5 Proposed taxonomy

This chapter represents the main contribution of this research to the available body of research linking architecture and organization. A new taxonomy of interactions between product building blocks will be proposed, one that can be viewed as linking engineering and organizational knowledge. The taxonomy is designed such that the characteristics of system-level coordination can be understood, and basic insights into options for improving the design process are provided. Broadly speaking, it is a means for understanding system-level requirements during the design processes by analyzing the structure of an existing product. The representation of a particular product architecture must trigger the generation of more suitable coordination devices that fit particular types of identified interactions, or suggest how the most difficult interaction can be manipulated in order to reduce future need for coordination.

The interaction types that will be introduced are built upon well-known architectural definitions and constructs. In that sense the taxonomy is not new. However, compared to available taxonomies the categorization and choice of constructs makes it much more effective for analyzing and combining available knowledge. The taxonomy and its features will be introduced according to the following framework:

- Introduction to the taxonomy’s main concept.
- Proposal and discussion of the three types of interaction.
- Linking of each type of interaction to coordination.
- Discussion of the taxonomy.
- Summing up.

5.1 Introduction to the taxonomy’s main concept

To this point, the idea of documenting interactions between building blocks as a useful way of representing product architecture has been proposed. Identifying the interactions between all possible pairs of building blocks will provide a clear overview of product architecture. Moreover it will provide an opportunity for analyzing system-level coordination, if it can be assumed that the structure of the project matches the structure of the product. Interactions between building blocks can be considered as the rationale behind system-level coordination, and understanding such a rationale will in turn promote understanding of what system-level coordination is actually required.

In the previous chapter it was argued that the effectiveness of such an analysis is critically dependent on what types of interactions can be identified between building blocks. For a useful analysis it must be possible to recognize what the technical reason or reasons behind an interaction are, and by which technical decision(s) an interaction can be manipulated. Moreover it is essential to be able to deduce the impact of a type of interaction on the system-level coordination characteristics. The question that remains is what these types look like. For a solution, this chapter will reexamine and link some of the important findings in Chapters 2 and 3 of this thesis.
First, Ulrich’s original definition will be looked at anew. According to him, three types of technical decisions determine product architecture. These can be put under the headings of functional arrangement, mapping from functions to building blocks, and specification of the physical interfaces. It will be argued that since these decisions determine architecture, and architecture is about interactions between building blocks, each of these decisions (and these decisions only) determines the interactions between building blocks.

This chapter will demonstrate that the role of architectural decisions is to achieve good integration between the building blocks. In fact, each decision can be interpreted as a set of conditions, technical and otherwise, that need to be satisfied by the building blocks in order to create a properly functioning final product. Each architectural decision refers to different technical conditions and can be thought of as a type of interaction. The proposed taxonomy must thus facilitate translation of the technical architectural decisions into types of interactions between building blocks.

This has at least two advantages:
- We will know the underlying reason (which technical decision) for each type of interaction.
- We will know how an interaction (by which technical decision) can be manipulated.

In addition, as will be shown in 5.2 the ‘architectural’ decisions can be used to identify the coordination characteristics. For each type of interaction it will be decided whether goal-setting (as specified in Chapter 3) between the teams is possible or not. The main idea here is that the hierarchical structure of the technical decisions (design goals versus physical solutions) can be linked to the hierarchical concept of goal-setting (goal or task versus detailed activities/decisions) between teams.

In the next section a taxonomy of three types of interactions between building blocks will be introduced: the functional, the mapping, and the physical type of interaction. These definitions will be based on the technical constructs as proposed in Chapter 2.3. Interpretation of the interaction types will be inspired by the prescriptive design methodologies, in particular axiomatic design.

### 5.2 The three types of interactions proposed

Under the following headings, the three types of interaction will be described in the order previously introduced. Each type will be discussed separately. In the discussion the relationship between the types will be addressed.

#### 5.2.1 The functional type of interaction between building blocks

Two building blocks have a functional type of interaction when their functions are connected by flows of energy, material, or information. Figure 5.1 shows this particular interaction type. A change to one building block that affects the specifications of its functional output is sufficient to force a modification of the specifications of the required functional input of another building block, and vice versa. This interaction type clearly relates to functions that can be expressed as inputs and outputs of energy, material, or information, and is neutral in respect to particular physical building block solutions. This interaction type is identical to the relationships described in the functional scheme of Pahl and Beitz or Ulrich and Eppinger. However, it differs from the exchange type of interactions of Pimmler and Eppinger since physical realization is not included.
5.2.2 The mapping type of interaction between building blocks

Two building blocks have a mapping type of interaction if they are mapped to the same function that is not initially decomposable, in line with the discussion in Chapter 2. This type is illustrated in Figure 5.2 (that is based on the example of Figure 2.12). Accordingly, a change to one building block that affects realization of the shared function requires a change to the other building block to properly realize their function, and vice versa. The building block physical characteristics will together result in a working interrelationship (Pahl & Beitz 1996) that fulfills the function.

A building block has no mapping interaction if its mapping (from functions to building blocks) is one-to-one or N-to-one. In these cases, the block fulfills one or more functions itself without direct interference from alternative blocks.

This type of interaction is obviously derived from mapping from functions to building blocks (according to Ulrich), and is also similar to the theory of axiomatic design. Nevertheless, available taxonomies do not distinguish the mapping type of interaction from the functional one. This distinction will be dealt with in the discussion.

5.2.3 Physical interactions

In addition, there are physical interactions between the building blocks since the blocks have to be physically put together (assembled). In contrast to the previous two types, these do not directly refer to desired functionality. Three sub-types can be identified: interactions due to a physical interface, global constraints, or side effects. These are all based on ‘interface coupling’ that according to Ulrich greatly affects modularity. Figure 5.3 shows the three sub-types, which are described below.
Interfaces
Two building blocks physically interact if a change to one that affects the other is necessary in order to realize their interface. This interaction not only refers to a physical connection but also (where such is relevant) to the physical realization of a functional interaction.

Global constraints
Physical building blocks physically interact if they are subject to the same global constraint. Accordingly, a useful change to one building block that results in exceeding the global constraint requires a change in the other block in order to satisfy the constraint. For instance, each building block has a size, shape, and position, and the use of space excludes the use of another building block. If the total space is determined (limited) then the blocks interact. Because all building blocks interact to some extent in this way (they all contribute to weight, space, and so on) only those useful changes to building blocks that have a significant (to be specified) impact will be considered here.

Available taxonomies do not include global constraint interactions. Spatial interactions (Pimmler & Eppinger 1994), or a need for orientation (Sanchez 1999a) can, however, be a consequence of this interaction type.

Side effects
Two building blocks physically interact if a change to one block that affects its side effects requires a change to the other in order to function correctly. Building blocks generally generate heat, vibration, magnetism and so on as a side effect of the design parameters for realizing a desired function. Since these interactions depend on detailed physical parameters or combinations of some (i.e. position) in respect of both blocks, and are difficult to define in a function structure, side effects can best be categorized as a 'physical' type of interaction.

It should be noted that the three sub-types have fundamentally distinct reasons for interface coupling. These reasons do not come into Ulrich’s definition but are included within the corresponding examples discussed in Chapter 2.

5.2.4 The relationship between the types: discussion
To this point the three interaction types have been introduced separately and they will now be discussed in relation to each other.

All interaction types should be considered as equally important. The fact that the global constraint interactions were dealt with relatively late does not necessarily mean that these should always be considered at a later stage than the others. Not all interaction types, however, are fully (sequentially) independent in the sense that one type of interaction also causes another type of interaction. This will be described below.
The functional and the mapping types are independent. Both result directly from the mapping of the functional scheme to the building blocks, but a functional type does not imply a mapping type of interaction between two blocks, or vice versa.

The physical interface interaction depends on the functional and mapping interactions. If there is a functional or a mapping interaction between two blocks there is also by definition an interface interaction that establishes the physical realization of the exchange. Conversely, an interface interaction between two blocks does not necessarily imply a functional or mapping type of interaction between two blocks. For instance, two blocks that are physically attached do not necessarily have a ‘functional’ relationship (think back to the example of the bottle and the cap in Chapter 2).

Due to global constraints, the physical interaction does not have a relationship to the other types. Note however, that realization of a mapped or other function may indirectly affect global constraints. For instance, the aesthetic design of the shaver housing may cause a limited amount of space for remaining building blocks.

Finally, due to side effects the physical interaction does not have a relationship with the other types. In the final discussion of the taxonomy, the dependencies between the types will be looked at in greater detail.

### 5.3 Impact on coordination per interaction type

This section will illustrate how the interactions of an existing product can in theory be translated into system-level coordination characteristics. The following illustration will throw light on the rationale behind this.

Essentially, a project team has the task of defining a whole range of detailed design parameters that together satisfy all product functions and constraints. When the project team is split up into smaller design teams, each of these teams has the task of specifying the detailed design parameters necessary to establish a physical building block. Obviously, if all of these design teams do their work in complete isolation it is extremely unlikely that their joint achievements will result in a functioning end product. System-level coordination is needed to compensate for the fact that the teams will act as if they are fully independent.

Furthermore, it is also evident (after Galbraith) that it is impossible to draw up a detailed protocol that specifies what detailed decisions each team has to make in each situation in order to realize the overall product. Goal-setting is a beneficial mechanism for achieving coordination. Teams ideally specify goals at a relatively high level of abstraction and are able to implement their decisions concurrently. As was argued in the chapter on organization (based partly on the work of Thompson), whether it is possible to apply goal-setting will depend on the conditions that are included by an interaction. The more conditions there are and the more detailed they are, the fewer conditions for constructing a building block will be independent of what is done by other teams.

Because architecture is being mapped on organization, these conditions can be identified with precision. In fact, the ‘conditions’ that need to be coordinated are embedded in the interactions between the building blocks of the product. The more detailed the interactions between the building blocks are, the less the ability to apply goal-setting, and the more intense system-level coordination becomes. This section will consider to what extent goal-setting is possible for each type of interaction. For this, chapter 2 will be largely relied on, and the prescriptive literature directed at efficient design.
It should be noted that a pair of building blocks may engage in multiple interactions (of several types). In this section, however, the characteristics of each type will be described as if these are the only ones between those blocks. In the event of multiple interactions occurring, each of these interactions has to be coordinated and the consequences are (at the very least) additive. In the discussion the possibility of the various interactions interfering with each other will be considered.

Finally, it should be stressed that the aim is not to explain the system-level coordination effort from the very beginning of a design project. It will be assumed that for each product that will be analyzed its functions, mapping, building blocks, and constraints are known. In fact, the period analyzed will start from after the architecture has been determined and extend to the final design (see the discussion at the end of this chapter).

The impact on coordination will be described below for each interaction type in the same order as introduced in the previous section. For each type there will be a brief reference to how the goals between the teams are set (and communicated), to what extent the teams can work in parallel, and a brief note on the likelihood of exceptions to the planned goals (similar to Galbraith’s theory).

5.3.1 Functional interaction and coordination
Two teams whose blocks are engaged in functional interaction are able to apply goal-setting to a considerable extent and require little system-level coordination. Both teams can make detailed design decisions in full isolation from each other as long as they each achieve the appropriate functional specifications (inputs and outputs). A power supply design team, for instance, is allowed to do what it likes as long as the design establishes the appropriate specifications for energy output.

A functional scheme of specification and agreement can, in fact, be described as specification of the goals that each team has to meet. It should be noted that the generation of such a scheme (specification of the goal itself) may involve many iterations and require intense communication (Pahl & Beitz 1996) to find suitable goals that can reasonably be expected to be achieved by the teams.

Figure 5.4 conceptualizes the coordination of a functional interaction. Once a functional scheme has been devised it can be readily communicated to the teams involved. The teams can each try to generate and test detailed design parameters concurrently in order to achieve their ‘goal’. As long as the design of each building block meets its planned specification no additional coordination is required and the teams can work concurrently. One team failing to reach its planned goal may have a disruptive effect. In that case additional system-level coordination is required to solve this problem. In general, the more ‘difficult’ it is to meet a functional specification, the more failures may be expected to occur.
5.3.2 Mapping interaction and coordination

Two teams whose blocks have a mapping type of interaction will be hampered in their goal-setting to a certain extent. Since their blocks have to jointly achieve a function, detailed design decisions cannot be made in full isolation of each other and mutual adjustment is a prerequisite.

Coordination of a mapping interaction relates to agreement on a functional scheme where two blocks need to jointly satisfy one function. Each team’s particular goal specification will include very detailed specifications of design parameters that each team needs to realize. In effect, the search for these separate goals will go hand-in-hand with the jointly evolving design of the two blocks.

As shown in Figure 5.5, generate-test cycles will occur between the two design teams and they have to exchange information at a highly detailed level. Based on the logic of axiomatic design, the following consequences can be deduced. In order to find an appropriate setting for both blocks, team 1 specifies a set of design parameters that (in their judgement) provide a contribution to the joint function. These specifications must be communicated to team 2. Doing so may be very complex since the detailed design parameters chosen may include multiple and complex expressions of detailed characteristics, possibly with hard-to-communicate sensitivities and behavior (Whitney 1996). Subsequently, team 2 adds to the design parameters of block 2 in such a fashion that it is expected that the blocks will jointly fulfill the function. However, whether the function is correctly fulfilled cannot be seen on paper but has to be tested in collaboration with the other team. If the results are negative the cycle starts again. If the results are satisfactory the required goals for each team are specified separately, but at the same time these are also realized in the relevant design parameters of both blocks.
When for some reason (for example, other goals having to be met) one team fails to meet its design goal, then additional system-level coordination is required. Since this type of interaction includes many detailed conditions, failure is likely to occur, as was concluded with respect to Thompson's work.

It should be noted that the way that the cycles are conceptualized here is somewhat primitive. The cycles will probably be performed within multiple layers, i.e. first the selection of a working principle and then the selection of the detailed design parameters, as in the work of Pahl and Beitz. In any case, this reasoning highlights the expected need for considerable mutual adjustment between teams if they have to deal with a mapping type of interaction. It would appear that such an interaction differs in this respect from a functional interaction where specification of the goals and realization of the goals can be cleanly separated for each block.

5.3.3 Physical interactions and coordination

The impact on coordination of the physical interaction type will differ per sub-type. Except where there are global constraint interactions, it is more difficult to generalize about the coordination effort required. Globally constrained interaction will thus be dealt with first and then the side effects and physical interface described.

Global constraints and coordination

Two teams whose blocks are involved in a global constraint interaction with global constraints are able to apply goal-setting. A global constraint can be decomposed into a constraint for each block. As long as each team satisfies this constraint, the teams are able to make all detailed design decisions independently of each other.

Specification of all sub-constraints can be very difficult and require considerable coordination effort, though once specified, the conditions can be easily communicated across the organizational units, and it is easy to check whether each blocks is staying within the constraint. As long as the design of each building block fits within its planned sum of space, weight or costs, the design can be executed concurrently and the overall constraints are satisfied. The tighter the constraints for each block the more difficult it obviously is to satisfy

---

3 Note the similarity between functions and global constraints here. In fact, both can be decomposed into 'smaller elements' that can each be considered as a 'design goal' by each of the teams. As a logical consequence of this assumption, it is possible to think of a strategy where two teams have to jointly satisfy (stay within) the same constraint. In such a case, decomposition of the constraint has to take place at a low level of detail in close team collaboration, similarly to a mapping interaction. Nevertheless, mapped constraint interactions will not be considered here. The reason is that technically speaking, global constraints are easy to decompose, in contrast to (mapped) functions. Global constraints thus allow for goal-setting. This is not to say that it is easy to find a decomposition of a global constraint that is appropriate (realistic) for each building block.
the conditions, and the more likely the occurrence of failure and additional system-level coordination effort.

**Physical interfaces and coordination**
Two teams whose blocks have a physical interface need to perform system-level coordination at a detailed level. However, it is difficult to generalize about such an interaction. Because it has so many physical aspects, this type of interaction probably has many of the characteristics of a mapping type of interaction. The problem, however, is that it may be difficult to locate a physical interface. Can it be seen as a part of one of the two blocks or is it to be considered a shared feature? Testing of an interface will obviously require two blocks and has to satisfy a large number of production and assembly constraints and wishes.

**Side effects and coordination**
The coordination of side effects involves reacting to unplanned or unintended effects. According to design literature (Pahl & Beitz 1996, Ulrich & Eppinger 2000), management of side effects involves close coordination between the organizational units involved and requires trial-and-error testing of small changes to physical parameters. A side effect may be considered an unexpected consequence of the original specifications and, as a logical consequence, cannot be planned for. It is simply impossible to set any rules for such occurrences. The amount or type of coordination will depend on what type of specification has not been met.

5.4 **Discussion of the taxonomy: its role, comparison, and restrictions**
A taxonomy of three types of interactions between physical building blocks of a product has been proposed. The role of the types will be briefly considered below, and the alternative taxonomy (of Pimmier and Eppinger) looked at a second time, and its restrictions highlighted.

5.4.1 **The role of the taxonomy**
The functional, mapping, and physical types have been delineated on the basis of Ulrich’s general definition of architecture. The underlying reasons for each interaction are embedded in the definition of the constructs, and the available literature and knowledge of product architecture can be used to interpret each interaction type. This can be expected to be beneficial for (1) a general understanding of what interactions involve (2) generation of options for improvement, and (3) indicating the contingencies inherent in any structure where interaction takes place.

It will first be argued that understanding the technical reasons behind an interaction will increase team member understanding in general. As such, the way that they perceive the interaction will not depend on background factors, temporary matters or individual factors.

Second, insight into the background to the interactions provides useful information on how an interaction can be manipulated and what the potential difficulties are. In fact, changing the structure of interactions means that the technical decisions that give rise to the interactions need to be altered. For instance, a mapping interaction can be manipulated by changing the way functions are mapped to building blocks.

Third, understanding the contingencies inherent in an interaction structure can be obtained by structurally discussing the why of underlying decisions. As argued in 2.4, there
may be many considerations underlying an architectural decision that subsequently gives rise to an interaction between building blocks. Some of the interactions may thus be seriously embedded in broader contingencies such as production structure, available technology, unit cost price, special priorities, or traditional ways of problem solving. Consequently, it is not only imperative to show how an interaction can be technically manipulated, but also how manipulation may be hampered by much broader considerations.

How each type can be individually manipulated will not be dealt with any further since this will depend on the definition. In the case study, though, this aspect will be thoroughly illustrated.

It has been shown that it is theoretically possible to deduce the characteristics required of the system-level coordination that matches each type of interaction. Prescriptive design models aimed at efficient design provide the framework to do so. Real-life projects, however, are generally not as clear and pre-structured as the prescriptive models. Hence engineers may apply more intense or different ways to handle interactions, but these are never less than the prescriptive characteristics. These minimum requirements can be expressed in terms of propositions. Their role is twofold. In the first place, they represent a step in the direction of a Thompson-like theory able to explain the differences in system-level coordination effort during a design process, based on the characteristics of the product alone. Second, they provide guidelines for understanding how the management of a specific interaction can be improved. On the one hand, actual coordination devices may be altered such that these correspond better to the propositions (ideal or less so) per type of interaction. On the other hand the propositions can be used to illustrate which effort is by definition embedded in an interaction, and hence to indicate when further improvement of coordination cannot be reasonably expected without changing the interaction.

To sum up, it has been argued that the proposed three types of interactions satisfy the criteria that were suggested in the formulation of the problem in the previous chapter. The taxonomy will now be compared to that of Pimmler and Eppinger (1994) so as to demonstrate this taxonomy's particular advantages.

5.4.2 Comparison with Pimmler and Eppinger

When the types of interaction were defined, it was mentioned that there were some differences with those in the taxonomies in the literature. Using the taxonomy of Pimmler and Eppinger, these will be illustrated in greater depth and the consequences discussed.

In Chapter 2 it was mentioned that the taxonomy of Pimmler and Eppinger is not easily translatable into Ulrich's architectural decisions. In contrast to the interaction types in this thesis, their taxonomy does not provide a clear decision-making hierarchy. The exchange types may refer to functional aspects as well as physical aspects (the physical realization of the interface). Moreover, spatial interactions seem to refer to a physical solution (the location of a building block/component). Consequently it is not possible to deduce what decision resulted in a constraint on location. This may have involved a decision at a high level of abstraction (i.e. managing a global constraint) or at a more detailed level (i.e. a solution to a mapping interaction).

Each of the constructs thus may refer to different levels of detail and it is not possible to make a statement about the required type of coordination. It should also be remembered that
poor alignment with the architectural decisions hampers clear understanding and the
generation of options for manipulation.

Most remarkably, though, is that in contrast to the taxonomy proposed here, Pimmler and
Eppinger do not include a mapping type of interaction. It has been shown that introduction of
the mapping type of interaction contributes to a meaningful analysis of coordination require-
ments and options for manipulation. The question thus remains as to how existing
taxonomies model mapping interactions. There are two possible explanations.

The first may be that the spatial interaction takes care of all mapping interactions. However, this does not seem adequate since many more detailed parameters than proximity
alone may be involved in the fulfillment of a function. Moreover, proximity may even be a
result of the exchange types (the physical realization of exchange of material energy or
information). When, for instance, an interface between two blocks is fixed, then these must
also be in proximity to each other.

A second explanation may be that mapping interactions are modeled by exchange interac-
tions. Within the taxonomy in this thesis, the distinction between the functional and the
mapping types is simply based on the assumption that a function cannot be fully decomposed
(to the most detailed level) without considering ‘physical solutions’ (as concluded in Chapter
2). If for argument’s sake it is assumed that (after Pahl and Beitz) functions can be
decomposed in a solution-neutral fashion, then a mapping type can be written as a collection
of functional interactions at a lower level of abstraction. A mapped function is split up into
such levels of detail that each sub(sub)function (transformations of input and output) can be
completely allocated to a block. As a consequence, the mapping type of interaction is
transformed into a set of input-output exchanges between the sub(sub)functions of both
blocks. Perhaps this reasoning explains why available taxonomies mainly consist of exchange
types and do not include mapping types of interaction.

However, in line with the conclusions in Chapter 2.2, decomposition of functions depends
strongly on chosen physical solutions. As a result, expressions of input and output between
sub(sub)functions strongly depend on detailed physical characteristics and probably involve
many and complex expressions of input and output (including force). Furthermore, many
design goals are practically impossible to decompose. As a result it will be argued that the
introduction of the mapping type gives this taxonomy a clear advantage over other available
taxonomies.

To sum up, the proposed taxonomy provides a contribution to available taxonomies
because, unlike those in the literature, the types of interactions described here can be linked
to architectural decisions.

5.4.3 The taxonomy’s limitations
Three comments on the proposed taxonomy should be made. The first involves the fact that
building blocks are considered as being black boxes, the second deals with the fact that only
the final interaction structure is dealt with, and the third has to do with the lack of time and
cost issues relating to these constructs.

Black box
This taxonomy of interactions is purely for indicating interactions between building blocks.
The contrary is true. The interactions within the blocks are generally much stronger than the interactions between
the building blocks. The inner interactions are not modeled, however, since we are only interested in system-level interactions and coordination. The blocks are thus considered as being black boxes. The limitation that this imposes is that it is not possible to define potential relationships between the types of interactions at a lower level of abstraction. While it was previously argued that a side effect interaction does not have a relationship with the other types, at another level specific design parameters that are required to fulfill a functional interaction may also refer to a side effect. A single design parameter may thus play a role in multiple interactions. The way in which one interaction is satisfied by a design parameter may thus have an impact on the way another interaction can be solved.

Since including all these details would have been at the expense of a general overview of system-level interactions, these aspects will not be taken into account within the current taxonomy. During the case-study discussion, this issue will be looked at in greater depth.

**Final interaction structure**

Product design is an evolving process, and consequently interaction structures may be expected to change during the course of the design project. As a result, there may have been system-level coordination of interactions that are not included in the final set of interactions preceding completion of the documented product.

The design of a product is an evolving process, and as a result the structure of interactions probably is dynamic during the course of the design project. Consequently, there may have been system-level coordination effort for interactions that did not return in the final set of interactions of the almost finished product that we documented. Since coordination is mirrored on the final set of interactions a portion of coordination that took place maybe missed. However, it is argued that (after the architecture was specified) most of the types of interactions are the same for the whole period and only the exact specifications were subject to dynamics. The final set will therefore be reasonably representative of the process as a whole.

Furthermore, it should again be stressed that the coordination activities that took place to define the basic architectural decisions have not been addressed. According to Henderson and Clark (1990) this involves a period of considerable experimentation and intense coordination and it is in fact impossible to speak about building blocks at all. For this analysis, what this implies is that for projects involving radical innovation (see Chapter 3), only a small part of the coordination of the whole design project can be analyzed. However, for projects involving incremental design, a large part of the processes can be analyzed since the major architectural decisions will already have been made early in the project.

**The interaction constructs only define the technical conditions**

This taxonomy of interactions applies to the technically inevitable system-level coordination that takes place between the design teams. In addition, there are many other aspects that affect system-level coordination. Besides resulting in a technically correctly functioning product, the project has to be finished within a specific schedule and has to be feasible in terms of the financial budget and number of designers. To that end, design teams also need to satisfy goals with respect to these issues. The tighter the restrictions, the higher the probability of failure, and the more system-level coordination probably is required. However, this taxonomy does not include such managerial factors. The focus of this study is solely on the unavoidable underlying technical product structure. This issue will be briefly reexamined in the case-study discussion.
5.5 Summing up and field of application

To recapitulate, in contrast to the available taxonomies, a taxonomy of technical interactions between building blocks that can be clearly linked to architectural decisions has been formulated. The benefits of the proposed interaction types are that they facilitate understanding of the reasons for documented interactions, and options for manipulation can be easily derived. Moreover, the impact on the coordination of each interaction type can be derived from the characteristics of the underlying technical constructs.

Perhaps it is no accident that the variety of technical decisions contained in the prescriptive design literature can be translated into coordination characteristics. The ‘layers’ of technical decisions not only have to facilitate effective problem solving, but must also facilitate structured coordination across designers. In fact, if the various technical constructs were not suitable for the working together of a great many people, the prescriptive literature cannot have been based on good practice.

Finally, those situations in which the suggested approach is likely to be the most appropriate one will be described. Although it can be argued that the technical theories will be valid across a whole range of physical products, some conditions can be specified for which the interaction approach is particularly relevant. These include:

- Products that are large and complex enough to be decomposed into building blocks.
- Products that consist of all three types of interactions (not fully modular products).
- Project teams where many designers are involved, and the design teams are best organized around the product building blocks.
- Design projects that are under pressure to improve their performance.

It would appear that in those cases where these four aspects apply, the theoretical constructs are particularly relevant. This is not to say that the theories are not appropriate for fully modular products. However, a focus on interaction is far less useful since very few interactions (and none of the mapping type) will occur between the building blocks and these are thus a variable of little relevance in improving design processes.

The question now remains as to how the proposed constructs will apply in practice. This will be illustrated and tested in the next chapters.