A simulation-based method to evaluate the impact of product architecture on product evolvability

Res Eng Design (2015) 26:355–371
DOI 10.1007/s00163-015-0202-3

ORIGINAL PAPER

A simulation-based method to evaluate the impact of product architecture on product evolvability

Jianxi Luo[1]

Received: 10 June 2014 / Revised: 28 August 2015 / Accepted: 31 August 2015 / Published online: 8 September 2015

Abstract Products evolve over time via the continual redesigns of interdependent components. Product architecture, which is embodied in the structure of interactions among components, influences the ability for the product to be subsequently evolved. Despite extensive studies of change propagation via inter-component connections, little is known about the specific influences of product architecture on product evolvability. Related metrics and methods to assess the evolvability of products with given architectures are also under-developed. This paper proposes a simulation-based method to assess the isolated effect of product architecture on product evolvability by analyzing a design structure matrix. We define product evolvability as the ability of the product's design to subsequently generate heritable performance-improving variations, and propose a quantitative measure for it. We demonstrate the proposed method by using it to investigate a wide spectrum of model-generated DSMs representing products with varied architectures, and show that *modularity* and *inter-component influence cycles* promote product evolvability. Our primary contribution is a repeatable method to assess and compare alternative product architectures for architecture selection or redesign for evolvability. A second contribution is the simulation-based evidence about the impacts of two particular product architectural patterns on product evolvability. Both contributions aim to aid in designing for evolvability.

✉ Jianxi Luo
luo@sutd.edu.sg

[1] Engineering Product Development Pillar and SUTD-MIT International Design Centre, Singapore University of Technology and Design, Singapore, Singapore

Keywords Product architecture · Product evolvability · Design structure matrix · Simulation

1 Introduction

Products or engineering systems continually evolve over time via heritable design changes, which can be either initiated or emergent (Otto and Wood 1998; Eckert et al. 2004; Fricke and Schulz 2005; Rajan et al. 2005). Product evolvability reduces long-term difficulties for design advancements over the product's life span. It is particularly desirable in industries where innovation dynamism is high and product redesign is a norm (Suh 1990; Silver and de Weck 2007; Beesemyer et al. 2011), and for start-up companies for which markets are uncertain and frequent redesigns are foreseeable. Different product designs may inherit different degrees of evolvability. Knowledge about the determinants of product evolvability and the methods to assess it are needed to guide *designing for evolvability*.

The engineering design literature has suggested that product architecture, i.e., the pattern of interdependences following which components influence each other, can affect the ability of a product's design to be evolved in the future (Ulrich 1995; Whitney et al. 2004; Tilstra et al. 2012). In particular, many studies on product architecture and design changes have been based on the analysis of design structure matrix (DSM) in different forms (Clarkson et al. 2004; Eppinger and Browning 2012; Tilstra et al. 2012). However, quantitative methods to assess alternative product architectures in terms of product evolvability are still lacking. As a result, our knowledge of the impact of product architecture on product evolvability is limited.

In this paper, we focus on the evolvability of a product and investigate how it can be conditioned by product

architecture. The concept of evolvability arose in biology. In a Darwinian evolutionary process, variations that have fitness improvements are most likely to be selected and heritable (Wagner and Altenberg 1996; Kirschner and Gerhart 1998). Following an analogy between how the interactions of genes condition their variations in biological evolution and how the interdependences among components condition their design changes in product evolution, we define "product evolvability"[1] as the ability of a given product design to subsequently generate heritable performance-improving variations in design configuration, i.e., alternative combinations of design choices of all components.

Our aimed contribution is a simulation-based method that analyzes a design structure matrix to assess the isolated effects of product architecture on product evolvability. The core of the method was drawn from the "*NK* model," which was originally created to study organism evolution via genome mutations (Kauffman and Weinberger 1989; Kauffman 1993) and later adopted into the field of organization sciences (Levinthal 1997). We also propose a metric of *product evolvability*, based on the *NK* framework. The model and metric are based on simulating and analyzing the overall landscapes of performances (or fitness) mapped from the total design choice space of a given product. This method is then applied to assessing the evolvability of a wide spectrum of model-generated networks (or DSMs) that represent products with gradually varied architectures.

The simulation exercises lead to our second contribution—evidence about the impacts of two particular product architectural patterns, including component influence cycles and interaction density, on product evolvability. "Cycle" is the set of components which have an influence or dependency path to every other[2] (MacCormack et al. 2006; Sosa et al. 2013). Prior research has shown that component cycles require iterative problem solving and give rise to product defects (Smith and Eppinger 1997a, b; Mihm et al. 2003; Sosa et al. 2013). Instead, herein we are interested in the impact of component dependence cycles on product evolvability. Component interaction density denotes the number of dependences among a given set of components and has been used as a proxy of product modularity in the literature (Martin and Ishii 2002; MacCormack et al. 2006; Sosa et al. 2007, 2013). If a component is influenced by or dependent on fewer other components (implying a lower interaction density), it is more modular.

The remainder of the paper is organized as follows. In Sect. 2, we review the relevant literature. Section 3 introduces the simulation-based method to assess a product's evolvability by DSM analysis. Section 4 applies the method to assessing a wide spectrum of simulated DSMs that represent products with gradually varied architectures, followed by a discussion of theoretical and practical implications in Sect. 5. Section 6 concludes with the contributions and limitations of the present paper, and future research directions.

2 Literature review

This article aims to contribute to the extensive quantitative studies on design change propagation based on design structure matrix (DSM) analysis and the relatively small and scattered literature on engineering system design for evolvability.

2.1 Change propagation through component interactions

A major stream of change propagation analysis has been based on the view that the design change of one component can propagate through the interdependence relationships between components, requiring redesigns of many other components until all components can work together to perform the intended function (Clarkson et al. 2004; Jarratt et al. 2011; Hamraz et al. 2013). Many change propagation studies use the component-based design structure matrix (DSM) to model the linkages between components in a complex product (Eppinger and Browning 2012).

In an early paper, Cohen et al. (2000) used matrices to represent the inter-influences between design decisions related to the key attributes of a product design to predict change propagation when an attribute is changed. To predict the amount of redesign effort for future changes, Martin and Ishii (2002) assessed the direct dependencies between components using a component-based DSM that

[1] Biological and technological evolution processes are not exactly the same. One major difference is that technologies are indeed consciously "designed" by "intelligent designers," whereas biology evolution relies on natural selection. The technology evolution process may experience more occasions of non-sequential inheritance and leaps then biological evolution because of the role and decisions of designers on technologies, even though both processes are incremental and generally slow. Readers interested in contrasting the biological and technology evolution processes may refer to Kelly (2010) and Beesemyer et al. (2011). In the present paper, we do not study processes, dynamics and influences from designer's choices, but the evolvability of a product at a time, given by its architecture at the time. Our analogy focuses on (1) variation of elements (genes vs. components), (2) how inter-element interactions constrain variations (to make use of the NK model) and (3) preferential selection of fitness-improving variations (to define our evolvability metric). Section 2 provides more detailed review of related concepts of biological and product evolutions.

[2] For example, if the design choice of component A influences the working of B, which influences C, which influences A, components A, B and C form a cycle.

captures degrees of coupling between components. Suh et al. (2007) proposed a change propagation index (CPI), calculated as the difference between the total numbers of changes propagated from and received by a component. Smaling and de Weck (2007) developed a component-based Change DSM, i.e., ΔDSM, to quantify the amount of design changes required to accommodate a new technology or invasiveness of new technology. These studies do not assess change propagation via indirect dependencies of components.

The Change Prediction Method (CPM) developed by Clarkson et al. (2004) was likely the first to evaluate indirect change propagation via the influence paths between components. CPM also considers the *likelihood* and *impact* of change propagation from one component to another, using a probabilistic path-finding algorithm to analyze DSMs whose entries capture both likelihood and impact of change propagation between pairs of components. Since Clarkson et al. (2004), there has been a surge of publications on DSM-based assessment of change propagation, many of which can be considered as derivatives of CPM.

For example, Rutka et al. (2006) considered the indirect paths of change propagation, with specified types and degrees of change for the linkages in a DSM. Koh et al. (2012) integrated the house of quality and CPM to model the effects of potential change propagation brought about by different change options. Hamraz et al. (2012) and Ahmad et al. (2013) applied procedures similar to CPM to different types of multi-domain or domain mapping matrices (Danilovic and Browning 2007) that capture the interdependences within and between multiple domains, such as components, functions, requirements, processes and organizations. The step-based CPM of Koh et al. (2013) considers the reachability of change propagation, i.e., the ability of a change-initiating component to propagate changes to a sink component, to limit the maximum length of change propagation paths to be examined.

Recently, network-based techniques and metrics have been increasingly adopted to investigate change propagation. MacCormack et al. (2006), Sosa et al. (2007) and Cheng and Chu (2012) used graph-theoretic metrics, such as degree and betweenness centrality, reachability and clustering coefficients, as indicators of a component's propensity to propagate changes to other components through direct linkages and indirect paths. Giffin et al. (2009) analyzed various motifs and graph-theoretic indices of the network of change requests connected by parent–child or sibling relationships. Pasqual and de Weck (2012) proposed a repository of network-based techniques and graph-theoretic metrics to assess change propagation within and between the coupled product, change and organizational domains.

In general, these studies have focused on measuring the changeability of components by considering change propagation via direct or indirect influences between them. Part of the measurement efforts is the development of indices, such as the Change Propagation Index (CPI) (Suh et al. 2007) and the Incoming Change Likelihood (ICL), Incoming Change Impact (ICI) and Outgoing Change Risk (OCR) indices (Koh et al. 2013), which are used to differentiate components that exhibit different propagation behavior, such as multipliers, absorbers, carriers and constants (Eckert et al. 2004). In turn, knowledge about the differentiated change-related properties of components is useful to support design decisions, such as which components to standardize, modularize or embed flexibility to address design changes (Martin and Ishii 2002; Eckert et al. 2004; Suh et al. 2007; Cardin et al. 2013; Koh et al. 2013), as well as what change modes are most suitable (Rajan et al. 2005; Keese et al. 2009).

For comprehensive reviews of change propagation research, please refer to Jarratt et al. (2011) and Hamraz et al. (2013). In brief, changeability of components has been the main level of analysis in prior studies, despite the consideration of inter-component interactions. However, the changeability of a system is not a simple sum of the changeability of components, because their interdependence relationships are often complex, intricate and nonlinear. Despite DSM data on component interdependences being used to assess the change-related properties of components, the overall architectural pattern (i.e., topology) of such interdependences or influences, which embody product architecture, has not been explicitly investigated as key variables affecting product changeability. Knowledge on how the architecture of a product affects its overall changeability is lacking. Relevant methods for assessment need to be developed. In addition, inquiry into changeability at the system level is relevant to the relatively less developed, but growing literature on engineering system evolvability that we will review in Sect. 2.2.

2.2 Product evolvability and design for evolvability

As stated in the introduction section, the focus of the present paper is on evolvability, which is a specific type of changeability. The concept of evolvability arose originally in biology. One formal definition of biological evolvability is "an organism's capacity to generate heritable phenotypic variations" (Kirschner and Gerhart 1998). Another definition is the "ability of a population to both generate and use genetic variation to respond to natural selection" (Colagrave and Collins 2008). These definitions and many similar others (Wagner and Altenberg 1996; Hansen 2003) emphasize variations with some level of heritability with

prior configurations and the selection of competing new variations to future generations according to their levels of *fitness* with the environment (Ziman 2000).

Evolvability of products or engineering systems can be defined analogically. For instance, Butterfield et al. (2008) considered evolvability as the "ability of the architecture to handle future upgrades." Beesemyer et al. (2011) formally defined evolvability of a technological system as its "ability to change an inherited design across generations over time." Fulcoly (2012) defined it as "the ability of an architecture to be inherited and changed across generations [over time]." Following them, herein, we generally consider evolvability as the ability of a design to generate heritable variations with improved fitness (i.e., performance, value).

The concept of "product evolvability" differentiate itself from the definitions of general design changeability (Fricke and Schulz 2005; Ross et al. 2008) and design flexibility (Rajan et al. 2005; de Neufville and Scholtes 2011; Tilstra et al. 2013) by emphasizing heritance, path dependences and fitness-improving selection from an evolutionary perspective. Traditional engineering design change literature concerns changes in general, not differentiating positive (i.e., performance-improving) and negative (i.e., performance-reducing) changes. Existing design flexibility literature has taken into account the effects from certain specific changes, but the assessment of the overall potential of a present product design to generate performance-improving changes is lacking. The concept of product evolvability is closely related to the "adaptability" that Engel and Reich (2013) defined as the ability of a system to be changed to fit varied circumstances, whereas Fulcoly (2012) specifically considered adaptability as "the ability of a system to be changed by a system-internal change agent with intent." Fulcoly's interpretation of "adaptability" addresses changes introduced by system-internal agent, while products evolve via the changes and redesigns introduced by designers, who are not internal in the product or system.

Ideally, the evolvability of a product design allows design engineers to continually and easily discover performance-improving (or value-creation in general) variants of the present products. In some cases, a design with weaker traditional functional performance but better evolvability than alternative designs may achieve a higher longer-term value (de Neufville et al. 2004; Engel and Reich 2013). Therefore, in addition to assessing and designing for traditional functional performances in a design iteration, engineers should also assess and design for evolvability because it affects the life cycle value of their products (Silver and de Weck 2007).

Assessing product evolvability requires proper metrics to inform the potential of a design to generate fitness-improving variations. Beesemyer et al. (2011) and Fulcoly (2012) analyzed the potential of a few existing metrics, which were originally developed to measure complexity, modularity, changeability, flexibility, etc., to be adopted as evolvability metrics. However, these metrics address general changeability, instead of evolvability which emphasizes heritance, path dependences and fitness-improving selection. Some existing metrics require well-defined comprehensive model of the system, such as the state functions of all system parameters. It is unlikely because engineers face many uncertainties in typical design processes. Taken together, in the literature there exists no metric that directly addresses the definition of product evolvability in the present paper and similar others (Beesemyer et al. 2011; Fulcoly 2012).

Designing for evolvability also requires related design principles to inform the design decisions and processes (Beesemyer et al. 2011; Ricci et al. 2014). Indeed, many design principles for changeability and flexibility are naturally relevant. The related literature provides a rich set of candidate design principles for evolvability, such as (just to name a few) *modularity* and *autonomy* (Ulrich 1995; Baldwin and Clark 2000; Dahmus et al. 2001; Hölttä-Otto et al. 2012), *scalability* (Fricke and Schulz. 2005; Tilstra et al. 2013), *redundancy* (Fricke and Schulz 2005; de Neufville and Scholtes 2011), *reconfigurability* and *extensibility* (Siddiqi and de Weck 2008; Singh et al. 2009; Saleh et al. 2009; de Neufville and Scholtes 2011), as well as *product platform* (Simpson et al. 2001, 2013). On that basis, Tilstra et al. (2013) proposed a categorization of actionable guidelines for implementing these design principles for product flexibility. Beesemyer et al. (2011), Fulcoly (2012) and Ricci et al. (2014) provide brief reviews of engineering system design principles for evolvability.

One general strategy for implementing these design principles is to apply them to certain components, and various techniques and indices for identifying suitable components for embedding the above-mentioned "*ilities*" have been proposed (Suh et al. 2007; Cardin et al. 2013; Koh et al. 2013). A complementary strategy is to design or choose the overall product architecture (Henderson and Clark 1990; Ulrich 1995; Fixson and Park 2008; Tilstra et al. 2012) that best facilitates later heritable variations. Product architecture is the arrangement of components interacting to perform specified functions (Ulrich and Eppinger 2011; Eppinger and Browning 2012). It is commonly accepted that the architecture of a product or system influences its later evolutionary paths and dynamics (Simon 1962; Ulrich 1995; Whitney et al. 2004), and a "modular architecture" is often favored for facilitating product variation and evolution (Ulrich 1995; Baldwin and Clark 2000; Dahmus et al. 2001).

Some techniques have been developed to assist the assessment of product architectures for design flexibility. For instance, Tilstra et al. (2012) proposed the high-definition design structure matrix (HDDSM) and illustrated its utilities in assessing and comparing product architectures for flexibility to facilitate later redesigns. Rajan et al. (2005; Keese et al. 2009) proposed the Change Mode and Effect Analysis (CMEA) methodology and showed how it can be practically implemented to examine detailed change modes (including the ones related to product architecture) for quantifying and comparing the flexibility of different products. Engel and Reich (2013) considered components' option values and interface costs to assess the adaptability of different architectural assignments of components into modules.

Taken together, the literatures on changeability and evolvability are summarized and compared in Table 1. The literature review reveals two opportunities for the research on product evolvability. First, despite vast research on changeability, flexibility and other related concepts, the methods and metrics that directly assess product evolvability are still lacking and need to be developed. Second, although we conceptually accept that product architecture influences product evolvability, quantitative methods to assess and guidelines to design product architectures for the interest of product evolvability are underdeveloped. Our knowledge on which specific product architectural patterns and how they affect product evolvability is still ambiguous.

The present paper addresses these gaps. Specifically, we will introduce a simulation-based method and metric to assess the isolated influence of product architecture on its evolvability in the next section. Our analysis of product architecture is based on DSM and primarily concerns the topology of dependences between components. Nonetheless, the functional dimension of product architecture is embodied in the physical components and their pattern of interactions. The pattern of component interactions is assessed as a design or decision variable, and two particular characteristic architectural patterns will be evaluated in a simulation exercise. The new understanding about the impacts of these architectural patterns on product evolvability can be used as product architecture design guidelines for evolvability.

3 Simulation-based assessment of product evolvability

Our evolvability assessment method is based on analyzing the shape characteristics of the simulated fitness landscapes mapped from the total design space of a product with a given architecture. In engineering practices, it is normally impossible to obtain performance data from experiments or calculations for all possible combinations of design choices of individual components in the total design space and forming the fitness landscape. Simulations can aid in exploring the total design space and the fitness landscapes mapped to it, and systemically assessing design potentials inherited in the architecture of a present design or prototype.

The architecture of a product is assessed based on a basic component interaction DSM. In such a design structure matrix, the cell (i, j) at row i and column j is 1 if the design choice of component j can affect the functional performance or value of component i, indicating the requirement for co-redesign or change propagation; the cell is 0 when there is no influence or change propagation from component j to i. The DSM is asymmetric by default because component i does not necessarily affect j, when j can influence i. Such an asymmetric DSM can be alternatively represented as a directed network (Keller et al. 2006), in which an arrowed link is created from node j to node i when the design choice of component j influences component i.

Our method of analyzing the DSM or network of components draws on the *NK* model originally created to study organism evolution (Kauffman and Weinberger 1989; Kauffman 1993) and later adopted into the field of management sciences (Levinthal 1997). In a potential *NK* model framework for products, a product has

Table 1 Comparison of changeability and evolvability studies

	Changeability	Evolvability
Definition	The ability to be changed	The ability to be changed with performance improvements
Type of change	Changes in general, with many studies addressing performance-destroying changes and associated risks	Implicit focus on performance-improving changes that designers would prefer and select
Level of analysis	Components	Product or system as a holistic whole
Variables of interest	Individual components' interactions	Overall product or system architecture
Theoretical foundation	Engineering and physics	Evolution theory
Metric	Various metrics	*No direct metric of product evolvability*
Design guideline	Modular design, flexible design, etc.	*No direct guideline for product evolvability*

N components, each of which has ω_i alternative design choices and K_i other components that it can influence. Components in actual products may have multiple design choices (i.e., alternative mechanisms such as chemical battery vs. ultra-capacitor for an electrical vehicle), and design choices can be continuous, such as length, weight and temperature. Without loss of generality, in our later analysis, we consider the simplest case that each component has two alternative design choices, 0 and 1 (indicating $\omega_i = 2$ for all i), for computational ease.[3] The design choice of a component is mutated between 0 and 1 whenever there is an emergent or initiated change that may significantly affect the functional performance of the component.

Thus, the configuration of a product of N components can be described by an N-digit string of 1 s or 0 s, denoted as $s_i = d_1 d_2 \ldots d_N$, with $d_i = 0$ or 1, for $i = 1, 2, 3 \ldots 2^N$. The combinatory design space of N components has a total of 2^N unique design configurations. The total design space includes all possible variations through the mutations of design choices of all components, following the same inherited architecture of interactions between components. That is, product architecture is preserved or inherited when design choices of components are mutated to generate variations in total design configuration, throughout the total design space.[4]

Within each design configuration denoted by the N-digit string, the fitness of a specific component is randomly drawn from a uniform distribution [0, 1] each time its design is changed or the design of any other component that can influence it changes. The fitness of the overall design configuration is evaluated as the mean of the fitness values of all components. These model settings are the standard ones of the general *NK* models in biology (Kauffman 1993) and organization sciences (Levinthal 1997; Rivkin and Siggelkow 2007).[5] Clearly in practices alternative fitness functions can be used if the engineer has accurate information about the functions. But in most cases engineers do not have such knowledge given many uncertain factors to affect fitness. For our purpose of comparing alternative product architectures in general, a uniform distribution [0, 1] of fitness values, as used in the standard *NK* model, delivers the simplest, tractable and general results. The fitness values of all design configurations in the design space constitute the fitness landscape. The "smoothness" or "ruggedness" of the fitness landscapes has been a central interest in prior *NK* analyses (Kauffman 1993; Levinthal 1997). A landscape with many "hills" and "valleys" is rugged.

Because the fitness values of components are randomly drawn under the only constraint imposed by their inter-influence relationships, *NK* simulations create a mapping from the structure of inter-component influences (indicating product architecture) to a fitness landscape, and more specifically, the shape characteristics of the fitness landscape, such as smoothness versus ruggedness. In other words, statistically, the architectural patterns of the inter-component influences are the only determinants of the fitness landscape, in the *NK* framework. For instance, Rivkin and Siggelkow (2007), using *NK* model-based simulation, showed that a shift in a few commonly observed interdependence patterns (e.g., centralization, small-world, scale-free, hierarchy) in general system architectures can significantly alter the ruggedness of their fitness landscapes.

Focusing on the impacts of network structures, the *NK*-based simulations only require the input of the most basic form of DSM data in which the inter-component influences are denoted dichotomously, i.e., 0 or 1, because it is sufficient to capture the topology of the inter-component influence structure. This topology is the single factor that our method aims to analyze with regard to its impact on product evolvability. In the meantime, the fitness or performance impacts of all other variables related to

[3] The number of design choices for individual components affects the size of the design space, but does not affect the qualitative results on the isolated influences of different product architectures on evolvability. The pioneers and leading scholars of *NK* model had written about the consequences of using $\omega_i = 2$ for all i. Kauffman and Weinberger (1989), who first published the *NK* model, wrote that, "although it is difficult to draw a picture of such high dimensional spaces, a sense of their structure can be captured by considering proteins with only two amino acids, e.g., alanine and glycine." Levinthal (1997), who introduced *NK* model to the field of organization sciences, wrote that, "the model can be extended to an arbitrary finite number of possible values of an attribute, but the qualitative properties of the model are robust to such a generalization." In this paper, the focus is to assess the isolated impact of product architecture rather than the size of design space, setting $\omega_i = 2$ provides the simplest and most tractable model for this purpose.

[4] That means the component design choices in consideration are those that do not alter the pattern of interactions among the components. In real-world design practices, potential design choices of a component may require new interactions or eliminate existing interactions with other components. Such design choices are not included in the design space resulting from a given architecture that we focus on to assess. That is, the design space given by architecture only constitutes of those design choices of individual components complying with the architecture.

[5] Our method follows the *NK* model specifically to use the random fitness function to simulate the fitness landscape. In theory, the fitness function can have other forms. If the engineer has a deterministic fitness function, he can obtain a fixed landscape given specific product architecture. The fixed landscape, rather than a sample of random landscapes, will be assessed using the evolvability metric in Eq. (2). In addition, if the engineer has total knowledge of the fixed fitness landscape, he/she can choose the global optimal design directly. However, this is normally not the case of real engineering practices. Often engineers are unable to have a deterministic fitness function. In such most cases, random fitness functions can be used to assess the influence of product architecture on evolvability.

component linkages, such as types (e.g., energy, information, material and spatial) (Pimmler and Eppinger 1994; Koh et al. 2012), physical laws of change propagation (Ollinger and Stahovich 2004), likelihood of impact (Clarkson et al. 2004) and sensitivity of impact (Yassine and Falkenburg 1999), are neutralized when we *randomly* draw fitness or performance values for component design choices. Thus, the impacts of additional factors in reality are neutralized in the statistical analysis of the association between product architecture and the simulated fitness landscapes, based on a large sample of simulation data.

Here, we use an example product of three components and three "influence" connections (Fig. 1) to demonstrate how to simulate a fitness landscape given its dichotomous DSM or network.

Following the network structure in Fig. 1, the sub-string that matters for each component is determined as follows:

- Component 1 is affected by itself, 2 and 3; the fitness of component 1 is randomly drawn from [0, 1] when the design of itself or the design of component 2 or 3 is changed, i.e., $p_1 = p_1\ (d_1d_2d_3)$;
- Component 2 is affected by itself and 1; the fitness of component 2 is randomly drawn from [0, 1] when the design of itself or the design of component 1 is changed, i.e., $p_2 = p_2\ (d_1d_2)$.

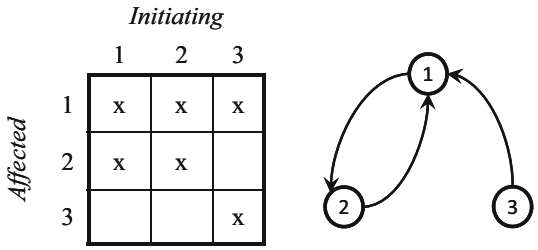

Fig. 1 DSM and network representations of an example product of three components and three influence interactions

- Component 3 is only affected by itself, despite the fact that it can affect component 1; the fitness of component 3 is randomly drawn from [0, 1] when the design of itself changes, i.e., $p_3 = p_3\ (d_3)$.

For each of the three components, we create a list of all possible configurations of the sub-string that affects its fitness and give a random fitness value in [0, 1] to each sub-string configuration. This procedure results in Table 2.

Then, Table 2 is used to search for the fitness value of each component, by matching its sub-string configuration with the corresponding digits in the 3-digit total product design configuration. For instance, when the entire product's design configuration is "101," the fitness of component 1 is determined by sub-string $d_1d_2d_3$ = "101" and Table 2-A gives $p_1 = 0.20$. Likewise, the fitness of component 2 is determined by sub-string d_1d_2 = "10," so Table 2-B gives $p_2 = 0.80$. Sub-string d_3 = "1" determines $p_3 = 0.60$ in Table 2-C. This procedure leads to results in Table 3, which lists the fitness of each component determined by the sub-string that is relevant to it under each total design configuration.

The rightmost column of Table 3 lists the fitness level of each total design configuration, calculated as the mean of all component fitness values, and constitutes a simulated "fitness landscape" for the given network or DSM which embeds a product architecture or topology. In the landscape, if the fitness of a configuration (e.g., 001) is better than any of its 1-mutant neighbors (e.g., 101, 011, 000), the configuration is a local peak. If a local peak's fitness is the highest among all configurations in the design space, it is also a global peak. For the landscape given in Table 3, alternatively represented in a 3D diagram in Fig. 2, design configuration 001 is a local peak, and 111 is a global peak.

The number of peaks indicates the "ruggedness" (i.e., opposite of "smoothness") of a fitness landscape (Kauffman 1993; Levinthal 1997). A better measure of ruggedness might be the density of peaks on a fitness landscape, i.e., peak density, which can be calculated as

Table 2 Sub-strings and randomly drawn fitness values for each component

(A) Component 1		(B) Component 2		(C) Component 3	
$d_1d_2d_3$	p_1	d_1d_2	p_2	d_3	p_3
000	0.50	00	0.20	0	0.25
001	0.90	01	0.30	1	0.60
010	0.80	10	0.80		
011	0.30	11	0.50		
100	0.10				
101	0.20				
110	0.05				
111	0.70				

Table 3 Fitness values corresponding to product configurations in the landscape

S_j	$d_1d_2d_3$	p_1	p_2	p_3	$P(S_j) = \frac{1}{N}\sum_{n=1}^{N} p_n(s_j)$
$j = 1$	000	0.50	0.20	0.25	0.32
$j = 2$	001	0.90	0.20	0.60	0.57 (local peak)
$j = 3$	010	0.80	0.30	0.25	0.45
$j = 4$	011	0.30	0.30	0.60	0.40
$j = 5$	100	0.10	0.80	0.25	0.38
$j = 6$	101	0.20	0.80	0.60	0.53
$j = 7$	110	0.05	0.50	0.25	0.27
$j = 8$	111	0.70	0.50	0.60	0.60 (global peak)

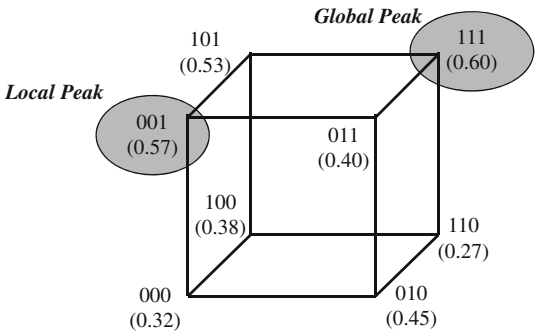

Fig. 2 A simulated fitness landscape for the example product

$$\text{Peak density} = \frac{\text{Number of peaks on landscape}}{\text{Size of landscape}} \tag{1}$$

For a general product of N components in which component i has ω_i design choices, the size of the landscape (equal to the size of the design space) is $\prod_{i=1}^{N}\omega_i$. In the most generic case when $\omega_i = 2$ for all i, the landscape size is 2^N. Earlier NK studies have commonly found that the interaction density of components gives rise to the ruggedness of the system's fitness landscapes (Kauffman 1993; Levinthal 1997; Rivkin and Siggelkow 2007). And in the extreme situation that all components are independent from each other, i.e., having no interaction, there is only one single peak on the fitness landscape.

For a highly rugged landscape where there are many local peaks, an average starting point (i.e., the initial design) will have a high chance to include a local peak in its neighborhood or be close to a local peak. Thus, "hill-climbing" adaptive redesigns via one-component-redesign-at-a-time for higher fitness will quickly reach and stabilize at a local peak closest to an average starting point (i.e., initial design configuration), get stuck at local optima and cease evolving. As a result, only modest performance is expected. An example of such design lock-in might be the automobile that has been stuck at a local optimum characterized by a dominant design of an internal combustion engine and accompanying subsystems and components, for more than a century, despite that there are potentially higher peaks elsewhere in the landscape, such as the ones characterized by hydrogen-powered or battery-powered powertrains.

In contrast, for a not-so-rugged or even single-peak landscape, the hill-climbing redesign process searching for performance improvements may sprawl over a wide portion of the fitness landscape for a long period of time, because it is more likely that the neighborhood of an average starting point includes a design configuration with higher fitness due to the lack of local peaks. This may lead to more sustainably foreseeable design improvement opportunities and a higher chance to reach a design configuration with sufficiently high fitness, than a process that quickly stabilizes at a local optimum.

In brief, the ruggedness of a fitness landscape given by specific product architecture implies a systematic constraint on its ability or potential to subsequently generate heritable design configuration variations with higher fitness. Therefore, peak density is a reverse indicator of product evolvability. Based on this understanding, we propose a metric for the evolvability of product architecture in the NK model framework as

$$\text{Product evolvability} = \frac{1}{\text{Peak density}} = \frac{2^N}{\text{Average number of peaks}} \tag{2}$$

For a generalized product in which each component i has ω_i design choices, the numerator is $\prod_{i=1}^{N}\omega_i$. The denominator is the average number of peaks of an ensemble of fitness landscapes simulated based on the same DSM, using the procedure described above. Each random landscape in the same ensemble differs in details from all the others, but all result from the same product architecture and must share common properties that are only determined by the architecture.

Note that, the product evolvability metric captures the potential of a product design, given its architecture, to have fitness-improvement design variations, instead of the extent to which the product design has evolved. In addition to the ruggedness of the landscape determined by only the

architecture, the scope of search of the designer, i.e., the number of components to be redesigned at a time, in the design space may also affect a product's actual evolutionary process and path. Intuitively, simultaneous redesigns of many more components, indicating more global or exploratory search, create a greater chance to discover a design configuration with dramatic fitness improvement. In contrast, a myopic local search process, in which only one or small number of components are redesigned at a time, exhausts the improvement opportunities and becomes stuck at a local peak quickly. In fact, the one-at-a-time redesign process is the most common in actual design practices, due to the natural limitations in experimentation resources and the designer's vision required for broader search, among many other factors.

Taken together, the structure of the landscape and the scope of components for simultaneous redesign together co-determine the eventual path of product evolution as a search process. In contrast to the scope of components for simultaneous redesign, which is a choice of the designer limited by his/her capability and resources and is extrinsic to the physical product, the shape characteristics of the landscape are determined by the product architecture, which is intrinsic to the product. Landscape ruggedness imposes a systemic constraint on the product's evolvability. Lower evolvability of a product's architecture will require broader exploration of the designer, i.e., redesigning more components at a time, and as a result requires more experimental resources, in order to improve the fitness of the product.

In the next section, we will apply the foregoing fitness landscape generation procedure and the evolvability metric to assessing the isolated intrinsic impacts of product architecture on evolvability of a wide spectrum of randomly generated networks with varied but controlled topologies representing different product architectures.

4 Simulation results: assessing evolvability of products with gradually varied architectures

4.1 Model-generated component interaction networks

We use a tunable network model to generate a wide spectrum of random directed networks with gradually varied network structures, which represent the component interaction networks of products with varied architectures (i.e., component interaction network topology). The mathematical details of the model and some properties of the model-generated networks are provided in "Appendix". One advantage of simulated component networks is that they can be controlled to have continually varied architectures and allow for the exploration of a wide spectrum of product architectures. In contrast, empirically we could only have small samples of and scattered data of actual products for inductive analysis.

In particular, these networks are randomly generated with controls for various (1) component interaction density and (2) amount of cyclic influences among components.

Interaction density (K) is the average number of components that each component influences. For a product with N components and M influence links, $K = M/N$. It equals the average nodal degree in network sciences (Newman 2003) and the average interaction density, denoted as K in the original NK model (Kauffman 1993). In the context of product design, it is an indicator of product integrality or reverse indicator of product modularity (MacCormack et al. 2006; Hölttä-Otto et al. 2012; Sosa et al. 2013). If there are few inter-influences between components (i.e., low K), the redesign of one component propagates few change to others. Such product architecture with low K is highly modular. A product's architecture is integral if there are many inter-influence links among components. As a result of high K, the redesign of one component will propagate changes to many other components.

To measure the amount of cyclic inter-component influences or dependences embedded in product architecture, we define a metric called "*cyclic degree (C)*" and calculate it as the percentage of directed influence links that are in at least one cycle,

$$C = \frac{\sum_{i=1}^{M} e_i}{M} \tag{3}$$

where M is the number of links in the network, and $e_i = 1$ if link i is in a cycle and 0 otherwise.

When cyclic degree $C = 0$, the network is purely acyclic (see examples A, B and C in Fig. 3). In such networks, components influence each other in a serial or sequential manner, implying that design changes can propagate in only one direction from upstream to downstream (Sosa et al. 2013). When $C = 1$, any influence link is in at least one cycle. Thus, the network is purely cyclic (example F in Fig. 3), and components in such a network can propagate changes to every other. Such a cyclic product architecture may call for more iterative problem solving (Smith and Eppinger 1997a, b; Mihm et al. 2003; Sosa et al. 2013). When $0 < C < 1$, the network is partially cyclic (see examples D and E in Fig. 3). Such a network has a mix of sequential and cyclic interdependences or influences.

Our network generation model incorporates *a tuning parameter* (see details in "Appendix") to control and adjust the amount of cycles to be generated in the networks. The network generation model begins by creating a random directed network with purely sequential (i.e., acyclic) dependence relationship among components and then randomly chooses some component links to rewire and form cycles to the extent determined by the tuning

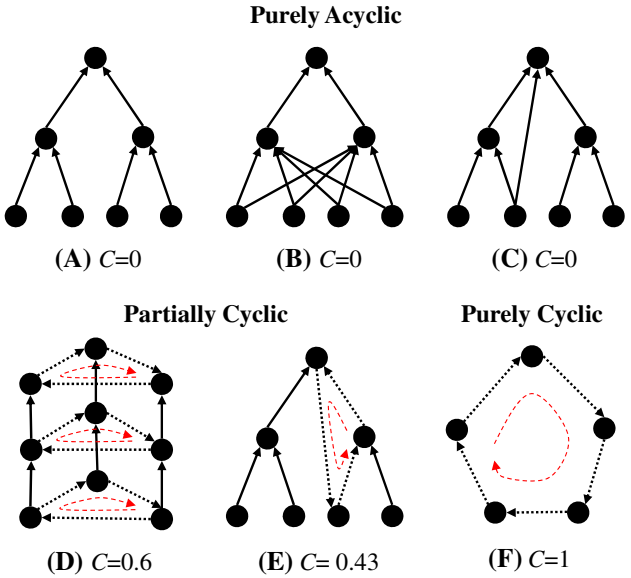

Fig. 3 Example networks and their cyclic degrees

parameter. "Appendix" includes mathematical details of the tuning parameter, the model and a sensitivity analysis of the model parameters, which shows that cyclic degree monotonically increases with the tuning parameter.

Figure 4 presents the DSMs of a series of prototypical networks simulated using the model ("Appendix") with varied cyclic degrees. On both horizontal and vertical axes, components are ordered according to "visibility," i.e., the count of all other directly and indirectly reachable components (MacCormack et al. 2006). In doing so, the ones in the same cycle will be placed together on the axes due to their equal visibility. The rectangular boxes drawn in the DSM encapsulate strong components (Newman 2003), in which all components and links are on cycles with one another. The simulated networks tend to develop a single strongly connected core component. In a purely acyclic DSM, all of the dots are above the main diagonal (see Fig. 4a). In (partially) acyclic DSMs, there are always some dots below the main diagonal regardless of the permutation of rows and columns (see Fig. 4b–d).

4.2 Simulation results

Now, we use the method introduced in Sect. 3 to assess the evolvability of a wide spectrum of model-generated component interaction networks with varied degrees of interaction density (K) and cyclic degree (C). In the simulation exercises, we fix $N = 12$ and tune K from 2 to 5 by step of 1. For each given combination of simulation controls or inputs, including K and the cycle tuning parameter, we simulate a sample of 2000 networks and calculate their average cyclic degree (C). For each network in the sample, we generate 200 fitness landscapes and calculate their average evolvability degree (E).

4.2.1 Impact of component cycles on product evolvability

The results first show that an increase in the extent to which inter-component influences are cyclic, i.e., cyclic degree (C), gives rise to product evolvability (E), when holding fixed interaction density (K) (Fig. 5). Vice versa, more acyclic product architectures tend to be less evolvable. That is, inter-component influence cycles promote product evolvability.

Two detailed patterns on the cycle–evolvability relationship are noteworthy. First, cyclic degree and evolvability are fairly linear correlated. The Pearson correlation coefficients between them range from 0.980 for $K = 5$ to 0.983 for $K = 2$. Second, when interaction density is lower, evolvability increases faster with the increase in cycles. The slope of the cycle–evolvability linear regression curve is 286.59 for $K = 2$ and much higher than the slope of 48.22 for $K = 5$. The expanding gaps between lines for the various K values shown in Fig. 5 indicate a constraining effect of interaction density on the promoting effect of cycles on product evolvability.

4.2.2 Impact of interaction density on evolvability

Figure 5 also shows that evolvability declines with the increase in the levels of interaction density (K). This relationship is consistent with prior NK analysis findings that local peaks proliferate when interaction density increases (Kauffman 1993; Levinthal 1997). This finding indicates that more modular product architecture, with lower component interaction density, gives rise to higher product evolvability. Our analysis here provides evidence for the prior argument in the literature that modular product architecture facilitates future product evolution (Ulrich 1995; Baldwin and Clark 2000). Conversely, more integral product architecture, implying that many components can influence each other, limits product evolvability.

These results from simulation exercises demonstrate the value of the proposed method to capture different influences of different product architectures, by showing that architectures with more component dependence cycles and higher modularity give rise to product evolvability. In the following section, we discuss the theoretical as well as practical implications.

5 Discussion

First, the negative impact of component interaction density on product evolvability is intuitive. It supports the assertion that modular product architecture promotes product

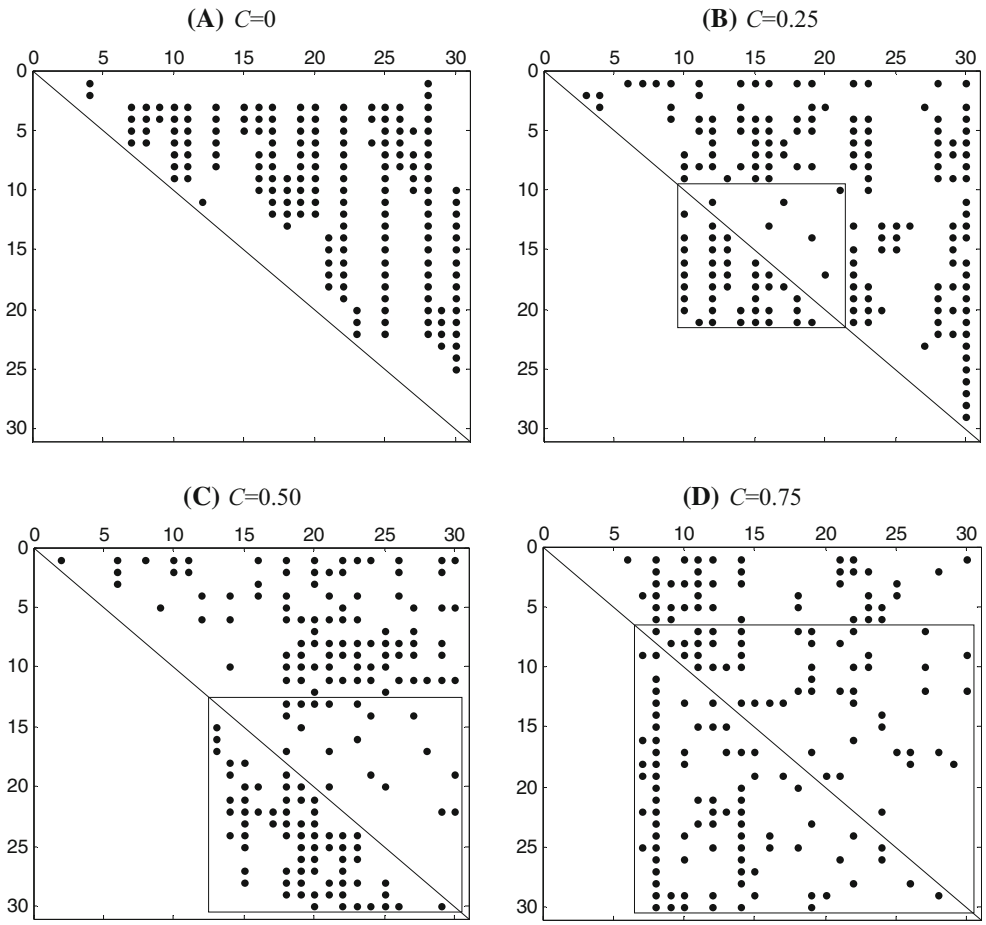

Fig. 4 DSMs of model-generated networks with varied cyclic degrees (fixed $N = 30$, $K = 6$)

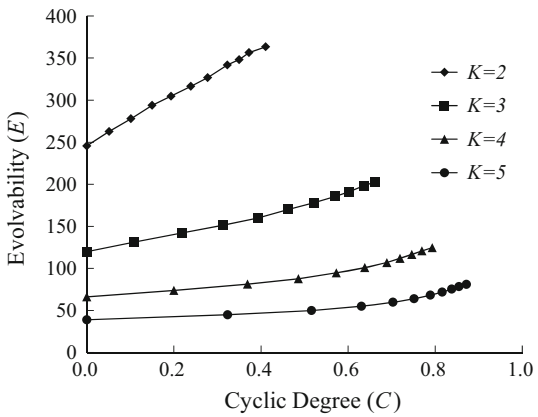

Fig. 5 Impact of inter-component influence cycles on product evolvability

evolution, whereas integral product architecture limits product evolution. When the design change of one component affects the performance of many other components or requires design changes of many other components, it will be more difficult for the overall system designers to find a heritable performance-enhancing variation in overall design configuration. One example is the slow evolution of automobiles given their highly integral architecture, which have a dominant design that has existed for more than a century, despite enormous R&D efforts aimed to improve automobile technologies.

Contemporary automobiles are highly integral complex systems (Whitney 1996; Clark and Fujimoto 1991). A single change of one component affects the functioning of and requires design changes of *many* other components due to the systemic requirements for energy efficiency, emission, safety, etc. The high inter-component influence density (indicated by high K defined in this paper) of automobile components and parts may provide a partial explanation to its limited evolvability. Thus, it is difficult to achieve higher "fitness" of the overall system without systemic and coordinated changes.

In contrast, contemporary electronics products are highly modular systems (Baldwin and Clark 2000), in which the design change of one component does not affect performances and does not require design changes of many other components. Without a high degree of

interdependences, individual component design changes that improve their own performances also improve overall system fitness and thus can be easily selected and inherited into future product generations. The low inter-component influence density, or modularity, of electronics products provides a partial explanation to their high evolvability.

Second, the impact of component influence cycles on product evolvability is relatively less intuitive than that of interaction density. Indeed, our finding about it is consistent with the assertion of Herbert Simon (1962) that hierarchy in a complex system facilitates the search process toward stability. Cycles imply reciprocal or cyclic interdependences that violate pure hierarchy, so reduce stability. An earlier empirical study of the call graph architecture of open source software also found that functional call cycles among code files in the Linux kernel diminished over years of evolution (Luo and Magee 2011). That indicates, as the product matures and evolves, product architectures (i.e., less cyclic ones) promoting stability rather than evolvability were increasingly favored and selected.

System designers and architects can potentially make use of the now understood impacts of (1) component interaction density and (2) influence cycles on product evolvability to adjust product architectures for desirable evolvability. They can be used as product architecture design guidelines for evolvability. First, concerning the impact of interaction density, system architects may purposefully pursue component interface standardization to reduce the inter-influences between components, improving modularity (Martin and Ishii 2002), in order to derive higher evolvability of their product designs. In addition, with understanding the impact of component cycles, system architectures may explore and maintain reciprocal or cyclic influence relationships between components to achieve higher product evolvability.

If product architectures are not to be changed, the designer's interest in product evolution may call for simultaneous and coordinated changes of a broader set of components. Changing a set of components at the same time, rather than one at a time, will allow searching beyond the direct neighbors and local peaks, exploring a wider area of the fitness landscape and sustaining the search process for a longer while, along with a higher chance to reach sufficiently high fitness configurations. For example, we might achieve a higher chance for the design of automobiles to evolve to a potentially higher optimum, perhaps the one characterized by hydrogen- or battery-powered electrical powertrains, if coordinated design changes related to energy source, transmission and infrastructure, etc., can be experimented simultaneously. In the other words, a product architecture that determines low product evolvability may require broader search for product evolution. However, it is often costly and difficult to coordinate the simultaneous redesign of many components or subsystems. Especially, the components of a product or system may be designed by various companies and organizations, implying the difficulty for coordination and co-design. This again implies the importance of appropriate product architecting for evolvability in early design phases.

Furthermore, evolvability is not always favorable for all engineers or companies. For instance, in such industries as the aerospace industry, where safety and reliability are of paramount importance, a high level of evolvability might be undesirable for system engineers. Instead, they might be more interested in tighter system integration, which results in a higher interaction density, and favor sequential interdependences among components by streamlining them through a hierarchical architecture, which, in turn, results in limited influence cycles among components.

6 Concluding remarks

Product evolvability is particularly important for start-up companies which face high uncertainty in market demands, and the companies in highly dynamic industries. This study makes both methodological and theoretical contributions to the studies of product evolvability, as well as changeability. Our proposed method and theories are useful for the assessment, comparison and selection of alternative or competing product architectures, in addition to the redesign of product architecture purposefully for either improving or reducing evolvability, depending on the interests of the engineers or system architects.

Methodologically, we have presented a repeatable method and a quantitative metric of product evolvability to assess the isolated impact of product architecture on the product's evolvability. The method and metric incorporate the emphases of the evolvability concept on inheriting the architecture and selecting fitness-improving variations, beyond the general changeability concept, and measure evolvability as the potential of a product design, given its architecture, to have fitness-improving variations, instead of the extent to which the product has evolved. The assessment method can be potentially embedded into CAD/CAE software and existing DSM software as an additional function of DSM analysis, to aid in designing for evolvability. Note that our simulation-based method is easy to implement because it only requires the most basic binary DSM data that capture the topological pattern of component interactions. With focusing on the impact of product architecture on evolvability, the impacts of other physical, social and probabilistic aspects of component interactions are all randomized in the simulations and thus neutralized when statistically associating product architecture with

evolvability. Such randomization reduces the barriers for the application of the proposed method in practices.

Theoretically, this paper also provides evidence on the specific impacts of two product architectural patterns on product evolvability. Specifically, product evolvability is promoted by component influence cycles, but is limited by component interaction density. Such understandings allow engineers and system architects to infer and predict the evolvability of their products using only the most basic DSM data. The new understanding also provides guidance on redesigning product architectures purposefully to adjust fitness landscape ruggedness for the desired level of product evolvability or guidance on product redesign strategies (e.g., one or many component changes at a time) for fixed product architectures. In the other words, such knowledge provides architecture design guidelines for evolvability. In brief, both of our methodology and theoretical contributions aid in designing for evolvability.

Note that, this paper examines the evolvability of a product at a given point of time, as determined by its component DSM. The present analysis is static in nature. The real product evolution process and dynamics are not investigated, and the evolution of evolvability itself and product architectures over time are also not investigated. As the next step, we plan to develop a dynamic model of product evolution processes to investigate alternative evolutionary trajectories and long-run performances of product designs conditioned by different product architectures and redesign strategies.

This paper has only analyzed two exemplary architectural lenses, i.e., cycle and density, to demonstrate the influences of product architecture on evolvability, whereas product architectures vary and be characterized in many more ways. Future research may investigate the influences of additional product architecture patterns on evolvability and develop more design guidelines for evolvability. We also hope this study can stimulate the development of more and better product evolvability metrics and calculation methods. By then, test cases will be needed to assess and compare alternative metrics and methods. In addition, future research should validate the proposed method in real engineering design practices and investigate the effectiveness of adding evolvability assessment of product architectures into the prototype evaluation and selection activities of engineers and system architects, in the prototyping process.

Acknowledgments I am grateful to Kristin Wood, Kevin Otto, Katja Otto, Richard de Neufville, Karen Wilcox, Christopher Magee, Daniel Whitney, Oliver de Weck, Jason Woodard, Carliss Baldwin and other colleagues at Singapore University of Technology & Design, Massachusetts Institute of Technology, and Harvard University. The enormous discussions with them have greatly inspired and shaped this research. This work was funded, in part, by the SUTD-MIT International Design Centre, http://idc.sutd.edu.sg.

Appendix

The network generation model is revised on the basis of a mathematical model that replicates ecological networks (Williams and Martinez 2000). The simulation model uses three input parameters, which may represent different architectural properties of a product.

(1) *Network size* (N): the total number of components in the product.
(2) *Interaction density* (K): the average number of components that each component influences. For a product with N components and M influence links, $K = M/N$. In the context of product design, it is an indicator of product integrality or reverse indicator of product modularity.
(3) *Influence diversity* (D): the scope of other components that an average component can influence in an ideal and hypothesized product hierarchy (Fig. 6a) or a "serial design chain," as Sosa et al. (2013) put it. Its reverse concept is "influence specificity," which indicates the degree to which a component's influences concentrate on a subset of components that are proximate to each other in the product hierarchy. As influence diversity increases, inter-influence relationships among components will gradually deviate from the pure serial or upstream–downstream manner.

Baseline Scenario ($D = 0$)

We begin by creating an ideal and hypothesized sequential (upstream–downstream) influence or dependence relationship between components in their interaction network, which will be rewired later to generate more general and cyclic networks. To do this, each of the N components is assigned to a uniformly distributed random position λ_i, along an axis ranging from 0 to 1 (Fig. 6a). Consider a focal component i with position value λ_i, the entire downstream interval for component i has a length $(1 - \lambda_i)$. The component's "influence niche" range r_i is the interval containing the components that it can influence, as defined by

$$r_i = X(1 - \lambda_i) \quad (4)$$

where X is a random variable between 0 and 1, and the probability distribution of X is component independent (to be set up later).

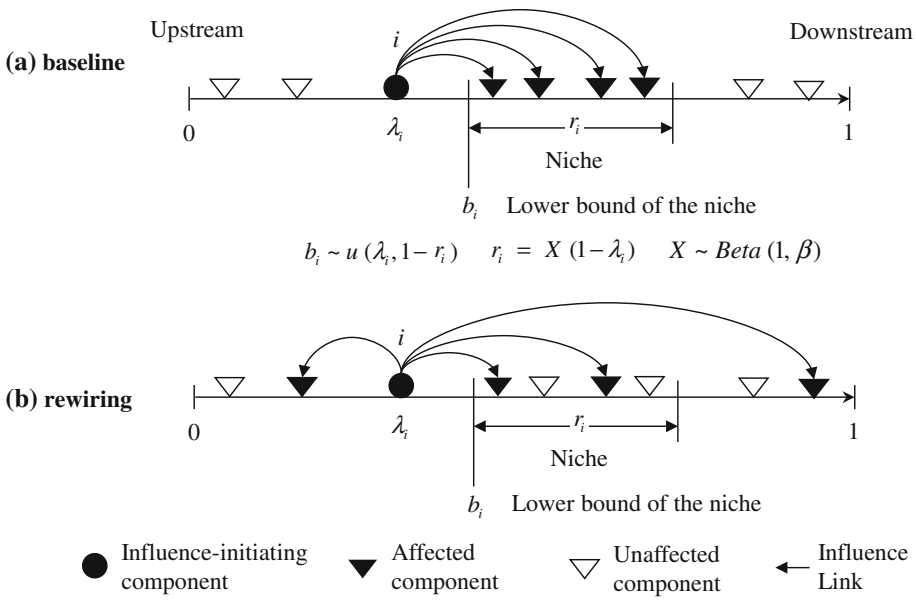

Fig. 6 Network rewiring model

The focal component's influence niche range can be located anywhere downstream. The position parameter b_i fixes the location of component i's niche range by defining its left most point. b_i is assumed to be uniformly distributed between λ_i and $(1 - r_i)$. Networks generated this way are strictly acyclic; there is no influence cycle among any components. Thus, they can be used as the basis for later rewiring to introduce cycles.

The niche range of a particular component r_i, as defined in Eq. (4), is a random variable whose statistical properties are affected by the number of components (N) and interaction density (K) of the product. Now we explain this association further. First, the density of components on the entire segment is N. Because the distribution of these components is uniform, the expected number of components in the niche for component i is as follows:

$$E(K_i) = N \cdot E(r_i) \tag{5}$$

For the entire system, excluding the rightmost component, the sum of the expected number of component that each component can influence is as follows:

$$E(M) = \sum_{i=1}^{N-1} E(K_i) = N \sum_{i=1}^{N-1} E(r_i) \tag{6}$$

In addition, the expected average number of components that each components influences is simply

$$E(K) = \frac{E(M)}{N} = \sum_{i=1}^{N-1} E(r_i) = \sum_{i=1}^{N-1} E(1-\lambda_i)E(X) = \frac{(N-1)}{2} E(X) \tag{7}$$

Thus, the random variable X is not only constrained to be between 0 and 1, but its expected value is

$$E(X) = \frac{2E(K)}{N-1} \tag{8}$$

$E(K)$ is given as the input variable K to the network construction model. Note that, although K_i is component-specific and randomly distributed, K is the average and an empirically measurable property of a given product's component network. To generate a network, we need to choose an appropriate functional form for the distribution of X and then impose the constraint of Eq. (8). For computational ease, a beta-distribution with parameters $(1, \beta)$ is used for the random variable X. This allows $E(X)$ to be in a computationally convenient form, $1/(1 + \beta)$. Given K and N as inputs, β will be determined by

$$\beta = \frac{N-1}{2K} - 1 \tag{9}$$

Then, a random niche range constrained by (4) can be given to each of the aforementioned array of components randomly organized between 0 and 1 on the axis. The focal component is then linked to each component in its influence niche.

Hybrid ($0 < D < 1$): Random rewiring

When "influence diversity (D)" is greater than zero ($D > 0$), a portion D of influence links of a component, which assumedly go into its hypothesized niche, become nonspecific and are wired to components anywhere in

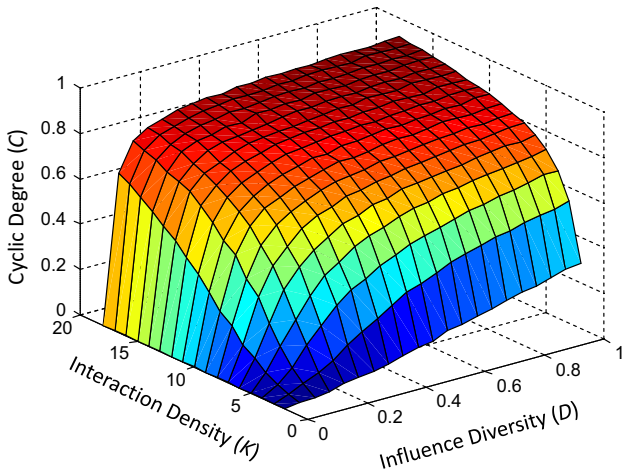

Fig. 7 Impact of influence diversity and interaction density on cycle degree

the product hierarchy (including the preassigned niche).[6] Figure 6b demonstrates a hybrid configuration after rewiring. Thus, influence diversity D is operationalized as the percentage of a component's influence links that can deviate away from its predefined niche of components as given in the baseline scenario, and is the control for the extent of rewiring. Now, cycles can emerge to the degree of rewiring determined by D.

Tuning parameters N, K and D, the model generates random networks with gradually varied cyclic degrees (C). We statistically assessed the basic regularity in the relationship between N, K, D and C via simulations[7] before using the simulated networks to investigate evolvability. First, the cyclic degree appears almost unaffected by changes in N when N is sufficiently large. Second, the cyclic degree is an increasing function of both K and D. The average cyclic degree of randomly generated network samples as a function of the inputs K and D (when $N = 100$) is plotted in Fig. 7. In particular, because the relationship between K or D and C is monotonic, one can infer the nominal "influence diversity" of an actual product from its empirically measurable interaction density (K) and cyclic degree (C).

[6] K_i of each component is still preserved, whereas the component's influence links are now totally nonspecific to any predefined niche. In addition, the resulted networks with $D = 1$ are not purely random. The components that are primarily in more upstream of a product hierarchy have a broader scope of influence than components that are more downstream. By a simple robustness checking simulation exercise, we found that if the networks are rewired without preserving each component's preassigned number of outgoing influences (K_i) when rewiring, the main conclusions of the paper still hold.

[7] The model is repeatedly run to simulate many network samples. For each given combination of inputs (N, K, G), we simulate 2,000 networks and calculate the average cyclic degree of each sample of 2,000 networks. To improve the fitness of the randomly generated networks, only the ones with the given N components fully connected and K within 3% of the target value were accepted as valid trials.

References

Ahmad N, Wynn DC, Clarkson JP (2013) Change impact on a product and its redesign process: a tool for knowledge capture and reuse. Res Eng Design 24:219–244

Baldwin CY, Clark KB (2000) Design rules, vol 1., the power of modularityMIT Press, Cambridge

Beesemyer JC, Fulcoly DO, Ross AM, Rhodes DH (2011) Developing methods to design for evolvability: research approach and preliminary design principles. In: 9th conference on systems engineering research, Los Angeles, CA

Butterfield MI, Pearlman JS, Vickroy SC (2008) A system-of-systems engineering GEOSS: architectural approach. IEEE Syst J 2(3):321–332

Cardin MA, Kolfschoten GL, Frey DD, de Neufville R, de Weck OL, Geltner DM (2013) Empirical evaluation of procedures to generate flexibility in engineering systems and improve lifecycle performance. Res Eng Design 24:277–295

Cheng H, Chu X (2012) A network-based assessment approach for change impacts on complex product. J Intell Manuf 23(4):1419–1431

Clark K, Fujimoto T (1991) Product development performance: strategy, organization, and management in the world auto industry. Harvard Business School Press, Boston

Clarkson J, Simons C, Eckert C (2004) Predicting change propagation in complex design. ASME J Mech Des 126(5):788–797

Cohen T, Navathe SB, Fulton RE (2000) C-FAR, change favorable representation. Comput Aided Des 32(5):321–338

Colagrave N, Collins S (2008) Experimental evolution: experimental evolution and evolvability. Heredity 100:464–470

Dahmus JB, Gonzalez-Zugasti JP, Otto KN (2001) Modular product architecture. Des Stud 22(5):409–424

Danilovic M, Browning TR (2007) Managing complex product development projects with design structure matrices and domain mapping matrices. Int J Project Manage 25:300–314

de Neufville R, Scholtes S (2011) Flexibility in engineering design. The MIT Press, Cambridge

de Neufville R, de Weck OL, Frey D, Hastings D, Larson R, Simchi-Levi D, Oye K, Weigel A, Welsch R (2004) Uncertainty management for engineering systems planning and design. In: MIT engineering systems division monograph, engineering systems symposium, Cambridge, MA

Eckert C, Clarkson P, Zanker W (2004) Change and customization in complex engineering domains. Res Eng Design 15:1–21

Engel A, Reich Y (2013) Architecting systems for optimal lifetime adaptability. In: International conference on engineering design, Seoul, Korea

Eppinger SD, Browning TR (2012) Design structure matrix: methods and applications. The MIT Press, Cambridge

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

Fricke E, Schulz AP (2005) Design for changeability (DFC): principles to enable changes in systems throughout their entire lifecycle. Systems Engineering 8(4):342–359

Fulcoly DO (2012) A normative approach to designing for evolvability: methods and metrics for considering evolvability in systems engineering. Master's Thesis, Massachusetts Institute of Technology

Giffin M, Keller R, Eckert C, de Weck OL, Bounova G, Clarkson PJ (2009) Change propagation analysis in complex technical systems. J Mech Des 131(8):081001

Hamraz B, Caldwell NHM, Clarkson PJ (2012) A multi-domain engineering change propagation model to support uncertainty reduction and risk management in design. J Mech Design 134(10):100905.01

Hamraz B, Caldwell NHM, Clarkson PJ (2013) A holistic framework for categorisation of literature in engineering change management. Syst Eng 16(4):473–505

Hansen TF (2003) Is modularity necessary for evolvability? Remarks on the relationship between pleiotropy and evolvability. Biosystems 69(2–3):83–94

Henderson RM, Clark KB (1990) Architectural innovation: the reconfiguration of existing product technologies and failure of established firms. Adm Sci Q 35:9–30

Hölttä-Otto K, Chiriac N, Suh ES, Lysy D (2012) Comparative analysis of coupling modularity metrics. J Eng Des 23:10–11

Jarratt TAW, Eckert CM, Caldwell NHM, Clarkson PJ (2011) Engineering change: an overview and perspective on the literature. Res Eng Design 22(2):103–124

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

Kauffman SA, Weinberger DE (1989) The NK model of rugged fitness landscapes and its application to maturation of the immune response. J Theor Bio 141:211–245

Keese D, Seepersad C, Wood KL (2009) Product flexibility measurement with enhanced change modes and effects analysis (CMEA). Int J Mass Cust 3(2):115–145

Keller R, Eckert CM, Clarkson PJ (2006) Matrices or node-link diagrams: which visual representation is better for visualising connectivity models. Inf Vis 5:62–76

Kelly K (2010) What technology wants. Viking, New York

Kirschner M, Gerhart J (1998) Evolvability. Proc Natl Acad Sci USA 95:8420–8427

Koh ECY, Caldwell NHM, Clarkson PJ (2012) A method to assess the effects of engineering change propagation. Res Eng Design 23(4):329–351

Koh ECY, Caldwell NHM, Clarkson PJ (2013) A technique to assess the changeability of complex engineering systems. J Eng Des 24(7):477–498

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

Luo J, Magee CL (2011) Detecting evolving patterns of self-organizing networks by flow hierarchy measurement. Complexity 16(6):53–61

MacCormack A, Rusnak J, Baldwin CY (2006) Exploring the structure of complex software designs: an empirical study of open source and proprietary code. Manage Sci 52(7):1015–1030

Martin MV, Ishii K (2002) Design for variety: developing standardized and modularized product platform architectures. Res Eng Design 13:213–235

Mihm J, Loch C, Huchzermeier A (2003) Problem-solving oscillations in complex engineering projects. Manage Sci 49(6):733–750

Newman MEJ (2003) Structure and function of complex networks. SIAM Rev 45:167

Ollinger GA, Stahovich TF (2004) RedesignIT—a model-based tool for managing design changes. J Mech Des 126(2):208–216

Otto KN, Wood KL (1998) Product evolution: a reverse engineering and redesign methodology. Res Eng Design 10(4):226–243

Pasqual MC, deWeck OL (2012) Multilayer network model for analysis and management of change propagation. Res Eng Design 23(4):305–328

Pimmler TU, Eppinger SD (1994) Integration analysis of product decompositions. In: Proceedings of ASME 6th international conference on design theory and methodology, Minneapolis, MN

Rajan P, Van Wie M, Wood KL, Campbell MI, Otto KN (2005) An empirical foundation for product flexibility. Des Stud 26(3):405–438

Ricci N, Rhodes RH, Roos AM (2014) Evolvability-related options in military systems of systems. Proc Comput Sci 28:314–321

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

Ross AM, Rhodes DH, Hastings DE (2008) Defining changeability: reconciling flexibility, adaptability, scalability, modifiability, and robustness for maintaining system lifecycle value. Syst Eng 11(3):246–262

Rutka A, Guenov M, Lemmens Y, Schmdit-Schaeffer T, Coleman P, Rivi'ere A (2006) Methods for engineering change propagation analysis. In: Proceedings of the 25th congress of the international council of the aeronautical sciences, ICAS, Stockholm, Sweden

Saleh JH, Mark G, Jordan NC (2009) Flexibility: a multi-disciplinary literature review and a research agenda for designing flexible engineering systems. J Eng Des 20(3):307–323

Siddiqi A, de Weck OL (2008) Modeling methods and conceptual design principles for reconfigurable systems. J Mech Des 130:101102–101115

Silver M, de Weck OL (2007) Time-expanded decision networks: a framework for designing evolvable complex systems. Systems Engineering 10:167–186

Simon HA (1962) The architecture of complexity. Proc Am Philos Soc 106:467–482

Simpson TW, Maier JRA, Mistree F (2001) Product platform design: method and application. Res Eng Design 13(1):2–22

Simpson TW, Jiao J, Siddique Z, Hölttä-Otto K (2013) Advances in product family and product platform design: methods and applications. Springer, Berlin

Singh V, Skiles SM, Krager JE, Wood KL, Jensen D, Sierakowski R (2009) Innovations in design through transformation: a fundamental study of transformation principles. J Mech Des 131(8):081010

Smaling R, de Weck OL (2007) Assessing risks and opportunities of technology infusion in system design. Syst Eng 10(1):1–25

Smith RP, Eppinger SD (1997a) Identifying controlling features of engineering design iteration. Manage Sci 43(3):276–293

Smith RP, Eppinger SD (1997b) A predictive model of sequential iteration in engineering design. Manage Sci 43(8):1104–1120

Sosa ME, Eppinger SD, Rowles CM (2007) A network approach to define modularity of components in complex products. J Mech Des 129(11):1118–1129

Sosa ME, Mihm J, Browning TR (2013) Linking cyclicality and product quality. Manuf Serv Oper Manag 15(3):473–491

Suh NP (1990) axiomatic design: the principles of design. Oxford University Press, Oxford

Suh ES, de Weck OL, Chang D (2007) Flexible product platforms: framework and case study. Res Eng Design 18(2):67–89

Tilstra A, Seepersad C, Wood KL (2012) High definition design structure matrix for quantitative assessment of product architecture. J Eng Des 23(10–11):767–789

Tilstra A, Backlund P, Seepersad C, Wood KL (2013) Principles for design products with flexibility for future evolution. Int J Mass Cust (forthcoming)

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

Ulrich KT, Eppinger SD (2011) Product design and development. McGraw-Hill, New York

Wagner GP, Altenberg L (1996) Perspective: complex adaptations and the evolution of evolvability. Evolution 50:967–976

Whitney DE (1996) Why mechanical design cannot be like VLSI design. Res Eng Design 8(3):125–138

Whitney DE, Crawley E, deWeck O, Eppinger S, Magee C, Moses J, Seering W, Schindall J, Wallace D (2004) The influence of architecture in engineering systems. Engineering Systems Monograph, MIT Engineering Systems Division, Cambridge

Williams R, Martinez N (2000) Simple rules yield complex food webs. Nature 404:180–183

Yassine AA, Falkenburg DR (1999) A framework for design process specifications management. J Eng Des 10(3):223–234

Ziman J (2000) Technological innovation as an evolutionary process. Cambridge University Press, Cambridge