Search for Architecture in Complex Worlds: An Evolutionary Perspective on Modularity and the Emergence of Dominant Designs[1]

SENDIL K ETHIRAJ
University of Michigan Business School
701 Tappan, D5203
University of Michigan, Ann Arbor MI 48109
Ph. 734-764 1230
Fax. 734-936 8716
Email: sendil@umich.edu

DANIEL LEVINTHAL
The Wharton School of Business
3620 Locust Walk, Suite 2000
University of Pennsylvania, Philadelphia PA 19104
Ph. 215-898 6826
Fax. 215-898 0401
Email: levinthal@wharton.upenn.edu

November 2002

[1] We thank seminar participants at Stanford University, Northwestern University, New York University, the University of Michigan-Santa Fe Institute Workshop, and The Lake Arrowhead Conference on Computational Modeling in the Social Sciences for useful comments and suggestions.

Search for Architecture in Complex Worlds: An Evolutionary Perspective on Modularity and the Emergence of Dominant Designs

Abstract

The problem of designing, managing, and coordinating the efforts of different parts of large-scale complex systems is central to the management and organizations literature. While there is significant attention directed at understanding how and why modular design structures help manage complexity (Baldwin and Clark, 2000; Langlois, 2002), there is little understanding on whether and how an appropriate modularization is achieved and how this process interacts with the evolution of the complex system. We set up four archetypal complex systems that vary along the two dimensions of hierarchy and decomposability. The key task of the designer(s) is to discover the true structure of the complex system, i.e., correct number of modules and correct mapping of policy choices to modules. Our results indicate that boundedly rational designers, with fairly simple rules guiding their behavior, are able to converge and stabilize on the true structure as long as systems are hierarchical. We also investigate the interaction of the process of architecture evolution (i.e., redrawing of module boundaries) with the process of module innovation (i.e., local adaptation and recombination). Not surprisingly, we find that module innovation is increasing (both in rate and final asymptote) in the ability to discover the true architecture. A more surprising result is that the process of module innovation generates performance improvement even when the system has not achieved the true structure. This suggests that discovering the true structure is not a necessary pre-condition for realizing the benefits of modularity. Indeed, the effectiveness of recombination of modules across organizations is primarily driven by organizations identifying a common architecture rather than the identification of the correct architecture. This raises the interesting possibility that recombination, rather than being an outcome of the emergence of dominant designs, is itself an engine that facilitates the emergence of dominant designs.

1. Introduction

Simon's (1962) the Architecture of Complexity provides the foundation for viewing organizations, as well as other systems such as products or technologies, as complex adaptive systems (Cohen and Axelrod, 1999). A central element of Simon's argument is that the fundamental features of organizations are hierarchy, the fact that some decisions or structures provide constraints on lower-level decisions or structures, and near-decomposability, the fact that patterns of interactions among elements of a system are not diffuse but will tend to be tightly clustered into nearly isolated sub-sets of interactions. These dual properties of hierarchy and near-decomposability are argued to enhance the evolvability of such systems.

We do not take issue with Simon's argument. Indeed, we wish to return to it as a starting point for a consideration of how, in a world of boundedly rational actors, the conscious structuring of interactions might evolve. That is, even if we accept Simon's postulates that effective adaptation requires hierarchy and near-decomposability, if we are mindful not only of Simon the "system designer" but also of Simon (1955) the forceful spokesperson for bounded rationality, how can we presume that organizations will be able to identify and uncover some true, latent structure of hierarchy and near-decomposability?

This is a question that has become particularly salient in light of recent interest in the power of modular designs (Baldwin and Clark, 2000; Sanchez and Mahoney, 1996; Schilling, 2000). Inspired by Simon's (1962) work, researchers have advocated the adoption of modularity as a design principle to manage complexity (cf., Baldwin and Clark, 2000). The adoption of modular design principles entails the creation of semi-autonomous modules or units with stable and visible rules for communication and interaction between them. As long as the integrity of between-module interaction is preserved, module designers are free to engage in local (i.e.,

within-module) adaptation or innovation. Following this observation, research on modular structures generally documents two[1] main benefits of adopting modular designs. First, modularity allows module designers to engage in parallel innovation efforts while maintaining overall conformance to the interface specifications. For instance, in the context of product design, modular design principles allow the hard disk manufacturer and the microprocessor designer in the personal computer product system to engage in parallel and simultaneous innovation efforts. Similarly, organizational design or restructuring efforts can occur in parallel or in isolation as long as departments are relatively autonomous and inter-departmental interactions are well documented. The key pay-off here is in the reduction of design, development, testing, and integration time (Baldwin and Clark, 2000; Loch*, et al.*, 2001; Ulrich and Eppinger, 1999).

A second benefit of modular designs is the ability to mix-and-match modules to create best-in-class designs. With greater module variety, innovation is made possible simply through the recombination of existing modules to create a range of systems with different performance outcomes (Baldwin and Clark, 2000). Such recombination is a very important design instrument in the case of complex assembled products such as personal computers and automobiles. In the case of organizations, the transfer of "best practices" from one organization or sub-unit to another is the closest analogue of recombination (Szulanski, 1996).

Much of the research on modular designs has focused on uncovering their benefits, while relatively little research has grappled with the question of whether and how designers might

[1] Some authors (see Garud and Kumaraswamy, 1995) suggest a third benefit of modularity – the selective upgrading of modules. We believe that this is subsumed under the two benefits highlighted here since selective module upgrading minimizes disruption to other modules (first benefit) while simultaneously improving system performance through recombining the existing modules of the system with the upgraded module (second benefit).

discover the true modularization of a complex system.[2] In a notable exception, Schaefer (1999) concludes that the problem of optimal complex system design is isomorphic with a general class of problems in computer science called NP-complete[3]. Representing the problem of complex system design analytically, Schaefer (1999: 325) infers that, "it would seem unlikely that a firm could ever hope to uncover an optimal modular design partition for a complex product". Designers are simultaneously searching on the matrix of design parameters at four levels: (1) the "appropriate" number of modules; (2) the "appropriate" mapping of design elements to the modules; (3) the "appropriate" interfaces or interactions between modules; and, (4) the "appropriate" interactions among the design elements within each module. Under these circumstances, the ability to discover the true underlying structure cannot be taken for granted. In addition, if the benefits of modularity are conditional on the discovery of the true structure of the complex system, then the question of ease (difficulty) of complex system design becomes particularly salient.

We explore three interrelated questions regarding the design of complex systems. First, while Schaefer's (1999) work suggests that the problem of discovering the optimal structure of a complex system is, in general, impossible, there may be special cases of the problem for which

[2] Modularization, the act of modularizing, generally makes references to two processes (Baldwin and Clark, 2000; Langlois, 2002). One is to partition the elements of a system to correspond to the true, but typically imperfectly understood, structure of the pattern of interaction among the constituent elements of the system. The second is the specification of the interface among modules that encapsulates the interaction among the subsystems. In our formal analysis, we address the first of these two challenges. Consistent with this focus, much of the literature on modularity attends to the existence and value of partitioning elements of a system and not the issue of identifying the "language" of interface among modules (Baldwin and Clark, 2000; Parnas, 1972). We elaborate this point in section 3.

[3] An NP-complete problem belongs to a class of problems for which the time to achieve an optimal solution grows at a rate that is faster than a polynomial function of the number of variables (Garey and Johnson, 1990). For such problems checking a solution is much easier than finding a solution. For example, the subset sum problem is NP-complete: given a finite set of integers, determine whether any subset of them adds up to zero. It's easy to check a supposed answer to see if it's right, but there is no known way to solve the problem quickly. The minesweeper game in Windows computers is a well-known example of an NP-complete problem.

solutions are relatively easy. In addition, while identifying an optimal solution may be an intractable problem, it does not imply that a reasonable or *satisficing* solution is unattainable.

We examine how the ease (or difficulty) of discovering a "correct" solution to the design problem varies along the two dimensions – hierarchy and decomposability – of complex system structures. Our goal is to identify the boundary conditions for the possibility (or impossibility) of discovering the true structure or architecture of a complex system. The discovery of the true structure or architecture here is defined as discovering the correct number of modules and the correct assignment of functions to the respective modules.

Second, little research, if any, has devoted attention to understanding whether the realization of the benefits of modularity is contingent on achieving the correct modularization for a given system. If indeed, as Schaefer (1999) suggests, discovering the true structure of complex systems is not possible, then it is of enormous theoretical as well as practical importance to understand whether the benefits of modularity is contingent on the ability to discover the optimal or true structure.

Finally, if a critical benefit of modularity is the opportunity that modularity offers for recombination, then the evolution of modularity must be intimately intertwined with the question of the emergence of dominant designs. The literature on the evolution of firms, technologies, products, and industries has some bearing on this question (see Abernathy and Utterback, 1978; Clark, 1985; Klepper, 1997; Tushman and Anderson, 1986). A typical characterization of technological and organizational evolution suggests a:

> "transition from an early, fluid state, to one that is highly specific and rigid. In the early fluid state, performance criteria for new products are not well defined and market needs or process difficulties are approached through a variety of different product or equipment designs. Innovation is relatively rapid and fundamental…As development proceeds, however, *technological diversity gives way to standardization. Particular design approaches achieve dominance* [emphases

> added], production volumes increase, and performance criteria and processes are more clearly specified" (Clark, 1985: 235-236).

If the notion of a dominant design, at least in the case of products, refers to the architecture of a product and hence to a particular modular structure, the important question is what drives standardization and how do dominant designs emerge. Unlike existing arguments in the literature that posit recombination as a by-product of a shared design architecture (Baldwin and Clark, 2000), we explore the possible role of recombination in influencing which among the alternative system architectures emerges as the dominant design.

2. Complex Systems and Modularity

The term complex system is widely employed in the management and organizations literature to describe a range of entities from organizations to products. However, there is little consensus on the exact properties of a complex system. Following Simon (1962: 468), we adopt a working definition of a complex system as one made up of a large number of parts that interact in a non-simple[4] way. The complexity that stems from a large number of parts interacting in non-trivial ways is two-fold. First, the large number of parts creates difficulty in comprehending the structure that binds them. Second, even if one is able to uncover and comprehend the structure that binds the parts, anticipating the effects of the interactions on system behavior and performance is far from simple (Cohen and Axelrod, 1999; Kauffman, 1993). The complexity increases as the system gets larger since designers need to first discover which parts interact with which others and then discover the nature of the interaction relationship.

Even complex systems, Simon (1962) argues, are rarely characterized by saturated interactions among all its elements. They often resemble hierarchies of sub-systems and sub-sub-

[4] For instance, predicting system behavior is relatively easy if the interactions between the parts are linear. On the other hand, if parts interact such that the relationship within and between then is non-linear, i.e., positive over some range and negative or unrelated over other ranges, then predicting system performance becomes a difficult problem.

systems. The property of hierarchy is not found by chance, but favored by evolutionary selection processes. Using the fable of the two watchmakers, Simon (1962) suggests that complex systems that resemble hierarchies tend to evolve faster and toward a stable, self-reproducing form as compared with non-hierarchic systems.

The meaning of hierarchy here is less in terms of authority flows from top to bottom as often employed in describing organization structures. Instead, hierarchy is used in the sense of nested hierarchies (Baum and Singh, 1994) where there is a precedence ordering of modules or sub-systems in the system. In the context of design (either products or organizations), the line of hierarchy denotes the flow of information, or constraints, from the immediately higher module or unit. Similarly, in an organizational context, Simon's (1947) notion of decision premises has the property of a nested set of constraints on organizational decision-making. The important function of hierarchy is not only to resolve conflicts between sub-systems, but also to facilitate learning through trial and error by allowing systematic and orderly local search and exploration. On the other hand, non-hierarchic systems, which display no particular order in their configuration, make local search and trial and error learning much more difficult to accomplish. Hierarchy enables the recognition of progress towards one's goals and the evolution of a system through several intermediate stable configurations.

Second, Simon (1962) observed that it is also unlikely that complex systems can be decomposed into a small number of completely independent units or sub-systems. Instead, complex systems in the real world are more often nearly decomposable, i.e., they are characterized by strong interactions among some elements and weak interaction among others (Simon, 1962: 474). If this statement is converted into a design principle, the important goal of system designers is to discover the interactions among the elements, group all strongly

interacting elements and create a small number of weakly interacting modules (Thompson, 1967).

The following section describes a formal modeling structure that we set up to explore first, the adaptive search for modular structures within complex systems and, second, the implications of such search processes for system performance. Specifically, we sought to carefully examine the relationship between the structural properties of complex systems (i.e., hierarchy and decomposability), the ability to discover the true structure (i.e., system architecture), and the effectiveness of adaptive processes. We set up four archetypal structures of complex systems: (1) hierarchical and nearly decomposable (cf. Figure 1a); (2) non-hierarchical and nearly decomposable (cf. Figure 1b); (3) hierarchical and non-decomposable (cf. Figure 1c); and, (4) non-hierarchical and non-decomposable (cf. Figure 1d). In each of the four structures, we contrast and examine two performance outcomes. The first outcome of interest is the ability of boundedly rational designers to converge on evolutionarily stable system architectures. Second, we examine the efficacy of adaptation efforts since successful adaptation is ultimately an important objective in designing complex adaptive systems. We initially analyze the search problem by considering organizations (systems) as operating in isolation. Since the benefits of recombination are dependent on the interaction of a population of systems, we then examine how the problem of search for the true structure interacts with the process of recombination of modules among a population of systems and, ultimately, how the joint processes of the search for architecture and recombination impact average system performance.

3. Model

3.1. Modeling the "true" structure or architecture

Simon's (1962) exposition on the architecture of complexity forms the basis for our modeling the structure of a complex system with some degree of inherent modularity. We

assume that a system is composed of a set of N policy choices, some subset of which is interdependent. Figures 1a-1d characterize different interaction structures that might describe complex systems. In the figure, an alphanumeric notation represents each policy choice. The alphabetic portion denotes the module, while the numeric portion denotes the respective policy choice. The x's in each row-column intersection identify interdependence between policy choices. Reading across a row, an "x" indicates that the row variable is affected by the column variable. Conversely, reading down a column, an "x" indicates that the column variable affects the row variable. Therefore, x's positioned symmetrically above and below the principal diagonal represents reciprocal interdependence between policy choices.

Within each module, each policy choice is tightly coupled with other policy choices in the same module --- what Thompson (1967) termed reciprocal dependence. Figure 1a depicts a system that is *hierarchical and nearly decomposable*, i.e., modules 2 and 3 have a weakly coupled interface relationship with the next higher module, denoted by a single x below the principal diagonal. The presence (absence) of hierarchy is identified by the asymmetry (symmetry) in between-module interaction. The near decomposability is a function of the strength (magnitude) of between module interactions. If there are no interactions between modules, then the system is fully decomposable. In Figure 1a, policy a4 influences the payoff associated with policy b1. This corresponds to the hierarchical block task structure described in Baldwin and Clark (2000: 59-61) and describes sequential or hierarchical interdependence between modules (Thompson, 1967).

<< Insert Figures 1a-1d here >>

Since complex systems can vary along the dimensions of hierarchy and decomposability, we set up three additional structures. Figure 1b denotes a *nearly decomposable but non-hierarchical system*. The system is still nearly decomposable, since the interaction within modules is stronger than interaction between modules. However, the system is not hierarchical, since there is no precedence ordering between the modules. Modules 'a' and 'b' are characterized by reciprocal interdependence (Thompson, 1967) and symmetry in between-module interactions. Figure 1c describes a *hierarchical, but non-decomposable system*. The interaction between modules 'a' and 'b' and modules 'b' and 'c' is as strong as the interaction within the respective modules. In this system, however, hierarchy is preserved since there is only sequential interdependence between modules, i.e., module 'a' affects module 'b', but not vice versa. Finally, Figure 1d represents a *non-hierarchical and non-decomposable system*. The degree of non-decomposability in Figure 1c and Figure 1d is held constant as the total number of interactions off the principal diagonal is held constant. Whereas Figure 1c is still a hierarchical structure, Figure 1d is not.

A system or organization is represented as a row vector of N attributes or policy choices $[a_1, a_2, a_3, \ldots a_n]$. For simplicity, in our model, each policy choice can take on two possible values (0, 1). Thus, the universe of organizations or systems comprises 2^N possible forms. For instance, if we consider an organization to be a complex system and its incentive system to be a policy choice, then a setting of 1 might represent individual incentives and a setting of 0 might connote group incentives. It follows that different settings for the policy choices of the complex system have different performance implications. Continuing with the organizational example, choices about the incentive system are likely to have interactions with the accounting system, quality of employee effort, organization of work spaces, and so on. For instance, it is conceivable that

group-based incentives entail collective workspaces, which in turn requires better group level monitoring and regulation of employee effort. In other words, some combinations of policy choices may yield performance improvements while others may undermine it (Ichniowski*, et al.*, 1997; Macduffie, 1995).

The performance of the system depends on the settings (1s or 0s) of the policy choices and the interactions among them as described by Figures 1a-1d. When there are no interactions between policy choices, each policy makes an independent contribution to system performance. As the interactions between policy choices increase, the contribution of each policy choice to system performance becomes increasingly interdependent. The complexity of the system stems primarily from the presence of these interactions and the fact that the nature of the interactions are likely to be poorly understood, both in the sense of what elements of the system interact with what other elements and, even when the existence of interactions are known, what is the functional form guiding the interactions among the policy choices. For these reasons, the performance landscapes of such complex systems tend to be rugged and non-linear with actions on one policy choice having ripple effects on other policy choices (Levinthal, 1997).[5]

Returning to Figure 1a with this discussion in mind, we see that the contribution of an individual attribute, a_i, depends on other attributes as indicated in Figure 1a. Thus, attribute a1 depends on attributes a2, a3, and a4. In contrast, attribute b1 depends on 4 other attributes (a4, b2, b3, b4). As a result, attribute a1 can result in 16 possible levels of performance, depending on its own value (a 0 or 1) and the value of the 3 other attributes on which it depends, while

[5] Our characterization of a performance landscape here bears a close link to the work of Kauffman (1993). A critical distinction though, lies in the fact that the systems that we explore have some unknown, but inherent structure, while Kauffman examines purely random structures of interactions. Thus, the modeling is a blend of the conceptual ideas of Simon (1962) and the technical apparatus of Kauffman (1993).

attribute b1 can take on 32 possible levels of performance, depending on its own value and the value of the 4 other attributes on which it depends.

A similar logic was applied in creating systems of different sizes. For instance, if N=90 and the true structure comprises nine modules, then near decomposability as in Figure 1a is characterized by an interaction between policy choice 10 of the first module and policy choice 1 of the second module, policy choice 10 of the second module and policy choice 1 of the third module and so on. The same principle was applied to create systems representing Figures 1b-1d.

The performance contribution (ω_i) of each policy choice (a_i) is determined both by the state (0 or 1) of the i^{th} policy choice and the states of the j other policy choices on which it depends. Thus,

$$\omega_i = \omega_i(a_i; a_i^1, a_i^2 \ldots a_i^j)$$

The value of ω_i is treated as an *i.i.d.* random variable drawn from the uniform distribution[6] U [0,1] for each configuration of a_i and the 'j' other policy choices on which it depends. System performance Ω is a simple average of the ω_i over the N policy choices.

$$\Omega = \frac{1}{N}\left[\sum_{i=1}^{N} \omega_i(a_i; a_i^1, a_i^2, \ldots a_i^j)\right]$$

In addition to initializing the performance landscape as characterized above, the states (0,1) of the vector of policy choices a_i are drawn at random at the start of an experiment. Since, any single run is sensitive to the inherent randomness in both the initial states of the policy choices and the initialization of the performance landscape, we replicated each experiment 100 times with different starting seeds for both the specification of the performance landscape and

[6] The results are robust to alternative distributional assumptions. In particular, we have run the analysis with exponential and log-normal distributions and obtain results that are qualitatively similar to those reported here. These results are available with the authors.

the starting state of the system to remove the stochastic component endemic to any single run. The reported results, unless mentioned otherwise, are averaged over the 100 runs.

3.2. Modeling the search for the "true" architecture

We take an evolutionary perspective on the identification of the latent or true architecture of complex systems. In particular, we postulate that designers search on the space of possibilities to discover the "true" structure. We implement three search operators that we believe have clear organizational analogues: (1) splitting; (2) combining; and, (3) re-allocation[7].

Splitting of modules may be seen as the breaking up of existing departments into two or more new departments. As organizations and their departments grow larger, the same piece of information is likely to have to pass through a larger number of potentially redundant individuals before resulting in an action or decision. In such cases, splitting an existing department can help economize on information flow (Arrow, 1974). Splitting is also necessitated when a department is engaged in a number of unrelated activities, each of which requires different skills, people, and/or resources. In such settings, splitting facilitates differentiation (Lawrence and Lorsch, 1967) among organizational units to allow specialization to their specific contexts.

Combining two or more modules is simply the opposite of splitting. This is akin to combining or integrating two or more departments. From the seminal work of Lawrence and Lorsch (1967), we know that organizational structures are constantly balancing the contrasting

[7] Baldwin and Clark (2000: 132-142), focusing mostly on the issue of product design, identify six evolutionary search operators: (1) splitting an existing module into two or more modules; (2) substituting one module for another; (3) augmenting (adding) a module; (4) excluding a module; (5) inversion, taking previously hidden information and moving it up the hierarchy; and, (6) porting, use of a hidden module in more than one system. The first of these, splitting, we include. The second, substitution, does not address the change in the architecture of the system, but a change in functionality of a particular module and we include such processes in our subsequent analysis of recombination. Their third and fourth operators are clearly important, but entail changes in the boundaries of the system and essentially pose "theory of the firm" questions that we do not address here. The fifth and sixth operators, inversion and porting, while important in the context of product technologies have less clear analogies in the organizational context. Finally, we include the possibility of combining modules, an operator not considered by Baldwin and Clark (2000), but clearly an important process in both the organizational and product technology context.

forces of differentiation and integration. Indeed, there is empirical evidence that organizations often cycle between extended periods of increasing decentralization (differentiation) and increasing centralization (integration) (Cummings, 1995; Mintzberg, 1979; Nickerson and Zenger, 2002) suggesting that the combining operator might be an important counterpart to splitting. In product design, Takeishi and Fujimoto (2001) found that European automobile designs were characterized by increasing modularization over time, i.e., evidence for the splitting operator. At the same time, the Japanese automobile designs have been exhibiting increasing integration over time, i.e., combining two or more modules to create larger sub-systems (Takeishi and Fujimoto, 2001). From the available empirical evidence it seems that combining modules might be an important instrument of complex system design.

In addition to the splitting and combining of departments, organizations often simply *transfer* or *reassign* functions from one subunit to another. Such reorganization[8] does not lead to a more (as in splitting) or less (as in combining) partitioning of tasks, but simply involves the re-allocation or re-assignment of functions between departments. For instance, the shift from an firm organized along geographic lines to a product structure (Bartlett and Ghoshal, 1989) or from a functional organization to a product organization (Chandler, 1962) has no clear implication for the number of subunits but obviously would involve the wholesale reconfiguration of organizational activity. We do not examine such radical and discrete points of restructuring, but rather the reallocation of individual elements from one subunit to another, or what Eisenhardt and Brown (1999) would refer to as "patching". However, it is important to note that such

[8] There is little consensus on the meaning of the term reorganization. While most agree that reorganization involves change, there is disagreement on the magnitude and nature of change to which it refers. Eisenhardt and Brown (1999: 74-75) suggest that reorganization connotes large change while "patching" connotes "frequent, routine and mostly small changes." Patching, however, includes splits, additions, combinations, transfers, and exits. Our use of the term reorganization is generally synonymous with one type of patching, namely transfers --- i.e., the movement of activities from one subsystem to another.

incremental efforts at reorganization can, over time, cumulate in broad changes in the organization of a firm's activities.

Collectively, these operations of combining, splitting, and transfer are the mechanisms of change in both the number of partitions within the organization and the assignment of individual activities to a particular module, what we consider to be an organization's architecture. The search over the space of possible architectures is modeled as a form of local, incremental learning.

The process was implemented as follows. The module designers assume that within each module there should be strong, reciprocal interaction between policy choices. When designers engage in search for module improvement by flipping a policy choice (from 0 to 1 or vice versa), they examine performance change within the module. In particular, they observe the presence or absence of interactions among policy choices within the module as a result of the change in a policy choice. For instance, suppose the controls for adjusting the brightness and contrast of a PC monitor is randomly assigned at start to a module that also contains a number of microprocessor related policy choices. The team responsible for this module then engages in search to improve and refine this module. Upon flipping a policy choice representing a function of the microprocessor, the team finds that it has no effect on the brightness and contrast of the monitor. As a result, they are able to group the policy choices into those that were affected by the search and those that were not.

Consider a set of policy choices that are perceived to belong to a module, $(a_i, a_{\neg i}) \in M_\gamma$, where a_i represents a focal policy choice and $a_{\neg i}$ are the remaining set of policy choices, M_γ is a module of the system, where the system is defined by a set of modules, M = $\{M_\alpha, M_\beta, M_\gamma, \ldots M_\kappa\}$.

The performance of the module is given by $\frac{1}{n_{M\gamma}}\sum_{n_{M\gamma}}\omega_i(a_i;a_{\neg i})$ where '$n_{M\gamma}$' is the number of policy choices in module M_γ. Now, consider a single policy choice, $a_j \in M_\gamma$, that is flipped to a_j' and the resulting performance of each policy choice in the module is observed. Then let A be defined as a set of all policy choices such that,

$$A = \left\{ a_i : \omega_i(a_i; a_{\neg(i,j)}, a_j) \neq \omega_i(a_i; a_{\neg(i,j)}, a_j') \quad for \quad all \quad i \in M_\gamma \right\}$$

The set A identifies the set of all policy choices in M_γ whose performance changes as a result of flipping policy choice a_j. The designers then adopt a simple rule that all policy choices that were unaffected by the search do not belong to the present module. All such policy choices are then either *transferred* to a randomly chosen different module or *split* into a separate module if it constitutes a large enough set. If A turns out to be an empty set it means that the performances of all remaining policy choices in M_γ, ($a_{\neg j}$), were unchanged by the flip in a_j suggesting that a_j does not belong to the module. Thus, a_j is transferred to a randomly chosen module M_κ.

On the other hand, if A is a non-empty set then it means that the subset of policy choices in A are interdependent with a_j and all the remaining policy choices in M_γ do not belong in the module. Thus, in this latter case, the unchanged policy choices in M_γ are *transferred* into a randomly chosen existing module M_κ if they constitute less than half the total number of policies in M_γ; otherwise, the unaffected policies are *split* into a new module $M_{\kappa+1}$.[9]

[9] More formally, if,

$$\text{If A} = \{\,\}, \text{ then } a_j \in M_\kappa, \text{ otherwise,}$$

$$If \quad nA \leq \frac{nM_\gamma}{2} \begin{cases} (\neg A \cap a_{\neg j}) \in M_{\kappa+1} & otherwise, \\ (\neg A \cap a_{\neg j}) \in M_\kappa \end{cases}$$

Observing such patterns of influence does not assume that the designers understand cause and effect processes at the module level. The only behavioral assumption made is that the designers are able to observe the effect of their actions and then able to trace the "road to root cause" (Macduffie, 1997) to isolate policy choices that were not affected by the experimentation[10]. This is akin to debugging a software program. When a program refuses to run, the programmer knows there is problem and uses the debugger to spot the problem. The only assumption about the designer is that he or she is able to observe the outcome (i.e., the program does not run) and trace the source of the problem (i.e., debug), but not necessarily know the solution to the problem.

In each period, we also consider *combining* each module with another randomly chosen module. This is akin to two teams deciding to pool their resources. Again, the simple rule employed in deciding whether or not to combine is to assess whether changing a policy choice in each module affects the performance of the other module. The modules combine if changes in each module affect the other and remain separate otherwise. For instance, consider two teams representing two modules, M_α with policy choices $[a_1, a_2, a_3, \ldots a_i]$, and M_β with policy choices $[b_1, b_2, b_3, \ldots b_j]$. A randomly chosen policy choice from each module, a_i and b_j respectively, are flipped. The modules M_α and M_β *combine* to create a new module, M_γ, if the performance of

---

where, nA and nM_γ represents the number of policy choices in A and M_γ respectively.

[10] This involves some assumptions about the level of aggregation at which performance can be reliably measured. The local search process implemented here assumes that performance can be tracked at the policy choice level, i.e., differentiate between policy choices that were affected by the flip and those that were not. The reality check is whether in organizations (or product systems) one can track performance at the individual (or activity) level. Most organizations do have performance measurement systems, albeit noisy, to track individual performance. In terms of introducing noise in our model we can conceive of two kinds: not perceive interactions when they actually exist (Type I), and perceive interactions when they actually do not exist (Type II). The introduction of Type I noise into our model will slow down the discovery process but the results will be robust. However, if Type II noise is introduced then the search rules employed here will never cease searching and hence never reach stability. This would affect our results regarding the time to identify the true module structure (see Section 4.1.), but our remaining analysis would be robust to such a modification.

module M_β is affected by the flipping of a_i and the performance of module M_α is affected by the flipping of b_j.[11]

3.3. Modeling the benefits of modular designs

The literature largely emphasizes two main benefits of modular designs. The first is the ability to engage in parallel and autonomous adaptation or innovation within modules and the second is the ability to mix-and-match or recombine equivalent modules from different systems.

3.3.1. Within-module innovation or local adaptation

The local adaptation for module improvement is implemented as follows. In each period of the experiment, the actors within each module attempt to enhance the performance of their particular module. Actors are assumed to "see" the performance of their given module and can anticipate what incremental changes from the existing policy string would imply for module performance. Thus, adaptation at the module-level occurs through a process of off-line, local search implemented simultaneously in each of the modules (Marengo*, et al.*, 2000; Rivkin and Siggelkow, Forthcoming)[12]. Within each module, a policy choice is selected at random and actors within each module evaluate the efficacy of flipping the policy choice (0,1) by the criterion of improvement in module performance. The innovation is implemented if there is a perceived performance increase in the module. On the other hand, if there is perceived

[11] More formally, if

$$\frac{1}{n_{M\alpha}}\sum_{n_{M\alpha}}\omega_i(a_i;a_{\neg i},b_j^{'}) \neq \frac{1}{n_{M\alpha}}\sum_{n_{M\alpha}}\omega_i(a_i;a_{\neg i},b_j) \wedge \frac{1}{n_{M\beta}}\sum_{n_{M\beta}}\omega_j(b_j;b_{\neg j},a_i^{'}) \neq \frac{1}{n_{M\beta}}\sum_{n_{M\beta}}\omega_j(b_j;b_{\neg j},a_i)$$

Then, $M_\gamma = M_\alpha \bigcup M_\beta$

[12] In contrast, adaptation *across* modules is assumed to require experiential or on-line search. This capability of module-level search as suggested by Rivkin and Siggelkow (Forthcoming), may be a function of effective accounting systems that are able to facilitate the evaluation of module level performance. Indeed, activity based costing is widely deployed to track the performance of organizational sub-units (Cooper and Kaplan, 1992).

degradation (or no improvement) in performance, then the innovation is discarded. Such innovation attempts occur in parallel in each of the modules.[13]

3.3.2. Recombination of systems or substitution of modules

We explore recombination (substitution) as follows. Two systems are selected at random and two modules are randomly selected for recombination. We calculate the fitness of the candidate modules for exchange and substitute the lower performing module with the higher performing module[14]. Note that this is not recombination in the sense of a biological exchange of genes. The process modeled here may be seen as intelligent recombination of systems where information about performance of individual modules is available to the designer. The organizational analogue of such recombination could be the transfer of "best practices" within and between organizations (Szulanski, 1996), partial merger of two organizations (e.g., merger of Hewlett-Packard and Compaq and the subsequent liquidation of redundant divisions or sub-units), or joint venture between two or more organizations staffed by teams from each (e.g., the joint R&D venture between Intel and Hewlett-Packard for the development of the Itanium chip staffed by teams from both firms).

---

[13] More formally, consider a policy choice $a_j \in M_\kappa$ that is flipped to a_j'. Then if,

$$\frac{1}{n_{M\kappa}}\sum_{n_{M\kappa}}\omega_i(a_i;a_{\neg(i,j)},a_j') > \frac{1}{n_{M\kappa}}\sum_{n_{M\kappa}}\omega_i(a_i;a_{\neg(i,j)},a_j) \text{ then } \begin{cases} M_\kappa = (a_{\neg j},a_j') & otherwise \\ M_\kappa = (a_{\neg j},a_j) \end{cases}$$

[14] More formally, consider two functionally equivalent modules M_α^s and M_α^t (i.e., the two modules contain the same set of policy choices) from two different systems, 's' and 't' respectively, and '$n_{M\alpha}$' is the number of policy choices in each module. Then if:

$$\frac{1}{n_{M\alpha}}\sum_{n_{M\alpha}}\omega_i(a_i^s;a_{\neg i}^s) < \frac{1}{n_{M\alpha}}\sum_{n_{M\alpha}}\omega_i(a_i^t;a_{\neg i}^t) \text{ then, } \begin{cases} M_\alpha^s = M_\alpha^t & otherwise \\ M_\alpha^t = M_\alpha^s \end{cases}$$

The recombination implemented here can be seen as 2-point crossover where the two crossover points are defined by the beginning and endpoints of a randomly picked module (Holland, 1992).

3.3.3. Selection

System selection processes are modeled as being proportionate to fitness. That is, the probability that a system will be selected equals its performance level divided by the sum of the performance of all systems in the population at that time. [15] This is a standard assumption in the modeling of biological processes (Wilson and Bossert, 1971) and has been used in a number of models of organizational selection (Lant and Mezias, 1990, 1992; Levinthal, 1997).

4. Analysis

4.1. Experiment 1: Boundedly rational search for the "true" design

The first experiment seeks to answer the question of whether boundedly rational designers can discover the true architecture of a complex system.[16] The problem of discovering the true architecture entails discovering the set of interactions among the N policy choices and clustering policy choices that seem to have strong interactions with each other. Schaeffer's (1999) analytical work and Kauffman's (1995) simulations suggest that it is virtually impossible for boundedly rational designers to discover the set of interactions as the number of policy variables and the number of interactions increase. However, their work assumed a random distribution of interactions among the policy choices, which made the application of systematic

[15] We used the standard roulette wheel algorithm (Goldberg, 1989) for modeling selection. More formally,

$$p(s_i) = \frac{\Omega_i}{\sum_{i=1}^{S} \Omega_i}$$

where, $p(s_i)$, the probability of selecting the ith system is given by the ratio of the performance, Ω_i, of the ith system to the total performance of all 'S' systems in the population. The cumulative probability, $P(s_i)$, is then computed as,

$$P(s_i) = \sum_{j=1}^{i} p(s_j)$$

A total of 'S' random numbers 'r_S' distributed *i.i.d.* in the interval [0, 1] are drawn and the systems whose cumulative probability spans a random draw are selected according to the rule, $P(s_{i-1}) \le r_S \le P(s_i)$.

[16] The system architecture refers to the number of modules and their composition, i.e., grouping of policy choices. The true architecture refers to the architectures described by Figures 1a-1d.

and simple search rules ineffective. On the other hand, if complex system structures can be described along the twin dimensions of hierarchy and decomposability (Simon, 1962), then it is useful to examine the variation of complex systems along these two dimensions and examining whether and how different degrees of hierarchy and decomposability alter the possibility of discovering the "true" structure.

In the experiments reported here, we describe in detail the results from the hierarchical and nearly decomposable structure (Figure 1a), which is our baseline case and compare the results from the three alternative structures (Figures 1b-1d) against this baseline. In all cases, we vary the number of policy choices, N, the interactions among them, and hence the number of true underlying partitions, M. We make the simplifying assumption that the true underlying structure, for all values of N, comprises modules of equal size in order to reduce the combinatorics of the possible true latent designs we need to consider as we vary N and M. Therefore, if N=90 and the complex system structure comprises 10 modules then each module contains 9 policy choices.

In each set of experiments, designers are confronted with a system of a fixed number of N policy variables. The number of modules (M) is randomly specified subject to the constraint that $M \leq N$. As part of this initialization, policy choices are randomly allocated among these M modules. Figure 1e illustrates a perceived interaction matrix at the start of a typical simulation run. As can be seen from the figure, the starting design contains unequal sized modules with no systematic pattern of interactions among policy choices.

<< Insert Figure 1e here >>

The key performance variable we track is the number of simulation periods for the system to converge on the true underlying architecture, i.e., the correct number of modules and the correct mapping of policy choices to the modules. This is an important performance metric since, as Simon (1962) rightly points out, an important goal of complex system design is to reach evolutionary stability. In the absence of any degree of stability, the efficacy of adaptation efforts is impaired. Intelligent adaptation requires stability as well as change (Levinthal, 1991). A stable architecture is also a necessary condition for effective recombination; if the structure of modules is in flux, then the population of organizations (systems) lacks the basis for compatibility that recombination requires.

Figure 2 plots the percentage of systems (over 100 runs) that converge on the true structure as the simulation progressed in each of the four archetypal structures. The figure shows that as long as the system is hierarchical, even at extreme levels of non-decomposability (Figure 1c) where all policy choices of one module affect all policy choices of the immediately succeeding module (i.e., all policy choices of module 1 affects all policy choices of module 2 and so on), the process of local adaptation is always able to discover and stabilize on the true structure. On the other hand, when we introduced reciprocal interaction between modules, i.e., system structure is non-hierarchical (Figures 1b and 1d), the system never manages to reach a stable state. In the non-hierarchical and nearly decomposable system (Figure 1b), the system converges on the true architecture most of the time, but the violation of hierarchy triggers a process of continual search as reflected in the non-monotonic plot. In the case of the non-hierarchical and non-decomposable system (Figure 1e), the violation of both principles (hierarchy and decomposability) results in the true structure never being identified.

<< Insert Figure 2 here >>

We examined the sensitivity of the results in Figure 2 for changes in the size of the system (i.e., N) and the number of underlying modules (i.e., M). Table 1 presents the results of the search for the true structure for hierarchical and nearly decomposable systems (Figure 1a) as M and N are varied. The first column, M, presents alternative "true" structures for a given N. For instance, a system of 30 policy choices can be decomposed into 3 modules of 10 policy choices each, 5 modules of 6 policy choices each and so on. We explored the sensitivity of our results to alternative true structures. The set of columns, labeled N, reflect different settings for N ranging from 30 to 900. Each of the entries under these columns represent the average number of time periods (standard deviation in parentheses), averaged over 100 distinct runs, for convergence on the true structure (cf., Figure 1a) from a random starting point (cf., Figure 1e).

<< Insert Table 1 here >>

We find that the results are quite robust to changes in both the size of the system and the number of underlying modules. Examining across columns, we find that as N increases from 30 to 900, the time taken for discovering the true structure increases fairly linearly. This suggests that for hierarchical and nearly decomposable systems, discovering the correct modularization is not an NP-complete problem. We also explored interaction patterns that were non-hierarchical and non-decomposable, i.e., greater levels of between-module interaction, for robustness to

alternative settings for 'N' and 'M'[17]. As seen in Figure 2, in cases where the underlying system was hierarchical, designers were always able to converge on the true structure.

In Tables 2-3, we characterize the emergence of architecture for a typical simulation run (N=30, M=5) under each of the four structural archetypes. The tables present a snapshot of the module composition at different periods of the simulation to highlight the process by which the search rule converges on the true structure. The first column denotes the period in the simulation when the snapshot was taken, the second column the number of modules. The set of columns labeled "module composition" capture the policy choices representing each module. Two-digit numbers represent the policy choices. In all cases, the true structure represents the grouping of six policy choices numbered sequentially from 01-30, i.e., policy choices 01-06, 07-12, 13-18, 19-24, and 25-30 clustered together signifies the discovery of the true structure.

Table 2 describes the module evolution in the two nearly decomposable structures. The top half presents the results for the hierarchical system (Figure 1a). The module starts with a randomly specified starting structure of 8 modules and then evolves to the true structure by period 12 and stabilizes on it, with no further changes in module structure after period 12. The lower half presents the results for the non-hierarchical system (Figure 1b). Again, the true structure is identified by period 12. However, in contrast to the results in the top half, the organization does not stabilize on this configuration. In period 19, modules 4 and 5 combine as a result of the reciprocal interaction of the two modules via policy choices 24-25. However, in period 20, the true structure is reestablished when it is observed that all policy choices in the combined module do not share a strong interaction. This process of cycling continues

[17] These results are not attached due to space constraints. They are available from the authors.

indefinitely. However, the frequency of cycling is quite low since the structure is still nearly decomposable.

<< Insert Table 2 here >>

Table 3 presents the module evolution for the non-decomposable systems. The top half presents the results for the hierarchical system (Figure 1c). The results are quite similar to those observed in the case of the hierarchical and nearly decomposable system (see top half of Table 2). The system converges on the true structure by period 12 and stabilizes on it. The lower half of the table presents the results for the non-hierarchical system. In this case, the system reaches a structure where modules 1 and 2 are united, module 3 is correctly identified and modules 4 and 5 are united. In each period after this, there is continual cycling where one or more modules are combining and/or splitting.

<< Insert Table 3 here >>

The above results taken together suggest that the search rules for discovering the true system structure is robust to all alternative structures when two conditions are met: (1) there is strong interaction within modules, i.e., all policy choices within modules are reciprocally interdependent; and, (2) there is a hierarchical precedence structure underlying between-module interactions. The first condition is a reasonably accepted premise that guides the design of modular systems (Alexander, 1964; Baldwin and Clark, 2000; Simon, 1962) and needs little justification. The second condition requires that between module interactions should be hierarchical, i.e., the policy choices in module 2 can depend on policy choices in module 1 or

vice versa but not each other simultaneously. This finding formalizes Simon's (1962) intuition that complex systems that are hierarchical tend to evolve faster and toward more stable structures. Hierarchy appears to be a necessary and sufficient condition for the successful search for the true system structure. Nevertheless, as a practical matter, the search for the true structure is reasonably successful even when complex systems do not conform to one of the two principles (i.e., hierarchy or decomposability). However the search process is significantly less effective when the true structure of the system violates both principles.

4.2. Experiment 2: Adaptation in complex systems

As highlighted earlier, the extant literature advocates modular design principles since it affords two main benefits: (1) accelerating the rate of innovation by allowing localized and parallel innovation efforts in each module; and, (2) allowing mix-and-match or recombination of modules to produce best-in-class system designs. However, much of this discussion implicitly assumes that designers have achieved an appropriate modularization. The purpose of the subset of simulation runs reported here is to examine the extent to which the purported benefits of modular design structures are conditional on discovering the true structure of the system. We examined the realization of the two benefits, singly and in combination.

Local adaptation in nearly decomposable systems

We investigate the interaction between the search for the true structure and local adaptation for module improvement. Clearly, prior to the establishment of a given modular structure, adaptation may take place within the context of the still evolving modules. It is an open question as to whether module-level innovation efforts confound the process of identification of module architecture or whether the two processes of adaptive search can effectively exist simultaneously. To explore this question, we preserved the search for the true structure as

implemented in experiment 1. In addition to this process of search for the appropriate module structure, we add the process of module-level innovation.

Figure 3 plots the average performance results from 100 runs of four models where the complex system was nearly decomposable, both with and without hierarchy (i.e., Figures 1a-b) with a true underlying structure of 5 modules (N=30). In these two settings, there is a simultaneous process of search for the true structure as well as local adaptation for module improvement. In the other two settings, only the process of local adaptation for module improvement occurs and the organization persists in the initial random initialization of the structure of modules.

<< Insert Figure 3 here >>

Comparing the four models in Figure 3 suggests that local adaptation shares strong complementarities with the search for the true structure. Performance not only increases faster but also asymptotes at a higher level than in the case where there is local adaptation alone. However, a surprising and, in some sense, re-assuring finding is that the process of local adaptation continues to be quite useful even when the design structure is misaligned with the true structure. The process of local adaptation alone reaches a performance level of 0.63 by the 100th period[18] as compared with a performance asymptote of 0.68 in the case where there is a joint process of architecture discovery and module improvement. On the one hand, this result suggests that the efficacy of local adaptation is amplified when the true structure is discovered.

[18] Note that the performance asymptote is not reached by the 100th period. The process is still moving uphill.

On the other hand, the result also highlights that discovering the true structure is not a necessary condition for effective local adaptation.

We also find no significant differences between the hierarchical and non-hierarchical systems. With departures from hierarchy, local adaptation without architecture discovery tends to be less monotonic. Actors engage in local innovation efforts that, *ex post*, turn out to be damaging to system performance as a whole. This is a function of the unanticipated and unknown reciprocal interactions between modules. Overall, in nearly decomposable systems, it appears possible to realize the benefits of parallelism and localized adaptation/innovation even when the complex system design does not correspond to true structure of the system.

Local adaptation in non-decomposable systems

We also ran a number of models in which the complex system was non-decomposable (Figures 1c-d). Figure 4 indicates the results of local adaptation in four non-decomposable complex systems: hierarchical, with and without search for the true structure; and, non-hierarchical, with and without search for the true structure. The results here are significantly different from the case of local adaptation in nearly decomposable systems. First, the non-monotonicity in system performance over time is much greater. This results from the fact that the likelihood of incorrect local adaptation efforts is increasing in the degree of non-decomposability. Second, we find that the complementarity between search for the true system structure and local adaptation is robust even in non-decomposable systems. Local adaptation in conjunction with search for the true structure tends to outperform local adaptation alone.

<< Insert Figure 4 here >>

Third, local adaptation and search for the true structure in the hierarchical system results in the highest performance asymptote. This result lends robustness to the findings of the first set of simulations. As long as the condition of hierarchy is met, the search for the true structure is quite effective and local adaptation is then useful in generating performance improvements. Interestingly, we find that the performance asymptote (0.68) here is not statistically significantly different from the performance asymptote (0.68) in the case of nearly decomposable systems (see Figure 3). This suggests that if the complex system is hierarchical and designers engage in search for the true structure, the success of local adaptation efforts is not sensitive to the degree of decomposability.

Finally, the violation of hierarchy tends to be damaging to local adaptation efforts. The consequences of the violation of hierarchy, however, is somewhat mitigated by the search for the true structure. This suggests that even when hierarchy is violated the search for the true structure is an extremely useful design activity since it results in more orderly structures that facilitate the process of local adaptation. At the extreme, when the system is non-hierarchical and non-decomposable, local adaptation without search for the true structure is largely fruitless with performance changes being little different from random change.

In sum, the results indicate that the search for the true structure shares strong complementarities with local adaptation efforts. In the extreme case when complex systems are neither hierarchical nor decomposable, local adaptation efforts without search for the true structure is largely futile. In general, the violation of decomposability is less critical to local adaptation efforts than the violation of hierarchy. These results support our prior findings that in situations where designers have some latitude over the design of system structures, deviations

from near decomposability may be more easily tolerated than deviations from the principle of hierarchy.

Recombination in nearly decomposable systems

The second benefit of modularity is the opportunity that modularity offers for the recombination of modules from two or more systems (Baldwin and Clark, 2000). To explore the effect of recombination, we need to introduce a population of systems that are capable of exchanging modules. The experiments underlying Figures 5-6 model the interaction among 10 systems (N=30) in each of the four alternative system structures (see Figure 1a-d) with a true module structure of 5 modules. The 10 systems have randomly specified starting design structures and their initial policy choices are also independently specified by random assignment.

Figure 5 plots the results for the process of recombination in the context of nearly decomposable systems (Figures 1a-b). Recombination without search for the true structure yields no performance improvement. In the absence of search for the true structure, no two modules in the 10 systems are compatible for recombination to occur. This accounts for the lack of performance improvement. Upon introducing search for the true structure, we observe significant performance gains from recombination. When systems search for and discover the true structure, the module architectures become compatible allowing for recombination to occur. We also find that when the systems are nearly decomposable, the violation of hierarchy has little significant impact on the effectiveness of recombination.

<< Insert Figure 5 here >>

Recombination in non-decomposable systems

Figure 6 indicates the result of carrying out the same analysis of recombination in non-decomposable systems (Figures 1c-d). As expected, we find that when designers do not engage in search for the true structure, modules across different systems are incompatible, resulting in unsuccessful recombination attempts and no change in system performance. However, when designers engage in search for the true structure, recombination is facilitated and we observe system performance improvement. The hierarchical system results in a marginally higher performance asymptote (0.623) as compared with the non-hierarchical system (0.618), though the rate of performance ascent is significantly higher in the hierarchical system. This again lends robustness to the results observed in experiment 1. In systems that are hierarchical, the search for the true structure is successful, which in turn facilitates recombination efforts. On the other hand, in non-hierarchical systems, search for the true structure is less effective in producing modules that conform to the true structure and indeed, such systems never settle on a fully stable configuration. Nevertheless, such a process yields some similarity in module structures which in turn makes recombination possible. Having the true structure does not appear to be a necessary condition for performance enhancing recombination. The necessary and sufficient condition is that two or more systems should share one or more identical module structures. We further explore this possibility of recombination in incorrect system structures in experiment 3.

<< Insert Figure 6 here >>

In sum, the results[19] suggest that, in nearly decomposable systems, the effectiveness of recombination is insensitive to the violation of hierarchy. However, in non-decomposable systems the violation of hierarchy renders recombination marginally less effective.

4.3. Experiment 3: Recombination and the emergence of dominant designs

The experiments reported immediately above suggest that similar system and/or module architectures are a necessary condition for recombination to occur. However, the results do not indicate that the true module structure is a necessary basis for recombination, merely that common module architecture is necessary. As a result, given that recombination offers a powerful adaptive mechanism, it is possible that the aggregation of an organizational population on a particular set of organizational, or technological, forms can be self-reinforcing. If a sub-population of systems share a common architecture, even if there is no particular wisdom in that architecture, it is still possible that that architecture may proliferate.

We examine this possibility in the following analysis. We created a population of 10 systems (N=30; M=5) along the lines described in experiment 1. We also specified initial module architectures randomly for each of the 10 systems. This comprises the baseline setting in which no recombination occurs since the design architectures and modules are all different. We created a setting in which two systems are identical by replicating the system architecture of a randomly picked system and assigned it to the tenth system. This resulted in 8 systems with random architectures and 2 with identical ones[20]. This procedure was repeated to create 3-identical (7 random), 4-identical (6 random), and 5-identical (5 random) systems respectively.

[19] We also explored the effectiveness of local adaptation and recombination efforts in conjunction. We found that local adaptation and recombination shared strong complementarities with each other and with the search for the true structure. In other aspects, the results were similar to that observed for these processes in isolation. These results are available with the authors.

[20] Note that only the policy choice groupings, i.e., system architectures, are identical, not the policy choices (1 or 0) themselves. In other words, the bundle of functions provided by each module is identical, but the level of functionality varies with the actual policy choices.

Figure 7 reports the results of the model runs for hierarchical and non-decomposable systems (Figure 1c). The baseline results reflect the performance level of the random systems in each of the five model runs (i.e., 10-, 8-, 7-, 6-, 5-random systems respectively). The lines labeled 2-identical, 3-identical and so on graph the average performance over time of the subset of systems that are identical in each model. As the graph indicates, when the system architectures are random, there are no performance gains since there is no room for recombination to occur. However, even if only 2 systems are identical, recombination provides significant gains as compared with the eight random systems. The value of recombination rises as the number of identical systems in the population increases from 2 to 5. This suggests that recombination generates tangible performance benefits even when the design architecture is not aligned with the true architecture of the system. Figure 8 plots the results for runs with non-hierarchical and non-decomposable systems (Figure 1d). The results are substantially similar lending robust support for the possibility that recombination among structurally similar systems, even if they do not conform to the true structure, can generate performance improvements.

<< Insert Figures 7 and 8 here >>

In the recombination implemented above, the diversity of the population of systems was kept constant, i.e., selection of systems in each period was without replacement. However, if recombination between similar systems can generate performance improvements then it can provide the fodder for selection forces. In other words, performance improvements arising from recombination can provide the architecturally similar systems with an evolutionary advantage over dissimilar systems. This can ultimately result in emergence of the similar systems as the

dominant design in the industry. We investigate this possibility by re-running the analysis above with selection operating on a population of 100 systems[21]. We performed different analyses, setting the number of similar systems at 20, 30, and 40 respectively, and then tracked the proportion of similar systems as the simulation progressed. With all random systems, asymptotically, every system has an equal probability of emerging as the dominant design as a function of purely random selection.

Figure 9 plots the number of runs in which a subset of systems emerged as the dominant design. In the case of randomly specified systems, each system has an equal probability of emerging as the dominant design through a purely random selection process and at the asymptotic limit (i.e., the number of periods approaches infinity) we would expect to see the domination of a single system. However, by period 100, no single system emerged as the dominant design. On the other hand, by period 100, we find that in 29 runs out of 100 the 20-identical systems emerged as the dominant design, i.e., multiplied into 100 systems. In 36 runs out of 100, the 30 identical systems dominated, and 48 runs out of 100 in the case of 40-identical systems.

<< Insert Figure 9 here >>

This provides some robust evidence that even when a common architecture is randomly specified, the benefits of recombination are sufficient to give them an evolutionary advantage

[21] We increased the number of systems to 100 since with a smaller number of systems a sub-population could early on be eliminated by the stochastic nature of the selection process before the benefits of recombination were realized.. For instance, with 2 identical systems out of 10, selection can randomly eliminate the identical systems before they have a chance to engage in recombination; whereas, with 20 out of 100, the systems have a better chance of recombining before selection can cause a premature loss of diversity. Note however that the result depends more on proportions than on the absolute numbers.

over other random and incompatible systems in the population. It seems that the benefits of recombination are not sensitive to the design architecture as long as two or more architectures are identical to allow recombination. Most importantly, it seems that recombinant benefits can serve as the trigger for the emergence of a dominant design or architecture in the industry.

5. Discussion

The main objective of this paper was to reconcile Simon's ideas on individual limits to rationality (Simon, 1947; Simon, 1955) and the architecture of complex systems (Simon, 1962), particularly how cognitively limited designers can hope to achieve effective and well functioning designs of complex systems. The results of our analyses, in addition to formalizing the intuitions in Simon's (1962) architecture of complexity, provides the foundation for a theory of complex system design that does not rely on omniscient, or even rational, agents to achieve effective designs.

First, we find that for systems whose true structure is hierarchical, local search rules are adequate to discover this structure. In addition, the solution time increases only linearly in problem size. This finding deviates importantly from Schaefer's (1999) finding that the problem of complex system design is NP complete, i.e., cannot be solved even in polynomial time. A key difference between Schaefer's (1999) model and that reported here accounts for the divergent finding. Schaefer (1999) makes no assumptions about the true nature of complex systems. If indeed complex systems were not hierarchical (i.e., between module interactions occur both below and above the principal diagonal), then our simulation results agree with Schaefer's (1999) findings. Under such a circumstance, policy choices randomly cycle between modules since there is no simple rule to stop the search process. On the other hand, if we assume, per Simon (1962), that complex systems in the real world tend to be hierarchical, then the search

process is able to converge on the true structure. Therefore, our finding may be stated with a caveat that, if indeed complex systems are hierarchical then even boundedly rational designers with simple search rules can converge on the true structure.

Although we find that designers can solve the problem of complex system design through an evolutionary search process, a question remains as to what constitutes a time period in our computational experiment and how does it map onto real time. While there is no established mapping from computation time to real time, some qualitative assessments are interesting and suggestive.

If we think of a complex system modeled here as a product, such as a microprocessor, then the experimentation in each period is akin to architectural change in the underlying product technology. In order to calibrate the real time frequency of architectural change in microprocessors, we tracked the architectural change within and between six generations of microprocessor design between 1981-1998[22]. We inferred architectural change if a module of the microprocessor was changed (re-allocation of policy choices), a new module was created (splitting), or a module was integrated with another (combining). We found some evidence for architectural change *within* microprocessor generations. For instance, within generations 4, 5, and 6 there were changes in the number of transistors embedded on the microprocessor suggesting that some architectural design changes were made.

We also tracked the architectural design changes *across* microprocessor generations. Our analyses suggest that each generation involved significant degree of architectural change and an average of 2-3 years separated the first design of each generation (see footnote 22). This suggests that, at least in products, it could take between 1-3 years for introducing any architectural design

[22] The tables describing the relevant data supporting the inferences are available with the authors.

change. In the results presented in Table 1, we found that for a complex system of size N=900 and correct modularization of 30, it took an average of 178 periods to discover the true modular structure. If this stylized complex system is assumed to be a microprocessor, it seems that discovering the right modularization would take between 178-632 years. Given the pace at which architectural change occurs in real world complex systems, it may indeed take very long periods of time to discover the true modularization. In this sense, the dilemma of intractability posed by Schaefer (1999), as a practical matter, re-asserts itself even in the case of hierarchical systems. Indeed, this challenge is compounded to the extent that the true structure of interactions itself changes over time.

In view of this result that progress is possible but its full realization possibly quite distant in time, the finding that, though the benefits of modularity (i.e., local adaptation and recombination) are increasing in the ability to discover the true structure the identification of the true structure is not a necessary condition, becomes quite important. The upside to adopting modular design structures is robust to the choice of a variety of incorrect design architectures. This finding is re-assuring to the advocates of modular design principles. While there is some penalty in terms of lost performance when designers are unable to discover the true structure, much of the benefits of modularity can be obtained even in the absence of the identification of the true modular structure.

Finally, we find that recombination may itself be an engine for the emergence of dominant designs. The extant literature, both on dominant designs (Anderson and Tushman, 1990) and modularity (Baldwin and Clark, 2000) seems to suggest that recombination is a viable mode of module performance improvement only after the emergence of dominant designs or standardization among components. For instance, Anderson and Tushman (1990: 614) suggest

that "dominant designs permit firms to design standardized and interchangeable parts and to optimize organizational processes for volume and efficiency…if the product or process is part of a larger system, industry standards permit system-wide compatibility and integration." In contrast, our results suggest that recombination may actually be a potentially powerful engine for establishing dominant designs. If two or more firms agree on a common architecture then the performance gains from recombination can provide the fodder for selection pressures to favor this architecture over other dissimilar or incompatible ones. Much of the literature on technological evolution relies on selection forces to accomplish a sorting of designs and contributing to the emergence of dominant designs. However, in the early stages of technological evolution, especially in new technologies, as indeed Anderson and Tushman (1990: 612) argue, "when a technology builds on a completely new knowledge base, many rival designs appear, and it will take longer for market forces to sort out these variants…the process of converging on a standard is hampered by a lack of common understanding among technical experts about how the new technology operates and where its economic performance limits lie." In contrast, our analysis suggests that recombination between identical system designs may be the systematic driver of performance differences for selection forces to accomplish such sorting. While the process of recombination may facilitate the emergence of dominant designs, this fact does not preclude the existence of other mechanisms. Indeed the social and political processes described elsewhere in the literature might be necessary to establish the shared architectural platform for recombination to occur (Anderson and Tushman, 1990; Phillips, 1987).

In sum, our analysis provides a useful micro-foundation for work on modularity --- work that has tended to take for granted the existence of well-specified module structures. Relatively local and incremental processes are capable of identifying useful, if not optimal, modules in

systems that have some inherent hierarchy and decomposability. Recombination will tend to reinforce the dominance of a more common modular structure independent of its true value (i.e., the degree to which the modular structure captures the true underlying structure of the system). As a result, recombination is found to be a powerful engine for the emergence of dominant designs and not merely a by-product of the presence of a dominant design.

Collectively, these are an important set of findings that bring to the surface the question of how boundedly rational actors are to identify the modular structures that are intended to economize on the coordination capabilities of these actors. Modularity need not be the product of divine design but may derive from an evolutionary process. At the same time, the work leaves unanswered many other important questions. The focus here has been on modularity in the sense of the partitioning of a complex system. As a result, the problem of the design of interfaces among modules has been restricted to the constraint imposed by sharing interactions across modules. We do not explore the subtler question of the development of a "language" of communication among modules. In a related vein, we have treated the problem of modularity as one of discovering the unknown but latent modular structure of a system. We have not addressed the possible endogeneity of the underlying interaction structure of a system. Nevertheless, we view the current work as an important step forward. If modularity is a partial solution to the problem of boundedly rational processes of adaptation, then our models of adaptation must address the evolution of the module structure itself rather than takes its existence, and indeed in many cases its optimality, as given. The search for the architecture of a complex system is a central element in the process of individual and collective change.

References

Abernathy, W. J. and J. Utterback
1978 "Patterns of industrial innovation." Technology Review, June/July: 40-47.

Alexander, C.
1964 Notes on the synthesis of form. Cambridge: Harvard University Press.

Anderson, P. and M. L. Tushman
1990 "Technological Discontinuities and Dominant Designs: A Cyclical Model of Technological Change." Administrative Science Quarterly, 35: 604-633.

Arrow, K. J.
1974 The limits of organization. New York: W. W. Norton and Co.

Baldwin, C. Y. and K. B. Clark
2000 Design rules: The power of modularity. Cambridge, MA: The MIT Press.

Bartlett, C. A. and S. Ghoshal
1989 Managing across borders: The transnational solution. New York: McGraw-Hill.

Baum, J. A. C. and J. V. Singh
1994 "Organizational hierarchies and evolutionary processes: Some reflections on a theory of organizational evolution." In J. A. C. Baum and J. V. Singh (eds.), Evolutionary dynamics of organizations: 3-20. New York: Oxford University Press.

Chandler, A. D.
1962 Strategy and structure: chapters in the history of the industrial enterprise. Cambridge, MA: MIT Press.

Clark, K. B.
1985 "The Interaction of Design Hierarchies and Market Concepts in Technological Evolution." Research Policy, 14: 235-251.

Cohen, M. and R. Axelrod
1999 Harnessing complexity: Organizational implications of a scientific frontier. New York: The Free Press.

Cooper, R. and R. S. Kaplan
1992 "Activity-based systems: Measuring the costs of resource usage." Accounting Horizons, 6: 1-13.

Cummings, S.
1995 "Centralization and decentralization: The never-ending story of separation and betrayal." Scandinavian Journal of Management, 11: 103-117.

Eisenhardt, K. M. and S. L. Brown

1999 "Patching: Restitching business portfolios in dynamic markets." Harvard Business Review, 77: 72-82.

Garey, M. R. and D. S. Johnson
1990 Computers and Intractability: A Guide to the Theory of NP-Completeness. San Francisco: W.H. Freeman Company.

Garud, R. and A. Kumaraswamy
1995 "Technological and organizational designs for realizing economies of substitution." Strategic Management Journal, 16: 93-109.

Goldberg, D. E.
1989 Genetic algorithms in search, optimization and machine learning. Reading, MA: Addison-Wesley Longman.

Holland, J. H.
1992 Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control & artificial intelligence, 2nd ed. Cambridge, MA: MIT Press.

Ichniowski, C., K. Shaw, and G. Prennushi
1997 "The effects of human resource management practices on productivity: A study of steel finishing lines." American Economic Review, 87: 291-313.

Kauffman, S. A.
1993 The origins of order: Self-organization and selection in evolution. New York: Oxford University Press.

1995 At home in the universe: The search for the laws of self-organization and complexity. New York: Oxford University Press.

Klepper, S.
1997 "Industry life cycles." Industrial and Corporate Change, 6: 145-181.

Langlois, R.
2002 "Modularity in technology and organization." Journal of Economic Behavior and Organization, 49: 19-37.

Lant, T. K. and S. J. Mezias
1990 "Managing Discontinuous Change: A Simulation Study of Organizational Learning and Entrepreneurship." Strategic Management Journal, 11: 147-179.

1992 "An Organizational Learning Model of Convergence and Reorientation." Organization Science, 3: 47-72.

Lawrence, P. R. and J. W. Lorsch
1967 Organization & environment: Managing differentiation & integration. Cambridge, MA: Harvard University Press.

Levinthal, D. A.
1991 "Organizational adaptation and environmental selection - Interrelated processes of change." Organization Science, 2: 140-145.

1997 "Adaptation on rugged landscapes." Management Science, 43: 934-950.

Loch, C. H., C. Terwiesch, and S. Thomke
2001 "Parallel and sequential testing of design alternatives." Management Science, 47: 663-678.

Macduffie, J. P.
1995 "Human resource bundles and manufacturing performance: Organizational logic and flexible production systems in the world Auto industry." Industrial & Labor Relations Review, 48: 197-221.

1997 "The road to "root cause": Shop floor problem-solving at three auto assembly plants." Management Science, 43: 479-502.

Marengo, L., G. Dosi, P. Legrenzi, and C. Pasquali
2000 "The structure of problem-solving knowledge and the structure of organizations." Industrial and Corporate Change, 9: 757-788.

Mintzberg, H.
1979 The structuring of organizations. Englewood Cliffs, NJ: Prentice Hall.

Nickerson, J. and T. Zenger
2002 "Being Efficiently Fickle: A dynamic theory of organizational choice." Organization Science, 13: 547-566.

Parnas, D. L.
1972 "On the criteria to be used in decomposing systems into modules." Communications of the ACM, 15: 1053-1058.

Phillips, A.
1987 "The role of standardization in shared bank card systems." In H. L. Gabel (ed.), Product standardization and competitive strategy: 263-282. Amsterdam: North-Holland.

Rivkin, J. and N. Siggelkow
Forthcoming "Choice interaction and organization structure." Management Science.

Sanchez, R. and J. T. Mahoney
1996 "Modularity, flexibility, and knowledge management in product and organization design." Strategic Management Journal, 17: 63-76.

Schaefer, S.
1999 "Product design partitions with complementary components." Journal of Economic Behavior & Organization, 38: 311-330.

Schilling, M.

2000 "Toward a general modular systems theory and its application to interfirm product modularity." Academy of Management Review, 25: 312-334.

Simon, H. A.
1947 Administrative behavior. New York: Macmillan.

1955 "A behavioral model of rational choice." Quarterly Journal of Economics, 69: 99-118.

1962 "The architecture of complexity." Proceedings of the American Philosophical Society, 106: 467-482.

Szulanski, G.
1996 "Exploring internal stickiness: Impediments to the transfer of best practice within the firm." Strategic Management Journal, 17: 27-43.

Takeishi, A. and T. Fujimoto
2001 "Modularization in the auto industry: Interlinked multiple hierarchies of product, production, and supplier systems." Boston, MA: IMVP, MIT Sloan School of Management.

Thompson, J. D.
1967 Organizations in action. New York: McGraw-Hill.

Tushman, M. L. and P. Anderson
1986 "Technological Discontinuities and Organizational Environments." Administrative Science Quarterly, 31: 439-465.

Ulrich, K. T. and S. D. Eppinger
1999 Product design and development, 2nd ed. New York: McGraw-Hill.

Wilson, E. and W. Bossert
1971 A primer of population biology. Sunderland, MA: Sinauer Associates Inc.

Table 1 Mean and standard deviation of time periods to discover the "true" structure under varying N and M for hierarchical and nearly decomposable systems (100 runs)

M	N				
	30	**48**	**60**	**90**	**900**
3	7.60 (3.56)	-	-	-	-
5	12.90 (6.20)	-	-	-	-
6	13.43 (6.46)	-	-	-	-
10	8.82 (4.16)	-	-	-	-
15	9.47 (4.56)	-	-	-	-
3	-	10.64 (5.11)	-	-	-
4	-	12.17 (5.89)	-	-	-
6	-	18.61 (9.13)	-	-	-
8	-	25.13 (12.36)	-	-	-
12	-	37.59 (18.59)	-	-	-
16	-	13.45 (6.52)	-	-	-
24	-	15.09 (7.48)	-	-	-
3	-	-	10.83 (5.23)	-	-
5	-	-	19.02 (9.31)	-	-
6	-	-	19.55 (9.63)	-	-
10	-	-	28.55 (14.13)	-	-
15	-	-	44.85 (22.19)	-	-
20	-	-	16.50 (8.08)	-	-
30	-	-	19.00 (9.49)	-	-
3	-	-	-	11.63 (5.65)	-
5	-	-	-	21.42 (10.64)	-
6	-	-	-	26.34 (13.01)	-
9	-	-	-	36.49 (18.07)	-
10	-	-	-	42.39 (21.00)	-
15	-	-	-	55.63 (27.70)	-
18	-	-	-	67.13 (33.39)	-
30	-	-	-	25.89 (12.91)	-
45	-	-	-	28.67 (14.70)	-
30	-	-	-	-	178.50 (92.56)
180	-	-	-	-	632.06 (337.30)

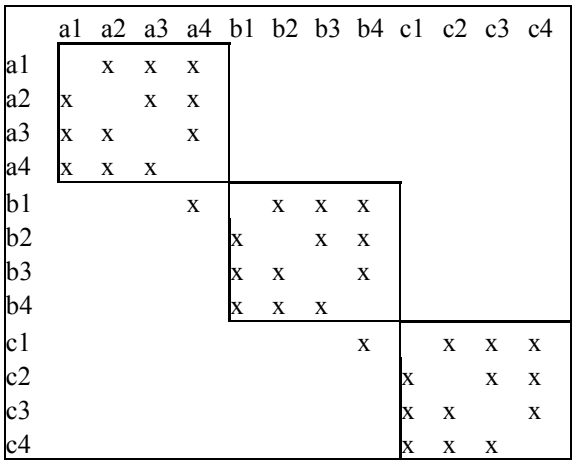
Table 2. Evolution of modules and emergence of structure in a typical single run with nearly decomposable systems

Period	# of modules	Module composition									
Hierarchical											
1	8	12181916	051123	061415	10132530	010928	2122262729	172024	0203040708		
2	10	181609	05230619	14151217	2530	0128	262729	2024	0203040708	1011	212213
3	10	181613	0506	141517	253026272928	0112	2024	020304	10110708	2122	231909
4	7	18161309	0506	141517	253026272928	20240112020304	10110708	23192122			
5	8	181613	0506	141517	253026272928	01020304	10110708	231921222024	1209		
8	7	181613	0506	141517	253026272928	01020304	231921222024	120910110708			
10	6	181613141517	0506	253026272928	01020304	231921222024	120910110708				
12	5	181613141517	253026272928	010203040506	231921222024	120910110708					
14	5	181613141517	253026272928	010203040506	231921222024	120910110708					
Non-hierarchical											
1	5	03061718202824	0108091115	0712212227	02051419232526	041013162930					
2	7	171828	01080911	07122122152024	02051923252627	131614	102930	030604			
3	8	1718	0809110712	131614	030604	21222024	1923	1528102526272930	020501		
4	7	171815	1316140809110712	030604	21222024	1923	282526272930	02050110			
5	7	17181510	0809110712	131614	212220241923	282526272930	020501	030604			
8	7	171815	080911071210	131614	212220241923	282526272930	020501	030604			
10	6	171815131614	080911071210	212220241923	282526272930	020501	030604				
12	5	171815131614	080911071210	212220241923	282526272930	030604020501					
14	5	171815131614	080911071210	212220241923	282526272930	030604020501					
19	4	171815131614	080911071210	282526272930212220241923	030604020501						
20	5	171815131614	080911071210	282526272930	030604020501	212220241923					

Table 3. Evolution of modules and emergence of structure in a typical single run with non-decomposable systems

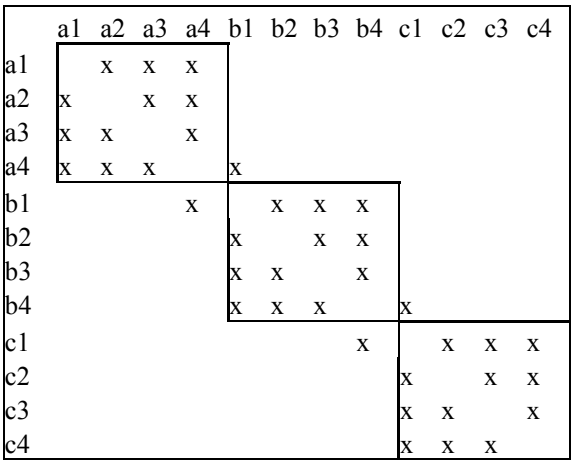
Period	# of modules	Module composition									
Hierarchical											
1	8	1013202628	021122	081823	061516192530	051214	0409	2129	010307172427		
2	7	2628272909	1923	05120717	04141122	0103	06151625300818021013	242120			
3	10	262827290602	1923	12071709	0405	0103	2530	242120	1422	15161813	081011
4	7	26282729	0602	120709	04050103	2530	242120	151618131417	081011	192322	
5	7	26282729	0602	120709	04050103	2530	242120	151618131417	081011	192322	
8	7	26282729	0602	120709081011	04050103	2530	242120192322	151618131417			
10	6	0602	120709081011	04050103	253026282729	242120192322	151618131417				
12	5	060204050103	120709081011	253026282729	242120192322	151618131417					
14	5	060204050103	120709081011	253026282729	242120192322	151618131417					
Non-hierarchical											
1	10	15162126	1924	040514	301122	202729	061218	080917	030107	25101328	0223
2	8	1924	0405251028	30221426	202729151621230618	1211	0301	02130809	0717		
3	7	1924	301426	2021232225282729060213	0301	08091211	0405100717	151618			
4	7	192406020405	3026	20212322	03011007	151618	25282729	13141708091211			
5	9	06020405	3026	20212322	1007	25282729	08091211	131417	1924151618	0301	
8	3	3026	19242021232225282729	131417151618100708091211030106020405							
10	4	1924202123223026	25282729	0301060204051007080912 11	131417151618						
12	3	030106020405100708091211	19242021232225282729 3026	131417151618							
14	3	131417151618100708091211	030106020405	19242021232225282729 3026							
17	3	252827293026192420212322	100708091211131417151618	030106020405							
18	4	252827293026	131417151618	192420212322	100708091211030106020405						

Figure 1a Hierarchical & nearly decomposable system



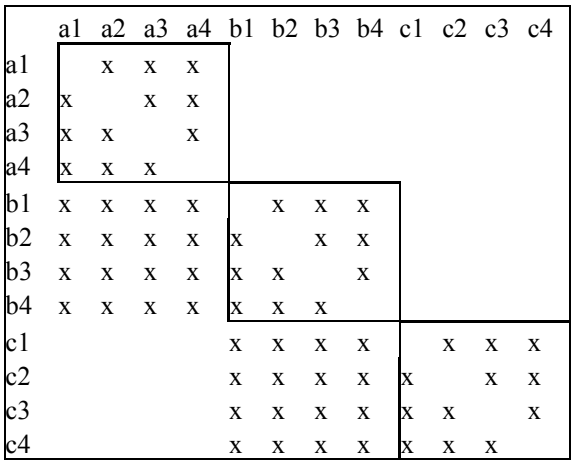
	a1	a2	a3	a4	b1	b2	b3	b4	c1	c2	c3	c4
a1		x	x	x								
a2	x		x	x								
a3	x	x		x								
a4	x	x	x									
b1				x		x	x	x				
b2					x		x	x				
b3					x	x		x				
b4					x	x	x					
c1								x		x	x	x
c2									x		x	x
c3									x	x		x
c4									x	x	x	

Figure 1b Non-hierarchical & nearly decomposable system



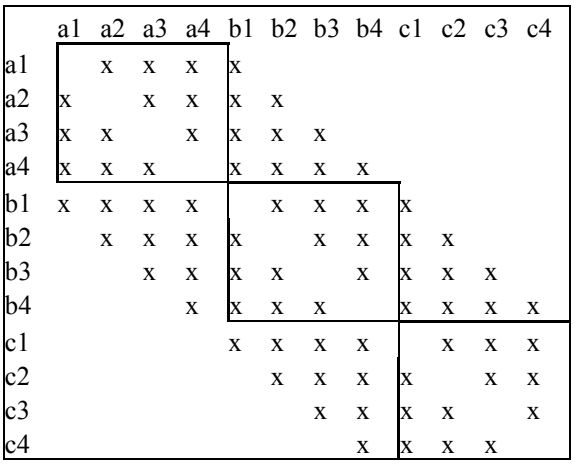
	a1	a2	a3	a4	b1	b2	b3	b4	c1	c2	c3	c4
a1		x	x	x								
a2	x		x	x								
a3	x	x		x								
a4	x	x	x		x							
b1				x		x	x	x				
b2					x		x	x				
b3					x	x		x				
b4					x	x	x		x			
c1								x		x	x	x
c2									x		x	x
c3									x	x		x
c4									x	x	x	

Figure 1c Hierarchical & non-decomposable system



	a1	a2	a3	a4	b1	b2	b3	b4	c1	c2	c3	c4
a1		x	x	x								
a2	x		x	x								
a3	x	x		x								
a4	x	x	x									
b1	x	x	x	x		x	x	x				
b2	x	x	x	x	x		x	x				
b3	x	x	x	x	x	x		x				
b4	x	x	x	x	x	x	x					
c1					x	x	x	x		x	x	x
c2					x	x	x	x	x		x	x
c3					x	x	x	x	x	x		x
c4					x	x	x	x	x	x	x	

Figure 1d Non-hierarchical & non-decomposable system



	a1	a2	a3	a4	b1	b2	b3	b4	c1	c2	c3	c4
a1		x	x	x	x							
a2	x		x	x	x	x						
a3	x	x		x	x	x	x					
a4	x	x	x		x	x	x	x				
b1	x	x	x	x		x	x	x	x			
b2		x	x	x	x		x	x	x	x		
b3			x	x	x	x		x	x	x	x	
b4				x	x	x	x		x	x	x	x
c1					x	x	x	x		x	x	x
c2						x	x	x	x		x	x
c3							x	x	x	x		x
c4								x	x	x	x	

Figure 1e Typical perceived interaction matrix of policy choices at the start of the simulation

	a1	a2	a3	a4	b1	b2	b3	b4	c1	c2	c3	c4
a1				x								
a2							x					x
a3					x				x	x	x	
a4	x											
b1			x						x	x	x	
b2								x				
b3		x										x
b4						x						
c1			x		x					x	x	
c2			x		x				x		x	
c3			x		x				x	x		
c4		x					x					

Figure 2. Number of runs that converge on the "true" structure where M=5 and N=30 (100 runs)

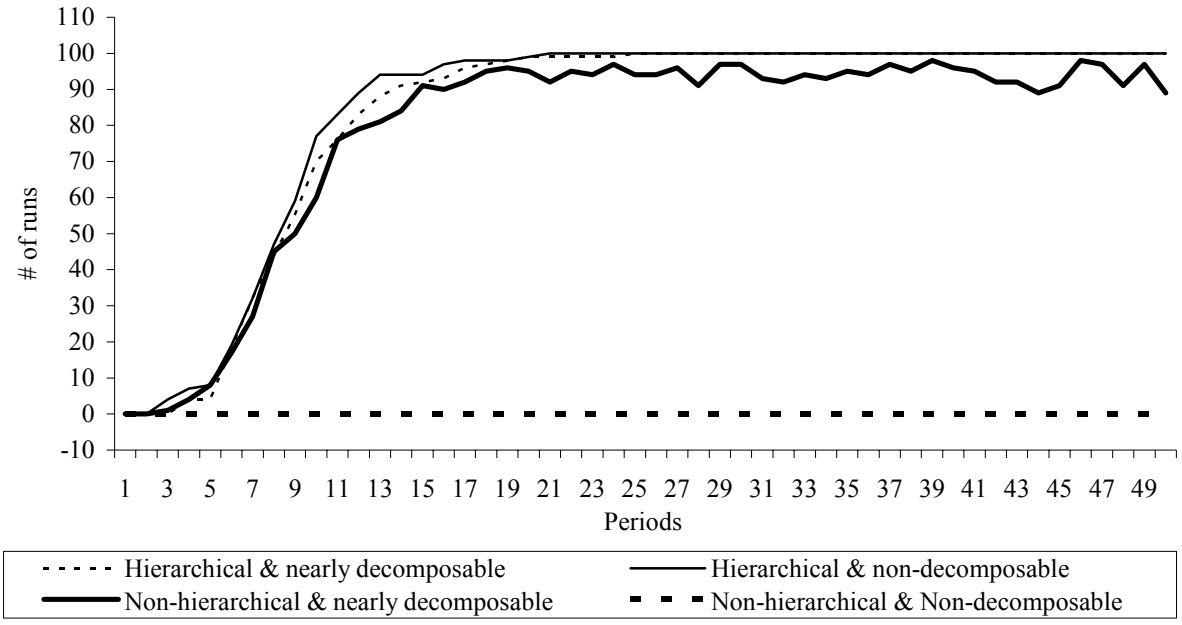


Figure 3. Local adaptation in nearly decomposable systems (100 runs)

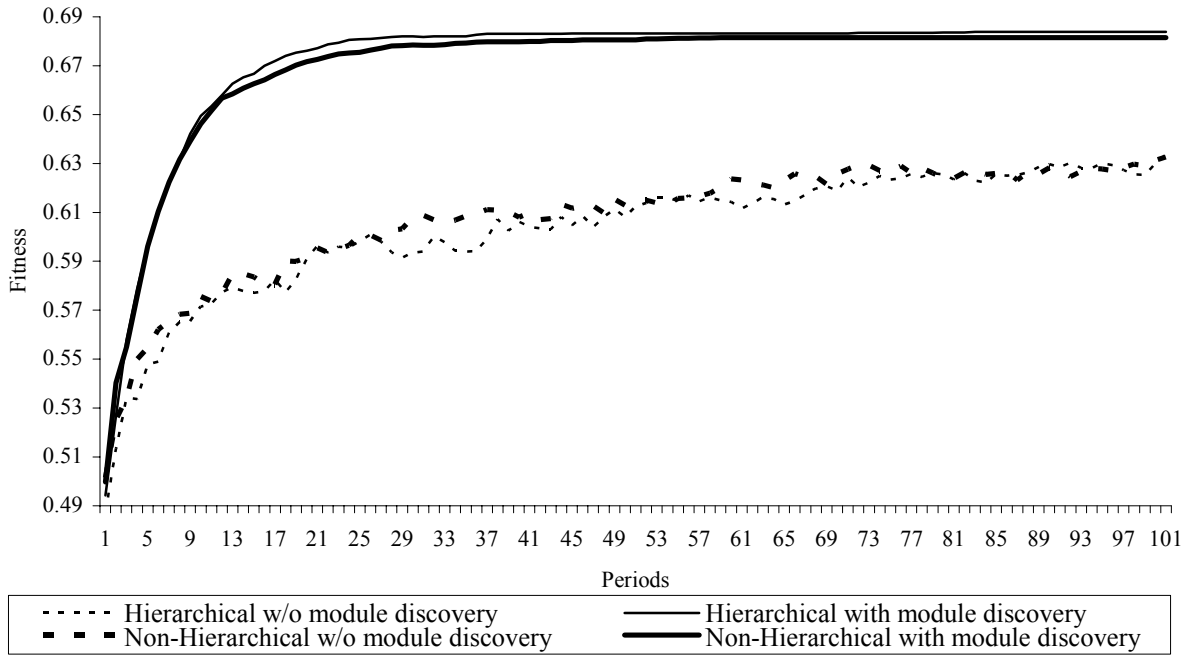


Figure 4. Local adaptation in non-decomposable systems (100 runs)

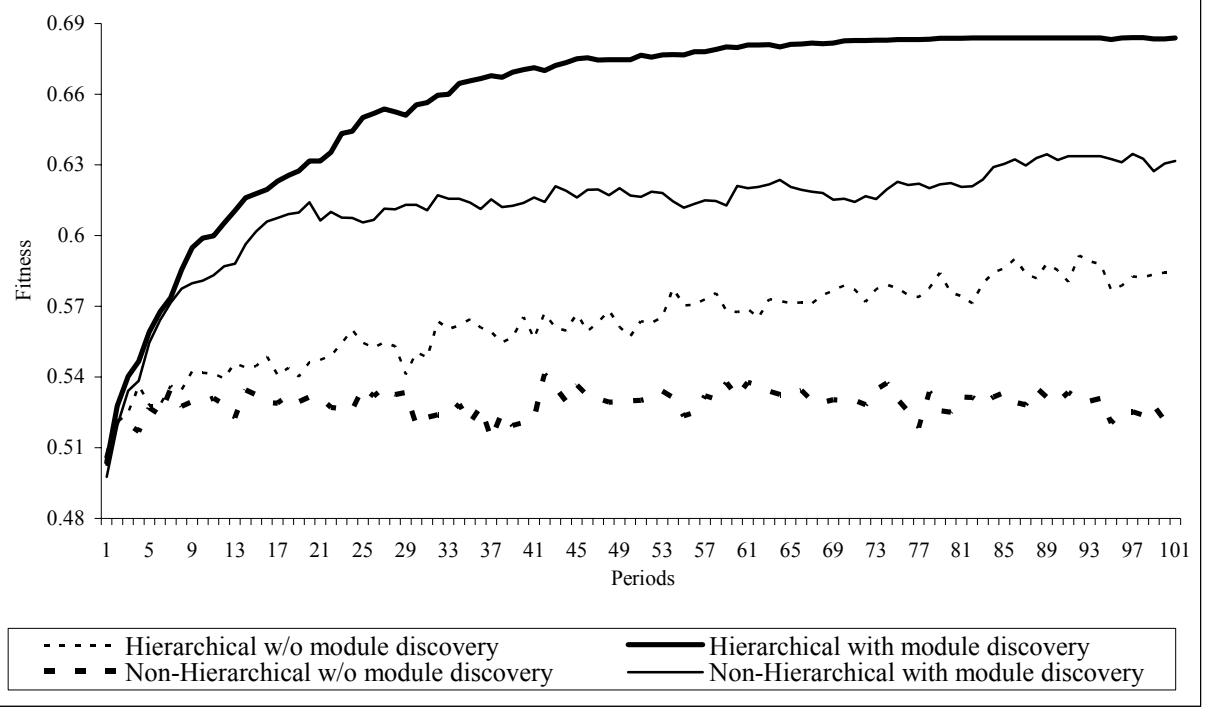


Figure 5. Recombination in nearly decomposable systems (10 systems 100 runs)

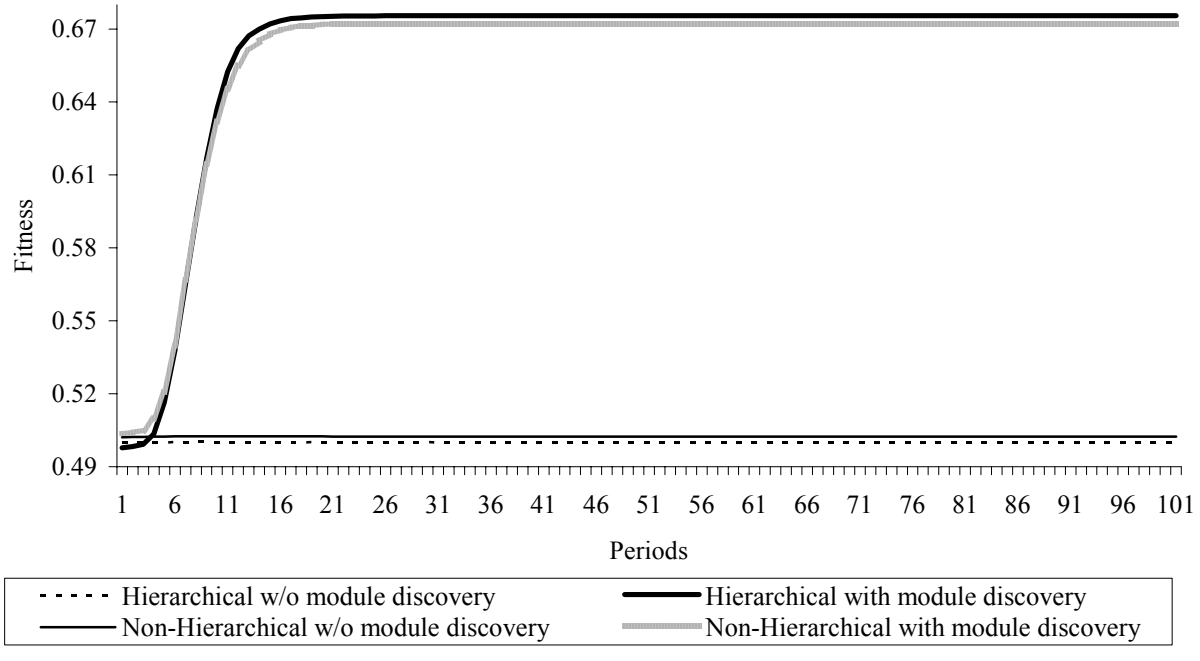


Figure 6. Recombination in non-decomposable systems (10 systems 100 runs)

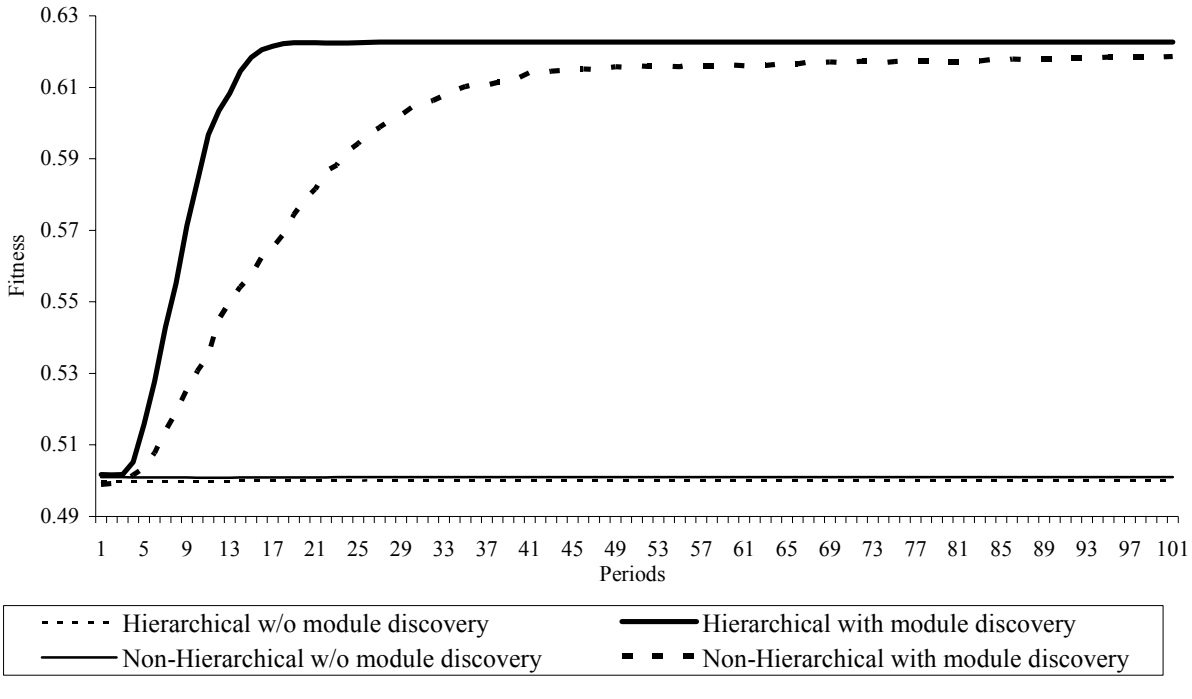


Figure 7. Recombination in hierarchical and non-decomposable systems with no search for the true structure (10 systems 100 runs)

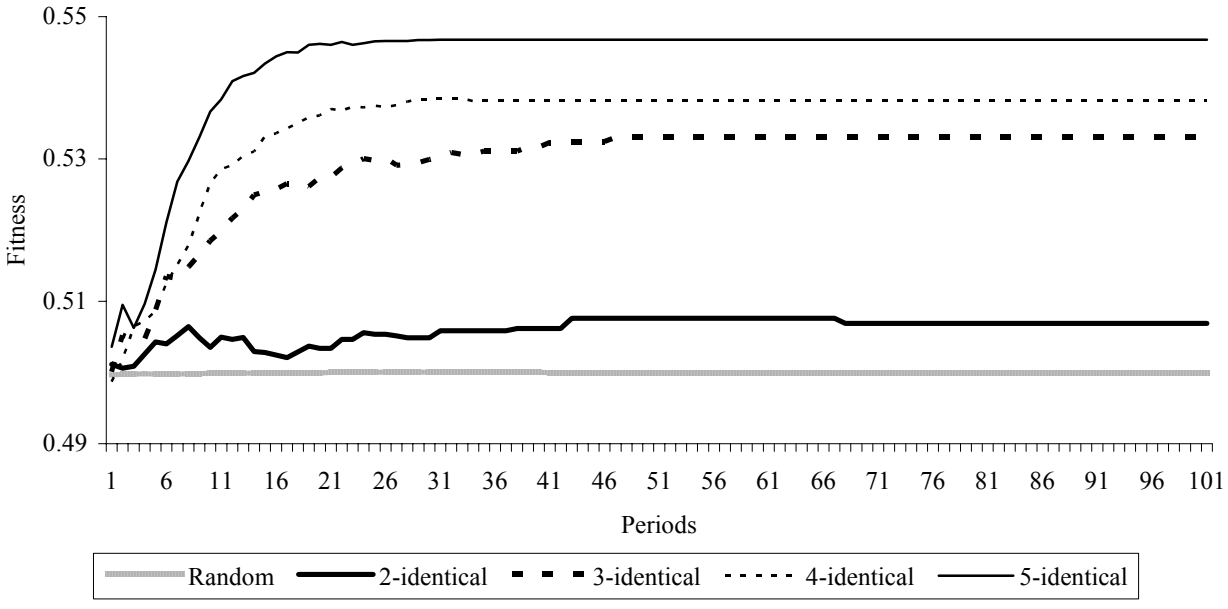


Figure 8. Recombination in non-hierarchical and non-decomposable systems with no search for the true structure (10 systems 100 runs)

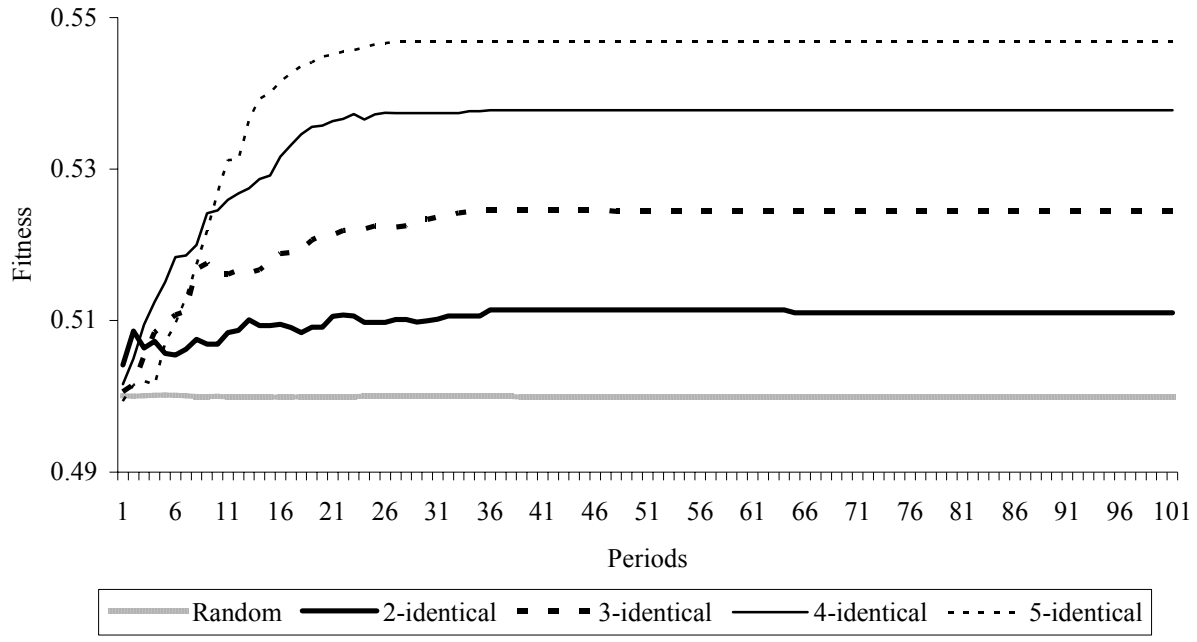


Figure 9. The emergence of dominant designs as a function of recombination (100 systems 100 runs)