HARNESSING EVOLUTION FOR ORGANIZATIONAL MANAGEMENT

JIM HINES & JODY HOUSE

MIT, Sloan School of Management
E53-309
77 Massachusetts Avenue
Cambridge, MA 02139-4307
JimHines@Interserv.Com, Jhouse@ece.ogi.edu

Synopsis. Organizations are too complex to "solve". Consequently, corporate improvement programs are as likely to be deleterious as beneficial. Evolutionary mechanisms offer an alternative approach to organizational improvement. Organizational policies correspond to biological genes; policy innovation corresponds to genetic mutation; and learning corresponds to genetic recombination. These correspondences form a foundation for creating mechanisms that will foster more rapid organizational evolution in the direction managers desire. Computer simulation offers a laboratory environment for exploring issues in organizational evolution including learning drift, hierarchy, and team learning.

1. Introduction: The importance of organizational evolution

Managers hope to improve their companies. Ultimately, they seek higher profitability for shareholders, more satisfaction for employees, and better service for customers. Unfortunately, managers who improve one part of a complicated business system, run a significant risk of creating problems in another part.

Kofman, Repenning and Sterman (1997) recently investigated a case of improvement gone wrong at Analog Devices. Analog's TQM program succeeded in slashing variable manufacturing costs. However, people in marketing continued to set prices using the traditional policy of multiplying variable costs by 2 to arrive at the price[1]. The old pricing formula, applied to the new lower variable costs, triggered price reductions which no longer covered fixed costs. The company's performance slipped and then slid as Analog touched off a price war with competitors.

Fortunately, Analog has recovered. The example, though, is telling: An existing trait – fixed costs low relative to variable costs – worked adequately well at the local, manufacturing level and was well integrated into other and widely separate areas of the company and industry. An improvement effort inadvertently modified the trait, resulting in lower performance for the company as a whole.

Once revealed connections between one part of an organization and another can seem either subtle or obvious. In either case, however, these connections are often only revealed when something goes wrong.[2] Connections that knit local activities together into an entire company form an invisible web. Like

[1] The factor of two was gleaned from the history of past prices and variable costs. In an interview an Analog manager said they multiplied by 315%. The researchers believe that perhaps the manager meant 315% of labor costs. This probably translates to about twice variable costs. (Nelson Repenning, 1997, personal communication).

[2] In the case of Analog the key link between the TQM program and the pricing policy was not revealed until it was clear that something had gone drastically wrong. Even today, the link between the TQM program and the financial problems remains highly controversial at the company.

unmapped, long-buried gas mains in a city, hidden organizational connections make improvement efforts hazardous.

Organizations are too complex to map. But this does not mean organizations cannot be improved. Evolution is a process that in the biological realm has consistently improved extraordinarily complex organizations (organisms) – sometimes in remarkably short periods of time as the unexpected emergence of drug-resistant bacteria attests. Computer simulations of evolutionary processes, strongly suggest that biological evolution is only one example, albeit the archetypal example, of more general processes that can improve complex systems (Holland 1995, Koza 1992, Goldberg 1989). There is every reason to believe that evolutionary processes can be harnessed by managers to create organizations that are ever better.

2. The Elements of Organizational Evolution.

Understanding evolution – whether organizational or biological -- means understanding three things: (1) The "genetic material" of evolution, (2) how novel genetic material can arise and (3) how genetic material can be manipulated so that "children" can surpass "parents".

Genes and Policies: the Genetic Material of Evolution. A significant achievement this century in genetics was pinning down precisely what fundamentally is changing during evolution. The answer is genes, meaningful segments of DNA molecules.[3] A gene forms a template which is used to create a stream of proteins in the cell[4]. These proteins form cellular structure, raw materials and enzymes that speed reactions. A gene produces a continuing flow of activity in the cell.

The organizational counterpart of a gene is an *idea*. Evolutionary biologist Richard Dawkins uses the term "meme" for an idea that can evolve and be passed from person to person (Dawkins 1990). Today, the new field of "memetics" is concerned with all such ideas from little tuneful commercial jingles to the idea of a wheel. We do not have to be quite this inclusive, since we are concerned only with ideas that control organizations. Jay Forrester has called these ideas "policies" (Forrester 1961).

By "policy" Forrester means any rule – implicit or explicit – that produces decisions. For example, a pricing policy might be: "Multiply variable costs by 2". A manufacturing policy might be: "Decrease production when inventories rise". Some policies are explicit, perhaps even written in policy manuals, but most policies are implicit, rarely if ever articulated and stored only in the heads of a company's employees. Policies guide pricing, marketing, budgeting, accounting, production, research, development, construction, acquisition and every other category of activity undertaken by an organization. Policies, like genes, produce a continuing stream of action. Policies, like genes, are a fulcrum on which evolution can operate. If policies evolve, so will the company.

Mutation and Innovation. In the popular press and even among some evolutionary biologists, mutation is considered the key to evolution. In fact, most mutations are deleterious. As a consequence, even primitive cells possess sophisticated "error checking" processes that make mutations in genes exceedingly rare. Estimates vary, but roughly a mutation occurs only once in every 1 million to 100 million cell divisions (Brown 1995, Smith 1989, Nei 1987).

---

[3] Some organisms, particularly certain viruses, carry their genes in RNA. Even in these cases, however, RNA is usually converted to DNA within the host cell so that it can be transcribed using the usual machinery.

[4] Arguably the term "gene" could also include control regions on a DNA molecule.

The problem with biological mutation is that it not only can create a slightly better gene, it can also produce a much worse gene. This problem becomes more severe as the fitness of the gene increases. Evolutionist John Holland has recently offered a homey illustration of this point: Randomly changing an ingredient of your favorite recipe is unlikely to improve the dish (Holland 1995).

Turning to human organizations, consider the business policy: "When inventories are low raise prices". A random mutation might be to replace the word "low" with the word "yellow": The policy "When inventories are yellow raise prices" is unlikely to improve matters, unless, perhaps, the business happens to involve selling ripening bananas. Most changes to policies are either misunderstandings or improvement attempts. Changing "low" to "yellow" might result from a mistake or misunderstanding and is clearly similar to a biological mutation. Less obviously, perhaps, improvement attempts, too, are similar to mutations: Because we don't understand our complex organizations, "improvements" have unpredictable – that is random – effects with respect to the functioning of the business as a whole. Improvement attempts usually "make sense" – such as Analog's TQM program. This means that of all possible "random" changes, some are more likely to be pursued, but still are no more likely to succeed. The case is similar for biological mutation where certain kinds of mutations are also more likely than others.

If innovations in policy correspond to mutations in genes it is likely that organizations need to control innovation for the same reason that organisms control mutations, and for the same reason that you stick to the recipe: A policy innovation is more likely to disrupt a well functioning business policy than it is to improve it.

Recombination and Learning. Compared to their miserly attitude toward mutation, cells are profligate with recombination. Consider your own cells. Half the chromosomes in each of your cells comes from your mother and half from your father. In recombination, a chromosome from your mother crosses with a similar chromosome from your father to create a chromosome which is part Mom's and part Dad's. You package these combo-chromosomes into your sperm or your eggs (depending on obvious parameters) and pass them on to your children. In the process you might just produce a child who is better than either of your parents, just as combining the best part of eggs Benedict with the best part of a sandwich produces an Egg McMuffin, which is better than either "parent", at least if you happen to be driving in the morning.

A numerical example might help. Say we have two "chromosomes" made of 5 digits each *4,987* and *7,329*. A recombination event at the coma will produce two children: *7,987* and *4,329*. Assuming that bigger numbers are better numbers, one of the children is better than either parent (or course, the other one is not as good). The process of recombination can speed up evolution by combining good parts of different parents to produce offspring that are even better. Holland (Holland 1992, Goldberg 1989) has termed this the "building block" hypothesis.

The organizational analog of recombining chromosomes or culinary dishes is the recombination of policies. "Policy recombination" goes under the more common name "learning". Say that my pricing policy is "increase prices when inventories are low", and your pricing policy is "set prices at a margin of 200% over variable costs." My policy is good because it responds well when demand exceeds supply causing inventories to fall. Your policy is good because you will never price below costs. Perhaps you will learn from me, combining my idea with your own. Your policy afterwards might be "set prices at a margin over variable costs, but raise that margin when inventories are low". Your new policy still ensures that prices never drop below costs, but it also responds to supply and demand.

3. Simulations and insights

Recombination is learning. It occurs whenever one person adopts a policy or part of a policy from another person; and under the right conditions, it can be the workhorse of organizational evolution.

Unfortunately, organizational evolution is difficult to track in the "wild": Learning is difficult to measure, policies difficult to explicitly articulate, and the process occurs over a period of months or years. Computer simulations make it possible to study evolutionary learning under "laboratory conditions" where the pace can be quickened, policies can be explicitly specified, and learning can be closely monitored. As we shall see learning by itself is not sufficient to guarantee improvement.

Learning drift. We will simulate a company that is somewhat like a Microsoft or a Lotus Development: The simulated company has a number of different products – say, word processor, spread sheet, data base, presentation, and organizer. The company develops and releases a sequence of new versions of each product. Each release of each product is modeled as a simple system dynamics project model (Abdel-Hamid and Madnick 1991, Cooper 1980, 1993). As programmers code, they empty the stock of unwritten code, producing correctly written code. Unfortunately, the programmers also produce bugs, which eventually are discovered and then need to be rewritten. Mathematically,

$$\text{Code to write} = \int (\text{Discovering bugs} - \text{Writing code})\,dt$$

$$\text{Writing code} = \text{Programmers} * \text{Productivity}$$

$$\text{Programmers} = \{\textit{determined by agent-managers}\}$$

$$\text{Productivity} = 2$$

$$\text{Discovering Bugs} = \text{Undiscovered bugs} / \text{Bug discovery time}$$

$$\text{Bug Discovery Time} = 0.25$$

$$\text{Undiscovered bugs} = \int (\text{Creating bugs} - \text{Discovering bugs})\,dt$$

$$\text{Creating Bugs} = \text{Writing code} * (1 - \text{Quality})$$

$$\text{Quality} = 0.7$$

$$\text{Correct code} = \int (\text{Writing code correctly})\,dt$$

$$\text{Writing code correctly} = \text{Writing code} * \text{Quality}$$

The dynamics of a release are simple: The stock of correctly written code increases until it reaches (almost) 100%. The more programmers, the faster the project is completed. After the version is released, programmers begin work on a new release.

As the third equation might suggest, the number of programmers is determined by managers (agents) which learn from one another through recombination. Each manager has in mind a number between 0 and 15 which specifies the number of programmers he would like to hire. A manager learns in a process of "plasmid-type" recombination: The learner recombines his pre-existing policy (expressed as a base-2 integer) with a random portion of his teacher's policy. This recombination is similar to plasmid dissemination and recombination in that (i) the donor's plasmid (policy) is unaffected, (ii) both the recipient and the donor survive the process, and (iii) no children are created (Summers 1996, *cf.* Banzhaf 1998). To phrase this more operationally, two managers might engage in conversation and one of them might come away with a modified idea.

If we are interested in shipping as quickly as possible it is intuitive that the highest number of programmers (fifteen in our simulation) is best. However, we do not let the managers in the model have this intuition. What we want to know is whether they converge on the right policy by learning from one another. A typical simulation appears as Figure 1.

Two observations can be made: (1) Managers are not tending toward the optimum number of programmers (i.e. fifteen), but (2) the managers do converge.

These results can be explained as follows. Learning is not moving managers toward the proper answer, because learning is undirected. The simulated managers are in the position of real managers, who operate in a complexity significantly beyond their comprehension. In the simulation managers copy parts of each other's ideas, but without direction and without foreknowledge of what is optimal. Which ideas they copy is random with respect to the performance of the company.

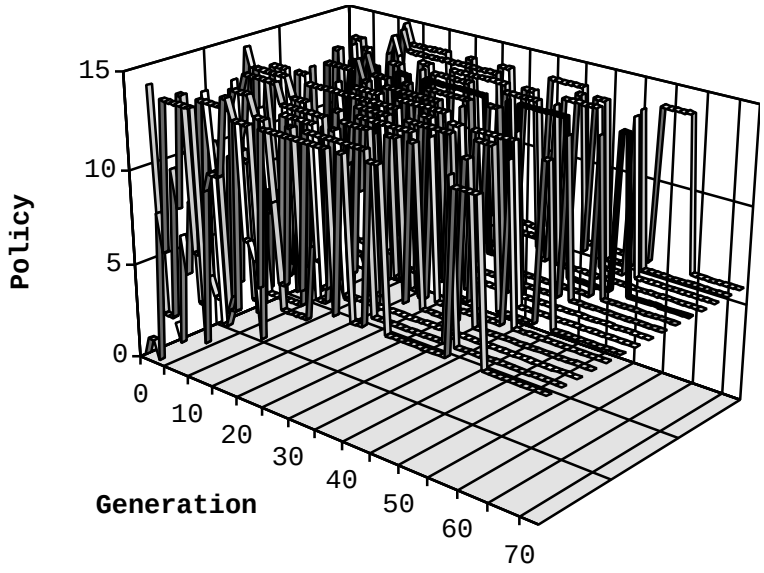

Figure 1: Learning drift: Each line represents the evolution of one managers beliefs about the proper number of programmers over the course of 50 learning generations.

Convergence is a mathematical property of the simulation [5]. But convergence is also an organizational property. Put a group of managers together in a room to decide something and a consensus is sure to emerge eventually, as long as the managers have some common ground. Consensus is dependable enough that consultants can make a living from it by guaranteeing that they will bring a group of managers to agreement. The nature of the consensus (and whether it is good or bad) is unpredictable, but consensus is assured.

Evolutionary systems operating without knowledge or direction are subject to random convergence. Geneticists call the biological case "genetic drift". We label the organizational case "learning drift". A

[5] Lost alleles are not re-introduced (there is no mutation in the simulation), and there is a finite probability that an allele in any given position will be lost in each generation. When all alleles except one are lost for a particular position, recombination simply continues to reproduce the surviving allele in each learner. The probability reaches 100% that eventually only one allele will survive in each position.

pure learning company drifts because the company is too complex for people to know what they should be learning.

Pointing and Pushing. How can evolution create a better company if no one can be relied upon to specify what people should be learning? Consider nature's answer.

Nature never specifies *what* to inherit; it specifies *who* to inherit from. Nature does not tell baby tigers which genes to choose, it tells them *whose* genes to choose – namely, its parents'. More conventionally, nature does not tell an amorous adult tiger *which* tiger to (re)combine with, instead it limits the choice to those other tigers who happen to be alive, those who have been fit enough to survive. Once Nature selects with *whom* tigers should should team, it *pushes* team-formation (so to speak) via the sex drive.

Similarly, organizations do not need to specify *what* to learn, but only *who* to learn from. Having pointed out the good teachers, the organization then only needs to push learners to learn. We call organizational features that promote learning from selected teachers "pointing and pushing mechanisms". These mechanisms are the organizational counterparts of natural selection and the sex drive.

Organizations must somehow find or invent pointing and pushing mechanisms that will single out people who have been successful and will encourage others to learn from (or imitate) them. As one example of a potentially powerful pointing and pushing device, consider position in the management hierarchy. Managers expend an enormous amount of time trying to figure out why their bosses do the things they do, trying to understand what their bosses think, and what attitudes they have. People seem *pushed* to imitate or learn from people *pointed-out* by high position. People want higher position and they believe they can get it by imitating (or learning from) the people who have it. If the people who have position attained it because they were successful, we have the makings of an effective evolutionary system.

We can add position to our simulation : Initially, a random positions is assigned to each manager. As the simulation proceeds, position changes based on performance, measured by how long it would take that manager to ship a product. The manager with the best performance (i.e. the manager who wants the most programmers) is promoted by a factor p^1, and the manager with the worst performance is demoted by a factor of p^{-1}. Managers performing in between these two extremes are promoted or demoted by a distribution of powers of p ranging between 1 and -1. In the following simulation $p=4$. Hence, a manager with position of 2 who performed best would be promoted to a new position of $8 = 2*4^1$. Had that same manager performed the worst, his new position would have been $0.5 = 2*4^{-1}$

In each product generation, each manager has an opportunity to learn from another. A learner chooses a teacher probabilistically based on the teacher's position. The probability of a manager learning from fellow-manager j is.

$$p(\text{learning from } j) = \frac{\text{position}_j}{\sum_i \text{position}_i}$$

where i ranges over all managers.

A typical simulation run now shows both convergence and improvement. In Figure 2, managers converge at the maximum number of programmers, fifteen, which is optimal in this situation.

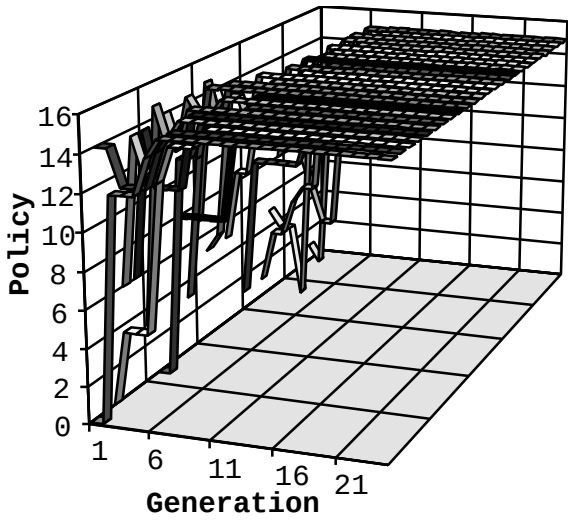

Figure 2: When managers tend to learn from a superior and when promotions are based on successful performance, improvement results. The graph indicates the policy of twenty-five managers in the simulation over a number of generations.

We do not mean to suggest that hierarchy is the only or even a desirable pointing and pushing device. We *do* mean to suggest that *a* pointing and pushing device is necessary for organizational evolution.

Defining success and team learning. A pointing and pushing device depends on the ability to tell whether someone has succeeded or not. The definition of success is up to the organization and could include such elements as personal improvement and happiness as well as the more traditional profit-based criteria.

For evolution to work organizations must point to people who are successful. In most organizations, individual success is difficult to gauge. Most people's performance depends on the performance of others. Individuals in organizations work in teams.

One of the more hopeful insights of our simulations to date is that team-based pointing and pushing mechanisms can be as effective as individual-based ones. Consider one last simulation : Instead of having individual managers managing each product, we will let teams manage the products. Promotions will now be based on team performance : All individuals on the best performing team will be promoted to a position which is a factor of p greater than their prior position. And, all individuals on the worst performing team will be demoted by a factor of p^{-1}. At the end of each product generation, teams will be mixed randomly. Even though wise and foolish policies on the same team receive the same (factor) promotion ; mixing teams provides enough discrimination to pick out the better policies, as shown in the following simulation.

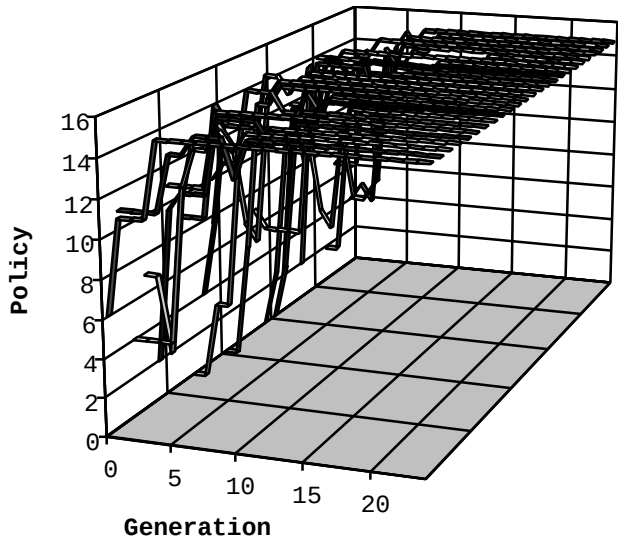

Figure 3. Convergence and accuracy are similar whether the pointing and pushing mechanism is individual-based or team-based.

4. Conclusion: Evolutionary management

In this paper we have laid a foundation that utilizes advances in evolutionary algorithms to understand what is possible in organizational evolution. Much remains to be done, however, these early results suggest that managers can create mechanisms which will make their companies evolve quickly in whatever direction desired. We term this activity "evolutionary management". Evolutionary management replaces the impossible task of understanding a complex organizations with the merely difficult task of understanding organizational evolution.

References

Abdel-Hamid, Tarik and Stuart Madnick. 1991. *Software Project Dynamics: An Integrated Approach*. Englewood Cliffs, NJ: Prentice Hall.

Banzhaf, Wolfgang, Peter Nordin, Robert E. Keller, Frank D. Francone. 1998. *Genetic Programming*. San Francisco, CA: Morgan Kaufmann Publishers.

Brown, TA. 1992. *Genetics: A Molecular Approach.* New York: Chapman & Hall.

Cooper, Kenneth G. 1993, "The Rework Cycle: Vital Insights Into Managing Projects." IEE Engineering Management Review. Vol. 21, Number 3, pp. 4-12.

Cooper, Kenneth. 1980. "Naval Ship Production: a Claim Settled and a Framework Built". *Interfaces*. Vol. 10, No. 6.

Dawkins, Richard. 1990. *The Selfish Gene.* Oxford: University of Oxford Press.

Forrester, Jay W. 1961. *Industrial Dynamics.* Portland, OR: Productivity Press.

Goldberg, David E. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning.* Reading, MA: Addison_Wesley Publishing Company, Inc.

Holland, John H. 1992. *Adaptation in Natural and Artificial Systems , Second Edition.* Cambridge, MA: The MIT Press.

Holland, John H. 1995. *Hidden Order: How Adaptation Builds Complexity.* Reading, Ma: Addison-Wesley Publishing Company, Inc.

Kofman, Fred, John D. Sterman, and Nelson P. Repenning. 1997. "Unanticipated Side Effects of Successful Quality Programs: Exploring a Paradox of Organizational Improvement". *Management Science*. Vol. 43, No. 4: 503-521.

Koza, John R. 1992. *Genetic Programming.* Cambridge, MA: The MIT Press.

Nei, Mastoshi. 1987. *Molecular Evolutionary Genetics.* New York: Columbia University Press.

Smith, John Maynard. 1989. *Evolutionary Genetics.* New York: Oxford University Press.

Summers, David K. 1996. *The Biology of Plasmids.* Oxford, U.K.: Blackwell Science Ltd.