fascinating example of the interplay between synergy incentives (placing papers online allows synergy threads in the forms of citations to grow more quickly) and publication only in specific journals (which limits synergy). The temptation to publish online is clearly growing as the number of papers in CiteSeer continues to expand, and the result will more pressure on traditional scientific publications. CiteSeer is at the leading edge of exploring how these incentives will play out in the long term.

Software Documentation

The tips of synergy threads grow only when external innovators link into them. In terms of software modules, this implies that the most powerful documentation is understandable to as wide an audience as possible, provides clear and unambiguous references to the earlier modules (synergy threads) on which they depend, and remains inextricably tied to the module itself. The implications of these constraints are radically different from more traditional approaches to documenting software, in which a closed context allows the development both of independent documenation storage locations and specialized terminologies that are largely incomprehensible to outside users. Not surprisingly, software cooperatives have gravitated towards approaches that support this concept of *synergy-first documentation*. Examples include more reliance on internal documentation and full display of the source code ("open source" [22]). A synergy thread interpretation of documentation provides some insights on why it is difficult to move proprietary software into a software cooperative. Even when the original documentation is high in quality, its proprietary style of documentation will make it signficiantly harder for a broader community to understand and use, which limits synergy and makes the code harder to use and integrate.

Linguistics

Synergy and synergy loops that operate on human-decision time scales are a historical novelty that did not exist until the arrival of the Internet. For most of the human history, synergy has operated at a much slower and less obvious time scale of years, decades, or even centuries. Viewed in this slower perspective, natural languages can be interpreted as Coase condensations of memory-access synthestructures of entire cultures. Intuitively, the greatest possible synergy in

most human endeavors occurs when everyone shares a single language, which allows maximal free-market exchange of concepts and information to take place. However, rising transportation costs such as those that occurred in the later days of the Roman empire can lead to geographical Coase condensations that help concept-exchange synergy to continue within isolated cultures [26] — that is, they can lead to new languages. The effects of the Internet on language over the next century or so should be fascinating, since the Internet simultaneously removes many of the barriers that led to earlier linguistic Coase condensations, but also adds a new ability to speed automated translation between existing languages.

Psychology

One of the more famous aspects of scientific insight is the "Eureka!" experience, in which an experience as mundane as taking a bath in an overfilled tub results in an unexpected insight into a difficult problem. Such small but invaluable synergy events appear to be an important component of how the human mind develops its own synergy. As anyone who has tried to move a project from their own mind to a team of developers knows, the human brain is adept at making connections in ways that teams of people can emulate only with experience and luck. Much research has gone into the problem of how the brain recognizes patterns, but less attention has been paid to the remarkable ability of the brain to create its own internal economy of easily shared and cross-linked concepts. The applicability of a synergy thread model is less obvious for this topic area, yet there are hints that applying such a model might lead to new ways to represent and analyze how the human brain supports and pursues innovation.

Artificial Intelligence

What is the smallest possible synergy event? This is an interesting question because synergy events in general are usually decomposable into smaller synergy events. A better understanding of the smallest possible synergy event might lead to insights on how larger synergy problems could be elaborated effectively until they can be translated into understandable processes. The identifying characteristic of a synergy event at any scale is its ability to make some task easier.

Genetics

Moving down to an even slower time scale of millennia, Coase condensation appears to provide an interesting approach to analyzing the formation of subspecies and regional groups in biology. In this case, genes represent synergy threads, gene pools represents synthestructure, and the slow spread of genes throughout gene pools represents the dispersal of synergy threads. When the gene pool is large and dispersal of genes easy, the stabilizing effects of a large cooperative come into play and species resists major differentiation, even while innovation presumably continues at a rapid pace within those limits. When barriers such as isolation by geography are introduced, the increased costs of gene exchanges encourage Coase condensations in various spaces, including in particular geographical space. Innovation within smaller regional groups continues, but without the stabilizing effects of the larger community. The result is a broader range of form experimentation. As with entrepreneurial firms, such experimentation can

be both interesting and risky. The dodo was, after all, a decidedly interesting bird, but not one that was particularly adept at survival once it was reconnected back into a larger community.

A notable implication of the genes-as-synergy-threads analysis is that it implies that the cause of isolation diversity is not the introduction of new needs, but rather the removal of the inhibiting effects of a larger genetic cooperative. It also implies that even when change is not apparent in outward forms, it is likely continuing vigorously in more subtle ways. The removal of inhibitions does not increase this rate of change, but rather allows ongoing changes to extend their reach deeper into the genetic infrastructure, where they can give rise to more obvious changes in form.

Acknowledgments

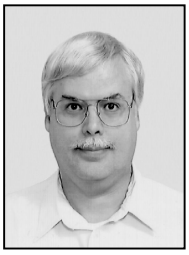
I am deeply grateful to Tom Greenfield, Dawn Meyerriecks, and Fritz Schulz of the Defense Information Systems Agency for their support in looking at use of open source software in the U.S. Department of Defense. I would also like to thank Christopher Bollinger for suggesting improvements to diagrams.

References

1. R. McOrmond, "Software models for Cooperatives," Sept. 1999; weblog.flora.ca/article.php3?story_id=112&section=flora.ca
2. R.C. Pavlicek, *Embracing Insanity: Open Source Software Development*, SAMS, Indianapolis, Sept. 2000.
3. R.H. Coase, *The Firm, the Market, and the Law* (Reprint ed.), University of Chicago Press, Chicago, June 1990.
4. R.W. Hahn (ed.), *Government Policy toward Open Source Software*, AEI-Brookings Joint Center for Regulatory Studies, Washington, 2002; www.aei.brookings.org/admin/authorpdfs/page.php?id=210
5. D. Ricardo, *Principles of Political Economy and Taxation* (Reissue ed.), Everymans Library, Dutton NY, Sept. 1992.
6. P. Aghion, P.W. Howitt, *Endogenous Growth Theory*, MIT Press, Cambridge Mass., Dec. 1997.
7. L. Lessig, *The Future of Ideas: The Fate of the Commons in a Connected World*, Random House, New York, 2001.
8. A. Smith, *An Inquiry into the Nature and Causes of the Wealth of Nations*, Indypublish.Com, June 2002.
9. A. Marshall, *Principles of Economics* (8th ed.), Porcupine Press, Philadelphia, Dec. 1982.
10. M. Bishop, *Pocket Economist: The Essentials of Economics from A-Z*, Profile Books Limited, Sept. 2000.
11. D.K. Rosenberg, *Open Source: The Unauthorized White Papers*, John Wiley & Sons, New York, Jan. 2000.
12. C.J.P. Moschovitis, H. Poole, T. Schuyler, T.M. Senft, *History of the Internet: A Chronology, 1843 to the Present*, ABC-CLIO, Santa Barbara, CA, Apr. 1999.
13. C. Dibona (ed.), *Open Sources: Voices from the Open Source Revolution*, O'Reilly Open Source, Sebastopol CA, Jan. 1999.
14. J. Sandred, *Managing Open Source Projects: A Wiley Tech Brief*, John Wiley & Sons, New York, February 2002.
15. J. Feller, B. Fitzgerald, E.S. Raymond, *Understanding Open Source Software Development*, Addison-Wesley, London, 2002.
16. B. Gates, N. Myhrvold, W.H. Gates, P.M. Rinearson, *The Road Ahead: Completely Revised and Up-to-Date*, Penguin, New York, Nov. 1996.
17. D. Spinellis, *Code Reading: The Open Source Perspective*, 3rd ed., Addison-Wesley, Boston, May 2003.
18. L. Torvalds, D. Diamond, *Just for Fun: The Story of an Accidental Revolutionary*, HarperBusiness, New York, May 2001.
19. R. Stallman, "The Free Software Definition," 1996-2003, Free Software Foundation, Boston; http://www.gnu.org/philosophy/free-sw.html.
20. R. Stallman, "Selling Free Software," 1996-2001, Free Software Foundation, Boston; http://www.gnu.org/philosophy/selling.html

21. J.M. Gonzalez-Barahona, P. de las Heras Quiros, and T. Bollinger, "A Brief History of Free Software and Open Source," IEEE Software (16:1) 1999, pp 32-34.
22. Open Source Initiative (OSI); http://www.opensource.org/
23. W. Scacchi, "Free/Open Source Software Development Practices in the Computer Game Community," draft submitted to *IEEE Software* on April 2003.
24. M. Fink, *The Business and Economics of Linux and Open Source*, Prentice Hall PTR, Upper Saddle River NJ, Sept. 2002.
25. S. Lawrence, "Online or Invisible?" Nature, Volume 411, Number 6837, p. 521, 2001; http://www.neci.nec.com/~lawrence/papers/online-nature01/
26. E. Gibbon, *The Decline and Fall of the Roman Empire*, Knopf, New York, Oct. 1993.

About the Author



Terry Bollinger is an IEEE Millennium Medal Winner and a former Assistant Editor-in-Chief for *IEEE Software*. He was also a special editor for an *IEEE Software* issue on Linux, and is currently a special editor for an upcoming November 2004 issue on Persistent Software Attributes. He wrote the Wiley Encyclopedia of Software Engineering 2nd edition article on Linux and open source software, and was primary author of Software Construction section of v1.0 of the international *Software Engineering Body of Knowledge* (SWEBOK) standard. He also wrote a 2002 report documenting widespread use of free/open source software (F/OSS) in the U.S. Department of Defense. Contact him at www.terrybollinger.com.