Manuel E. Sosa
Technology Management Area, INSEAD,
77305 Fontainebleau, France
e-mail: manuel.sosa@insead.edu

Steven D. Eppinger
MIT Sloan School of Management,
Cambridge, MA 02142
e-mail: eppinger@mit.edu

Craig M. Rowles
Advanced Engine Program,
Pratt & Whitney Aircraft,
East Hartford, CT 06107
e-mail: rowlescm@pweh.com

Identifying Modular and Integrative Systems and Their Impact on Design Team Interactions

The typical approach to developing complex products is to decompose the product into systems, and these into components. We introduce a new notion of system modularity based upon the way components share design interfaces across systems. Modular systems are those whose design interfaces with other systems are clustered among physically adjacent systems, whereas integrative systems are those whose interfaces are physically distributed or functionally integrative across all or most other systems. Our research method allows us to study how system modularity impacts design team interactions. Our approach is illustrated by analyzing the development of an aircraft engine.
[DOI: 10.1115/1.1564074]

1 Introduction

Understanding how product development organizations manage the knowledge associated with the architecture of the product they design has been recognized as a critical challenge for established firms facing architectural innovation [1]. This paper presents a method that allows engineering managers to identify modular and integrative systems within a product, based on how systems share design interfaces. Furthermore, by studying the coupling between the product architecture and the development organization we enhance our understanding of the differences in designing modular versus integrative systems. The results of this work highlight the importance of identifying design interfaces during the project planning stage so that corresponding design teams are managed efficiently during project execution.

We are particularly interested in the development of complex products, such as automobiles, computers, or aircraft engines. The general approach when developing complex products is to decompose the product into systems, and if the systems are still too complex, decompose these into smaller components [2–4]. In the organizational domain, design teams in complex product development are commonly organized around the architecture of the product. In most technical products we have observed a clear mapping between the product architecture and the development organization which designs it [4]. However, little research has focused on understanding the effects on team dynamics involved with designing various types of architectures.

Product Architecture Literature. Ulrich [5] defines product architecture as ''the scheme by which the function of a product is allocated to physical components.'' A key feature of product architecture is the degree to which it is modular or integral. In modular architectures functional models of the product map one-to-one to its physical components. On the other hand, in integral architectures a large subset of the product's functional models map to a single or small number of components.

Other researchers have investigated the implications of product modularity on various aspects of product development. Newcomb et al. [6] study the effects of product modularity on product life cycle. Other researchers have studied the relation between product architecture and product portfolio definition [7–9]. The link between product architecture and supply chain has been addressed in the operations and management science literature [10–14]. Few other researchers have studied how product modularity may affect testing strategies of design alternatives [15,16].

In the engineering design field a large stream of research has focused on methods and rules to map functional models to physical components. Researchers have developed several architecting rules to map function to physical modules [17–20]. Other approaches view the functional model of a system as being described by an abstract functional decomposition that may, but do not need to, have a direct mapping onto physical decomposition of assemblies and subassemblies [21–24].

Ulrich and Eppinger [25] claim that the product architecture is also the scheme by ''which the chunks[1] [of a product] interact.'' In complex products, the chunks of physical components are complex systems as well. They also argue that the challenge of establishing the architecture of these systems ''is essentially identical to the architectural challenge posed at the level of the entire product.'' [[25], pp. 183] However, we believe that the challenge of establishing the architecture of systems (the chunks) is compounded by the fact that components have design interfaces not only with other components within the system but also with components that belong to other systems that comprise the product (see Fig. 1).

By using established concepts in the current product architecture literature we can categorize systems as modular or integral based on how their corresponding components share design interfaces (within the system). However, we also need to categorize systems according to how they share design interfaces with other systems as a result of the product architecture. Hence, at the system level in order to define system modularity we need to specify whether we are looking at the system *internally*, as an independent entity, or *externally*, as an entity that interacts with other systems comprising the product.

We introduce here the concepts of *modular* and *integrative systems*, from an external perspective, that is, based on the existence of design interfaces between components of the same product that belong to different systems. We define modular systems as those whose design interfaces with other systems are clustered among a few physically adjacent systems, whereas integrative systems are those whose design interfaces span all or most of the systems that

Contributed by the Design Theory and Methodology Committee for publication in the JOURNAL OF MECHANICAL DESIGN. Manuscript received July 2000; rev. October 2002. Associate Editor: J. Cagan.

[1] *Chunks* refer to the physical building blocks in which the components of a product are organized. We will use the term *systems* instead.

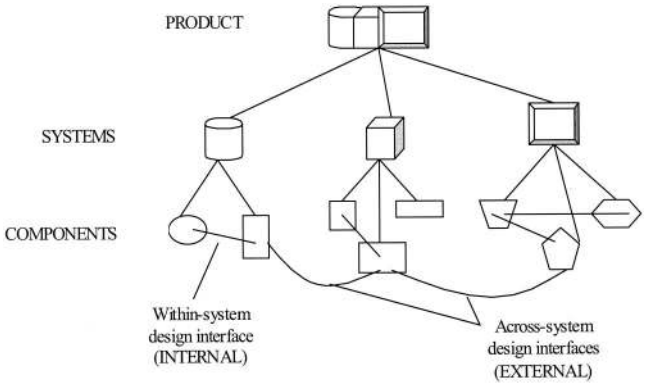

Fig. 1 Internal and external system design interfaces

comprise the product due to their physically distributed or functionally integrative nature throughout the product. Note that integrative systems may span design interfaces with other systems not only due to physical proximity (e.g., the main shaft system of an aircraft engine spatially interfaces with the fan, compressor, diffuser, and turbine systems) but also due to functional dependency with other components (e.g., engine external plumbing design is defined by material transference needs of internal engine component requirements across all systems).

Based on these definitions we can argue that a ''hypothetically perfect'' modular system would be one whose components do not share design interfaces with components that belong to other systems. Moreover, such a perfect modular system could potentially exhibit an integral architecture internally depending on how its components share design interfaces among themselves. On the other hand, a ''hypothetically perfect'' integrative system would be one whose components are completely physically distributed throughout the product resulting in components that share interfaces with all the systems that comprise the product, even if it exhibits a modular architecture internally. Examples of (real) modular systems can be the engine block of a car, the microprocessor system of a computer, and the fan system of an aircraft engine. Examples of (real) integrative systems are the chassis of a car, the rotors, cases and airfoils of a compressor system, and the fuel system of an aircraft.

Distinguishing modular and integrative systems can play a critical role in planning the interactions of development teams. Some engineering teams may not be fully aware of or motivated to address the design interfaces and needs for system-level interaction. In this context, one of the hypotheses we test in this paper is that *team interactions with design teams that develop integrative systems are more likely to be predicted by design interfaces than are team interactions between design teams that develop modular systems*.

Product Development Organizations. From an organizational viewpoint, complex development projects usually involve the efforts of hundreds or even thousands of team members. A single team does not design the entire product at once (it is too complex). Rather, many teams develop the components, or systems, and work to integrate all of these components to create the final product [2,4,26,27]. We call *modular design teams* those which design modular systems while *integrative design teams* are those which design integrative systems.

An important challenge faced by development organizations is product integration [28]. Design teams face two important levels of integration during the development of complex products:

- Function-level integration takes place within each cross-functional design team when they have to coordinate efforts in order to design their respective components.
- System-level integration takes place across design teams in order to integrate the components (designed by each team) to assure the product works as an integrated whole.

This paper focuses on understanding the system-level integration efforts faced by both modular design teams and integrative design teams. In particular, we seek to better comprehend how modular and integrative design teams face barriers imposed by architectural system and organizational boundaries. We also want to investigate whether design teams must apply greater effort toward addressing certain types of design interfaces.

2 Our Research Approach

We summarize our research method in three steps. First, we capture the product architecture and identify modular and integrative systems. Second, we capture the development organization, and third, we compare the product architecture with the development organization to study the impact of system modularity on team interactions.

1. **Capture the product architecture**. By interviewing design experts who have a deep understanding of the architecture of the product, we identify how the product is decomposed into systems, and these further decomposed into components. We then ask them to identify the design interfaces between the components required for their functionality. Lastly, we identify modular and integrative systems by analyzing how design interfaces between components that belong to different systems are distributed throughout the product.
2. **Capture the development organization**. We first identify the design teams responsible to develop the product's components. We then survey key members of each team to capture the frequency and importance of the technical interactions between them, and thus assess the integration effort of the development organization.
3. **Compare the product architecture and the development organization**. We compare the design interfaces with the team interactions in order to study how the product architecture drives technical communications in the development organization. More specifically, we study how modular and integrative systems differ in the way they drive the system-level integration efforts of the development organization.

3 An Example

We apply our approach to the design of a large commercial aircraft engine (Pratt & Whitney PW4098). The project chosen was a complex design that exhibited explicit decomposition of the engine into systems, and these into components. Furthermore, the design project involved the development of both modular and integrative systems. The engine was decomposed into eight systems, and these further decomposed into 54 components. Figure 2 exhibits a cross-section diagram of the engine which highlights the eight systems that comprise the engine and Table 1 shows the decomposition of the engine into systems and the systems into components.

On the organizational side, the development team was organized according to the architecture of the product, with 54 cross-functional teams assigned to design the 54 engine components. In addition, there were six design teams responsible for system integration who were not responsible for the design of any of those components.

The direct mapping between the architecture of the engine and the organizational structure facilitated the implementation of our approach. For more details of the project description and the data collection process refer to Rowles [29].

Capturing the Product Architecture. The engine studied was decomposed into eight systems. Each of these systems was further decomposed into five to ten components each (see Table 1).

After documenting the general decomposition of the product, we proceeded to identify the design interfaces between the 54

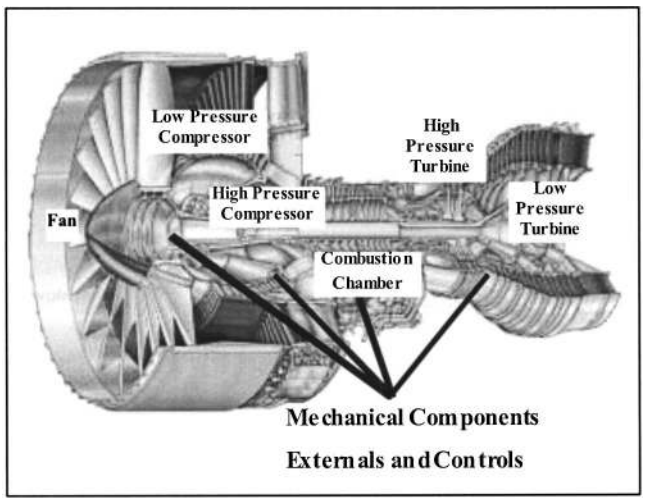

Fig. 2 Large commercial aircraft engine

components of the engine. Researchers in engineering design [30,31] have modeled functional requirements of product designs in terms of exchanges of energy, materials, and signals between elements. Building upon the methodology presented by Pimmler and Eppinger [3] we identified five types of design dependencies between the 54 components which comprise the engine. The five types are explained below, along with an example of each type. Note that some interface relationships are symmetric, while others are not.

- **Spatial** dependency indicates that physical adjacency is needed for alignment, orientation, serviceability, assembly, or weight. For example, the clearance between the tips of the high-pressure turbine (HPT) blades and the HPT blade outer air seals (BOAS) must be minimized for optimum performance. Note that in this example the dependency is symmetric as the blades spatially depend on the seals and vice versa.
- **Structural** dependency indicates the existence of a functional requirement for transferring design loads, forces or containment. For example, the HPT casing geometry is dependent on HPT blade loads and trajectory for containment capability, even though the parts are not adjacent to one another. However, blades are not structurally dependent on the casing (asymmetrical dependency).
- **Energy** dependency indicates a functional requirement related to transferring heat energy, vibration energy, electrical energy, or noise. For instance, when studying the energy dependency between fan blades and fan exit guide vanes we found that abatement of fan flutter and noise (produced by the fan blades) drives the airfoil and platform design of both the fan and the fan exit guide vane. Both airfoil counts and axial spacing are designed to optimize fan vibration and noise, which results in a symmetrical dependency.
- **Material** dependency indicates a functional requirement related to transferring air, oil, fuel, or water. For example, fuel nozzles are dependent on fuel from the fuel control, and air from the diffuser, but not vice-versa (asymmetrical dependencies).
- **Information** dependency indicates a functional requirement related to transferring signals or controls. For example, oil system controls such as the fuel/oil cooler valve and oil tank valve are dependent on oil temperature and pressure sensors (symmetrical dependencies).

Design dependencies were captured by interviewing design experts who had a deep understanding of the product architecture but were not directly involved in the design of the engine. Moreover, for each design dependency we asked design experts to assess the criticality of each dependency using a five-point scale as suggested by Pimmler and Eppinger [3]. Table 2 shows the scale used during the data collection. Note that the scale used captures not only *desired* and *required* dependencies (the positive ones) but also *undesired* and *detrimental* dependencies (the negative ones). An example of a negative dependency would be the undesired transfer of energy (vibration) from the LPT vanes to the LPT blades. However, the blades and vanes are positively co-dependent for proper inlet and exit gas conditions to achieve optimum aerodynamic efficiency. Hence, an important challenge these two design teams face is how to manage these conflicting design interfaces that are critical in opposing directions.

We mapped the design-interface data into a square (54×54) design interface matrix. (The design interface matrix can be described as a special form of design structure matrix (DSM). For a formal introduction to DSM refer to Steward [32] and Eppinger et al. [33].) The identically labeled rows and columns of this ma-

Table 1 Engine decomposition into systems and components

System	Components
Fan	• Fan containment case • Fan exit guide vanes & cases • Fan blades • Fan hubs • Fan stub shafts • Spinners & Nose caps • Fan blade platforms
Low-Pressure Compressor (LPC)	• LPC airfoils • LPC stator • LPC drum • LPC splitter • LPC liner • Bleed BOM • Intermediate case
High-Pressure Compressor (HPC)	• HPC blades • HPC inner shrouds & seals • Variable Vanes • HPC fixed stators/cases • HPC rubstrips & spacers • HPC disks & drums • Giggle tubes & blade locks
Combustion Chamber (CC)	• Burner • Diffuser • Tobi duct • Diffuser tubes • Fuel nozzle
High-Pressure Turbine (HPT)	• HPT blades • HPT 1V • HPT 2V • HPT rotor • HPT case & blade outer air seal BOAS
Low-Pressure Turbine (LPT)	• LPT shaft • LPT case • TEC • LPT vanes • LPT blades • LPT OAS/Ducts insulation
Mechanical Components	• Mainshaft • Gearbox • Breather valve • Oil pump • Internshaft seal • PMA • Mech. Components of oil system
Externals and Controls	• External Tubes • 2.5 Bleed Butterfly • Externals/Controls air systems • Externals/Controls oil system • External/Controls fuel/drain • Harness • Ignition • Electrical Controls • Mechanical Controls • Sensor Controls

Table 2 Design dependency criticality

Level of design dependency criticality	Description	Measure
Required	Dependency is necessary for functionality	+2
Desired	Dependency is beneficial, but not absolutely necessary for functionality	+1
Indifferent	Dependency does not affect functionality	0
Undesired	Dependency causes negative effects, but does not prevent functionality	−1
Detrimental	Dependency must be prevented to achieve functionality	−2

trix correspond to the 54 components of the engine, and their (original) sequencing follows the physical arrangement of the systems within the product (The 54 components are listed in the second column of Table 1.) Each off-diagonal cell of the design interface matrix contains a five-component vector corresponding to the five types of design dependencies which categorize each design interface.

For graphical simplicity, Fig. 3 displays only the binary form of the design interface matrix of the engine studied. The off-diagonal elements of the binary matrix are marked with an ''X'' when a given component has at least one design dependency with another component. Reading across a row corresponding to a particular component indicates the other components upon which it depends for functionality. The diagonal elements of the matrix are meaningless and are shown to separate the upper and lower triangular portions of the matrix.

The nonreciprocal (asymmetric) nature of some design dependencies was captured during the data collection as we asked the respondents how a given component depends on the other ones. The structural and material dependency examples described above show typical cases of asymmetrical dependencies found in the design interfaces of the aircraft engine. This results in a design interface matrix that is not completely symmetric with respect to the diagonal. Previous work in functional modeling [21,30,34,35,36] suggests the possibility of non-reciprocal design dependencies. Asymmetrical dependencies can be associated with some assembly functions such as hold and support. For instance, the frame of a gearbox may depend structurally on the weight of the gears to support, but not vice versa. Some design dependencies can be asymmetrical due to functional constraints. For instance, the diameter of the shaft of an electric motor might be constrained to the internal diameter of the bearings to be used (asymmetrical spatial dependency in which the shaft depends on the bearing as its dimensions are fixed), which in turn usually depends on the loads transmitted from the shaft (resulting in an asymmetric structural dependency).

For consistency between the physical product and the design interface matrix, we first sequence the design interface matrix following the spatial arrangements of the systems throughout the engine. More specifically, we sequence the matrix following theairflow through the engine (from left to right in Fig. 2). The important point about sequencing the design interface matrix is to cluster components that belong to the same system together so that system boundaries could be easily identified. At the compo-

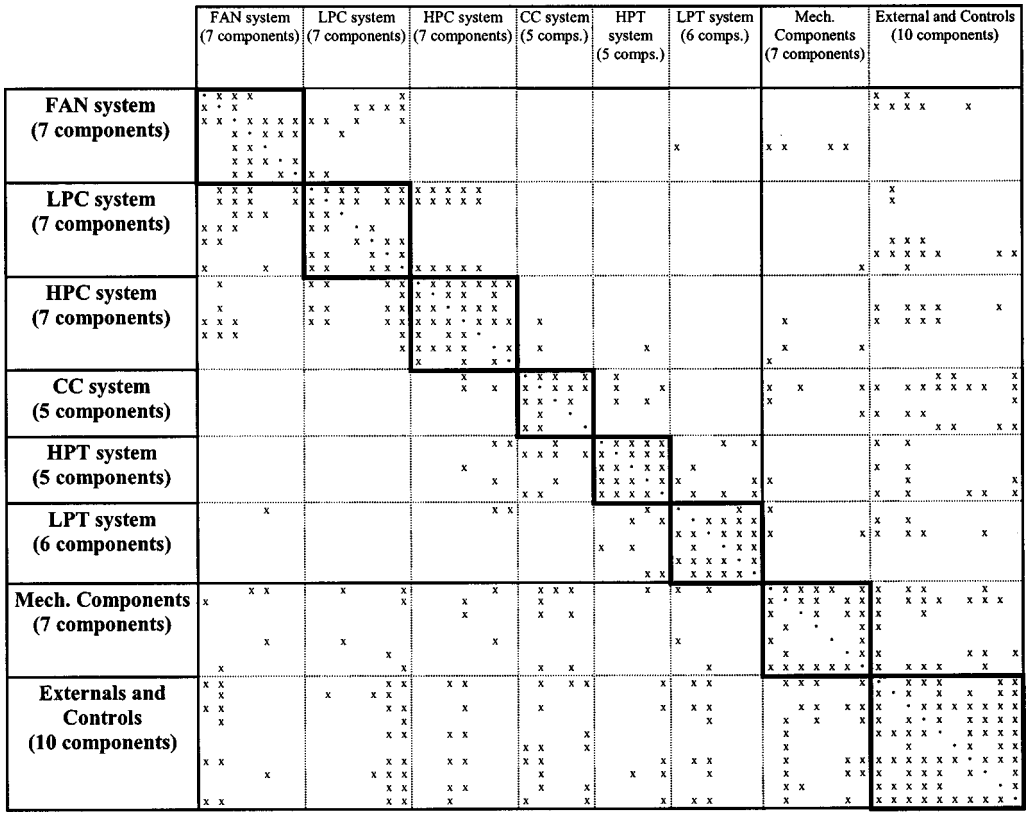

Fig. 3 Design interface matrix

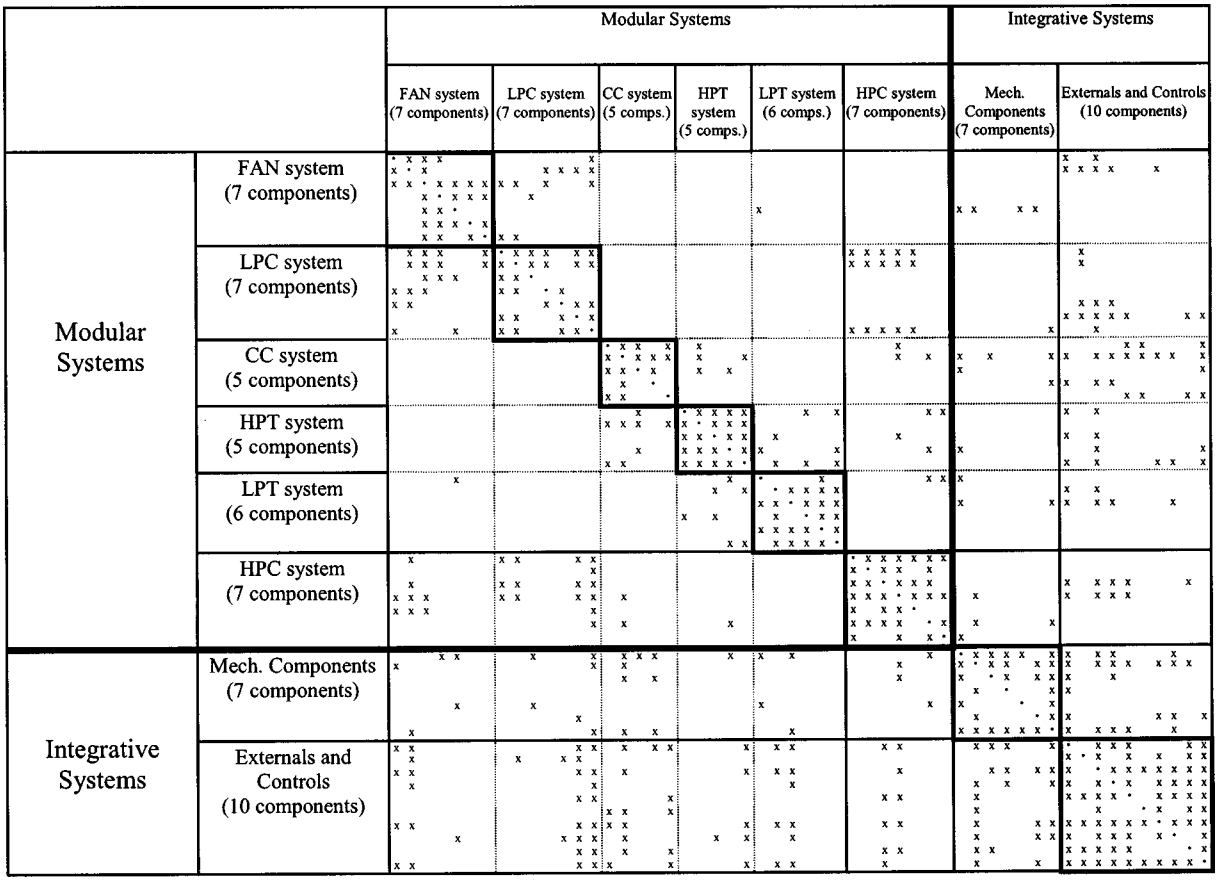

Fig. 4 Re-ordered design interface matrix

nent level (i.e., within each system), rows and columns were arbitrarily ordered. We highlight system boundaries with boxes along the diagonal of the design interface matrix. Marks inside the boxes represent design interfaces between components of the same system, whereas marks outside the boxes indicate interfaces between components of different systems (cross-boundary design interfaces). Light boxes throughout the matrix enclose the cross-boundary design interfaces between any two given systems.

Identifying Modular and Integrative Systems. As mentioned above, modularity at the system level can be categorized from an internal viewpoint by studying the design interfaces within the system, or from an external perspective by studying the design interfaces across systems. In this paper we are interested in the latter categorization.

We define modular systems as those systems whose cross-boundary design interfaces are concentrated among a few other systems (usually spatially contiguous systems). On the other hand, we define integrative systems as those systems whose design interfaces are scattered among components in (almost) all the systems that comprise the product. An example of a modular system is the low-pressure compressor (LPC) whose components share design interfaces with components of the fan and high-pressure compressor (HPC) systems located adjacent to the LPC. An example of an integrative system is the main shaft system (which is a subsystem of the mechanical components systems) which has design interfaces with components of all other systems in the engine.

Our approach to identify modular and integrative systems is facilitated when the design interface matrix is sequenced to reveal the system boundaries. In order to do so, we must sequence components that belong to the same system together. Then, we suggest re-ordering the matrix so that the systems sharing a larger number of design interfaces across systems are sequenced last so that the matrix can be divided in two sections, the first columns and rows corresponding to modular systems and the last ones assigned to integrative systems. In our example, the systems with more cross-boundary design interfaces were the externals and controls, mechanical components, and high-pressure compressor systems. Figure 4 shows the re-ordered design interface matrix exhibiting the high-pressure compressor system sequenced third-last in the matrix.

By visually inspecting the design interface matrix we observe that the first five systems are those in which cross-boundary design interfaces are primarily clustered among adjacent systems (modular systems). On the other hand, it is clear that the externals and controls system has a significantly large number of cross-boundary design interfaces well distributed among the other systems (integrative system). The question becomes: where do we draw the line to separate modular systems from integrative systems? Are the mechanical components and the high-pressure compressor modular or integrative systems?

In order to answer these questions we compare the frequency distribution of cross-boundary design interfaces of the mechanical components and the high-pressure compressor systems against the externals and controls system. Hence, we assume that the externals and controls system is an integrative system and we use it as our basis of comparison to determine whether the two other systems in question are modular or integrative. In general, we should statistically test whether the difference in the proportions of cross-boundary design interfaces between any two given systems is large enough to be attributed to a difference in system modularity rather than just random variation in the data. Hence, when comparing a modular system and an integrative system's frequency distribution of cross-boundaries design interfaces the difference should be statistically significant.

Figures 5, 6 and 7 show the number of cross-boundary design interfaces of the externals and controls system, the mechanical components system and the high-pressure compressor (HPC), respectively. By visually inspecting these three frequency distribu tions we can observe that the first two frequency distributions are very similar to each other but different than the third one. This visual inspection allows us to conjecture that *both externals and controls system and mechanical components system are integrative systems whereas the high-pressure compressor is a modular system*. To formally test that the difference in frequency distributions of cross-boundary design interfaces is due to system modularity rather than just random variation in the data we perform a chi-square (χ^2) test analysis on these frequency distributions.

Chi-square (χ^2) tests are commonly used in analysis of frequencies. Analysis of frequencies are characterized by (1) data consisting of counts or frequencies, and (2) implied or stated hypotheses which determine the expected frequencies with which we can compare the actual frequencies[2] [37]. In this analysis we assume that the cells in each of the matrices shown in this paper are statistically independent. Using log-linear network analysis Sosa [38] shows that such an assumption does not affect the results of the analysis presented in this paper.

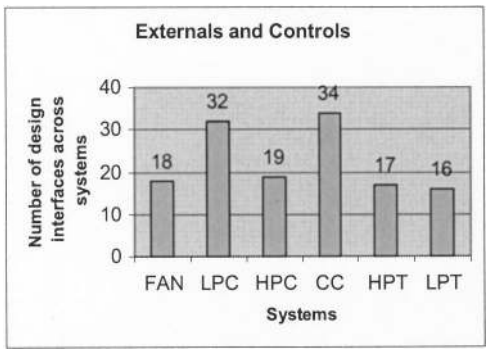

Fig. 5 Frequency distribution of design interfaces of the externals and controls system

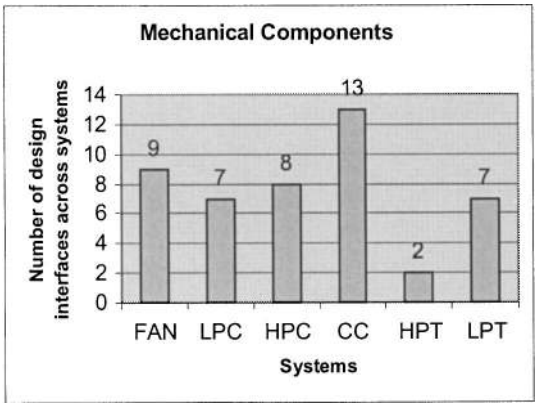

Fig. 6 Frequency distribution of design interfaces of the mechanical components system

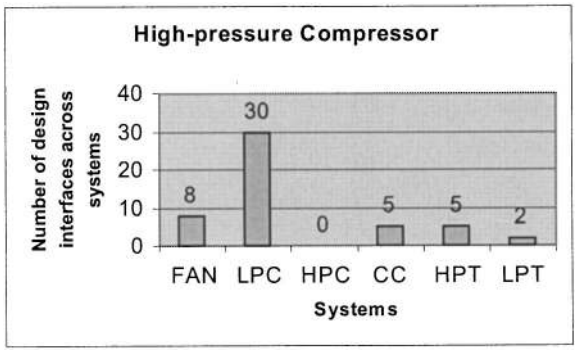

Fig. 7 Frequency distribution of cross-system design interfaces of the high-pressure compressor (HPC) system

Table 3 Chi-square test results—Comparing externals and controls system with high-pressure compressor system

System	Expected fraction of design interfaces based on Ext/Controls	Expected number of design interfaces of HPC†	Actual number of design interfaces of HPC	χ^2
FAN	11.04%	6.403	8	0.398
LPC	19.63%	11.385	30	30.436
CC	20.86%	12.099	5	4.165
HPT	10.43%	6.049	5	0.182
LPT	9.82%	5.696	2	2.398
MC	28.22%	16.368	8	4.278
Total	100.00%	58.000	58	**41.857**

†The fraction of design interfaces between the external and controls system and the fan system is 18 out of a total of 163 design interfaces, that is, 11.04%. Hence, the expected number of design interfaces between the HPC system and the fan system, under the null hypothesis, is the 11.04% of a total of 58 design interfaces, that is, 6.403. The rest of the expected values are determined in a similar way.

Table 3 shows the results of the chi-square test performed to test the null hypothesis that "*the frequency distribution of design interfaces of the externals and controls system is statistically equivalent to the frequency distribution of the high-pressure compressor system*". The test resulted in a χ^2 equal to 41.857 which is greater than the critical value of 11.070 (for α=0.05 and five degrees of freedom). This result allows us to reject the null hypothesis stated above. The expected values shown in Table 3 are determined based on the frequency distribution of the design interfaces of the externals and controls system. That is, according to the null hypothesis stated above, the expected fraction of design interfaces between the high-pressure compressor and the other systems is equal to the fraction of design interfaces between the externals and controls and the other systems.[3] The actual values are the number of design interfaces of the high-pressure compressor system with each of the other systems. Similar results were found when comparing the frequency distribution of cross-system

[2]Chi-square (χ^2) statistic is computed as [Observed value−Expected value]^2/Expected value

[3] Fraction design interfaces $_{\text{external and controls, system } i}$

$$= \frac{\text{Number of design interfaces}_{\text{external and controls, system } i}}{\Sigma_{\text{all other systems}} \text{ number of design interfaces}_{\text{external and controls, system } k}}$$

Table 4 Chi-square test results—Comparing externals and controls system with mechanical components system

System	Expected fraction of design interfaces based on Ext/Controls	Expected number of design interfaces of Mech. Comps.†	Actual number of design interfaces of Mech. Comps.	χ^2
FAN	13.24%	6.088	9	1.393
LPC	23.53%	10.824	7	1.351
HPC	13.97%	6.426	8	0.385
CC	25.00%	11.500	13	0.196
HPT	12.50%	5.750	2	2.446
LPT	11.76%	5.412	7	0.466
Total	100.00%	46.000	46	**6.237**

†The fraction of design interfaces between the external and controls system and the fan system is 18 out of a total of 136 design interfaces, that is, 13.24%. Hence, the expected number of design interfaces between the mechanical components system and the fan system, under the null hypothesis, is the 13.24% of a total of 46 design interfaces, that is, 6.088. The rest of the expected values are determined in a similar way.

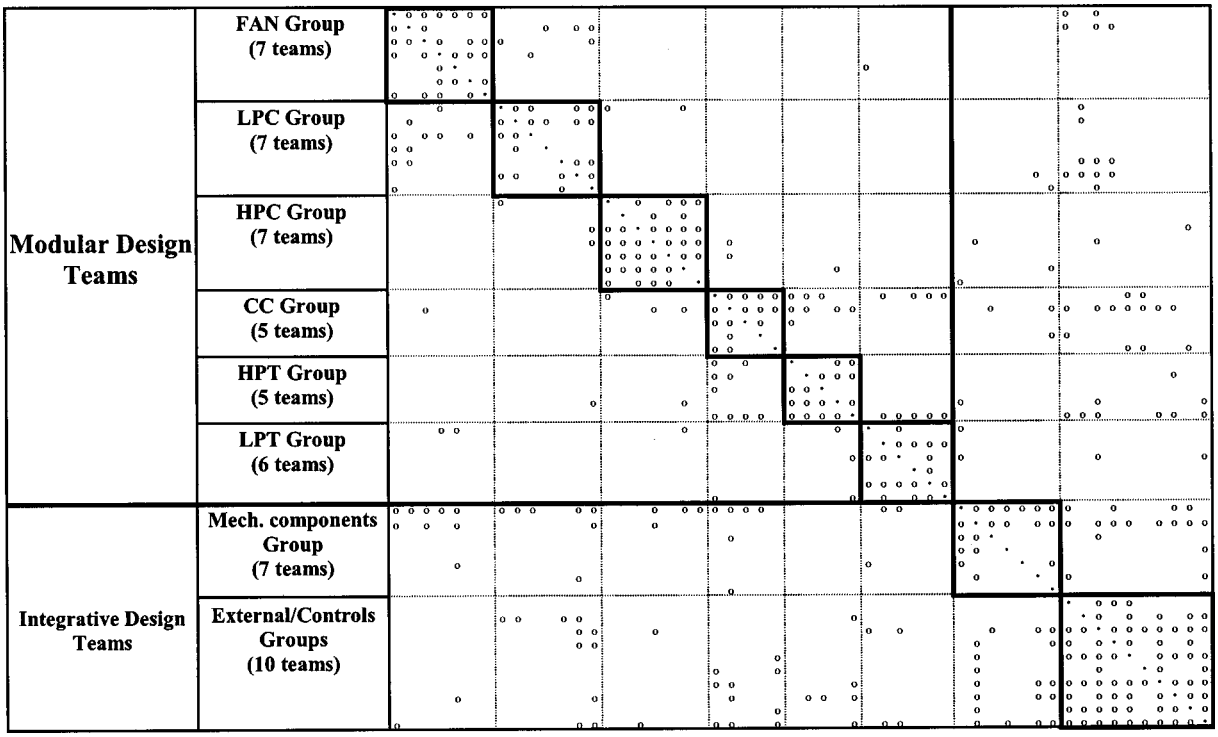

Fig. 8 Team interaction matrix

design interfaces of the externals and controls system and the other five modular systems. (For more details refer to Sosa [38].)

We found that the design interfaces of the externals and controls system and the mechanical components system are similarly distributed among the first six modular systems. Table 4 shows the results of the chi-square test performed to test the null hypothesis that "*the frequency distribution of design interfaces of the mechanical components system and the externals and controls system are statistically equivalent*". Note that the expected values are determined following the same method used in Table 3. The test resulted in χ^2 equal to 6.237 which is smaller than the critical value of 11.070 (for $\alpha=0.05$ and five degrees of freedom). We cannot, therefore, reject the null hypothesis of no difference in the frequency distribution of design interfaces for the two systems.

These results confirm our initial conjecture regarding systems modularity, that is, the mechanical components and externals and controls are integrative systems while the high-pressure compressor and other five systems are modular systems.

Capturing the Development Organization. The organization responsible for the development of the aircraft engine was divided into sixty design teams. Fifty-four design teams were grouped into eight system design groups according to the architecture of the engine described above. Each of those teams was responsible for developing one of the 54 components of the engine. The remaining six design teams were system integration teams, which had no specific hardware associated with them and whose main responsibility was to assure that the engine worked as a whole. Examples of the system integration teams are the rotordynamics team and the secondary flow team. The six system integration teams were excluded from the analysis presented in this paper because they did not design either modular or integrative systems. Moreover, Sosa [38] shows that the presence of these teams did not influence the results presented in this paper.

We capture the system-level integration effort of the organization (both within groups and across groups) by documenting the technical interaction between the design teams involved in the development process. We surveyed at least two key members from each design team and asked them to rate the criticality and frequency of their interactions with each of the other teams during the detailed design phase of the engine's development project. This method is similar to the approach illustrated by Eppinger [4].

We organize the team-interaction data in a square (54×54) team-interaction matrix (see Fig. 8). The labels of the rows and columns of this matrix contain the names of each of the design teams. For comparison, we must sequence the matrix to match the sequence of the design interface matrix. The binary team interaction matrix shows off-diagonal cells marked with an "O" to indicate each non-zero team interaction revealed. Reading across a particular row indicates with which other teams the surveyed team interacted.

It is important to emphasize that we documented coordination-type communications only. Morelli et al. [39] identified three types of technical communication in development organizations: coordination-type, knowledge-type and inspiration-type. Since we were interested in capturing the integration effort of the development organization we focused our surveys upon the coordination-type interactions which took place during the design process to address task-related issues. Since we captured the interactions reported from the respondent's point of view, the matrix exhibits an asymmetric structure with respect to its diagonal.

As shown in Fig. 8, associated with the six modular systems are corresponding six groups of design teams. Similarly, the two integrative systems have their two corresponding groups of design teams. The highlighted boxes along the diagonal indicate the organizational boundaries between the eight design groups. Marks inside the boxes indicate within-boundary team interactions, which we associate to within-boundary system-level integration effort. On the other hand, marks outside the boxes indicate cross-boundary team interactions, which we associate to cross-boundary system-level integration effort.

Comparing the Product Architecture with the Development Organization. The one-to-one assignment of the 54 components to the 54 design teams allows the direct comparison of the design interface matrix with the team interaction matrix. Figure 9

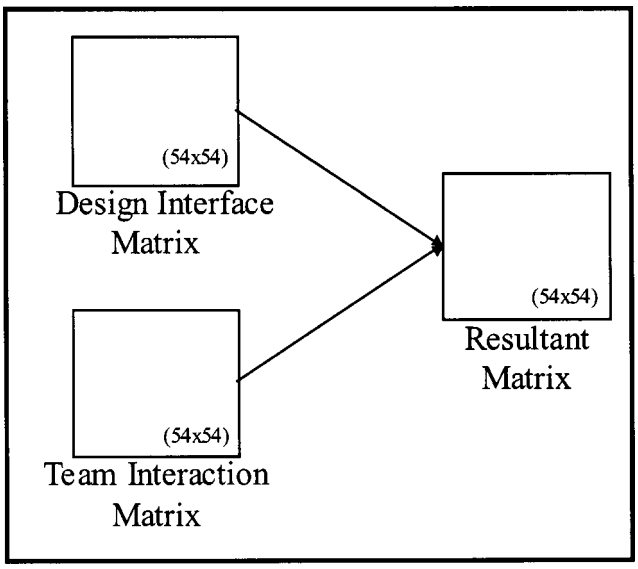

Fig. 9 Comparing design interfaces and team interactions

Team Interactions	NO (2439)	**X** (220)	(2219)
	YES (423)	# (349)	**O** (74)
		YES (569)	NO (2293)
		Design Interfaces	

Fig. 11 Results of the resultant matrix

shows how, by merging the design interface matrix with the team interaction matrix, we obtain the resultant matrix. Note that in order to generate a valid resultant matrix both design interface matrix and team interaction matrix have to be sequenced identically. In our example, we use the original design interface matrix shown in Fig. 3.

The resultant matrix is exhibited in Fig. 10 and the four possible outcomes for each cell on that matrix are shown in Fig. 11. The number of cases in each category is given in parentheses. As expected, the majority of the cells (90% of the cells) are the cases where known design interfaces were matched by team interactions (349 "#" cells), and the cases with no design interfaces with no corresponding reported team interactions (2219 blank cells). The unexpected cases accounted for 10% of the cells; those were the cases when known design interfaces were not matched by team interactions (220 "X" cells), and the cases when reported team interactions were not predicted by design interfaces (74 "O" cells).

Sosa [38] presents detailed analyses to test some of the hypotheses that explain the existence of the unexpected cases ("X" and "O" cells of Fig. 11). The analysis presented in this paper focuses on the effects of system modularity on the communication patterns of design teams. By analyzing the resultant matrix, we investigate the effects of architectural system and organizational boundaries, and the effects due to the types of design interfaces on the development of modular and integrative systems.

4 Analysis

Effects of System Boundaries. System boundaries are the result of product decomposition and are defined by the way components are grouped into systems. Similarly, organizational boundaries are defined by the way the design teams are grouped into system teams. Boundaries are highlighted in the resultant

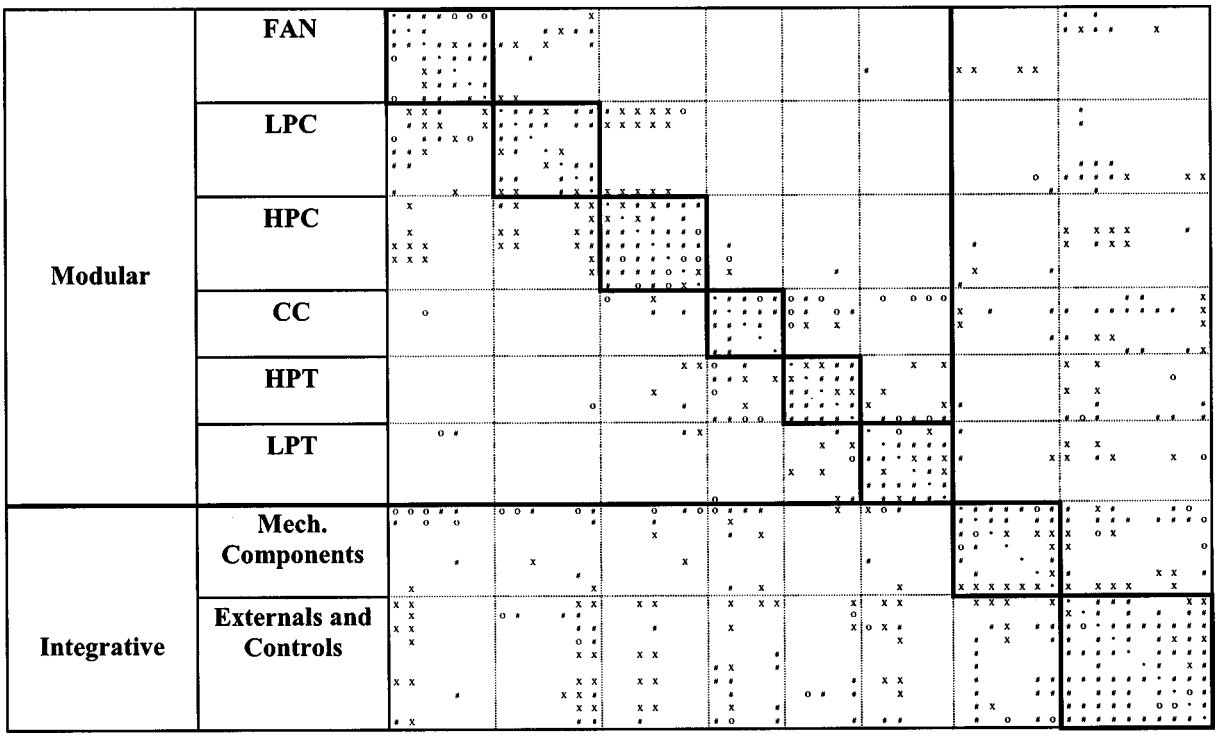

Fig. 10 Resultant matrix. Legend: X: Design interface with no team interaction; O: Team interaction with no design interface; #: Both design interface and team interaction; *: Diagonal elements (meaningless).

Table 5 Chi-square test results—effects of system boundaries†

	Total	Expected cases of predicted team interactions	Expected cases of unpredicted team interactions	Actual cases of predicted team interactions	Actual cases of unpredicted team interactions	χ^2 of predicted team interactions	χ^2 of unpredicted team interactions
Design interfaces **within** system boundaries (Modular systems)	124	111.5 (89.9%)	12.5 (10.1%)	110 (88.7%)	14 (11.3%)	0.020	0.175
Design interfaces **within** system boundaries (Integrative systems)	84	75.5 (89.9%)	8.5 (10.1%)	77 (91.7%)	7 (8.3%)	0.029	0.259
Total	208	187.0	21.0	187	21	**0.049**	**0.434**
Design interfaces **across** system boundaries (Modular systems)	65	49.0 (75.3%)	16.0 (24.7%)	40 (61.5%)	25 (38.5%)	1.645	5.029
Design interfaces **across** system boundaries (Integrative systems)	150	113.0 (75.3%)	37.0 (24.7%)	122 (81.3%)	28 (18.7%)	0.713	2.179
Total	215	162.0	53.0	162	53	**2.358**	**7.208**

†Expected values are determined with the pooled data which indicates that 89.9% of the 208 design interfaces within system boundaries predict team interactions while 75.3% of the 215 design interfaces across system boundaries predict team interactions. $\chi^2_{\text{within-boundaries}} = 0.483$, $\chi^2_{\text{across-boundaries}} = 9.566$, $\chi^2_{\text{critical}(0.95,1)} = 3.841$.

matrix by the boxes along the diagonal. Cells inside the boxes represent the interactions within (organizational and system) boundaries while cells outside the boxes represent the interactions across boundaries. System boundaries impose architectural knowledge barriers which inhibit design experts' understanding of certain design interfaces. This results in some team interactions that are not predicted by design interfaces. Indeed, Sosa [38] shows that team interactions within system boundaries are more likely to be predicted than team interactions across system boundaries. In this paper we are interested in testing the hypothesis that *cross-boundary team interactions with integrative design teams are more likely to be predicted by design interfaces than are cross-boundary team interactions between modular design teams.*

We use the sample formed by the 423 team interactions ("#" and "O" cells from Fig. 11) to test whether the way design interfaces predict team interactions is statistically equivalent for modular and integrative design teams. The chi-square tests shown in Table 5 resulted in a χ^2 equal to 0.483 for the cases within system boundaries, which is well below the critical value of 3.841 (for $\alpha = 0.05$ and one degree of freedom). On the other hand, for the team interactions across system boundaries the χ^2 equaled 9.566, which is well above the critical value.

These results do not allow us to reject the null hypothesis that *the portion of unknown design interfaces within system boundaries is equivalent for both modular and integrative systems.*

Table 6 Chi-square test results—effects of organizational boundaries†

	Total	Expected cases of design interfaces matched by team interactions	Expected cases of design interfaces not matched by team interactions	Actual cases of design interfaces matched by team interactions	Actual cases of design interfaces not matched by team interactions	χ^2 of cases of design interfaces matched by team interactions	χ^2 of cases of design interfaces not matched by team interactions
Design interfaces **within** organizational boundaries (Modular systems)	137	110.9 (81.0%)	26.1 (19.0%)	110 (80.3%)	27 (19.7%)	0.007	0.031
Design interfaces **within** organizational boundaries (Integrative systems)	94	76.1 (81.0%)	17.9 (19.0%)	77 (81.9%)	17 (18.1%)	0.011	0.046
Total	231	187.0	44.0	187	44	**0.018**	**0.077**
Design interfaces **across** organizational boundaries (Modular systems)	110	52.7 (47.9%)	57.3 (52.1%)	40 (36.4%)	70 (63.6%)	3.070	2.826
Design interfaces **across** organizational boundaries (Integrative systems)	228	109.3 (47.9%)	118.7 (52.1%)	122 (53.5%)	106 (46.5%)	1.481	1.363
Total	338	162.0	176.0	162	176	**4.551**	**4.189**

†Expected values are determined with the pooled data which indicates that 81.0% of the 231 design interfaces within organizational boundaries are matched by team interactions while 47.9% of the 338 design interfaces across organizational boundaries are matched by team interactions. $\chi^2_{\text{within-boundaries}} = 0.095$, $\chi^2_{\text{across-boundaries}} = 8.740$, $\chi^2_{\text{critical}(0.95,1)} = 3.841$.

Table 7 Chi-square test results—effects due to type of design interfaces†

	Total	Expected cases of design interfaces matched by team interactions	Expected cases of design interfaces not matched by team interactions	Actual cases of design interfaces matched by team interactions	Actual cases of design interfaces not matched by team interactions	χ^2 of cases of design interfaces matched by team interactions	χ^2 of cases of design interfaces not matched by team interactions
Spatial-type design interfaces (Modular systems)	24	11.7 (48.8%)	12.3 (51.2%)	15 (62.5%)	9 (37.5%)	0.926	0.882
Spatial-type design interfaces (Integrative systems)	17	8.3 (48.8%)	8.7 (51.2%)	5 (29.4%)	12 (70.6%)	1.307	1.245
Total	41	20.0	21.0	20	21	**2.233**	**2.127**
Transfer-type design interfaces (Modular systems)	41	13.2 (32.1%)	27.8 (67.9%)	8 (19.5%)	33 (80.5%)	2.024	0.957
Transfer-type design interfaces (Integrative systems)	40	12.8 (32.1%)	27.2 (67.9%)	18 (45.0%)	22 (55.0%)	2.074	0.980
Total	81	26.0	55.0	26	55	**4.098**	**1.937**

†Expected values are determined with the pooled data which indicates that 48.8% of the 41 spatial-type design interfaces are matched by team interactions while 32.1% of the 81 transfer-type design interfaces are matched by team interactions. $\chi^2_{\text{spatial-type}}=4.360$, $\chi^2_{\text{transfer-type}}=6.035$, $\chi^2_{\text{critical}(0.95,1)}=3.841$.

However, for interactions across system boundaries we were able to reject the corresponding null hypothesis. Indeed, Table 5 shows that the portion of unknown design interfaces across modular systems is statistically significant larger than for integrative systems. Note that unknown design interfaces are characterized by unpredicted team interactions (''O'' cells of Fig. 10). Specifically, 38.5% of the modular design team interactions across system boundaries were not predicted by design interfaces whereas only 18.7% of the integrative design team interactions across system boundaries were not predicted by design interfaces.

Effects of Organizational Boundaries. Organizational boundaries impose communication barriers which inhibit design teams interactions [40,41]. However, given the distributed nature of integrative systems we expect integrative design teams to be less affected by this type of barrier. Since the design interfaces of integrative systems are distributed throughout the engine, we anticipate integrative design teams to be more accustomed to crossing organizational boundaries than are modular design teams. Indeed, we hypothesize that *integrative design teams handle a statistically significant larger portion of design interfaces across organizational boundaries than do modular design teams*.

We use the 569 cases in which design interfaces were identified (''#'' and ''X'' cells of Fig. 11) to test whether both modular and integrative design teams handle a statistically equivalent portion of design interfaces. Note that we are implicitly assuming that the coordination-type communication reflects handling the corresponding design interface.

The chi-square tests shown in Table 6 resulted in a χ^2 equal to 0.095 for the cases *within* boundaries, which is well below the critical value of 3.841 (for $\alpha=0.05$ and one degree of freedom). These results do not allow us to reject the null hypothesis that *design interfaces within organizational boundaries are equally handled by teams that design modular systems as by teams that design integrative systems*. On the other hand, for the cases *across* organizational boundaries χ^2 equaled 8.740, (well above the critical value) which allows us to reject the corresponding null hypotheses. Indeed, Table 6 shows that integrative design teams do handle a statistically significant larger portion of design interfaces than do modular design teams. Specifically, integrative design teams matched 53.5% of the cross-system design interfaces while modular design teams matched 36.4% of their cross-system design interfaces.

Effects of Types of Design Interface. According to the type of design dependency, we classify design interfaces into two major categories:

- **Spatial-type** design interfaces, which involve spatial dependencies only.
- **Transfer-type** design interfaces, which involve structural and/or energy and/or material dependencies. (Information dependencies are not included in this analysis because they are not present in the modular systems.)

Henderson and Clark [1] refer to ''communication filters'' as the mechanism for screening the most crucial information. Adapting this concept to our context, we expect some design teams to handle a larger proportion of some types of design interfaces than other ones. Hence, we want to investigate whether there is a difference in the way modular and integrative design teams handle these two types of design interfaces.

A subset of 122 design interfaces, which could be categorized as either spatial-type or transfer-type, were used to answer this question (see Table 7). The chi-square test resulted in a χ^2 equal to 4.360 for spatial-type design interfaces, and a χ^2 equal to 6.035 for transfer-type design interfaces. Both are greater than the critical value of 3.841 (for $\alpha=0.05$ and one degree of freedom). These results allow us to reject the null hypothesis that *spatial-type design interfaces and transfer-type design interfaces are equally handled by modular design teams and integrative design teams*.

These results allow us to state that *teams designing modular systems have a stronger preference, ability, or willingness, to deal with spatial-type design interfaces than do teams designing integrative systems*. As shown in Table 7, 62.5% of the modular spatial-type design interfaces analyzed were matched by team interactions, while 29.4% of the integrative spatial-type design interfaces were matched by team interactions. Similarly, we found that *teams that design integrative systems are more likely or willing to deal with transfer-type design interfaces than modular design teams*. Table 7 shows that 45.0% of the integrative transfer-type design interfaces analyzed were matched by team interactions, while only 19.5% of the modular transfer-type design interfaces were matched by team interactions.

5 Discussion

By studying the coupling of the architecture of an aircraft engine and the development organization that designed it we have gained important insights about the difference between designing modular versus integrative systems. While we cannot claim the generality of the results before completing similar studies in other types of products in different industries, we expect to obtain analogous findings in other projects developing complex systems and where the development teams are organized according to the product architecture. The analysis presented in this paper provides three important results (subject to further verification) which are summarized here:

1. When analyzing the effects of system boundaries, we found a statistically significant larger proportion of unpredicted team interactions (''O'' cells of Fig. 10) associated with modular systems. Since unpredicted team interactions represent unrecognized design interfaces, we conclude that design interfaces across modular systems are more difficult for design experts to recognize than interfaces with integrative systems. In the context of the project studied, one of the practical outcomes of this project was the creation of a new design team dedicated to handle the critical cross boundary design interfaces that were unattended before. This newly created design team was implemented in the design process of the subsequent engine to be developed.
2. The statistically significant differences in the way integrative design teams handle design interfaces across boundaries suggest that these teams are more effective at overcoming the barriers imposed by organizational boundaries. The distributed nature of the integrative systems forces these design teams to overcome organizational barriers in order to handle design interfaces with all the systems.
3. The existence of various types of design interfaces and the statistically significant difference in the way they were handled by modular and integrative design teams provide empirical support to the notion of ''communication filters'' introduced by Henderson and Clark [1]. We found that spatial-type design interfaces are largely addressed in the design of modular systems while transfer-type design interfaces are more likely to be handled in the design of integrative systems.

These results suggest that managers should pay particular attention to identifying modular and integrative systems so that the critical design interfaces between modular systems are identified. Since modular systems may not be perfectly modular, managers should suspect the existence of cross-boundary design interfaces. It is important to understand that greater effort is needed to identify and handle these design interfaces due to the effects of system and organizational boundaries. Hence, managers should put special emphasis on identifying critical cross-boundary design interfaces occurring between modular systems to facilitate technical interactions between the corresponding design teams. On the other hand, design teams that develop integrative systems appear to be less vulnerable to organizational and system boundaries due to the physically distributed nature of the systems they design. Managers should also identify critical design interfaces of unexpected type (i.e., transfer-type for modular systems, and spatial-type for integrative systems) in order to facilitate the technical interactions associated to those design interfaces.

We recommend the method to capture the product architecture illustrated in this paper in order to identify critical design dependencies during the planning stage of a development project. This method can be summarized in five steps:

1. **Document product decomposition**. By interviewing design experts capture the decomposition of the product into systems, sub-systems, and components.
2. **Identify design dependencies**. Capture the various types of design dependencies (spatial, structural, material, energy, and information) among the product's components and document the level of criticality of each of them (required, desired, indifferent, undesired, and detrimental).
3. **Construct design interface matrix**. Construct a design dependency matrix for each dependency type. Each cell of these matrices should contain the level of criticality of the corresponding design dependency. Sequence the matrices by clustering components of the same system together. Then, aggregate all design dependency matrices into one binary matrix for compact representation (similar to the design interface matrix constructed in our example, Fig. 3). Outline the system boundaries so that cross-boundary design interfaces are easily determined.
4. **Identify modular and integrative systems**. By visually inspecting the design interface matrix identify the systems with larger proportion of cross-boundary design interfaces. If necessary, re-sequence the matrix (at the system level) so that potentially integrative systems occupy the last rows and columns of the matrix. Clearly separate modular versus integrative systems. Perform chi-square analysis of frequencies for the cases where it is not clear whether it is a modular or integrative system.
5. **Highlight critical cross boundary design interfaces**. By examining the aggregated level of criticality of each design interface highly critical design interfaces across systems can be identified. Additionally, determine the type of dependency (spatial-type or transfer-type) of those critical design interfaces. This step is particularly important to be carried out between modular systems, as we learned that technical interactions across their design teams are more difficult to manage.

In general, however we believe engineering managers should be able to apply the insights provided in this paper without carrying out any detailed analysis. They only need to know which systems are modular and which are integrative ones, based on their expertise and understanding of the types of interfaces each system has. Then, this paper suggests key factors that they need to ''watch out for'' while planning and managing the process.

It is important to mention that a significant portion of design interfaces were not addressed by team interactions (the ''X'' cells in the resultant matrix), yet components involved in these interfaces still got designed and the entire system worked as a whole. Some of the effects that explain the existence of these ''unexpected'' cases are (refer to Sosa [38] for details):

- **Design escapes**. These design interfaces, indeed, do not get addressed in the design phase, but they are addressed in the test and refinement phase resulting on unplanned design iterations. In some cases design escapes are found during customer use as well, which is expensive in terms of product life cycle cost and customer satisfaction.
- **Noncritical design interfaces**. Some design interfaces are desired, or undesired, but not required, nor detrimental, for component functionality. These noncritical interfaces are less likely to be addressed by design teams. In many of these cases design teams assume the state of the interface instead of interacting directly with the corresponding design team.
- **Carry-over effect**. Some design interfaces did not change with respect to previous engine generation and design teams could have known about it, therefore no team interaction was needed. We observed few instances of this effect, but not enough to be statistically significant.
- **Design interface standardization**. Some design interfaces may be considered standard for certain product designs and teams know about it. In our example, and probably similarly

for other such complex products, we observed very little of this effect due to the multi-dependent nature of the design interfaces.

- **Predefined design interfaces**. Some design interfaces are specified within interface control documents, boundary conditions, or product design requirements. Design teams assume that the other team involved in such interface will meet the predefined criteria, or will initiate an interaction otherwise.
- **Indirect team interactions**. Some design teams do not interact directly, however technical information related to their design interfaces flows indirectly via other design teams with which they interact.
- **Measurement errors**. Lastly, there are two kinds of measurement errors that could have happened during the data collection. First, there is the possibility that some of the design experts we interviewed might have erroneously reported to us certain design interfaces that are really not present in the current engine design. Hence, no team interactions were required in these cases. Second, some team interactions that indeed took place during the design process to address the corresponding design interfaces might not have been reported during the survey. This could have happened because the respondent was not aware of other team members' interactions or simply forgot to report them.

6 Conclusions and Future Work

The research method presented in this paper provides a useful approach to investigate the coupling of the product architecture and the development organization. Our research method can be summarized in three steps: 1) capture the product architecture by documenting design interfaces, 2) capture the development organization by documenting team interactions, and 3) couple the product architecture with the development organization by comparing design interfaces with team interactions. This method is particularly applicable to projects where the architecture of the product is well understood and the development team is organized around the product architecture.

In addition to the research method itself, this paper makes three important contributions:

- First, we extend the product architecture literature by introducing the concepts of modular and integrative systems based on the way components share design interfaces with components across systems (rather than within systems). Indeed, we illustrate how to formally identify modular and integrative systems by analyzing the frequency distribution of design interfaces across system boundaries.
- Second, we propose a structured method to capture the architecture of complex products whose decomposition is known in advance. This method can be used by managers wishing to predict technical team interactions that would require special managerial attention during the execution of a development project.
- Third, we enhance our understanding of the difference between designing modular versus integrative systems by studying the moderating effects of systems modularity. We present limited empirical evidence showing that the effects of system and organizational boundaries, and the effects due to the type of design dependencies, are different for modular systems versus integrative systems. These results highlight the importance of distinguishing modular and integrative systems during the planning stage of a development effort so that critical technical interactions can be managed more effectively during the project execution stage.

The type of analysis illustrated here may outline the study of other issues related to the coupling of product architecture and organizational structure. A challenge for future research work is to extend this method to explore the evolution over time of both design interfaces and team interactions for several generations in a product family.

This study takes advantage of the direct mapping of the product architecture and the development organization in the project studied. What if this were not the case? Which types of barriers are most severe (organizational or system barriers)? Is an organizational design that mirrors the architecture of the product an optimum one? Studying various mappings of product architectures to development organizations may help answer these research questions.

Acknowledgments

Funding of this research has been provided by the MIT Center for Innovation in Product Development. We appreciate the assistance of the engineers from Pratt & Whitney Aircraft during the development of the example presented in this paper. Finally, we would like to thank three anonymous reviewers for their insightful comments on a previous version of this paper.

References

[1] Henderson, R., and Clark, K., 1990, "Architectural Innovation: The Reconfiguration of Existing Product Technologies and the Failure of Established Firms," Adm. Sci. Q., **35**(1), pp. 9–30.
[2] Alexander, C., 1964, *Notes on the Synthesis of Form*, Harvard University Press, Cambridge, MA.
[3] Pimmler, T. U., and Eppinger, S. D., 1994, "Integration Analysis of Product Decompositions," *Proceedings of the ASME Design Theory and Methodology Conference (DTM' 94)*, DE-Vol 68, pp. 343–351.
[4] Eppinger, S. D., 1997, "A Planning Method for Integration of Large-Scale Engineering Systems," *International Conference on Engineering Design (ICED 97 Tampere)*, August.
[5] Ulrich, K. T., 1995, "The Role of Product Architecture in the Manufacturing Firm," Res. Policy, **24**, pp. 583–607.
[6] Newcomb, P. J., Bras, B., and Rosen, D. W., 1998, "Implications of Modularity on Product Design for the Life Cycle," ASME J. Mech. Des., **120**, pp. 483–490.
[7] Yu, J. S., Gonzalez-Zugasti, J. P., and Otto, K. N., 1999, "Product Architecture Definition Based Upon Customer Demands," ASME J. Mech. Des., **121**, pp. 329–335.
[8] Gonzalez-Zugasti, J. P., Otto, K. N., and Baker, J. D., 2000, "A Method for Architecting Product Platforms," Res. Eng. Des., **12**, pp. 61–72.
[9] Robertson, D., and Ulrich, K., 1998, "Planning for Product Platforms," Sloan Manage. Rev., **39**(4), Summer, pp. 19–31.
[10] Lee, H. L., 1996, "Effective Inventory and Service Management through Product and Process Re-Design," Oper. Res., **44**(1), pp. 151–159.
[11] Lee, H. L., and Tang, C., 1997, "Modeling the Costs and Benefits of Delayed Product Differentiation," Manage. Sci., **43**(1), January, pp. 40–53.
[12] Ulrich, K. T., and Ellison, D. J., 1999, "Holistic Customer Requirements and the Design-Select Decision," Manage. Sci., **45**, May, pp. 641–658.
[13] Gupta, S., and Krishnan, V., 1999, "Integrated Component and Supplier Selection for a Product Family," Productions and Operations Management, **10**, pp. 291–308.
[14] Fine, C. H., 1998, *Clockspeed: Winning Control in the Age of Temporary Advantage*, Perseus Books, Reading, MA.
[15] Thomke, S. H., 1998, "Managing Experimentation in the Design of New Products," Manage. Sci., **44**(6), pp. 743–762.
[16] Loch, C. H., Terwiesch, C., and Thomke, S., 2001, "Parallel and Sequential Testing of Design Alternatives," Manage. Sci., **47**(5), pp. 663–678.
[17] Rechtin, E., and Maier, M., 1997, *The Art of System Architecting.*, Boca Raton, CRC Press.
[18] Stone, R. B., Wood, K. L., and Crawford, R. H., 2000, "A Heuristic Method for Identifying Modules for Product Architectures," Des. Stud., **21**(1), pp. 5–31.
[19] Liu, Y. C., Chakrabarti, A., and Bligh, T. P., 1999, "Transforming Functional Solutions into Physical Solutions," *Proceedings of the ASME Design Theory and Methodology Conference*, DETC/DTM-8768, Las Vegas, NV, September.
[20] Shapiro, V., and Voelcker, H., 1989, "On the Role of Geometry in Mechanical Design," Res. Eng. Des., **1**, pp. 69–73.
[21] Kirschman, C. F., and Fadel, G. M., 1998, "Classifying Functions for Mechanical Design," ASME J. Mech. Des., **120**, pp. 475–482.
[22] Baxter, J. E., Juster, N. P., and de Pennington, A., 1994, "A Functional Data Model for Assemblies Used to Verify Product Design Specifications," Proceedings IMechE, Part B, **208**(B4), pp. 235–244.
[23] Umeda, Y., Ishii, M., Yoshioka, M., Shimomura, Y., and Tomiyama, T., 1996, "Supporting Conceptual Design Based on the Function-Behavior-State Modeler," Artif. Intell. Eng., **10**, pp. 275–288.
[24] Szykman, S., Racz, J. W., and Sriram, R. D., 1999, "The Representation of Function in Computer-based Design," *Proceedings of the ASME Design*

Theory and Methodology Conference, DETC/DTM-8742, Las Vegas, NV, September.
[25] Ulrich, K. T., and Eppinger, S. D., 2000, *Product Design and Development*, second edition, McGraw Hill, New York.
[26] von Hippel, E., 1990, ''Task Partitioning: An Innovative Process Variable,'' Res. Policy, **9**, pp. 407–418.
[27] Allen, T. J., 1977, *Managing the Flow of Technology*, Cambridge: MIT Press.
[28] Iansiti, M., 1998, *Technology Integration: Making Critical Choices in a Dynamic World*, Harvard Business School Press, Boston, MA.
[29] Rowles, C. M., 1999, ''System Integration Analysis of a Large Commercial Aircraft Engine,'' Master's Thesis in Engineering and Management, Massachusetts Institute of Technology.
[30] Pahl, G., and Beitz, W., (Wallace, K., ed.), 1991, *Engineering Design, A Systematic Approach*, Springer-Verlag, New York.
[31] Suh, N. P., 1990, *The Principles of Design*, Oxford University Press, New York.
[32] Steward, D., 1981, ''The Design Structure Matrix: A Method for Managing the Design of Complex Systems,'' IEEE Trans. Eng. Manage., **EM-28**(3), pp. 71–74.
[33] Eppinger, S. D., Whitney, D. E., Smith, R. P., and Gebala, D. A., 1994, ''A Model-Based Method for Organizing Tasks in Product Development,'' Res. Eng. Des., **6**(1), pp. 1–13.
[34] Lai, K., and Wilson, W. R., 1987, ''FDL-A Language for Function Description and Rationalization in Mechanical Design,'' *Proceedings of the ASME International Conference and Exhibition*, pp. 87–94.
[35] Sturges, R. H., O'Shaughnessy, K., and Reed, R. G., 1993, ''A Systematic Approach to Conceptual Design,'' *Concurrent Engineering: Research and Applications*, **1**, pp. 93–105.
[36] Collins, J. A., Hagan, B. T., and Bratt, H. M., 1976, ''The Failure-Experience Matrix—A Useful Design Tool,'' ASME J. Ind., **98**, August, pp. 1074–1079.
[37] Rice, J. A., 1994, *Mathematical Statistics and Data Analysis*, Duxbury Press, Belmont, CA.
[38] Sosa, M. E., 2000, ''Analyzing the Effects of Product Architecture on Technical Communication in Product Development Organizations,'' Ph.D. Thesis in Mechanical Engineering, Massachusetts Institute of Technology.
[39] Morelli, M. D., Eppinger, S. D., and Gulati, R. K., 1995, ''Predicting Technical Communication in Product Development Organizations,'' IEEE Trans. Eng. Manage., **42**(3), pp. 215–222.
[40] Van den Bulte, C., and Moenaert, R. K., 1998, ''The Effects of R&D Team Co-location on Communication Patterns among R&D, Marketing, and Manufacturing,'' Manage. Sci., **44**(11), pp. S1–S19.
[41] Sosa, M. E., Eppinger, S. D., Pich, M., McKendrick, D. G., and Stout, S. K., 2001, ''Factors that Influence Technical Communication in Distributed Product Development: An Empirical Study in the Telecommunications Industry,'' IEEE Trans. Eng. Manage., **49**(1), pp. 45–58.