ARCHITECTURAL SEARCH AND INNOVATION

Daniel Albert
LeBow College of Business
Drexel University
daniel.albert@drexel.edu

Nicolaj Siggelkow
The Wharton School
University of Pennsylvania
siggelkow@wharton.upenn.edu

Abstract

Product innovation can result from the novel design and combination of product components as well as from changing the underlying architecture, that is, the way components interact with each other. Even though previous studies have shown that architectural change can constitute a powerful source of innovation, little insight exists on how organizations should engage in architectural search itself. In this paper, using computer simulation, we explore underlying mechanisms of architectural search. We find that contrary to search for component combinations, architectural search provides greater performance improvements the narrower the search scope, regardless of product complexity. Moreover, our theory and findings suggest a more differentiated typology of architectural innovation. While narrow architectural search often leads to *pure architectural innovations* that do not require substantial component changes, broader architectural search often leads to *composite architectural innovation*, i.e., architectural innovations that typically render existing component designs suboptimal, but allow for new high-performing component combinations to arise. Lastly, while narrow architectural search outperforms broad architectural search in the long run, in the short run, broad architectural search can have performance advantages.

Keywords: Architectural Innovation, Product Design, Organizational Learning, Complex Systems, Computer Simulations

Acknowledgements: The authors are thankful to Felipe Csaszar, Martin Ganco, Giovanni Gavetti, Rahul Kapoor, and three anonymous reviewers for helpful comments. The authors are also grateful for generous high performance computing resources made available at the University of Wisconsin-Milwaukee and Drexel University as well as generous financial support provided by the Mack Institute for Innovation Management at the University of Pennsylvania.

1. INTRODUCTION

Product innovation is a crucial means for organizations to sustain and improve performance. One central way in which firms innovate is by searching for new designs of the components that make up a product. Besides such component innovation, firms can also innovate by finding novel ways in which existing components interact, a process called architectural innovation (Henderson and Clark 1990). While substantial work exists on how firms should search for new components and combinations of components (Baumann and Siggelkow 2013, Billinger et al. 2013, Ethiraj and Levinthal 2004b, Holland 1975, Katila and Ahuja 2002, March 1991), surprisingly little research has examined how organizations should search for new architectures. The literature on architectural innovation has mainly focused on organizations' inability to recognize changes in linkages among components (Brusoni et al. 2001, Henderson and Clark 1990) and has treated architectural change as largely exogenous (Levinthal and Warglien 1999). This is particularly surprising, considering empirical accounts that have documented architectural redesign efforts and performance benefits associated with new product architectures (Baldwin and Clark 2000, Christensen et al. 1998, Fixson and Park 2008, Kapoor and Adner 2012).

A well-documented insight from the literature on search for component combinations holds that a broad search scope — a large number of components that are considered simultaneously to be redesigned — is helpful when components are highly interdependent, i.e., when product architectures are complex (Holland 1975, Levinthal 1997, March 1991, Milgrom and Roberts 1995). One natural question to pose is whether this insight also applies to architectural search: When exploring new architectures, are redesigns that involve many changes to interactions in the existing architecture more beneficial than redesigns that involve few changes? More broadly, what are the consequences with respect to product innovation when architectures change by different degrees?

To explore these questions, we create an agent-based computer simulation model, using a common imagery of the innovation process as a search across a rugged "landscape" of possible innovations, where each point of the landscape represents a particular component combination and the height on the landscape represents the performance of this configuration (Fleming and Sorenson 2001,

Levinthal 1997). While search for new component combinations can be thought of as a search over a specific landscape (determined by the product's architecture), architectural innovation can be visualized as actively changing and then searching across *different* landscapes. To capture these processes, our model contains product designers who search for novel, higher performing product designs along two dimensions: (1) the way the product's components are designed and combined (component innovation), and (2) the way components interact with one another (architectural innovation). A key focus of our research is on the effects of different levels of breadth of the architectural change, i.e., the number of simultaneous changes in the interactions among a product's components.

The first contribution of our paper emerged from translating the verbal theory of architectural innovation into a simulation model. This exercise, combined with a deep review of the empirical literature on architectural innovation, revealed a number of different types of architectural innovation that had not been distinguished before in the literature. As a result, we propose a new typology that distinguishes architectural innovation along three dimensions: the trigger of innovation; the degree of required subsequent component innovation; and whether the redesign objective is to achieve a particular pattern of interactions among components, or not.

Our simulation analysis surfaces two important insights into the mechanics of architectural change that are robust across our typology of architectural innovation. First, contrary to component search within architectures, the long-run performance increase through architectural innovation is highest when architectures are changed incrementally, regardless of the initial complexity of the product. While incremental architectural change leads to high-performing product designs in the long run, it does require, however, many more successful architectural adaptations than broader architectural change. As a result, in the short run, broad architectural change can outperform narrow architectural change.

Second, our analysis reveals two distinct mechanisms of how architectural change can lead to performance improvements: architectural change can lead to performance improvement by increasing the value of existing component designs; or architectural change can enable new, higher-performing component combinations to be discovered. Which mechanism is at play depends on the degree of the

architectural change. Incremental architectural change tends to lead to performance increases of existing components by linking them in new ways, i.e., a form of *pure architectural innovation*. In contrast, broader architectural change tends to yield new architectures that often render the existing component combination suboptimal, but leads designers to discover new high-performing component combinations (*composite architectural innovation*).

2. PRIOR RESEARCH AND A NEW TYPOLOGY FOR ARCHITECTURAL INNOVATION

2.1. Product design and innovation as complex systems

Product innovation has frequently been portrayed as a matter of designing and coordinating a complex system that is composed of a set of components which interact in nontrivial ways. The number of interactions is typically referred to as the "complexity" of a system (Simon 1962). Likewise, a product design is considered of greater complexity the more interdependencies between component decisions need to be taken into consideration (Baldwin and Clark 2000).

An innovation process is commonly triggered when a designer (or a design team) faces a problem that needs to be solved. Designers may attempt to improve product performance in at least two ways. The first, and most widely studied, refers to designers' efforts to solve a design problem by finding better-performing components (Nelson and Winter 1982). The second refers to designers' efforts to solve a problem by changing the underlying interactions among components, termed *architectural innovation* (e.g., Galunic and Eisenhardt 2001, Henderson and Clark 1990).

2.2. Component innovation

Component innovation occurs when an individual component in a product is redesigned. Interactions among components can complicate component innovation, as more than one component may need to be redesigned to lead to overall higher product performance. Such large-scale change raises challenges given that boundedly rational decision makers often are unable to consider all interdependencies among components and tend to focus on incremental changes (Cyert and March 1963, March and Simon 1958).

Prior studies have substantially advanced our understanding of how organizations can mitigate the constraints of boundedly rational search under complexity. Most notably, broad exploratory search, i.e., experimenting with multiple component changes simultaneously, has been shown to help discover distant solutions (Levinthal 1997, March 1991, Tushman and Romanelli 1985), especially when followed by incremental search (Billinger et al. 2013, Ethiraj and Levinthal 2004b). One line of research has investigated how to induce broader search through different organizational designs and incentive systems (Siggelkow and Rivkin 2005), or through imitation of other firms' designs (Csaszar and Siggelkow 2010).

Another line of research, building on the concept of the "design structure matrix" (Eppinger et al. 1994, Steward 1981), has explored how the underlying pattern of interactions among components may affect the search for new component combinations. In particular, products that are made up of modules (or clusters) of components that contain highly interdependent components within and little interdependence between such modules facilitate the search for high-performing component combinations (Baldwin and Clark 2000, Ethiraj and Levinthal 2004a, b, Sanchez and Mahoney 1996, Simon 1962). With such modularization, innovations can occur at the module level, while at the same time leading to more distant solutions at the aggregated system-level. This line of work often implicitly assumes that the interactions among the various elements of the system are given and fixed. In these studies, product designers innovate on components, but not on the interactions among components.

2.3. A new typology of architectural innovations

Architectural innovation occurs when interactions among components change. A number of empirical studies have observed how firms have actively changed the linkages among a product's components (Baldwin and Clark 2000, Christensen et al. 2002, Fixson and Park 2008, Kapoor 2013, Ulrich 1995). These empirical accounts have been largely inspired by Henderson and Clark's (1990) seminal piece on architectural innovation as an understudied type of innovation. These authors' main insight was that innovation not only can happen at the component level, but also through changes of interactions among components. While they define architectural innovation "as innovations that change

the architecture of a product without changing its components" (Henderson and Clark, 1990, p. 9), they also note that "architectural innovation… does not mean that the components themselves are untouched" (p.12). A careful review of the literature and the process of translating a verbal theory into a formal model revealed that architectural innovations, as conceived by Henderson and Clark (1990), come in different forms that are related to the type of component changes that accompany a change of interactions. One type of component change is an incremental adjustment in components, which often may be required to fit components into a new architecture. These are such minor modifications to a component that other components, which interact with the changed component, are not affected. A second type of component change is a *substantial* change, which is a redesign of a component that does affect components that interact with this focal component. It is this type of component change we described in the last section (and it is the usual way how "component changes" have been modeled in the literature (e.g., Baumann and Siggelkow 2013, Ethiraj and Levinthal 2004a)).

Henderson and Clark discuss one further type of component change, a change that requires a new "core concept." A core concept overturn occurs when existing knowledge about a component becomes obsolete because the new component requires completely different scientific knowledge.[1] Given that changes in core concepts lead to radical innovations and that our focus is on architectural innovation, we conceptually bound component changes in our model to within their core concepts. It is important to note, though, that "substantial' component changes, as we defined them, do not require changes in core concepts, but can also occur within a core concept.

Observing whether and when substantial component changes accompany an architectural change allows us to be more differentiated with respect to architectural innovations. An architectural innovation that only involves minor modifications to components comes closest to the ideal of a *pure architectural innovation*. This innovation is triggered by a design insight that a rearrangement of components (that also changes interdependencies among them) could lead to a performance improvement or solve a particular

[1] For instance, if the function of "cooling" is not performed by a spinning rotor (as in a fan) but by compressing a refrigerant (as in an air conditioner), this would constitute a change in "core concept."

problem. (The "design insight" can be deliberate or could have arisen from chance experimentation.) For instance, the order in which sub-routines are executed in a software program can have substantial performance implications (as we found out in our own coding efforts for this paper). By re-arranging routines, and making minor modifications to incorporate the order in which routines exchange information with each other, the run time of code can be substantially improved.

A different type of architectural innovation is also triggered by a design insight, but in order to gain the full benefit from the new architecture, substantial changes to components have to be implemented subsequently. Indeed, it is the difficulty to recognize these substantial changes that has often been highlighted as the key problem that incumbent firms face when encountering an architectural innovation. We suggest to call this case a *composite architectural innovation*, as it is composed of both a change in architecture and of components.

Lastly, the architectural change itself can be preceded and triggered by a more substantial component innovation. For instance, Henderson and Clark (1990) note how the introduction of high-strength-low-alloy steel allowed new architectures to arise for automobile bodies. We term these types of architectural changes *component-driven architectural innovations*. That is, a substantial change in a component may free up previously not feasible linkage changes to this and/or other components and lead to the discovery of a new architecture (and possible further subsequent component changes.)

To illustrate our suggested typology, we provide a number of descriptions of architectural innovations that have been discussed in the literature on architectural change. An example of pure architectural innovation is provided by Christensen, Suarez and Utterback (1998), who illustrate how IBM introduced in 1973 the Winchester architecture, an architectural innovation to the disk-drive industry that provided superior performance. While previous architectures allowed for individual disk stacks to be removable, the Winchester architecture sealed "the entire disk drive-heads, disks, motors, bearings, and everything else – inside a dust-free housing" (1998:S210). Consequently, the reading and writing heads of the drive could be moved much closer to the disk surface, which improved recording density substantially. This innovation mainly changed the interaction among components, but did not involve any

substantial changes to components.

A more recent example of a seemingly pure architectural innovation, in which existing technology can be embedded in a new architecture, is the move from 2-dimensional to 3-dimentional chip architecture:

> "To demonstrate its progress, Intel showed off a new 3D packaging technology called Foveros, which for the first time brings the benefits of 3D stacking to enable logic-on-logic integration. It's a way to put a couple of chips together vertically in a package and shorten the distance electrical signals have to travel, thereby improving efficiency and power consumption for the whole system…. The technology provides tremendous flexibility as designers seek to "mix and match" intellectual property blocks with various memory and I/O elements in new device form factors." (Takahashi 2018).

The notion of "mixing and matching" existing components, but linking them differently, captures well the idea of a pure architectural innovation.

A good example of a "composite architectural innovation," where architectural change requires substantial subsequent component change, is Henderson and Clark's (1990) description of innovations in the photolithographic industry. They illustrate how Canon created a new, better performing product that involved a new interaction between the gap setting mechanism and the operation of the mask and wafer in manufacturing semiconductor devices. However, to gain the full benefit of the new architecture, substantial changes to the gap-setting components were required. Kasper, the previous leader in the industry, failed to realize that these component changes were required, and ultimately failed.

Similarly, studying the U.S. bicycle gear shifting market, Fixson and Park (2008) and Park et al. (2018) show how Shimano changed its bicycle drivetrain product architecture in the mid 80s to create a better performing product. In particular, "index shifting was an architectural innovation wherein … the linkages between components changed dramatically" (Park et al., 2018: 330). At the same time, though, in order to get the full benefit from the new architecture, "Shimano had to redesign four relevant components – shifter, derailleur, freewheel, and chain." (Fixon and Park, 2008, p. 1304).

The work by Kapoor and Adner (2012: 1235), who study innovations in the DRAM manufacturing industry, contains examples that can be coded as "component-driven architectural innovations." In some of the architectural innovations they analyze, substantial innovations in a

component (e.g., a change in the wavelength of the UV light) enabled a new architecture of the manufacturing process to arise.

Up to this point, we have described architectural innovations in the spirit of Henderson and Clark (1990): the main goal of developers in their search for architectures is to find an architecture that improves the performance of a product. In this case, designers are not concerned with what kind of pattern of interactions emerges among the components; i.e., the redesign objective is *performance-focused*.

If we define architectural innovations as innovations that involve changes in the interactions among components, then a second type of architectural innovation can be identified: Designer might seek a particular interaction pattern, e.g., an architecture that facilitates internal coordination efforts for production, component innovation, and problem solving. This redesign objective arises especially in decentralized innovation settings, in which designers seek to create an architecture that is more easily "searchable." This objective has been at the center of the modular design literature that not only studies the consequences of modularity (as discussed in 2.2.) but also the active creation of such modular systems. For instance, MacCormack et al. (2006) examine how the web browser Mozilla underwent an orchestrated architectural redesign toward a more modular pattern to promote greater code development from external contributors. While the ultimate goal of such redesigns is higher performance as well, the more proximate goal is to create a different, more desireable pattern of interactions. Consequently, we term this redesign objective *pattern-focused*.

To summarize, we can characterize architectural innovations along three dimensions, as shown in Figure 1. First, is the *redesign objective* performance-focused or pattern-focused? Second, what *triggers* an architectural innovation? Is it a design insight or a substantial component innovation? Third, what are the *subsequent component changes* that are necessary to achieve the full benefit from the architectural change? Are only minor adjustments necessary, or are more substantial component changes necessary?

[FIGURE 1 ABOUT HERE]

MODEL

3.1. Model Description

We develop a model of product innovation, in which a designer searches for performance improvements by testing alternative product designs. In line with prior research, we focus on the product design as the unit of analysis (Baumann and Siggelkow 2013, Henderson and Clark 1990, Kapoor and Adner 2012). Further, we adopt the literature's key distinction between the two dimensions of product design, that is, the choice of components (*component design*) and the linkages among components (*architectural design*). Accordingly, we define a component as "a physically distinct portion of the product that embodies a core design concept (Clark 1985) and performs a well-defined function" (Henderson and Clark 1990: 11). This definition of "component" is similar to Ulrich's (1995), who illustrates how physical components fulfill the functional requirements of a particular product. We further denote that the architecture of a product "lays out how the components will work together" (Henderson and Clark, 1990: 11). As a result, the performance of a particular product is a function of its component choices and the underlying architecture (i.e., the way components influence one another). Table 1 shows a full overview of the relevant terminology used in this paper.

[TABLE 1 ABOUT HERE]

An important part of our model of product innovation constitutes the search behavior of the product designer. Because performance improvements can result from changes to components as well as from changes to the linkages, product designers will experiment with both types of product changes. In line with the predominant focus on component innovation, a designer in our model first exhausts her search for component improvements, before testing a new architecture. When the designer cannot find any more component improvements, this product design (components and architecture) constitutes the "current design." At this point, the designer experiments with a prototype that involves changes to the architecture. With this new prototype in hand, the designer searches for component changes that may have become performance-enhancing given the new architecture. When there are no further component improvements to be found, the performance of the prototype is compared to the performance of the

current design and becomes the new current design if performance is higher. Otherwise the designer keeps the (original) current design and uses it as the starting point for a new prototype experiment with regards to architectural changes.

3.2. Modeling Product Design

3.2.1 Definition of architectural design

In our model, a product is made up of *N* components and a particular architecture that shows how the *N* components interact with each other, which can be visualized in the form of an *N* by *N* influence matrix. This matrix arrays the *N* components on the horizontal as well as on the vertical axis. An "x" in a cell of this matrix denotes that the value that the row component creates (i.e., how well the row component performs) is influenced by the design of the column element. For instance, an entry in the [1st row, 2nd column] would mean that the value of the 1st component is influenced by how the 2nd component has been designed. We use the term "influence matrix" (as opposed to "interaction matrix" or "interdependency matrix") to be explicit that influences are directional and do not need to be symmetric. Component A may influence component B, but not vice versa. The diagonal is always filled with "x" because the value of each component is influenced by itself. Influence matrices differ in the number and the position of the off-diagonal entries. We refer to the number of off-diagonal entries in an influence matrix as the *complexity* of the product design. (Gell-Mann 1994, Simon 1962). That is, complexity refers to the overall degree to which components' contribution values are influenced by the way other components are designed. The influence matrix contains $N^2–N$ off-diagonal positions that could be filled with an "x." For instance, in the case of $N = 12$, there are 132 off-diagonal positions, and *complexity* can range from 0 to 132.

3.2.2 Component combination search space (performance landscape)

To conceptualize search for good component combinations, we borrow assumptions from previous models of organizational search. In particular, we consider that each component can only be designed in two fundamentally different ways (e.g., Levinthal 1997, Rivkin 2000). We denote the design

choice for each component with a "0" or a "1" (we could also have used "A" or "B" to denote the two possible design choices; there is no rank order implied by "0" or "1"). Given two design choices for each component, a product of N components has 2^N different component combinations. In our example of N = 12 components, examples of different component combinations would be 000000000000, 100000000000, 010001000100, or 111111111111.

We also adopt previous studies' methodology of creating a performance value for each possible product design. The mapping of the performances of all 2^N possible component combinations for a given architecture is referred to as the *performance landscape* of this architecture. Specifically, we build on Kauffman's (1993) NK fitness landscape framework[2], which assigns randomly drawn contribution values to the specific design of component i depending on the designs of the k_i components that influence component i. For instance, if component 1 is affected only by component 2, then the value of component 1 could take on four different values, depending on how components 1 and 2 are designed: $c_1(00)$, $c_1(10)$, $c_1(01)$, and $c_1(11)$. Each possible contribution value is drawn from a uniform distribution U~[0,1]. The overall performance of the product is then computed as the average over all N component contribution values.

Given our implementation, one should note that a switch from "0" to "1" corresponds to a substantial component change as described in Section 2.2., since a change in a component leads to performance implications for other components that are influenced by the focal component. In contrast, minor component adjustments are not explicitly modeled as changes in the component combination. The impact of adjustments to the performance of components in the context of an architectural change is, however, captured (see next section).

3.2.3. Changes in the performance landscape after an architectural change

A change to the architecture, i.e., the way components interact, can be captured by a change of the product's influence matrix. An architectural change may take place in the form of existing interactions

[2] There is an increasing body of research that has empirically supported the suitability of the NK framework in modeling the innovation process (Billinger et al. 2013, Fleming and Sorenson 2001, Ganco 2017, Lenox et al. 2010).

becoming obsolete (i.e., they disappear) as well as new interactions emerging among existing components. We model the consequences of architectural change in the following way: if after an architectural change component *i* is affected by one more component, or is not affected anymore by a component that previously affected it, new contribution values for component *i* apply. Like all contribution values, these are drawn from a uniform distribution U~[0,1]. In other words, if there is a change in row *i* of the influence matrix, component *i*'s contributions change, taking the new influence pattern into account. All components that do not experience a change in their influence matrix row retain their current contribution values. One should note that the change in a component's contribution includes any changes due to minor adjustments to the component that are required by the architecture. (For an extended example of how we model architectural change, please see Appendix A. For more modeling details, please see Appendix B).

3.3. Designers' search processes

3.3.1. Search for component combinations

We follow a long tradition of modelling search as incremental performance improvements (Cyert and March 1963, Lant and Mezias 1990, Levinthal 1997), where a decision maker evaluates an alternative against the status quo on some performance dimensions. This logic has also been applied to models of component innovation in product designs (e.g., Baumann and Siggelkow 2013).

At the beginning of each time period, the designer of a product conducts an evaluation of all component combinations (within the current architecture) that require exactly one component change ("local greedy search"). For example, for an $N = 12$ product, there are 12 adjacent combinations in addition to the current component combination. The designer always chooses the highest performing out of the N+1, here 13, combinations. Each time period, the designer can implement one change to the component combination, after which this time period ends. This search will eventually lead the designer to a local optimum within the current architecture. If the designer cannot find any performance-enhancing change to the component combination, i.e., she has reached a local peak on the performance landscape,

architectural search is started in the same time period.

3.3.2. Architectural search

To model the different types of architectural innovations, we start with the first row of our typology in Figure 1. That is, we model architectural innovations that are triggered by a design insight and either only involve minor component adjustments or require subsequent substantial component changes. Similarly in line with Henderson and Clark (1990), we first analyze the case of performance-focused architectural redesign objectives. In Section 4.4, we extend our model to encompass component-driven architectural innovation and pattern-focused architectural change.

We conceive of architectural search as an online search process (Gavetti and Levinthal 2000), that is, designers experiment with changes to the current product architecture and realize the performance consequences upon implementation. To fix terms, we call the existing product the "current design" and refer to the experiment that embodies the new architecture as the "prototype." We model different design experiments that vary with respect to the magnitude of the architectural change. The variable *scope* reflects the number of changes in the influence matrix whenever an architectural change is considered. In the tradition of simulation studies of innovation (Lant and Mezias 1990, Levinthal and March 1981), we model the changes to the influence matrix as a random process by which the designer's architectural experimentation will cause *scope* off-diagonal matrix entries to change from their current values to the opposite, i.e., from empty to "x" (adding an interaction) or from "x" to empty (removing an interaction). After implementation of these changes, the time period ends. In the time periods following the architectural change, the designer again conducts local, greedy component combination search for the prototype (as described under 3.3.1) until reaching a local optimum. Once a local optimum is reached within the new architecture, the designer compares the final prototype performance to the performance of the current design (i.e., before architectural and subsequent component changes were implemented). If the final prototype performance is greater than that of the current design, the prototype becomes the new current design. That is, the new architecture and possible component changes are "accepted," which starts

a new round of experimentation with respect to architectural change. However, if the final prototype performance is lower, the designer discards the prototype and goes back to its current design. In the next period, the designer would launch a new experimentation using its current design as the starting point.

In this conceptualization of architectural search, we do not envision that every "accepted" architectural change corresponds to a new product release. Rather we consider the, for instance, 200 periods of search as a certain amount of time spent on product development that is used for both component and architectural experimentation. (Please see Appendix C for a detailed example of the architectural search process.)

In Section 4.4, in addition to modelling component-driven and pattern-focused architectural change, we also relax the assumption that the scope of changes to an architecture is constant for a particular designer, and that all architectures are feasible.

3.4. Model parameters

Initial complexity: In all simulations, the number of product components is set to $N = 12$. For each simulation, we first set an initial level of architectural complexity and generate a random influence matrix based on this value. We specified levels of initial architectural complexity of 12, 36, 72, 108, and 132. For example, an initial complexity of 12 denotes that 12 randomly picked off-diagonal positions in the influence matrix will be filled.

Architectural search scope: We model architectural search as a change in *scope* elements of the current architecture's off-diagonal influence relationships per prototype experiment. To understand the effects of *scope* across a broad range, we set *scope* to 1, 3, 6, 12, 36, and 96.

Run time: For each level of initial complexity, we ran 500 simulations. Each simulation lasted for 200 periods. If a designer was in the middle of an architectural search in period 200, we gave the designer additional time to finish the local search and/or reset the architecture. (In section 4.3, we also explore search over 10,000 periods.)

4. RESULTS

We discuss our results in four parts. We start with the results for narrow architectural search in section 4.1, followed by the results of increasing levels of architectural search scope in section 4.2. In section 4.3, we consider how performance changes over time, and we finish with a number of model extensions in section 4.4.

4.1. Results for narrow architectural search

Figure 2 contains the main results for a *scope* = 1 change, i.e., architectural search that changes only one interdependency at a time. (Full results of all metrics for all scope and complexity levels can be found in Appendix D.)

[FIGURE 2 ABOUT HERE]

For *scope* = 1, the final performance is remarkably high. Consider the top panel of Figure 2. The first bar in this panel represents the product performance in period 200 when the starting *complexity* = 12. The final performance in this setting is 0.912. Had this designer conducted search only over components within the starting architecture, i.e., stayed on the initial performance landscape, the performance would have been only 0.694, the performance of the first local peak found (see bar 2). Even if the designer could have found the global peak in the initial performance landscape, the performance would only have been 0.703 (see bar 3). (All performance differences that are discussed for the *scope* = 1 case are highly statistically significant.)

What explains this high performance? Since architectural search changes the level of complexity of the product over time, one might wonder whether designers simply create landscapes that have a level of complexity that yield higher performance. For instance, the designer in the first panel starts with *complexity* = 12 and ends on average with a product that has *complexity* = 24.3. If such a complexity level would lead to higher local search outcomes, this could explain the higher performance resulting from architectural change. However, this turns out not to be the case.

To investigate this effect, we estimate two benchmarks: (a) the average height of local peaks

found via greedy search on landscapes that have the same complexity as the final landscapes that are created after architectural search, and (b) the average height of the global peaks on these benchmark landscapes. (For more details, see Appendix B.) In the case of the designer in the first panel, the first benchmark yields a performance of 0.700 (bar 4), and the second benchmark 0.739 (bar 5). Thus, even if designers had always found the global peak in these benchmark landscapes with similar complexity, they would not have achieved the high performance of 0.913. In sum, the performance of the designers conducting architectural search is not just driven by changing the complexity of the architecture.

So what can explain the performance benefits of architectural search? Designers engaging in architectural search experience new values for the contributions of the affected components. If they find a better performing outcome, they will accept the new architecture; if not, they will retain their old performance and search a new landscape. Over 200 time periods, a designer with *scope* = 1 and starting *complexity* = 12 actually experiments with on average 88.9 architectures, of which on average 15.2 lead eventually to higher performance and thus are accepted.

To investigate whether being able to search across many landscapes over time, and then being able to pick the best outcome that was found across all landscapes, can fully explain the performance benefits of architectural search, we construct two more benchmarks. Whenever throughout a 200-period simulation a designer engages in an architectural change, we create an entirely new landscape that has the same complexity as the landscape that the designer in our simulation had just created. This landscape is "entirely new" in the sense that all contribution values are redrawn, not just those of the affected component (as under architectural change). For each of these new landscapes, we find the performance of a designer who engages in local greedy search (from a random starting point); in addition, we record the performance of the global peak. We then take the maximum value over all these new landscapes within a simulation. In each panel of Figure 2, we report these benchmark results in bars 6 and 7. For the designer with *scope* = 1 and starting *complexity* = 12, the best performance outcome of local greedy search would have been 0.832 (bar 6). Even if the designer could have found the global peak in each newly created landscape and then picked the highest global peak across all landscapes, performance would only have

been on average 0.845 (bar 7), still far below the 0.912 resulting from architectural search. In sum, in this case, the effect of architectural search is not just driven by being able to sample many performance landscapes.

Similar results can be observed for the *scope* = 1 designer for higher starting complexities (see the middle and bottom panel of Figure 2). For all levels of complexity, the performance after 200 periods during which the *scope* = 1 designer was allowed to search for new architectures yields higher performance than if the designer had been able to choose the highest global peak in an equivalent number of randomly created landscapes with similar complexity.

4.1.1 Mechanism analysis of narrow scope search

What is special about (low-scope) architectural search? Narrow architectural search allows a designer to investigate the space of architectures in a more productive way than just choosing random new architectures. By changing one interdependency at a time, only one component's contribution values are affected. If the designer had already found high-performing designs for other components, *scope* = 1 architectural change allows to retain these contribution values, in contrast to having to search a completely new landscape with all new contribution values (as in our benchmark experiment). Moreover, in cases of pure architectural innovation, the focal component enjoys its higher performance without having to be changed substantially. As a result, the contributions of components that are influenced by the focal component are not affected either. Thus, over time, with low-scope architectural search the designer is able to find architectures in which each individual component achieves a very high performance. As this process unfolds, and as each component's performance improves, it also becomes increasingly likely that the component combination that the designer finds represents the global peak of the current architecture. By period 200 practically all designers with *scope* = 1 reach the global peak in their final landscape (or put equivalently, the designers have created architectures for which the final component combination is the optimal solution.) To deepen our understanding of how architectural search improves performance, it is helpful to investigate the effect of search scope, the topic of the next section.

4.2. The effects of changing the scope of architectural search

What happens when designers experiment with a broader scope in their architectural search? In general, for any level of starting complexity, as scope increases, long-run performance declines. Figure 3 shows the performance after 200 periods for different levels of *scope* and for different starting complexities. The figure also shows the 99.9% confidence intervals for each mean level depicted. Lastly, for reference, we included the maximum greedy search benchmark for *complexity* = 132 (which for *scope* = 1 corresponds to bar 6 in the bottom panel of Figure 2). Once the architectural change is very broad (e.g., *scope* = 96), the architectural search becomes very similar to redrawing the entire landscape, as practically all contribution values are affected. As a result, as one would expect, at a very high scope, designers do not outperform the benchmark created by redrawing entire landscapes.

[FIGURE 3 ABOUT HERE]

To better understand the effects of scope, we look in more detail at how architectural changes create performance improvements. We distinguish between two sources of performance enhancements following architectural change: performance improvements of existing components, leading to pure architectural innovations; and performance improvements due to subsequent redesign of components, leading to composite architectural innovations. In our model, pure architectural innovations correspond to finding a new architecture in which the current component combination (including minor adjustments) has a higher performance than in the old architecture, and no other significant component changes can be found that would yield performance improvements. In other words, the existing component combination is also a local peak in the new performance landscape (and this local peak has a higher performance than the local peak found in the previous architecture). In contrast, in a composite architectural innovation, both the architecture and the component combination changes. Thus, after designers have created a new protoype architecture, they discover performance-enhancing changes to the existing component combination, which yield a higher performance than the performance of the previously accepted product design.

Figure 4 shows the fraction of accepted architectures that were pure architectural innovations and those that were composite architectural innovations. As we can see, different levels of scope have very different underlying mechanisms that create performance improvements. For low levels of scope, most accepted architectural changes lead to pure architectural innovations, while for high levels of scope most accepted architectural changes lead to composite architectural innovations. This pattern is particularly strong for high levels of starting complexity, as seen in the right panel of Figure 4. (For an in-depth explanation of the underlying mechanisms generating these patterns, please see Appendix E).

[FIGURE 4 ABOUT HERE]

4.3. The effect of search scope on performance over time

In this section, we investigate how the degree of search scope affects performance over time. The role of "time" in simulation models is quite tricky. Even though a "period" is well defined and comparisons across model specifications is straightforward, two complications arise. First, how many periods do constitute the "short-run," and how many periods represent the "long-run"? It has been common practice to consider the steady state results of a model as the "long-run." This requires, however, that the model converges. With large search spaces and/or some degree of stochastic search behavior, convergence may not occur (or only after a very large number of periods). In this section, we report our results over different time windows, up to 10,000 periods, and let the reader decide which time frames to call "short-term" or "long-term."

The second complication of "time" in simulation models is that assumptions have to be made concerning how long certain processes take that are part of the model. For example, in our model, we specified that creating and testing a new architecture takes one time period, the same amount of time it takes to search for a local component design change. To account for the possibility that architectural searches might be much more difficult and time consuming than component design changes, we also report performance results in this section that are a function of "number of attempted architectural changes," thereby compressing the time we count towards finding local component design changes.

The investigation of performance over time is interesting in our setting, because search scope affects the probability of accepting new architectures in a given time window. This result is driven by two factors. First, high-scope designers engage in more composite architectural innovations, which require component changes that take time, and, if changes to components are required, high-scope designers need more changes until they find a local peak. The many component adjustments that are required leave less time for the high-scope designer to test as many architectures compared to narrow scope designers.[3]

Second, whenever a new architecture is investigated, the likelihood of finding a performance-improving architecture is lower for high-scope designers than for low-scope designers, since high-scope designers need to discover larger sets of higher-performing contribution values, rather than a higher contribution value for a single component.

The scope of architectural search not only affects the overall probability of finding new performance-enhancing architectures, but also *when* these architectures are found. As component contributions increase, all designers have a harder time finding better architectures, but this is true especially for high-scope designers. As a result, high-scope designers find better architectures rather early and few in numbers. In contrast, low-scope designers accept new architectures more continuously over time, with a relatively high probability to find a new architecture even after several accepted changes.

To explore how performance evolves over time, we extend the simulation to 10,000 periods, and report results in Figure 5. *Scope* = 1 performance flattens out after ~1,000 periods, while for other levels of scope performance continues to improve over time. This can be explained by the smaller number of architectural alternatives that can be explored by the *scope* = 1 designer. Over time, the *scope* = 1 designer will almost exclusively innovate via pure architectural innovation (see Appendix E). In this case, there are only 132 alternative architectures available for a given component combination. That is, per

[3] For example, under a starting *complexity* = 12, a *scope* = 1 designer tests on average 88.9 architectures, while a *scope* = 96 designer tests on average 43.9 (see column 8 in Appendix D). Likewise, for instance, for initial *complexity* = 12, a *scope* = 1 designer changes only on average 1.28 components if a change to the component combination is required, whereas a *scope* = 96 designers changes on average 3.13 components.

component, there are only 11 changes to the current influence structure. Especially when existing component contribution values are already high, the 11 alternatives per component will increasingly less likely lead to any improvement and the designer will run out of architectural alternatives – as shown in the horizontal performance curve for *scope* = 1 designers late in the simulation.

[FIGURE 5 ABOUT HERE]

In the early parts of the search process, a different effect drives the performance differences across levels of scope. While high-scope architectural search leads to fewer architectural changes over a given time window, if a performance-enhancing architecture is found, the average performance impact is greater. For instance, for starting *complexity* = 12, the average performance gain of an architectural change is 0.014 for a *scope* = 1 designer while it is 0.032 for a designer with *scope* = 96. Early in the search process, when contribution values are still close to average, changing many contribution values at once can yield bigger performance increases than those achieved by a *scope* = 1 designer who only changes one component's contribution at a time (and thereby affecting only 1/Nth of product value). This argument suggests that early in the search process, high-scope designers can outperform low-scope designers.

In the inserts in Figure 5, we plot the currently accepted architecture's performance for the first 35 time periods given a starting complexity of 12 and 132, respectively. Under low complexity, all levels of scope perform quite similar within the first ~10 periods. Under high complexity, designers with larger levels of scope clearly outperform in the early stages while intermediate levels of scope start underperforming everyone else after ~15 periods. This is due to the fact that medium level scope designers require similar adjustments to components as high scope designers but their average performance gain is lower. Over time, medium level designers catch up and perform in-between narrow and large scope designers.

As noted at the beginning of this section, if experimenting with new architectures takes much more time than component changes, showing performance results as a function of "number of attempted architectural changes" (rather than by "periods") would be helpful as it eliminates differences due to

search for new component combinations. In Figure 6, we plot the number of attempted architectural changes against the performance of the currently accepted architecture for up to 3,000 architectural searches (shown on a logarithmic scale). The results of this figure clearly illustrate the short-run benefits of higher scope. Designers with narrow scope will initially underperform designers with higher levels of scope because of lower performance gains per accepted architecture. Especially with greater initial complexity, higher levels of scope become more valuable early in the search process.

In sum, in the short-run, designers with high scope have an advantage, especially for high levels of complexity. For longer search processes, designers with *scope* = 1 perform best; and for very long search processes, slightly larger levels of scope can be beneficial as *scope* = 1 designers run out of architectures to test. (One should note, though, that the last result truly requires many architectural innovation attempts. Designers with *scope* = 3 only outperform *scope* = 1 designers after more than 2,000 tested architectures.)

[FIGURE 6 ABOUT HERE]

4.4. Model extensions

4.4.1. Component-driven architectural change

So far, we only modeled substantial component innovations that followed an architectural change. In our first model extension, we investigate the effects of component-triggered architectural change, i.e., the bottom row of our typology depicted in Figure 1. To capture this type of architectural innovation, we adjust our model as follows: Once a designer cannot improve the current product design, the designer creates a prototype in which one randomly chosen component is changed ("the component change that inspires/enables the new architecture"). Only then the architecture of the prototype is altered (as before, this will cause a change in *scope* number of interdependencies in the influence matrix). Subsequently the designer continues with search for additional component improvements. Analogously to our core model, if the new final prototype that is found by this process leads to a higher performance than the current design, the prototype is accepted; otherwise the designer reverts back to its previously accepted design

and starts another round of the architectural innovation process spurred by a component change.

We find that this process of "component-driven architectural innovation" creates a very similar pattern of results to our core model (see Appendix F). In the long run, low-scope designers outperform high-scope designers; in the short run, high-scope designers have a performance advantage if initial complexity is high.

4.4.2. Probability-based search scope

To cleanly study the implications of different levels of search scope, we used a fixed scope for a given designer throughout the entire simulation. In real innovation settings, it is, however, quite unlikely that designers have always the ability to precisely control the number of changes in linkages that arise from an architectural innovation. To allow for less precise control, we extend our core model by making the number of interaction changes stochastic. In particular, when a new architecture is created, any off-diagonal matrix position is flipped (from an existing interaction to no interaction, and vice-versa) with probability *p*. We choose different values of *p* to have in expectation *scope* number of changes every time an architectural change is attempted. Thus, we model different design innovation processes that tend to lead to different degrees of architectural change on average, but can vary over time.

For instance, $p = \frac{1}{(12^2-12)}$ would correspond to *scope* = 1, as there are 12^2-12 off-diagonal entries. A designer with this level of *p* would change on average one interaction, but could change many (or none) in any period in which architectural changes are considered. The results we obtain with this probability-based scope rule are "blended" results of designers that have a fixed scope rule. Designers with very low levels of *p* tend to underperform their respective fixed scope counterparts; designers with medium levels of *p* outperform their fixed scope counterparts, while designers with high levels of *p* tend to have very similar results (please see Appendix G for details).

4.4.3. Restricting architectural search

In our main model set-up, we allow designers to change any of the entries in the influence matrix.

While many new architectures are rejected because they do not lead to a performance increase, designers still have a large number of possible new architectures to search over. What would happen if the space of possible new architectures is much smaller, because many influence patterns might not be feasible at all? To start exploring this issue, we investigated *scope* = 1 designers. In the current model, these designers have 132 possible new architectures available every time they search. In a modification to the model, we restricted this to 12 new architectures. Thus, every time designers are stuck on a local peak, they can only change one out of 12 entries in the influence matrix. If designers find a performance enhancing new architecture, they will accept it and have 12 new architectures available to search. However, if they cannot find any performance enhancing new architecture among the 12 architectures, designers stop their entire search effort. (For instance, for starting *complexity* = 12, about half of the *scope* = 1 designers are now stuck after 100 periods). Relative to our unconstrained model, performance falls between 3-7% for the *scope* = 1 designer; architectural search is still, however, outperforming the "maximum local peak benchmark" (bar 6 in Figure 2). The assumption that a designer would completely stop architectural search efforts, rather than switching to a larger scope, is of course also extreme. It is, however, reassuring that our main performance results for the *scope* = 1 designer hold even if the architectural search is severely restricted.

4.4.4. Pattern-focused architectural search

As noted in Section 2.3., one research stream in the prior literature has studied designers' attempts to redesign product architectures toward a particular pattern, for instance, a modular architecture. In our core set-up, we modeled search that was pattern agnostic and did not impose any constraint of a desired interaction pattern for the emerging architecture. (The final architectures, i.e., the landscapes that arose from this unguided process, had a number of interesting properties, though; see Appendix H). In our second main model extension, we modify our model to explore the effects when architectural search seeks to achieve a specific pattern. In particular, we follow Rivkin and Siggelkow (2007) in implementing two target patterns: (a) centralized and (b) modular (i.e., block-diagonal). In a centralized structure, there exist

a few central components that affect all other components, while in a modular structure interdependencies are mainly clustered within different sets of components. We chose to study a centralized structure, because this pattern has been shown to be particularly desirable for local search as it produces fewer local peaks than other patterns (Rivkin and Siggelkow 2007). We chose a modular structure, because of its ubiquitous relevance in the product design and coordination literature.

To study the effects of pattern-seeking architectural search, we modify our model in the following way. Each time designers search for a change to the current architecture they do not change randomly chosen entries in the influence matrix, but compare the current architecture with the desired (target) architecture (e.g., centralized). Entries that are not aligned with the target pattern will be considered for change.

We find that pattern-focused search qualitatively returns the same results as our core model. For both types of patterns (centralized and modular), lower levels of scope lead to higher performance in the long-run than higher levels of scope. (For more details, see Appendix I).

5. DISCUSSION AND CONCLUSION

Product innovation may result from a change in components, a change in the underlying product architecture, or both. While previous research explored how firms can improve their search for new component combinations, it had remained silent on how to search for new architectures. Our research provides a first step toward a better understanding of how the scope of architectural search affects product innovation. In the following, we discuss our findings and possible implications.

5.1 Formalizing a verbal theory of architectural innovation

We have taken a first step toward formalizing the concept of architectural innovation, which has its roots in verbal theory (Henderson and Clark 1990). In the process of formalizing this concept, we have surfaced important and often implicit details underlying architectural innovation, creating a new, more differentiated typology. Architectural innovations can be classified using three dimensions: the *redesign objective*, the *trigger* of architectural innovation; and the necessary *subsequent component changes* to

take full advantage of the new architecture. Consequently, as shown by Figure 1, architectural innovations can be classified as *pure architectural innovation*, *composite architectural innovation*, and *component-driven architectural innovation*. Moreover, each of these types can be pursued with a pure performance improvement in mind but no objective regarding the architectural structure that emerges (i.e., performance-focused), or with a desired design pattern (e.g., modular) in mind (i.e., pattern-focused).

Our proposed typology of architectural innovation suggests new research paths. For instance, the question arises how traditional sources of search (e.g., problemistic and opportunistic search) relate to the trigger of architectural change, i.e., design insight or component-driven architectural search. For example, problemistic search, often prompted by a relative performance decline, renders designers drawing on proven solutions (Cyert and March 1963), which may suggest an initial search focus on component changes, which then may trigger architectural changes. The relative dominance of component-driven architectural changes observed by Kapoor and Adner (2012) took place in the semi-conductor industry, where innovation is driven out of necessity from increasing competition and innovation speed. In contrast, insight-driven architectural change (e.g., rearranging existing components in a novel way) may be more of a creative process that requires an opportunity mindset (Jackson and Dutton 1988). A promising starting point may be to compare architectural innovation processes in well-established industries to those in newly emerging, more entrepreneurial industries.

Another interesting avenue to pursue is to further study the relationship between the different types of architectural innovation and the ability of other firms to react to such an innovation. For instance, are pure architectural innovations easier to copy as they do not involve any substantial changes to components (a frequent barrier noted in the literature)? Or are pure architectural innovations perhaps more difficult to spot by competitors, as they are even more subtle than architectural innovations that involve substantial component changes?

5.2 Architectural search as driver of product performance

In line with prior, empirical research, we find that architectural search can outperform pure component innovation (Kapoor and Adner 2012, Christensen et al. 1998, Fixson and Park 2008). Our model reveals the scope of the architectural search, i.e., how many changes to interdependencies are considered at once, to play an important role in the innovation process that has not been discussed previously. It is well documented that under high complexity designers typically benefit from larger scale experimentation when the objective is to find better solutions *within* a given architecture (Holland 1975, Levinthal 1997). Intriguingly, we find the opposite relationship when it comes to finding better architectures. Narrow architectural search yields higher performance enhancements over time as compared to broader levels of scope, regardless of product complexity. While narrow architectural search leads to high performance in the long run, it does require many changes over time though, each yielding only modest performance improvements. In contrast, we found that architectural search with broad scope – if it succeeds – can yield larger instantaneous performance enhancements. These performance-enhancements of broader scope search are a consequence of architectural change that unlocks new component combinations. However, especially over time, finding large sets of interaction changes that improve performance is increasingly difficult.

Because search scope can have important implications given the stage of a product and the type of innovation it leads to, future empirical work should explore the underlying drivers of architectural scope. For example, through which means may designers be able to "control" the magnitude of linkage changes and do these mechanisms differ for insight- and component-driven architectural changes? A particularly suitable setting to study these and related questions may be software architectures. Software products are version-controlled, i.e., source code traces back any changes by date and user. Moreover, the architecture of a software design has relatively few limitations compared to many physical products that typically have material-related and natural laws constraining viable architectures. An opportunity to capture different scopes of architectural search may exist in studying changes to function calls between source files (i.e., dependencies among them) in version-controlled software projects (MacCormack et al. 2006). Because version-controlled (open source) projects can track proposed as well as implemented

changes down to the individual developer, one may be able to study different scopes of such function call modifications, i.e., architectural changes.

5.3 The power of pure architectural innovation

The analysis of our model results also revealed an intriguing conceptual insight. We observed an interesting similarity between the power of modularized systems and pure architectural innovation. A common attempt to increase the performance of a product is for designers to search for a higher performing component design. This strategy of product improvement is, however, quickly exhausted in the presence of many interdependencies among components. When a designer redesigns one component in a higher-performing way, such change affects the performance of all other components that are dependent on that component. As a result, the performance improvement in the focal component might be negated by the changes of performance in the other components. Only if components are completely independent, or put differently, if the system is perfectly "modularized" (Baldwin & Clark 2000), would an improvement in a focal component not affect the performance of all other components.

Our research suggests that a different way to increase performance of a product is to find an architecture in which the components with their existing designs (plus some minor adjustments) reach a higher performance enabled by a pure architectural innovation. As we have seen in our results, these pure architectural innovations are difficult to find when many interdependencies are changed simultaneously. However, if a designer can experiment with architectural changes of a rather narrow scope, pure architectural innovations can more likely arise. The power of narrow architectural search is that it helps discover a higher performance of a focal component without requiring design changes to the focal component. As a result, the higher performance of the focal component can be realized without affecting the performance of other, interdependent components. In a sense, pure architectural innovation helps to "quasi-modularize" performance improvements despite rich interactions among components. Since the designs of components do not have to be changed, the effect of the interdependencies among the components does not come into play and the performance of each individual component can be improved

in an isolated fashion over time.

5.4 Model limitations and future extensions

The synthesis of the existing literature and its translation into a formal simulation have revealed modeling opportunities that could further advance innovation research. In particular, we have identified opportunities with respect to richer modeling of types of component changes, extended modeling of pattern-focused architectural change, assuming different distributions for performance implications of architectural innovations, and modeling of search cost. We will discuss each of them in the following.

First, practically all models of innovation have allowed for only one type of component change, which we term "substantial change" in our paper. The conceptual synthesis of our model and the underlying literature point to the opportunity of developing new models that would allow for a systematic study of various types of component changes. We chose the NK framework as a first step toward formalizing architectural search and innovation, because it allowed for a consistent and comparable model in reference to the prior literature on component search. However, future models may want to explore how the innovation process unfolds for different component changes (e.g., substantial, minor adjustments, and core-concept overturn), and how such an extended typology may qualify and advance previous findings.

Second, we believe that the distinction of architectural search for performance improvement versus architectural search for particular interaction patterns bears potential for future contributions. Especially when architectural search is conducted by multiple actors simultaneously, novel coordination needs may arise that are different from coordination solutions previously studied in the case of component search. For example, which architectural patterns are more feasible to actively search for? It is not immediately clear whether architectural search that requires at least some performance preservation (or even improvement) can identify more easily a modular, centralized, or any other pattern.

Third, we find that as product performance increases, architectural innovations become more difficult, especially those that also involve changes in components. In part this is driven by the

assumption that after an architectural change, the new contribution values are drawn from the same distribution as under the old architecture. If a new architecture "unlocks" possible, previously not available big performance gains for some components, we would see more of these composite architectural innovations in our model. From a modeling perspective, the problem, of course, is to specify reasonable assumptions for distribution changes that actually yield insightful results. The reason we chose the same distribution across all changes was to limit the risk of "biasing" results toward a search that merely takes advantage of a skewed distribution. Further, such "radical" performance changes may also be more appropriate in the context of core-concept changes instead of within core-concept component changes. However, in practice one could imagine that such "unlocking" of disproportional performance gains can happen in some form (at least for some architectural changes), which should motivate future modelling attempts.

Fourth, as with most simulation models in this domain, our model has deliberately refrained from any cost assumptions to provide results with respect to the (gross) benefit of a particular architectural search strategy. Modeling costs creates many free parameters, which depending on the modeler's choice can render core results stronger or make core results disappear altogether. For example, when modeling costs one would have to make an assumption whether two single changes to an architecture in sequence are more or less costly to a firm than one change to the architecture that involves two linkages. Consequently, our results may carry different implications for different cost-structures associated with architectural search. For example, when testing a prototype incurs high cost, e.g., because it requires costly manufacturing of a physical life-size prototype regardless of how many interactions have been changed, incremental architectural search will become very expensive and may lose some of its benefits accordingly. In contrast, if testing a new prototype is not incurring significant cost, e.g., because it is software based and requires only to compile an altered code, low-scope architectural change might be most beneficial.

The implications of our results further differ depending on the magnitude of the costs that are associated with changing a linkage. For example, when the implementation of a linkage change requires

changes to the organization as well, the firm might be better off with one wide-scope overhaul of the product and the organization compared to continuous, incremental changes to the product architecture, which would trigger ongoing and expensive reorganization.

Finally, our model's implications may be more appropriate the higher the cost associated with component redesign. Higher component redesign cost render wide-scope changes more costly because such changes go hand in hand with several changes to components. Consequently, our finding of benefits to narrow architectural search would be particularly applicable to such a scenario because incremental architectural change will more likely leave the component design intact. Component redesign cost may be high when new component designs require new supplier relationships and/or component knowledge is expensive to acquire. In contrast, component redesign cost may be low when alternative component designs are already available within the organization.

5.5 Conclusion

In sum, our study provides a first attempt to understand how designers' architectural search can influence the product innovation process. As our results suggest, different search practices with respect to the scope of interdepenceny changes have an impact on what kind of innovations tend to arise: pure architectural innovations or architectural innovations that also involve substantial component changes. Given that architectural innovations can be a source for creating competitive advantage, a deeper and more refined understanding of how different types of architectural innovations arise can be very valuable. We hope that this study has provided a promising starting point for a new research stream with a focus on different types of architectural search processes as an important lever for innovation and firm performance.

REFERENCES

Baldwin CY, Clark KB (2000) *Design Rules: The Power of Modularity* (MIT Press, Cambridge, MA).

Baumann O, Siggelkow N (2013) Dealing with Complexity: Integrated vs. Chunky Search Processes. *Organ. Sci.* 24(1):116–132.

Billinger S, Stieglitz N, Schumacher TR (2013) Search on rugged landscapes: An experimental study. *Organ. Sci.* 25(1):93–108.

Brusoni S, Prencipe A, Pavitt K (2001) Knowledge specialization, organizational coupling, and the boundaries of the firm: why do firms know more than they make? *Adm. Sci. Q.* 46(4):597–621.

Christensen CM, Suárez FF, Utterback JM (1998) Strategies for survival in fast-changing industries. *Manag. Sci.* 44(12-part-2):S207–S220.

Christensen CM, Verlinden M, Westerman G (2002) Disruption, disintegration and the dissipation of differentiability. *Ind. Corp. Change* 11(5):955–993.

Clark KB (1985) The interaction of design hierarchies and market concepts in technological evolution. *Res. Policy* 14(5):235–251.

Csaszar FA, Siggelkow N (2010) How Much to Copy? Determinants of Effective Imitation Breadth. *Organ. Sci.* 21(3):661–676.

Cyert RM, March JG (1963) *A behavioral theory of the firm* (Blackwell, Oxford).

Eppinger SD, Whitney DE, Smith RP, Gebala DA (1994) A model-based method for organizing tasks in product development. *Res. Eng. Des.* 6(1):1–13.

Ethiraj SK, Levinthal D (2004a) Modularity and innovation in complex systems. *Manag. Sci.* 50(2):159–173.

Ethiraj SK, Levinthal D (2004b) Bounded rationality and the search for organizational architecture: An evolutionary perspective on the design of organizations and their evolvability. *Adm. Sci. Q.* 49(3):404–437.

Fixson SK, Park JK (2008) The power of integrality: Linkages between product architecture, innovation, and industry structure. *Res. Policy* 37(8):1296–1316.

Fleming L, Sorenson O (2001) Technology as a complex adaptive system: evidence from patent data. *Res. Policy* 30(7):1019–1039.

Galunic DC, Eisenhardt KM (2001) Architectural innovation and modular corporate forms. *Acad. Manage. J.* 44(6):1229–1249.

Ganco M (2017) NK model as a representation of innovative search. *Res. Policy* 46(10):1783–1800.

Gavetti G, Levinthal D (2000) Looking forward and looking backward: Cognitive and experiential search. *Adm. Sci. Q.* 45(1):113–137.

Gell-Mann M (1994) *The Quark and the Jaguar: Adventures in the Simple and the Complex* (Freeman, New York).

Henderson RM, Clark KB (1990) Architectural Innovation - the Reconfiguration of Existing Product Technologies and the Failure of Established Firms. *Adm. Sci. Q.* 35(1):9–30.

Holland JH (1975) *Adaptation in Natural and Artificial Systems* (University of Michigan Press, Ann Arbor, MI).

Jackson SE, Dutton JE (1988) Discerning Threats and Opportunities. *Adm. Sci. Q.* 33(3):370–387.

Kapoor R (2013) Persistence of integration in the face of specialization: How firms navigated the winds of disintegration and shaped the architecture of the semiconductor industry. *Organ. Sci.* 24(4):1195–1213.

Kapoor R, Adner R (2012) What firms make vs. what they know: how firms' production and knowledge boundaries affect competitive advantage in the face of technological change. *Organ. Sci.*

23(5):1227–1248.

Katila R, Ahuja G (2002) Something Old, Something New: A Longitudinal Study of Search Behavior and New Product Introduction. *Acad. Manage. J.* 45(6):1183–1194.

Kauffman SA (1993) *The origins of order: Self-organization and selection in evolution* (Oxford University Press, New York).

Lant TK, Mezias SJ (1990) Managing discontinuous change: A simulation study of organizational learning and entrepreneurship. *Strateg. Manag. J.*:147–179.

Lenox MJ, Rockart SF, Lewin AY (2010) Does interdependency affect firm and industry profitability? An empirical test. *Strateg. Manag. J.* 31(2):121–139.

Levinthal D, March JG (1981) A model of adaptive organizational search. *J. Econ. Behav. Organ.* 2(4):307–333.

Levinthal DA (1997) Adaptation on rugged landscapes. *Manag. Sci.* 43(7):934–950.

Levinthal DA, Warglien M (1999) Landscape design: Designing for local action in complex worlds. *Organ. Sci.* 10(3):342–357.

March JG (1991) Exploration and Exploitation in Organizational Learning. *Organ. Sci.* 2(1):71–87.

March JG, Simon HA (1958) *Organizations* (Wiley, New York).

Milgrom P, Roberts J (1995) Complementarities and Fit - Strategy, Structure, and Organizational-Change in Manufacturing. *J. Account. Econ.* 19(2–3):179–208.

Nelson RR, Winter SG (1982) *An Evolutionary Theory of Economic Change* (Harvard University Press, Cambridge, MA).

Park WY, Ro YK, Kim N (2018) Architectural innovation and the emergence of a dominant design: The effects of strategic sourcing on performance. *Res. Policy* 47(1):326–341.

Rivkin JW (2000) Imitation of complex strategies. *Manag. Sci.* 46(6):824–844.

Rivkin JW, Siggelkow N (2003) Balancing search and stability: Interdependencies among elements of organizational design. *Manag. Sci.* 49(3):290–311.

Rivkin JW, Siggelkow N (2007) Patterned interactions in complex systems: Implications for exploration. *Manag. Sci.* 53(7):1068–1085.

Sanchez R, Mahoney JT (1996) Modularity, flexibility, and knowledge management in product and organization design. *Strateg. Manag. J.* 17:63–76.

Siggelkow N, Rivkin JW (2005) Speed and search: Designing organizations for turbulence and complexity. *Organ. Sci.* 16(2):101–122.

Simon HA (1962) The Architecture of Complexity. *Proc. Am. Philos. Soc.* 106(6):467–482.

Steward DV (1981) The design structure system: A method for managing the design of complex systems. *IEEE Trans. Eng. Manag.* (3):71–74.

Takahashi D (2018) Intel shows off its next big chip plans for 'architecture era.' *VentureBeat*. Retrieved (July 29, 2020), https://venturebeat.com/2018/12/12/intel-shows-off-its-next-big-chip-plans-for-architecture-era/.

Tushman ML, Romanelli E (1985) Organizational evolution: A metamorphosis model of convergence and reorientation. Cummings LL, Staw BM, eds. *Res. Organ. Behav.* (JAI Press, Greenwich, CT), 171–222.

Ulrich K (1995) The role of product architecture in the manufacturing firm. *Res. Policy* 24(3):419–440.

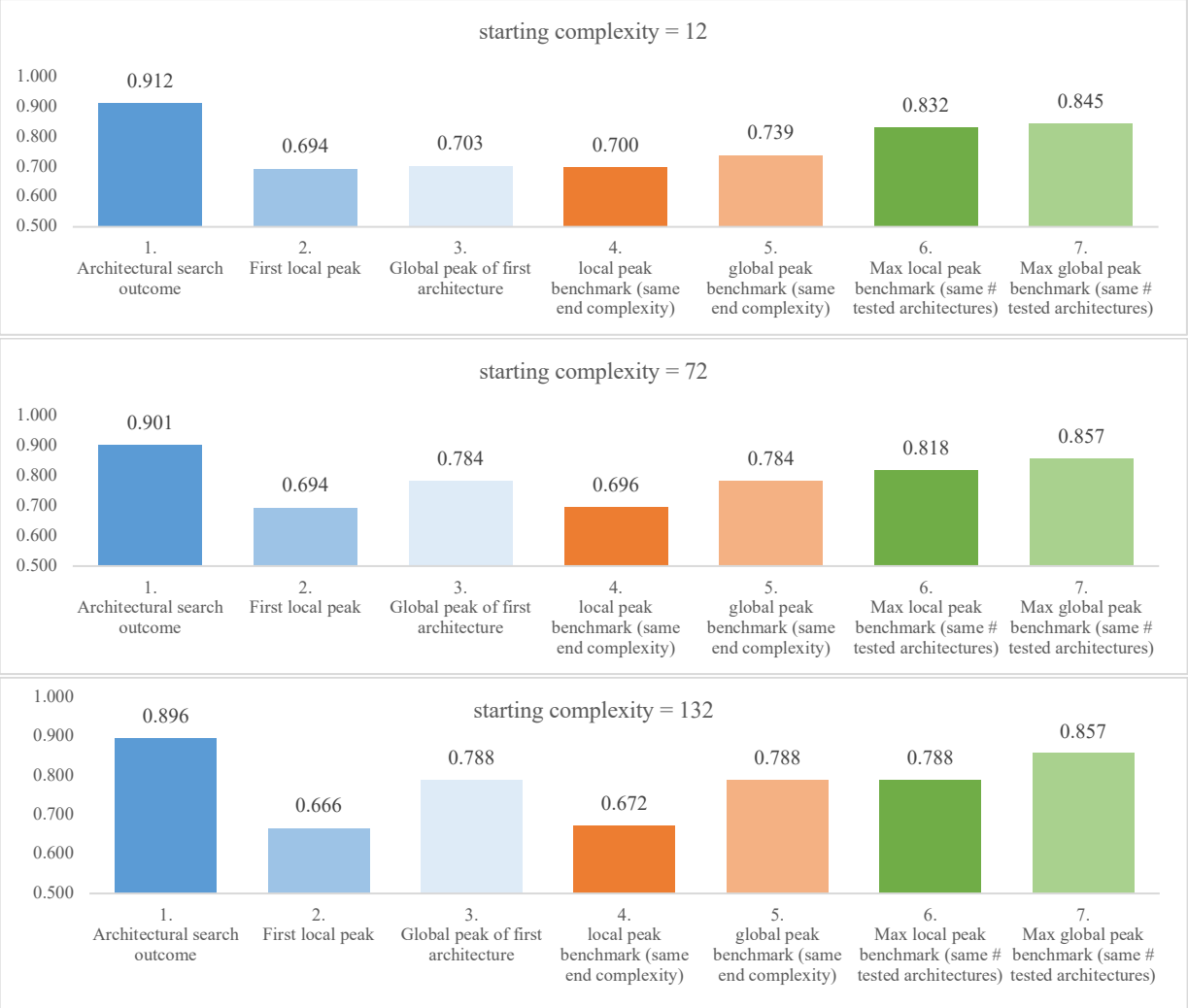
Table 1. Terminology overview

Construct	Definition
Component	A distinct element of the product that performs a specific function
Component combination	A particular set of design choices for each of the *N* components that make up a product. In our model, this is represented by a string of *N* "0s" and "1s"
Minor component adjustment	A component modification that does not affect other components that interact with the focal component
Substantial component change	A component redesign that affects components that interact with the focal component
Architecture	The pattern of how the *N* components of a product interact, represented by an *N* * *N* influence matrix
Architectural change	A change in the influence matrix (addition and/or removal of interactions)
Scope of architectural change	The number of simultaneous changes in the influence matrix
Pure architectural innovation	An architectural innovation that involves only minor component adjustments
Composite architectural innovation	An architectural innovation that requires substantial subsequent changes to components in order to gain the full benefit from the new architecture
Component-driven architectural innovation	An architectural innovation that is triggered by a substantial component innovation
Complexity	The number of off-diagonal interactions in the influence matrix
Product design	A product's component combination and its underlying architecture
Performance landscape	The mapping of all possible component combinations (for a given architecture) onto performance values

Figure 1: Typology of architectural innovations

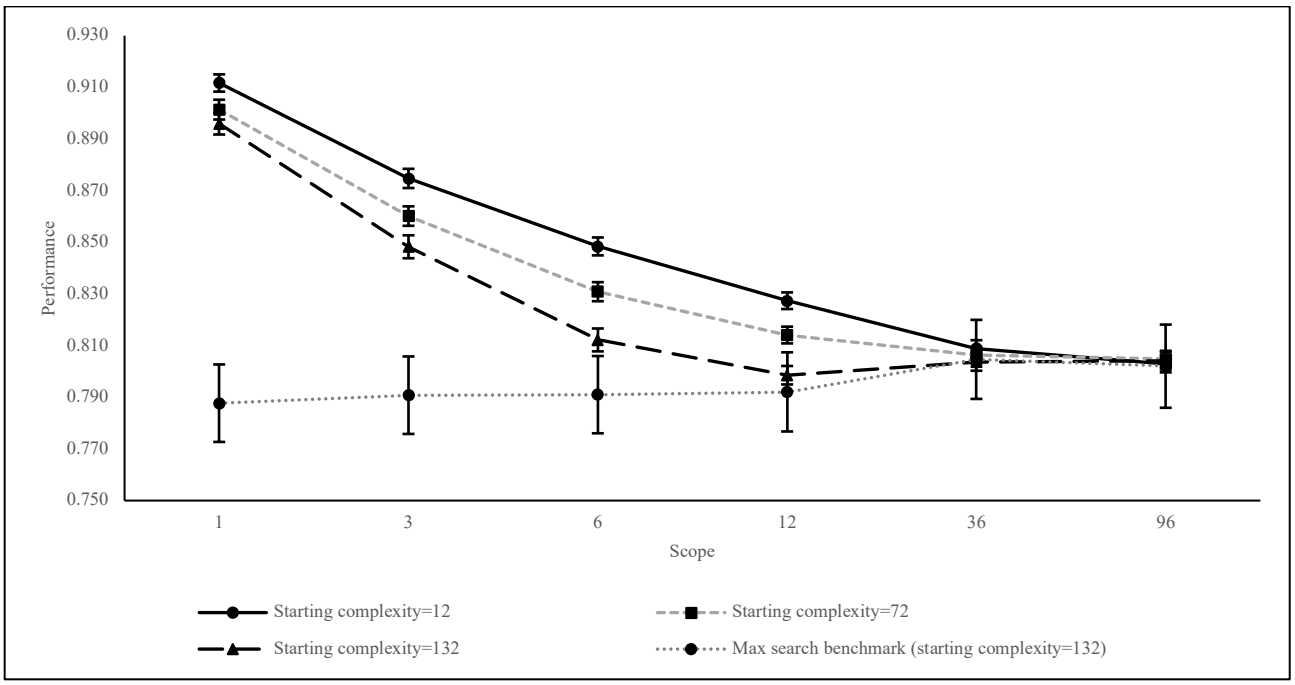
Redesign objective	Trigger	Subsequent component changes	
		minor adjustments	substantial
performance-focused	design insight	*pure architectural innovation*	*composite architectural innovation*
pattern-focused	substantial component change	*component-driven architectural innovation*	

Figure 2: Simulation results and benchmark statistics for narrow architectural search



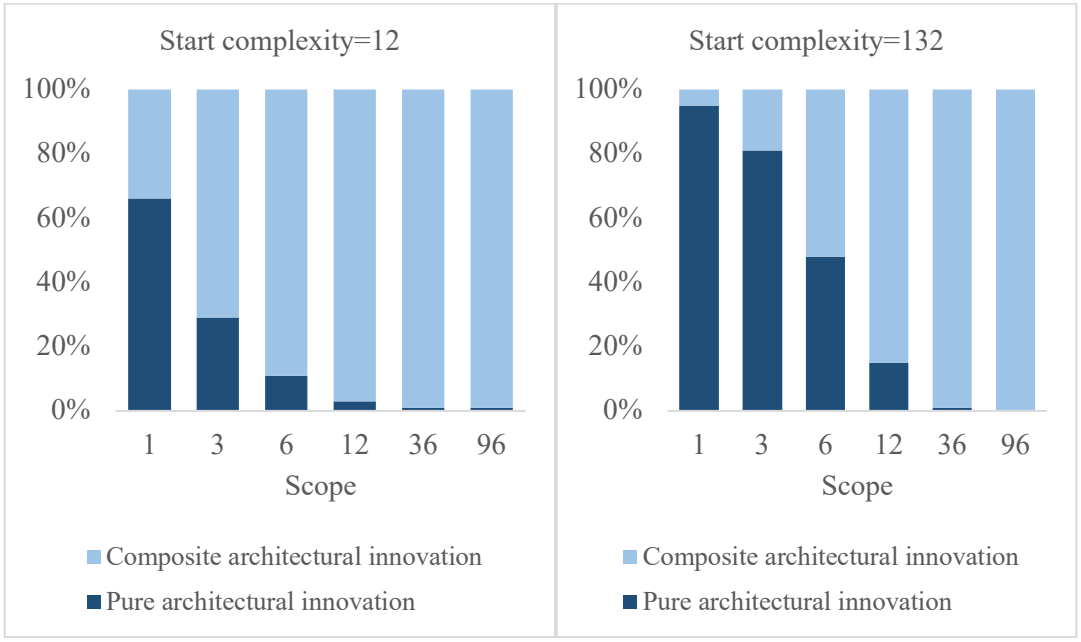
Note: Each of the three bar charts shows the average simulation results for a *scope* = 1 designer after 200 periods for a different starting complexity. For starting complexity = 12 (72 and 132), the average end complexity after 200 periods is 24.3 (71.2 and 117.7) and the designer tested on average 88.9 (101.2 and 105) architectures.

Figure 3. Architectural search performance for different levels of search scope



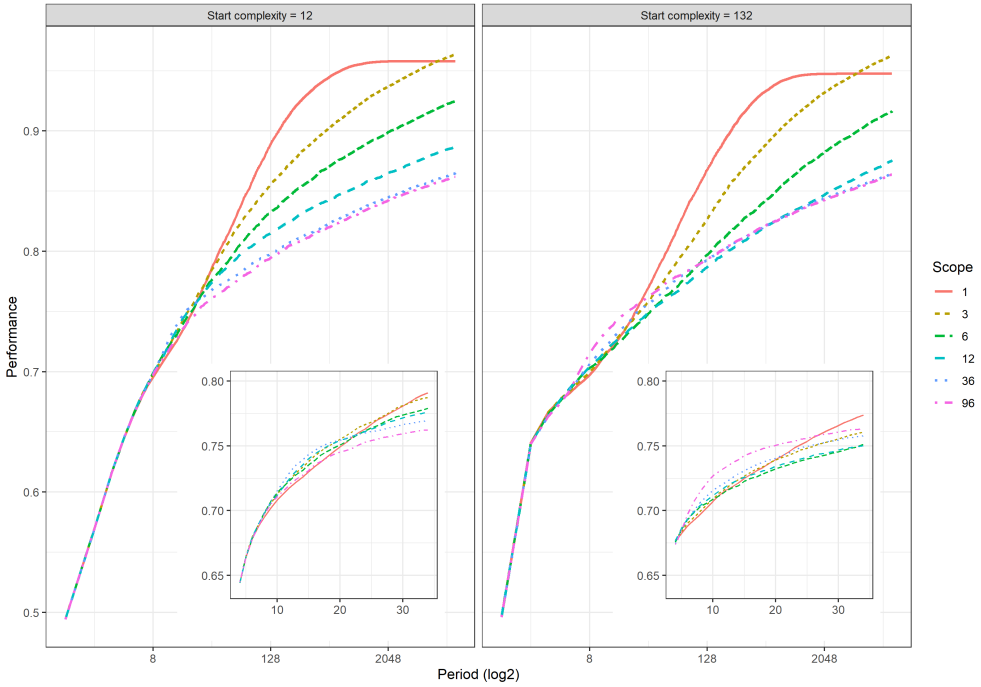
Note. Bars represent the 99.9% confidence interval.

Figure 4. Decomposition of accepted architectural changes into pure architectural and composite architectural innovations



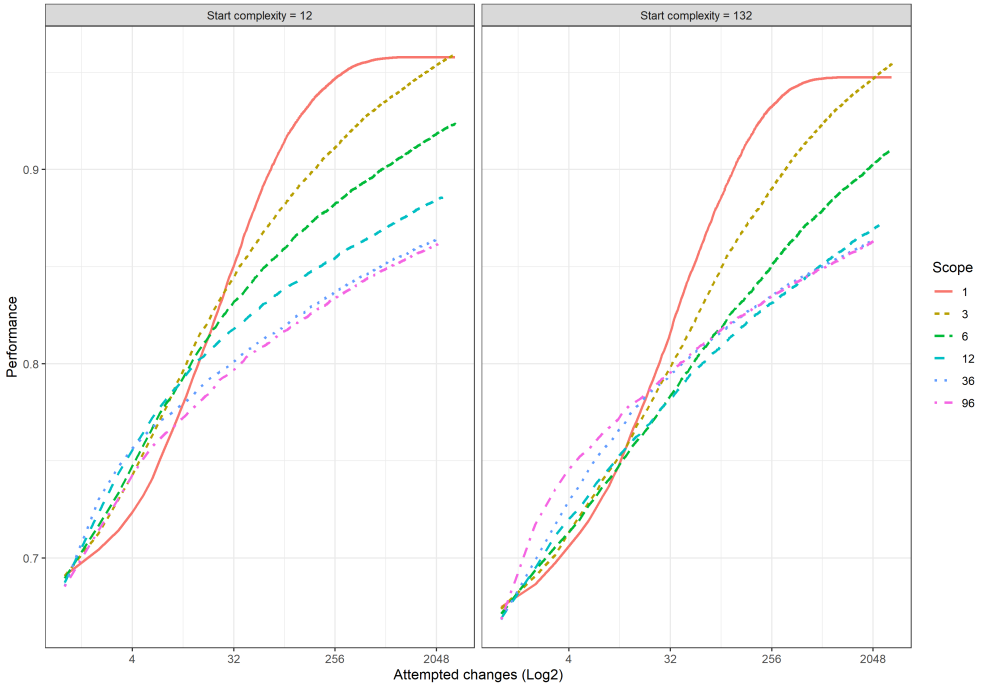
Note: Each bar represents 100% of all accepted architectural changes for a given scope and a given start complexity. The color filling in a bar represents the fraction of changes corresponding to composite architectural innovations (changes that lead to a new architecture and a change in component combinations) and pure architectural innovation (changes that lead to a new architecture and no change in component combinations).

Figure 5: Performance of architectural search over 10,000 time periods.



Note. The y-axis refers to the performance of the currently accepted product design in the t-th time period (shown on the x-axis). To conserve space, the x-axis is on a logarithmic scale. In order to improve readability of short-run performance results, in the insets, we show results for time periods 3-35, using a linear scale for time.

Figure 6. Performance of architectural search as a function of attempted architectural changes.



Note: The y-axis refers to the performance of the currently accepted product design at the time of the n-th attempted architectural change (x-axis). For readability purposes, the number of attempted changes are shown on a logarithmic scale.