Chapter 34
A Framework for Measuring the "Fit" Between Product and Organizational Architectures

Zoe Szajnfarber and Erica Gralla

Abstract This paper develops a framework for measuring the "fit" between product and organizational architectures. The ability to measure "fit" in an objective and systematic way is a necessary precursor to understanding the nature of the hidden costs associated with design support tools that are becoming commonplace enablers of complex system design. Specifically, these tools are enabled by significant upfront decomposition – the problem is a priori broken up into a set of loosely coupled tasks that can be worked in parallel, with interactions across tasks routinized and often encoded in computational tools. When this imposed structure fits the problem well, it can drastically speed up design cycles and enable intraorganizational collaboration, by hiding extraneous information and freeing up experts' time to focus on the hardest parts. However, even minor mismatches between the organizational decomposition (people and tasks) and product decomposition (the problem being solved) can cause designers to miss important trades and make poor choices. The proposed measurement framework builds on existing measures from the organizational design literature and systems engineering literature. Our contribution lies in unifying the level of analysis of the two disciplines and developing a novel strategy for tracking the interaction among the product and organizational system. The utility of this approach for observing influences in real systems is demonstrated with a "toy" case study example based on space system development at the Jet Propulsion Laboratory.

Keywords Architecture • Decomposition • Organization • Design structure matrix • Mirroring

---

Zoe Szajnfarber and Erica Gralla contributed equally to this work.

Z. Szajnfarber (✉)
Engineering Management and Systems Engineering and Elliott School of International Affairs, the George Washington University, Washington, DC, USA
e-mail: zszajnfa@gwu.edu

E. Gralla (✉)
Engineering Management and Systems Engineering, the George Washington University, Washington, DC, USA
e-mail: egralla@gwu.edu

A.M. Madni et al. (eds.), *Disciplinary Convergence in Systems Engineering Research*, https://doi.org/10.1007/978-3-319-62217-0_34

34.1 Introduction

As today's engineered systems become increasingly complex, effective design requires bringing diverse expertise and knowledge to bear on each design iteration, in an expedient way. Faster design cycles and better cross-disciplinary integration are enabled by the ever-increasing power of model-based approaches to concurrent design (e.g., NASA's Team X, EADS Astrium Satellite Design Office). However, these advantages come at a cost, and that cost has not received enough attention. Specifically, concurrent design and engineering is enabled by significant upfront decomposition – the problem is a priori broken up into a set of loosely coupled tasks that can be worked in parallel, with interactions across tasks routinized and often encoded in computational tools. When this imposed structure fits the problem well, it can drastically speed up design cycles by hiding extraneous information and freeing up experts' time to focus on the hardest parts. However, even minor mismatches between the organizational decomposition (people and tasks) and product decomposition (the problem being solved) can cause designers to miss important trades and make poor choices [1, 2]. As these design tools become more popular, it is important to understand their scope of applicability and potential costs.

A critical precursor to understanding the impact of lack of "fit" on the design process is an ability to empirically observe and measure the "fit" between the imposed organizational architecture and the natural decomposition of the technical system. In this paper, we take that first step, carefully defining what is meant by "fit," identifying its key dimensions, and proposing a framework for how it can be measured. The concept of "fit" has previously been explored in the management literature (as in the mirroring hypothesis; [3]) and in software engineering (as Conway's Law); however, we take the notion of matching much farther, tracking the nature of organizational-product interactions that drive goodness of fit. We illustrate (a) the need to adopt consistent levels in characterizing the architectures of the organization and the product and (b) the challenges in observing and tracking the interactions between them through application to an empirical setting.

34.2 Related Literature

Decomposition of complex systems has been studied extensively from a variety of perspectives. The most relevant streams are those that focus on the relationship between the decomposition of a technical product and the decomposition of the organization that designs or produces it.

Product Decomposition Technical products are decomposed because they are complex. Complexity is often managed through task and product decomposition [4, 5]. Specifically, since no individual can practically process the amount of specialized knowledge required to design a complex integrated system,

decomposition serves to break the problem into a set of (nearly) decoupled, individually tractable modules [6–8]. Intelligently selected modules are tightly coupled internally and loosely coupled to other modules in the system. This allows work on individual modules to proceed independently and in parallel [8–10].

Organizational Decomposition. Organizations employ decomposition for similar reasons: to manage complexity. If an individual is faced with more information than (s)he can feasibly process, (s)he is no less able to make a decision than if (s)he had too little. Tushman and Nadler [11] view an organization as an information processing system. Hierarchical structures in organizations serve to compartmentalize where decisions need to be made. In these structures, information is "hidden" inside business units, and only some information is flowed up the organization, to limit the scope of information each unit must deal with.

"Fit" Between Product and Organizational Decompositions The literature also addresses the relationship between product and organizational decompositions. The core finding of this literature is the so-called mirroring hypothesis, which states that: the structure of the organization that develops the technology (defined in terms of, e.g., communication links, collocation, team, and firm co-membership) will match the product architecture of the system under development [3, 6, 8]. Substantial effort has gone into demonstrating the empirical validity of the hypothesis [12]. An implication of the hypothesis is that successful design can occur when the organization is less decomposed than the product, but not the other way around.

Level of Decomposition. All technical systems can be decomposed to some degree at relatively low cost. For a given system, the precise level depends on the inherent structure of the system [6, 13]. Beyond this point, systems can be actively decomposed into progressively smaller subproblems by imposing design rules [8], global explicit rules about how a system will operate. The idea is to define key design parameters and guarantee that they will not change; this allows modules to depend on design rules and not on each other. Design rules can take the form of standards (e.g., 802.11b) or be embodied in interface control documents.

Impact of Decomposition on the Design Process While decomposition is necessary, and ensures tractability of the design process, it also constrains the trajectory of the design process in several ways. Because work happens within clearly defined module boundaries, new insights are developed inside particular modules rather than across them [14, 15]. In addition, the decomposition defines how coordination happens across interdependent tasks [4, 11, 16]. Information is "hidden" within modules and shared across them only when the need is clear, which may result in missed opportunities for design trades. For all these reasons, the decomposition influences the design process and the space of possible design solutions that can be explored. When chosen well, decompositions can streamline the design process substantially.

Costs of Over-Decomposing However, product and organizational decompositions are not always chosen well, and the costs of poor decomposition choices are

not well understood. While volumes have been written about the value of modular decomposition [8, 10, 17], its costs have only been discussed in a superficial way. For example, costs of structural overhead associated with splitting were discussed by Baldwin and Clark [8]. Ethiraj and Levinthal [1] find that under-decomposing leads to limited search and suboptimal designs, while over-decomposing leads to stalled improvement in design performance, but this is based on a simplified model rather than empirical studies.

The cost of getting a decomposition "wrong," in the sense that the product decomposition does not "fit" the organizational decomposition, has not been examined explicitly, nor has this concept of "fit" been well articulated. Our survey of the extant literature makes clear that the selection of a decomposition strongly influences the trajectory of the design process, so it is important to understand the potential problems that could result: the costs of decomposition.

34.3 Approach

This paper describes an initial attempt to develop a framework for measuring the fit between organizational and product architecture. We began by surveying the literature to identify how each of the organizational design and product architectures is represented in their respective literatures. We then assessed their mutual consistency – could representations of organizational design be contrasted directly with representations of product architectures? We built on concepts in both literatures to develop a framework that enables clear description of the key elements of both product and organizational architectures in the same "language," so that the representations could be directly compared.

We then applied these measures to an example from the domain of space system design to both identify areas where additional resolution is required and also to illustrate the kinds of insights that can be gained. This paper represents a first step toward empirical investigation of the effects of mismatches between organizational and product architectures. The framework will guide empirical data collection and support the analysis of empirical results.

34.4 Diagnosing "Fit" Between Organizational and Product Architecture

We next develop a screening tool to assess the quality of "fit" between an organization and its product architecture. While there are established frameworks for characterizing and representing both the "design" of an organization and the architecture of its product system, there is currently no common basis upon which to measure "fit." We build on existing approaches to develop an appropriate

framework for (1) representing (i) the product architecture and (ii) the organizational decomposition, and (2) assessing the "fit" between them and screening for potential issues. Each of these endeavors is described below. As a preliminary test of the screening tool, we apply it to characterize a spacecraft development team.

34.4.1 Representing Architecture

A common way of representing the structure of a system is in a Design Structure Matrix (DSM) [18, 19]. A DSM is an n × n matrix representation that lists variables on both rows and columns; when an "X" appears in the matrix, it signifies that the variable in its row requires an input from the variable in its column. A DSM is often analyzed and rearranged in order to group together tightly coupled sets of variables into modules. This is useful because these interconnected variables need to be designed simultaneously, usually by an individual or team, while work on different modules can proceed in parallel. The basic DSM structure has been used to document interdependencies among product components or design tasks [19]. We will use the basic DSM structure as a starting point to represent the architectures of both the product being designed and the organization carrying out the design.

34.4.1.1 Product DSM

One DSM will represent the design problem as a product DSM. Product DSMs are fairly standard and aim to capture the interdependencies among modules of the technical system. These DSMs must represent two components. (1-p) The elements of the DSM represent the standard product subsystems (in a spacecraft, e.g., subsystems would include thermal, configuration, power, etc.). (2-p) Many of the subsystems are interdependent (e.g., if the collecting area of the instrument is enlarged, there will be more data to process and downlink and the electronics will require additional thermal regulation). These interdependencies are abstracted as "Xs" in the off-diagonal cells of the DSM.

In the product DSM, more advanced representations also seek to distinguish between local and global dependencies among subsystems. Here we specify two special cases and discuss how they can be represented in a DSM. Traditional interface control documents (ICDs) capture the predefined dependency between two subsystems. For example, the definition of a VGA port guarantees that any monitor can be interfaced with any personal computer or laptop. It allows design decisions internal to each subsystem (i.e., those not affecting the interface) to be made completely independently. Design rules, as coined by Baldwin and Clark [8], explicitly define global dependencies that can be assumed for all subsystems. For example, North American power outlets provide power at 60 Hz. The key difference between these two types of interdependencies lies the ability to change them. While ICDs are intended to be fixed for the life of a project, if there's a compelling

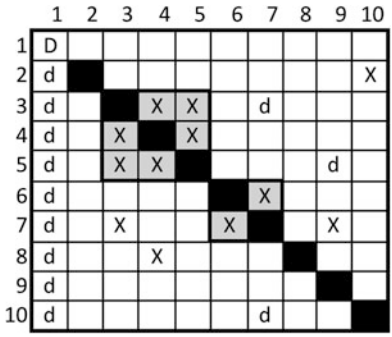

Fig. 34.1 Notional pDSM

reason to modify the standard, a change only needs to be agreed on by the two subsystems affected. If, on the other hand, a change needs to be made that affects a design rule, the effects would ripple through the whole system (and any past systems that relied on that design rule).

In the DSM, the ICD-type dependency (3-p) is represented by replacing the "X" with a "d" to indicate that the dependency has been fixed. A design rule-type dependency (4-p) shows up as a separate element at the top left corner of the DSM. Since it affects most of the other elements of the system, one would expect to see a "d" in nearly all rows of that column. Figure 34.1 illustrates a notional product DSM.

34.4.1.2 Organizational DSM

The second (separate) DSM will represent the structure of the organization doing the work. While product-level task-based DSMs are fairly common and have been used extensively to analyze project attributes like efficient work distribution and ordering (e.g., [18]), the DSM construct has not been used to analyze "standing organizations" (i.e., those designed to work on multiple distinct products that aren't known a priori) in any detail. This is the view of the organization taken in this research.

For our purposes, the organizational DSM needs to include four critical components.

(1-o) Similar to the pDSM, the elements of the oDSM represent the business units or task owners in the organization. These tend to map to subsystems in the physical system, but that is not always the case. For example, you would likely have a propulsion team that maps to the propulsion subsystem, but you might also have a quality assurance team that does not map directly to a single subsystem. Where in the pDSM, the important distinction is among implicit (2-p), local (3-p) and global (4-p) dependencies, in the oDSM, the important distinction is between passive (routinized) and actively managed (human) interdependencies.

(2-o) In the oDSM, the routinized aspects of organizational interdependencies are captured with "Rs." They can show up as off-diagonal "Rs," but more often they mirror the global design rule construct, with one subsystem automatically depending on another through a central (often artificial) layer. As an example of this kind of dependency, many organizations implement an IT backbone that automatically updates each business unit when decisions or commitments made in one unit impact another. For example, in situations where multiple groups draw from a common inventory, when one draws that inventory down, it affects what's available to the other unit. It is important to note a distinction between the "R" in the oDSM and the "D" in the pDSM: whereas a design rule "D" indicates a static value, so that all subsystems can proceed with their own designs without worrying that the interface with "D" will change, a routinized interdependency "R" indicates that the existence of the *relationship* is static, not necessarily the value passed within that relationship. Continuing the inventory example above, the "R" dependency does not mean that business unit A can rely on a static inventory level of, say, 4. It means that they can check their current allocation through the IT backbone without talking to business unit B, even if their level depends on B's use.

(3-o) In the oDSM, we also need to capture the common case where interdependencies are handled and negotiated in real time, i.e., actively managed by humans. These are represented as off-diagonal "Hs." For example, many organizations use cross-disciplinary high-performance teams to dynamically reallocate resources across business units and tasks. It can also happen less formally between two business units. For example, on a technical project, detailed design may have revealed that an assumed process won't be feasible in practice. This may affect requirements allocated to multiple subsystems, and necessitate a reallocation of that resource.

(4-o) The last aspect of the organization that needs to be captured is the physical layout of the organization or workspace, because it impacts the ease of coordinating across interdependencies (Allen 1997). This involves considering the collocation of particular aspects of work (e.g., in a multinational corporation, which subunits share a facility in a particular location). These constraints can be represented in the DSM as a fixed ordering of certain rows/columns, which means that there are limited options for reordering the DSM (a core aspect of traditional DSM analysis).

Figure 34.2 illustrates a notional oDSM. It takes the same general form as a pDSM, but as noted above, some of the elements take on different meanings. For a given product-organizational pair, the DSMs are often not the same size, since there may be extra organizational units or single organizational units that are responsible for more than one technical subsystem. Rows/columns may have a fixed ordering, represented by the brackets to the right of the matrix.

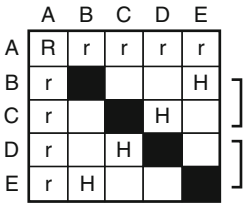

Fig. 34.2 Notional oDSM

34.4.2 Assessing Fit

The next step is to measure the "fit" between the pDSM and the oDSM. As noted above, previous attempts have been coarse and qualitative. For example, MacCormack et al. [3] compare the software architectures of two similar products, one developed by an open source team and the other by a traditional "closed" organization. The study found that "fit" was important because the traditional organization generated a product with a large interconnected core, while the open team produced a product architecture with a comparatively small core and many loosely coupled modules. Since our goal is to assess the impact of lack of fit, we need a more granular and systematic representation.

34.4.2.1 Conceptual Basis: Problems to Be Identified

Before discussing the mechanics of how to measure "fit," it is important to be clear about the conceptual intent of the proposed framework. It needs to be able to identify potential problems (which can then be further investigated) resulting from a lack of "fit" between the pDSM and the oDSM. Extant theory on fit provides some guidance on the types of problems we expect to find. In the following paragraphs, we use this theory to develop a list of the types of problems the framework must identify.

At the highest level, a good "fit" is defined as a perfect overlap between (a) the elements in each DSM and (b) their respective feedback structure. Mismatches occur when a technical element or feedback does not map to an organizational counterpart or vice versa. While it is more problematic for a need for technical feedback to exist where there is no organizational system to facilitate it, misplaced institutional structures (e.g., a cross-functional team where no tradeoff needs to be made) represent a wasteful overdesign. Therefore, the framework needs to be able to directly compare the pDSM and oDSM in terms of their joint ability to handle necessary feedback in the design process.

A structural mismatch does not necessarily indicate a problem. Evaluating the potential impact of that mismatch requires some understanding of its potential outcome. Prior research has highlighted several kinds of impacts that relate to mismatches.

First, it is known that routinizing information flow can have substantial advantages in terms of process efficiency [2]. Since everything happens automatically, the process tends to fade into the background and is unobservable except in terms of the process artifact. This is great when the routinized structures fit the natural communication flows of the product; however, even slight mismatches can cause problems if the organization forgets that hidden processes were enabling smooth operations, and fails to evolve [2]. An unaccounted-for divergence between the product (which changes from, e.g., one project to the next) and the organization (which stays the same) can lead to two types of problems in our framework:

- Problem type 1: Lack of institutional pathway where one is needed. Diagnosing this issue requires two levels of analysis, elaborated upon below. First, in terms of structural match between the pDSM and oDSM, this would involve an off-diagonal X in the pDSM, matched by only an "r" in the oDSM or potentially no structure at all. Second, the impact can be observed in terms of the "extra" communication required to deal with the technical feedback in the unplanned organizational location. This might be observed as pulling leads from several subsystems into an emergency meeting to resolve an issue discovered late in the process.
- Problem type 2: Overdesign (excess institutional pathways). Now instead of the change in product leading to a lack of institutional pathway where one is needed, an overdesign occurs where a legacy communication paths remains where it is no longer needed. In the structural sense, this would show up as an oDSM "h" with no counterpart in the pDSM. In terms of communication, one would observe little or none in a location where a lot is expected (e.g., a standing cross-functional team exists).

The product analog to the organizational routinization of information flow is embodied in the concept of a predefined design rule. An important intent of a product decomposition is to enable subtasks to be executed independently and/or in parallel [6, 8]. This can add substantial process efficiencies, as long as the decomposition is clean. Ethiraj and Levinthal [1] have shown that a poor decomposition – one where the divisions do not match the natural structure of the system – can do more harm than good. Technical issues don't surface until late in the design process or inefficient designs are chosen so that everyone can fit in their poorly allocated requirements.

- Problem type 3: Poor decomposition (not accounting for a technical interdependency). In this case, no structural mismatch between the pDSM and oDSM would surface. The issue is that a technical feedback was not represented in the pDSM when it needs to be. To identify a problem type 3 requires observation of the in-process communications. Here, one would observe high incidence of communication where none is expected (i.e., the pDSM is empty and the oDSM is likely empty too (though that is not required)).

In considering levels of communication, it is important to recognize that not all incidences of high communication are indicative of a problem. An important

systems engineering function involves brokering in a structured way across predefined interfaces. This is particularly important early in the design process. Therefore, if a high level of communication is observed – especially if it happens early – in a location where there is both a product feedback and an organizational feedback, it is simply evidence of the organization working as it should. Our framework enables the identification of these problems.

34.4.2.2 Generating a Structural Matching Matrix (mDSM)

The first step in assessing fit involves a direct comparison of the structural similarities of pDSM and oDSM. To do this first requires that the pDSM and the oDSM be matched in terms of size – accounting for the fact that some organizational elements don't have technical counterparts and that some organizational elements deal with more than one technical subsystem. For the notional example above, this might look like Fig. 34.3. Business units B and C are responsible for three technical subsystems each, and business unit D is responsible for two. The assignment of pDSM elements to oDSM elements should follow the actual assignment of work in the organization. If any pDSM element has no mapping to the oDSM, it can be added as a row/column in the oDSM; this would be a serious oversight on the organization's part and therefore is likely a rare occurrence in real organizations.

In order to create a matrix like Fig. 34.3, a second transformation may also be required. Recall that the order of the oDSM rows/columns is often fixed, to represent the collocation of some teams or personnel. Therefore, the pDSM may require reordering in order to map to the oDSM structure. In our example, we assume that the pDSM has already been reordered so that pDSM elements 2, 3, and 4 are all assigned to business unit B, etc. The difference between an optimized pDSM (e.g., one in which above-diagonal interfaces are minimized) and the pDSM required to match the oDSM is one measure of the cost of a decomposition. This will be explored in future work.

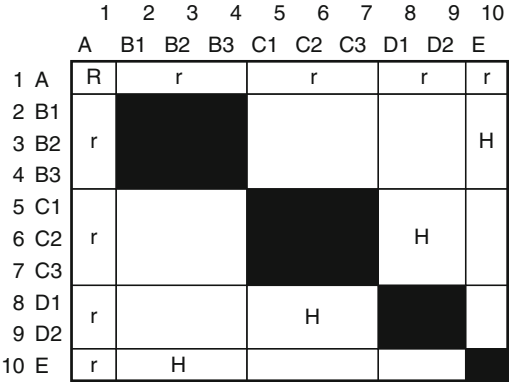

Fig. 34.3 Transformed oDSM

Fig. 34.4 mDSM

		1	2	3	4	5	6	7	8	9	10
		A	B1	B2	B3	C1	C2	C3	D1	D2	E
1	A	0	1	1	1	1	1	1	1	1	1
2	B1	0	0	1	1						0
3	B2	0	1	0	0	-1		-1			1
4	B3	0	1	0	0	-1					1
5	C1	0		-1	-1	0	1	1	1	0	
6	C2	0				1	0	0	1	1	
7	C3	0		-1		1	0	0	1	0	
8	D1	0			-1	1	1	1	0	1	
9	D2	0				1	1	1	1	0	
10	E	0	1	1	1			-1			0

With this transposition, the reordered pDSM can be "subtracted" from the modified oDSM. We will record the difference (oDSM-pDSM) as follows. Our example mDSM is shown in Fig. 34.4.

- 0 = an element or feedback was present in both structures.
- Blank cell = no elements were present in either structure.
- 1 = an element or feedback was present in the oDSM but not the pDSM.
- −1 = an element or feedback was present in the pDSM but not the oDSM.

34.4.2.3 Screening for Issues Using Information Flow (iDSM)

A mismatch among the pDSM and oDSM becomes an issue when communication that should have happened does not happen, and an inefficient design and/or design process results. Alternatively, an organization can be "overdesigned" to accommodate information flows that never materialize. Section 34.4.2.1 identified specific types of problems that might occur. In terms of the framework, these situations can be operationalized as follows:

- Problem type 1: Lack of institutional pathway where one is needed = −1 AND high level of information flow
- Problem type 2: Overdesign (excess institutional pathways) = 1 AND limited information flow
- Problem type 3: Poor decomposition = blank cell AND high level of information flow
- Normal process = 0 AND information flow

Clearly, a measure of information flow is needed to assess whether the mDSM indicates that there are problems. We propose the use of a parallel matrix, termed an iDSM, to track the intensity of information flow actually required during the design process. The iDSM distinguishes intense from limited information flow in the

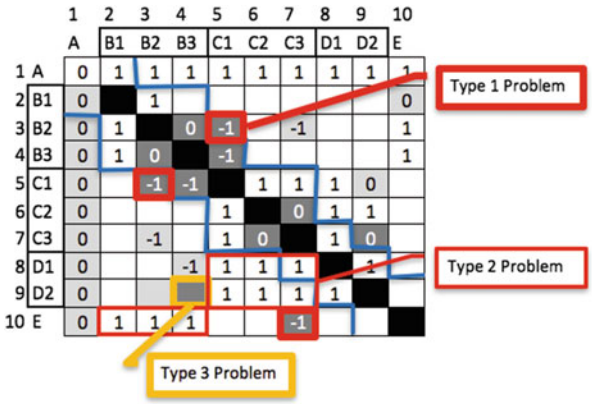

Fig. 34.5 iDSM overlaid on mDSM

interaction among the matrix elements. If we superimpose the iDSM over the mDSM, we can distinguish the problematic interfaces from those that are appropriately managed. Figure 34.5 provides an example. The iDSM provides the color for the cells, while the value of the cell is taken from the mDSM. Cells in light gray exhibit limited information flow, and cells in dark gray exhibit intense information flow.

Problems can now be identified. Most of the dark gray cells are not problematic, because they are appropriately managed by grouping tasks within business units or by actively managed interfaces. However, there are several dark gray cells that contain −1 entries. These indicate a lack of institutional pathways where one is needed (problem type 1). There is also a dark gray cell that is blank, which indicates a poor decomposition (problem type 3): the interdependency of these elements was not recognized in the pDSM. Another set of problems is indicated by the white cells that contain 1 entries: these may indicate overdesign (problem type 2), in which institutional pathways exist where they are not needed. In most cases, these are the result of organizational units managing more than one technical element, but not in all cases; these latter are likely more problematic.

A second layer should be considered in assessing these results. Recall that the ordering of the rows/columns indicates collocation, which eases communication. If two teams are collocated, they may be able to create an actively managed interface even if it was not planned for originally. The blue lines in Fig B outline a notional border within which such "on the fly" active management is feasible: they include entire business units (such as B1-B3) and rows/columns that are directly adjacent (such as B3 and C1) on the assumption that these subteams are next to one another. Under these conditions, a few additional problematic interfaces (C1-B3) may be appropriately managed.

34.5 Preliminary Validation of the Approach

For any screening tool to be useful, it must be implementable in a real-world context. This generally means that (1) the constructs both have meaning in and fully describe empirical settings and that (2) the required data is available or at least collectable. Therefore, in this section, we first consider data needs for each of the base DSMs (p, o, and i) and then apply the constructs in a very preliminary way to a setting we hope to study further in the future.

34.5.1 Data Inputs

The information needed to produce the pDSM is generally well documented in most organizations. It can therefore be constructed based on source documents such as blueprints, architecture drawings, or more generally design documentation. For example, Suh et al. did this for Xerox machines [20].

The same is generally true for oDSMs. The bulk of the information needed to construct an oDSM will be stored in documents like organization charts that define business unit leads or cognizant engineers (CogEs) or similar. Since the standing up of high-performance teams or the use of liaisons between groups can be somewhat ephemeral, it may be harder to find explicit documentation.

Where the pDSM and oDSM can be reconstructed from archival documents, the iDSM is more difficult because information flow is not generally captured in such documents. Instead, information flow is revealed by the meetings, emails, and other communication that occurs over the course of the project. There is a lot of communication that is internal to a business unit (e.g., unit B may lead subsystems 3–5 which strongly influence each other). To limit the scope of required data collection, we will intentionally focus on across-unit communication, because this is where mismatches tend to be most relevant.

Since this type of data is rarely documented, it must be gathered by observing the progression of a project. The specific content of the information will be project dependent – a large scale project will have a sequence of formal reviews and associated structured technical interchanges, where a conceptual design effort may be limited to a few team "huddles." Level of information flow is therefore a relative concept. For our present purposes, a simple categorization of information flow as "high" or "low" is sufficient. In the future, it may be important to track information flow over time. As noted above, high levels of flow late in the process can be important. It will also be interesting to record when information flow actually impacts the design. These issues will be recorded but not used at this stage of the framework development.

34.5.2 A Preliminary Test of the Tool

As an initial test of this framework, we have identified a suitable empirical setting to exercise it. JPL's Team X is widely considered an exemplary concurrent engineering team, and they have completed more than a thousand studies since 1994 [21]. Their success led to requests to use their services for a wider variety of design problems, including orbiters, landers, and rovers. While the mission type varies, the fundamental structure of Team X changes very little from study to study. As a result, we can observe the design process across studies with varying fit between the organizational and product architectures.

Team X consists of a set of subject matter experts (SMEs) covering the main subsystems required for spacecraft design, and a set of tools and facilities to support their work. The main facility is a room with workstations for each subsystem; each workstation has a set of Excel spreadsheets that capture the evolving design of the subsystem, and each sheet includes links that push and pull parameters from other subsystems. The spreadsheets function as models of the subsystem, and include assumptions about the relationships among parameters. For example, a spreadsheet might include an assumption that the mass of the spacecraft bus is proportional by some factor to the mass of the payload. The spreadsheets are designed by the subject matter experts based on best practices in subsystem design and data from past missions. Team X typically conducts three-day design studies that end with a feasible "point design" for a spacecraft. In other words, the team begins with a set of customer requirements (such as a mass limit, cost cap, pointing requirements, etc.) and a basic architecture (such as an orbiter or a lander), and designs a spacecraft that meets these requirements.

Going forward we intend to use the screening tool to study the impact of different levels of mismatches between Team X's organizational and product architectures. In this initial study, we have a more modest goal. Table below uses the Team X context to verify that our tool's constructs have meaning and describe the setting in a useful way.

From the table we can already see how the tool will allow us to quickly hone in on potential problem areas that merit additional follow-up. For example, the Team X facilitator plays an explicit brokering role to resolve issues across subsystems. Therefore, we need to be able to distinguish high communication led by a facilitator from a similarly high level of communication led by a technical SME. In the first case, the high communication is normal whereas the second case is indicative of a problem wherein a technical SME might be forced to fill the same brokering function due to a mismatch. Layering the iDSM constructs on the mDSM (oDSM-pDSM) will let distinguish these cases as intended.

In future work, we intend to refine the framework through observation of a pilot study, then use it to examine the issues discussed in this paper.

DSM	Construct	Generic meaning	Examples from application to team X	Information source
pDSM	Rows & columns	Elements of product	Technical spacecraft subsystems (e.g., power, thermal, data handling)	Design documents
	Off-diagonal "X"	Physical interdependency, not governed by pre-established rule	Dependency between two subsystems (e.g., larger collecting area drives larger downlink needs)	Design documents and spreadsheet assumptions
	Off-diagonal "d"	Defined interface between two subsystems	Parameter value fixed between two subsystems (e.g., instrument agrees to s/c connector)	Design documents
	"D" as a module	Global design rule fixed for relevant subsystems	Parameter value fixed for whole system (e.g., 5 V power)	Design documents
oDSM	Rows & columns	Units within the organization	Represented by a "chair" where a technical SME sits (e.g., systems engineer).	Room layout and team staffing
	"R" as a module	Routinized (static) relationships between units or among all units	Technical spreadsheets govern routinized interactions (e.g., when subsystem enters bandwidth, it impacts mass roll-up)	Spreadsheet tool
	Off-diagonal "H"	Actively managed (by a human) dependencies between units	The study facilitator calls scheduled check-ins where system level trades are made. Informal collaborations also crop up.	Planned discussions from schedule. Informal huddles.
	Brackets	Defined geographical spacing of units	Instrument team often sequestered to separate room. Chairs in fixed locations, intended to bring closely collaborating SMEs closer together.	Room layout and team staffing
iDSM	Light gray shading	Low information flow	The deputy systems engineer periodically checks in with SMEs slow to populate the spreadsheet, to get a sense of their progress	Observation of design work
	Dark gray shading	High information flow	The facilitator called an unscheduled meeting between three subsystems when it became clear that the design might not close.	Observation of design work

34.6 Conclusion

In this paper, we set out to develop a framework to assess the "fit" between organizational and product architectures. We built on existing constructs in the literature to generate comparable organizational and product DSMs. Our modest contribution in that context is a formal framework for capturing feedback (in the pDSM) and feedback management (in the oDSM) at a constant level of analysis suitable for cross-comparison. The main contribution is in moving beyond the established notion of structural similarity to consider how structural mismatches actually create problems. By layering observed information flows (iDSM) on the matching matrix (oDSM-pDSM), we are able to quickly identify potential problem areas – where, for example, the institutional feedback mechanism is insufficient to handle necessary product feedback. As a preliminary validation of the utility of the tool, we applied it JPL's Team X. In future work, we intend to use that setting to further probe the implications of the identified mismatches.

Acknowledgments This work was supported by the National Science Foundation Grant CMMI-1563408.

References

1. Ethiraj S, Levinthal D (2004) Modularity and innovation in complex systems. Manag Sci 50 (2):159–173
2. 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
3. MacCormack A, Baldwin C, Rusnak J (2012) Exploring the duality between product and organizational architectures: a test of the 'mirroring' hypothesis. Res Policy 41(8):1309–1324
4. Thompson JD (1967) Organizations in action: social science bases of administrative theory. McGraw-Hill, New York
5. Williamson OE (1991) Comparative economic organization: the analysis of discrete structural alternatives. Adm Sci Q 36(2):269–296
6. Simon HA (1962) The architecture of complexity. Proc Am Philos Soc 106(6):468–482
7. Nadler D, Tushman M (1997) Competing by design: the power of organizational architecture. Oxford University Press, Oxford
8. Baldwin C, Clark K (2000) Design rules, volume 1: the power of Modularity. MIT Press, Cambridge, MA
9. Morelli MD, Eppinger SD, Gulati RK (1995) Predicting technical communication in product development organizations. IEEE Trans Eng Manag 42(3):215–222
10. Ulrich KT, Eppinger SD (1995) Product design and development. McGraw-Hill
11. Tushman ML, Nadler DA (1978) Information processing as an integrating concept in organizational design. Acad Manag Rev 3(3):613–624
12. Colfer LJ, Baldwin CY (2016) The mirroring hypothesis: theory, evidence and exceptions
13. Szajnfarber Z, Vrolijk A, Crusan J (2014) Exploring the interaction between open innovation methods and system complexity. In: Proceedings of the 4th International Engineering Systems Symposium
14. Parnas DL (1972) On the criteria to be used in decomposing systems into modules. Commun ACM 15(12):1053–1058

15. Prencipe A (1997) Technological competencies and product's evolutionary dynamics: a case study from the aero-engine industry. Res Policy 25(8):1261–1276
16. Galbraith J (1977) Organization design: an information processing view. Interfaces 4(3):28–36
17. Schilling (2000) Toward a general modular systems theory and its application to interfirm product modularity. Acad Manag Rev 25(2):313–334
18. Steward DV (1981) The design structure system: a method for managing the design of complex systems. IEEE Trans Eng Manag 28(3):71–74
19. Eppinger SD, Browning TR (2012) Design structure matrix methods and applications. MIT Press, Cambridge, MA
20. Suh ES, Furst MR, Mihalyov KJ, de Weck O (2009) Technology infusion for complex systems: A framework and case study. Sys Eng 13(2)
21. Sherwood B, McCleese D (2013) JPL innovation foundry. Acta Astronaut 89:236–247