15. The Synthesis of Design Alternatives

15.2 Determining the boundary of the information system.

An important design decision that is made during the synthesis process is the determination of the boundary of the computer-based information system. By assigning a particular set of IPFs to specific components of the software system being designed, each information system design alternative defines the IPFs that will be automated and, hence, the boundary of the computer-based information system. It is through this boundary that the communication between the information system and its users takes place. All the inputs to the information system, including transactions to be processed, database updates, requests of queries and reports, and the input of control and parameter information are submitted to the information system through this boundary. Similarly, all the outputs from the information system including reports produced, answers to queries, errors detected, current parameter values, systems status, and other kinds of information are delivered to users through this boundary.

The boundary of the computer-based information system that corresponds to a particular design alternative defines a corresponding location and informational contents of the user interface. For a given design alternative, the user interface cuts across all the information flows that interconnect IPFs performed within the computer-based information system with those performed by its users and by other systems, internal or external to the organization. At the logical level, the user interface consists of the information structures conveyed by the information flows that go across it. Summarizing, any design alternative defines a boundary of the computer-based information system. Such a boundary encloses the information processing functions that will be performed by the computer-based information system. It also defines the information contents of the user interface in terms of the information structures composing the information flows that go through it.

Once the boundary of the information system is determined for a particular design alternative, the design of the information system architecture can be completed. Since the boundary of the information system determines the set of IPFs that can be assigned to a specific component of the information system architecture, it also determines in a precise manner the contents of the user interface and how exactly the user interface requirements apply to this particular design alternative. The focus, in this respect is on how the organizational unit that implements the function should work and
how the function itself is affected by each design alternative considered. The main goal of the synthesis process, however, is to synthesize valid design alternatives, that is, design alternatives that are consistent with the logical requirements of the information system as specified by the LIPS description.

Design is a predominantly creative activity. In fact, systematic design methods seem to be applicable in sufficient detail only to situations arising in narrow domains. Even in these cases the design processes should be under the control of humans and ultimately their responsibility. The synthetic approach to design gives designers the possibility of considering any number of design alternatives. This flexibility is essential to the design of function-supporting information systems and allows the designers to exercise their creativity. The synthetic approach also provides criteria for the evaluation of the design alternatives from the point of view of their ability to support effectively the functions of the organization. In the following sections I will describe the synthesis of design alternatives starting with the design of the architecture of the computer-based information system.

15.3 The synthesis process in more detail.

For each design alternative considered the synthesis process proceeds as follows:

1. The architecture of the software system is determined by identifying its main functional components and their interrelationships.

Once the architecture of the information system and its boundary have been determined for a given design alternative, the synthesis process proceeds as follows:

2. The structure of each computer program is determined. This involves decomposing the program into modules that in turn determines the interfaces between modules and their input/output characteristics.

3. The structure and logic of each module is then determined.

4. The validity of the proposed design alternative is determined. That is, it is consistent with the logical requirements of the information system. This involves verifying that all the IPFs that the computer-based information system will perform are incorporated into the appropriate programs and modules in a manner consistent with the logical requirements. A design alternative is consistent with the logical requirements of the information system if it preserves the five relations of the LIPS specification.

Each of these activities of the synthesis process is explained in more detail below.

15.4 Verifying the validity of design alternatives.
A primary concern of the synthesis process is the verification of the validity of design alternatives, that is, their consistency with the logical requirements of the information system. The synthesis process, in fact, can be considered as a process whereby a specification of the computational requirements of the information system, as expressed in the LIPS formalism, is translated into a specification of the design and implementation characteristics of a particular design alternative. The verification of the consistency of a design alternative with respect to the logical requirements of the information system lends itself to be done along with this translation. In fact, the verification of the consistency of a design alternative with the logical (computational) requirements of the information system should be done as early as possible in the synthesis process. An obvious advantage of doing this is that detecting and correcting inconsistencies of a design alternative with respect to the logical requirements practically eliminates the problem of dealing with design alternatives that ultimately will be discarded because of the existence of this kind of inconsistencies.

As each design alternative is synthesized, its validity, that is, its consistency with the logical requirements of the information system must be verified. The verification of the consistency with the logical requirements of the information system must ensure that the specification of the processing in each design alternative is correct and complete. The correctness of the computer-based information system means here that the assignment of IPFs to software system components preserves the five relations of the LIPS. Its completeness means that all of the IPFs that are assigned to the proposed software system for a particular design alternative is actually incorporated in one of its components. Clearly, any IPF that is part of the LIPS specification but is not incorporated in the information system being developed should be performed manually or by some other computer-based information system. The validity of the design alternatives considered guarantees that the information system will perform the right computations and deliver the information required for the effective support of the organization’s functions. In particular, in a valid design alternative the contents of the user interface are consistent with the particular assignment of IPFs to the computer-based information system used in that design alternative. The contents of the user interface in a valid design alternative correspond precisely to the information flows that interconnect the IPFs incorporated into the computer-based information system being developed with those performed by the users of the information system or by other information system applications. The three levels normally used for the description of computer-based information systems, namely, the system architecture, the program structure, and the module logic levels are all relevant to the verification of the consistency of the information system with its logical requirements. The verification of the validity of design alternatives is explained in more detail below.

*Preserving the input and output relations.* $\sigma_I$, the input relation, specifies the information structures that are inputs to each IPF included in the LIPS specification. Preserving this relation means that the inputs to each software system component, be it a high-level functional component, a program, or a module, are precisely the information structures input to the IPFs that such a component incorporates. In other words, it guarantees that the information structures that are input to each IPF indicated in the LIPS
specification; are also an input to the software-system component incorporating that IPF. Preserving the input relation in a design alternative therefore ensures that in such an alternative, each module operates on the correct inputs. Similarly, \( \sigma_O \), the output relation, specifies the information structures that are output of each IPF included in the LIPS specification. Preserving the output relation through the assignment of IPFs to the information system components considered in a given design alternative clearly means that the outputs of a software system component are precisely the information structures that are output to the IPFs it incorporates. Consequently, in a design that preserves the output relation, each information system component is designed to produce the correct output.

**Preserving the precedence relation.** An important relation in the LIPS specification is \( \sigma_P \), the precedence relation, a binary relation on the set of IPFs. Recall that according to this relation, an IPF \( p_i \) precedes an IPF \( p_j \), that is, \( \text{precedence}(p_i, p_j) \) if at least one information structure, \( x_k \) say, is both an output from \( p_i \) and an input to \( p_j \). Any design alternative preserving the precedence relation must be designed in such a way that, any pair of IPFs that it incorporates must relate to each other precisely in the manner specified by the precedence relation. For example, suppose that the IPF \( p_i \) that verifies product information has as an output an information structure \( I_k \) that is also input to the IPF \( p_j \) that checks the dollar amount of an order against the customer’s credit limit. Let us call \( M_i \) the module that incorporates \( p_i \) and \( M_j \) the module that incorporates \( p_j \). Clearly, in this case, according to the LIPS specification \( \text{precedence}(p_i, p_j) \) holds. Therefore, in the software system, for any design alternative to preserve the precedence relation, \( M_i \) should produce an output equivalent to \( I_k \) which will also be an input to \( M_i \). Verifying that the precedence relation is preserved in a design alternative amounts to proving an important aspect of its correctness. More specifically, it verifies that the inputs and outputs of the various IPFs are consistent with each other.

**Preserving the hierarchy relation.** \( \sigma_H \), the hierarchy relation specifies another essential aspect of the structure of IPFs. Briefly, it specifies how a particular computation is composed of various smaller IPFs. In particular, it specifies whether an IPF, say \( p_k \), performs part or all of the computation performed by another IPF, say \( p_i \). As you may recall, \( \text{hierarchy}(p_k, p_i) \) asserts that \( p_k \) defines part of the computation defined by \( p_i \). Notice, in particular, that \( \text{hierarchy}(p_k, p_i) \) means that all of \( p_k \) is included in \( p_i \). Preserving the hierarchy relation in a design alternative means that the software system component that incorporates \( p_k \) performs precisely the computation specified by \( p_k \) as specified in the LIPS. Conversely, it also means the software system component that incorporates \( p_i \), includes the computation specified by \( p_k \) as specified in the LIPS. In other words, in order for the hierarchy relation specified in the LIPS to be preserved in such an alternative, whenever \( \text{hierarchy}(p_k, p_i) \) holds in the LIPS of an information system, the same relationship must exist between the software system components incorporating \( p_i \) and \( p_k \) respectively. Let us say that, for example, that \( \text{hierarchy}(p_k, p_i) \) holds in an LIPS
specification, where the IPF $p_i$, that verifies product information, includes $p_k$, that verifies the product code. Preserving the hierarchy relation means, in this case, that the software system component that incorporates $p_i$ includes the computation specified by $p_k$ as part of its code, or calls a module that performs such a computation at execution time. In any case, the computation specified by $p_k$ should be a part of $p_i$ exactly as specified in the LIPS.

Preserving the trigger relation. $\sigma_T$, the trigger relation specifies necessary conditions for the activation of IPFs. These conditions represent events or situations occurring in the organization that signal the need of the information produced by the IPF they trigger. The trigger relation thus establishes a link between the information processing system and the dynamics of the organization. Recall that $\text{trigger}(t_m, p_i)$ means that the IPF $p_i$ should be performed whenever the condition $t_m$, its trigger, holds. A customer payment, for example, triggers the updating of customer accounts and the updating of the accounts receivable register; the end of the month triggers the production of the payroll register and the corresponding pay checks; a management decision may trigger a decision-making function aimed at solving some emerging problem or at taking advantage of some perceived opportunity. In all those cases, the IPFs that provide the information needed will have to be activated.

Preserving the trigger relation in a design alternative means that the execution of any software system component should be conditioned to the occurrence of the event associated with its trigger. A pair in this relation, such as $\text{trigger}(\text{Pay-Day}, \text{Compute-Payroll})$, must therefore result in a procedure that specifies that the computer program that performs the Compute-Payroll IPF is run whenever the event Pay-Day occurs.

The design alternatives that the synthesis process generates must be logically correct alternatives; these alternatives, referred to here as valid alternatives, must satisfy the logical requirements of the information system. The valid alternatives will perform exactly the computations described in the LIPS specification and, consequently, produce as a result exactly the information structures specified in the LIPS. Although they may differ in the way in which IPFs are assigned to specific software-system components, valid design alternatives will all in principle produce the correct information. Each valid design alternative forms the basis of a number of specific solutions to the information systems design problem. It represents a basic choice of architecture and detailed organization and logic of the computational contents of the software system. Each valid design alternative can also be implemented in a number of ways depending on make or buy decisions, choice of package or programming language, specific hardware and systems software options, and the like. Clearly, a considerable number of design alternatives can be considered for a given information system.

The synthesis process, however, cannot guarantee by itself that the design alternatives it generates also meet the performance and user interface requirements of the information system. Therefore, all valid alternatives, including of course all the possible implementations of a valid alternative must be verified as to their acceptability, that is,
their compliance with the performance and user interface requirements of the information system and their ability to contribute to the adaptability of the organization. This means that all of the options required from each dimension of the design space must be specified, so that it is clear, for each design alternative considered, how each aspect of the design is going to be implemented. In the next section we describe the verification of the acceptability of design alternatives.

16. The Design Principle and the Evaluation of the Acceptability of Design Alternatives

In the synthetic approach, the validity of all the design alternatives synthesized must be verified. In other words, their consistency with the logical requirements of the information system must be determined. Once the basic architecture of an information system design alternative has been determined through the synthesis process, the design of the processing part of the information system can be specified in detail. As shown in previous notes, the synthesis process, although based primarily on the logical requirements of the information system, takes into account its performance and user interface requirements as well. In the synthetic approach, however, the acceptability of design alternatives (recall that, to be acceptable, a valid design alternative must also meet the performance and user interface requirements and contribute positively to the adaptability of the organization) is not completely verified in the synthesis process. In fact, as it is also shown in Figure 13.1, the verification of acceptability fully takes place as a separate evaluation process after the design alternative in question has been found valid. An important reason for this is that it is indeed more costly and inefficient to verify the acceptability of design alternatives that might turn out not to be valid. Notice also that it is convenient to determine the validity of a design as early as possible in the synthesis process so that the designer’s attention can be directed toward the consideration of its acceptability. How this can be done depends on the particular information system being designed. A good design strategy is to focus first on those aspects of the processing design that are more critical from the standpoint of validity. This strategy facilitates the early detection of inconsistencies of proposed designs with the logical requirements of the information system. This detection can occur even at the stage of information system architecture design. This strategy also helps to signal when the process of evaluation for acceptability can start without too much of a risk of spending time and resources on an a design alternative that is likely to be found not valid.

As it is the case with the synthesis of design alternatives, the design principle provides also the necessary guidance for their evaluation for acceptability. The design principle helps the designers focus on the compatibility of the information system with the structure and the dynamics of the organization and its functions and with the adaptability of the organization. Meeting a set of information system requirements that reflect the information needs of organizational functions amounts to achieving the necessary compatibility with the structure and the dynamics of these functions and with the adaptability of the organization. It also amounts to achieving effective function support.
The process of evaluation of the acceptability of a design alternative starts in full when its validity has been determined. This happens when the processing to be done by each component of the information system has been specified in sufficient detail to verify its compliance with the logical requirements. Not only the processing must be correctly specified, but the specification must also be complete. This means that all the IPFs that will be performed by the computer-based information system must be included in the specification in a complete and correct manner. Detailed aspects of design, especially as it relates to process design, need therefore to be considered in order to evaluate properly the validity of a design alternative. But the more detailed the specification of process design is, the more that it is accompanied by physical design and implementation aspects. The incorporation of detail into a design thus makes the evaluation of its acceptability not only possible but also desirable and necessary.

Of course, the sooner that it is discovered that a design alternative is not acceptable, the less the resources that need to be invested in it. How detailed the specification of design and implementation features needs to be in order for the acceptability of a design alternative to be properly evaluated depends, to a large extent, on the kind of application in question. If the process to be supported by the information system has very stringent requirements, the need to specify design and implementation details is correspondingly greater. This may happen when the functions supported by the information system are critical in nature or when the information system has to interact in real time with its users.

A strategy that would help to reduce the amount of resources invested in the evaluation process is to evaluate first those components of the information system that are more critical form the standpoint of acceptability. This strategy calls for the early detection and avoidance of potentially critical problems of acceptability. In any case, once a design alternative has been specified in sufficient detail, the evaluation process can be carried out effectively.

17. A Predominantly Top-Down Approach to Design

The synthetic approach encourages a predominantly top-down design strategy. In the synthesis process, this strategy facilitates the early detection of design alternatives that have inconsistencies with the logical requirements of the information system by focusing first on those aspects of process design that are more critical from the standpoint of validity. This strategy also aims at the reduction of the amount of resources invested in the process of evaluation of the acceptability of design alternatives by focusing first on those aspects of the design that are more critical from the standpoint of acceptability. This top-down approach to design directs the designer’s attention first to the design of the architecture of the information system. This clearly helps the designers to focus first on design aspects that are more critical from the standpoint of both validity and acceptability.
A top-down approach is convenient partly because it is in general easier and more economical to correct, or change, a design that has not been worked out in detail. However, although certain detailed aspects of the design of a proposed information system architecture need to be determined before the evaluation of its acceptability can be done effectively, the proposed design strategy would allow the designers to explore specific design details in the context of particular information system architectures. This, in our opinion, is more focused and effective than the incorporation of detail into the design in a haphazard manner. The context of the information system architecture is particularly useful for the identification and study of detailed aspects of design that are critical to the acceptability of a design alternative. In this respect, the context provided by the consideration of the system architecture facilitates the study of the effect of the designs proposed for a particular component of the information system on other components of the information system, on the organizational functions it supports, and on the organization as a whole.

Let us consider in more detail the strategies for the synthesis and evaluation of information system design alternatives from the standpoint of the design principle. Especially important, in this respect, is that the design principle provides basic, function support-based criteria for the synthesis and evaluation of information system design alternatives. These function support-based criteria are the compatibility of the information system with the structure and the dynamics of the functions it is intended to support and with the adaptability of the organization. But the processes of synthesis and evaluation of design alternatives are, together, intended to translate a set of logical, performance, and user interface requirements into a set of acceptable solutions to the information system design problem. How can this be achieved effectively? The answer is that having in mind the structure and the dynamics of the processes that constitute the organizational units that implement the organizational functions to be supported by the information system is essential to the design of an information system that effectively supports these functions.

Earlier we expressed, as a client/server system, the architectural design of what we called Alternative 1 of the customer order processing system of ABC Manufacturing (according to Pressman [1992], for example, the software design produces a data design, and architectural design, and a procedural design). Let us consider a modular solution but focusing more on procedural design. Let us refer to it as Alternative 2. From the standpoint of the synthetic approach to design, this design alternative represents just another way of configuring a computer-based information system that aims at meeting the logical, performance, and user interface requirements of the customer order processing system in ABC Manufacturing. Notice that the fact that Alternative 2 has a conceptual basis different from that of Alternative 1 illustrates the design flexibility that characterizes the synthetic approach to design.

In the case of Alternative 2 the architectural design can be expressed as a hierarchical structure that in fact represents the structure of the software system, as shown in Figure 16.3. Notice that in the synthetic approach (being a part of the ASM) the
allocation of IPFs to software system components is an essential aspect of architecture design.

The program structure level describes the way in which computer programs are broken down into modules and how the modules composing a computer program relate to each other. More specifically, a computer program can be described in terms of a hierarchy of modules in which the top-level module, also referred to as the ‘main’ module, calls other modules at the next lower level which perform the basic functions of the program. The function of each of these modules can be performed with the help of lower-level modules that perform specific software functions. The ‘calling’ relationship between modules is usually described with structure charts, similar to the control hierarchy shown in Figures 16.4 and 16.5. A program module performs a subfunction of a program’s function. To do this it receives some input parameters and, when its job is completed, returns some results to its caller. The number and type of the input parameters it receives, and the number and type of each data element it returns as a result, constitutes the interface of such module. Module interfaces are clearly an important aspect of program structure. Knowledge about the structure of a program structure helps answer questions concerning the modules composing a program and the functions that each module performs, the interfaces to each module and how the different modules in a program exchange information, the composition of each module in terms of smaller units of processing, and which modules process external inputs and which produce external outputs.

A structured programming approach, similar at the one illustrated here, is typically used to improve maintainability and understandability of software. An object-oriented approach would instead favor portability, extensibility and reusability of code. With the object-oriented approach, however, process design has a distinct conceptual basis and uses its own terminology and notation. The synthetic approach to design allows for the consideration of any kind of alternative solutions to the design problem. Once a set of acceptable alternatives is available, design objectives like the ones just mentioned above can be considered for the selection of the design that will be implemented.

Figure 16.3. Architectural Design of Alternative 2 of the customer order and inventory system of ABC Manufacturing. Notice how the IPFs VO, AC1, PO, and PI are assigned to components of the architecture.
18. Information Systems Design and Structured Methods

In structured methods, the design process is partitioned into output design, input design, file or database design, design of the user interface, and design of data-entry procedures. This is covered in Chapters 15 through 19 in the book. The following are the basic guidelines provided by Kendall and Kendall for each of these design aspects:

*Output Design.* There are six objectives:

1. Design output to serve the intended purpose
2. Design output to fit the user
3. Deliver the appropriate quantity of output
4. Assure that the output is where it is needed
5. Provide the output on time
6. Choose the right output method.

In addition, the output content should be related to the output method used. The book considers a number of factors to be considered when choosing output technology.
**Input Design.** Typical objectives are:

1. Ease of use
2. Effectiveness
3. Accuracy
4. Attractiveness
5. Simplicity
6. Consistency

The book provides guidelines for the design of forms. This includes screen and Web forms, and Internet and Intranet form design.

**File or Database Design.** The general objectives in the design of data storage organization are the following:

1. Purposeful information retrieval
2. Efficient Data Storage
3. Data Availability
4. Efficient Updating and Retrieval
5. Data Integrity.

Conventional files remain a practical way to store data for some (but not all) applications. They however, have some disadvantages such as redundancy, and lack of data independence. Databases, on the other hand, have a database management system associated with them. Effectiveness objectives for databases include:

1. Ensuring that data can be shared among users for a variety of applications.
2. Maintaining data that are both accurate and consistent.
3. Ensuring that all data required for current and future applications will be readily available.
4. Allowing the database to evolve and the needs of users to grow
5. Allowing the users to construct their personal view of the data without concern of the way they are physically stored.

These objectives provide a reminder of the advantages of the database approach. Disadvantages of the database approach include the centralization of data and the difficulty of achieving time and storage costs efficiency.

**User interface design.** The main objectives are the following:

1. The interfaces must allow users to access the system in a way that is congruent with their individual needs
2. Speed must be increased, but errors should also be reduced
3. Must be suitable to users and provide feedback to users from the system
4. Productivity as measured by sound ergonomic principles of interface and workspace design.

Designing Accurate Data Entry Procedures. The main objectives are:

1. Effective and efficient data entry.
2. Effective coding.
3. Assuring quality through validation.

These objectives are adequate at the level of data entry, but they are not related in any way to the functions they support or to other aspects of the design of the system. The function support framework and the ASM, on the other hand, help to put these procedures and techniques and the specific objectives they help to achieve in the proper perspective. Remember that the basic design goals from the standpoint of the function-support framework are the compatibility (both structural and dynamic) of the computer-based information system with the functions it supports and its contribution to the adaptability of the organization as a whole.

19. The ASM and the Use of Traditional Tools of Structured Analysis and Design

The ASM provides tools for the description and analysis of information-processing related aspects of the structure and dynamics of organizations. It does so in a manner that facilitates the development of function-supporting information systems. First, key structural features of organizations that determine basic patterns of control and interaction between organizational functions are described with the help of the OCSM formalism. The information needs of the particular organizational functions are expressed using informational interaction diagrams in terms of the information flows that must exist between these functions and between them and the external environment in order for them to achieve their goals. This interpretation of the informational interaction diagrams constitutes a firm basis for the analysis and specification of the information needs of these functions. Moreover, it puts the information-processing dynamics of organizations in the context of the organizational functions using the information. The LIPS/LIPN formalism allows for the description of the computational structure needed to maintain the information flows shown in the informational interaction diagrams. But since these information flows reflect the needs of specific organizational functions, the information processing functions needed to sustain these information flows can be easily associated with these functions. The explicit relationship between information-processing and function that is so established clearly facilitates the development of information systems that aim at the effective support of specific organizational functions.

The structured methods, on the other hand, provide tools for the analysis and design of information systems that basically address data modeling and process modeling aspects of organizations. The main data modeling tool used in connection with structured methods is the entity-relationship diagrams. This tool allows for the representation of organizational entities that need to be described, and the data structures needed to describe them. Entity-Relationship diagrams also allow for the description of
relationships between entities. The data structures needed to represent the entities shown in the Entity-Relationship diagrams are typically described using algebraic (data dictionary) notation. This facilitates the use of these descriptions in database design, and other systems development activities, such as checking the consistency of data and process models. Data flow diagrams (DFDs) are used to model information processing aspects of organizations. As mentioned earlier, DFDs relate data flows, processes, data stores, and external entities. They can be drawn at various levels of detail and provide logical or physical description of the processes.

But all modeling tools are good if used in the right context and for the right purpose. In particular, the modeling tools used in connection with structured methods can be used in connection with the ASM and the function-support framework. Once the information needs analysis and the information system requirements analysis has been completed for a particular information system application, the tools of structured methods can be used to model specific aspects of data and processing. The function support framework and the ASM helps to put these procedures and techniques and the specific objectives they help to achieve in the proper perspective. Remember that the basic design goals from the standpoint of the function-support framework are the compatibility of the computer-based information system with both the structure and the dynamics of the organization and its functions and its contribution to the adaptability of the organization as a whole. This can be achieved with the synthetic approach to design with the guidance that the design principle provides. Once the architecture of the information system has been determined, as explained in earlier notes, specific design aspects such as those mentioned in the previous section can be tackled more effectively.

20. Process, Database, and Detailed Design in the ASM

I mentioned in earlier notes that we can consider three main levels of information system design: Information system architecture design, process and database design, and detailed design. I also discussed the advantages of a predominantly top-down approach to design, such as the one followed in the synthetic approach, in which information system architecture design is done first. More specifically, in the synthetic approach to design, the acceptability of the architecture of the information system must be verified as early as possible. There are several advantages to verifying that a proposed information system architecture is acceptable, that is, compatible with the structure and dynamics of the functions it supports and with the adaptability of the organization, before process and database design are undertaken. One of the advantages of this is that it allows the early detection of problems at the architecture level thus avoiding situations in which valuable time and resources are spent in process and database design for an information system whose architecture is later found not to be acceptable.

Verifying the acceptability of the architecture design before proceeding with process and database design has additional advantages in the sense that it an acceptable information system architecture leads naturally to the design of an information system that effectively supports the functions of the organization. For example, an information system architecture that is compatible with the structure of the organization would also
be consistent with the adequate support of the implicit (and the explicit) control available in the organization. An acceptable architecture would therefore provide a basis for the adequate distribution of data and processing. But since the structure of the organization influences its dynamics, an acceptable architecture would also provide a basis for the design of the processes, that is, the part of the dynamics of information processing that is realized by the computer-based information system. If the architecture of an information system is compatible with the structure of the functions it supports it actually defines the dynamics of information processing required to effectively support these functions. Among other things, this dynamics is characterized by the degree programmability of information processing that is appropriate, the kind of algorithms needed to implement the information processing functions assigned to the information system, the level of abstraction of the models used by, or incorporated into, the computer-based information system, and how the computational models should interact with their real or simulated environments, are just a few of the characteristics of the computer-based processes that support the organizational functions that the information system aims at supporting. The support of a high-level decision-making function, a strategic planning function for example, usually involves the use of very abstract models that relate high-level functions of the organization to high-level characteristics of the environment such as the state of the economy, the availability of labor, the technology available, etc, A lower-level organizational function, on the other hand, would require for its support computational models that interact with the environment they model. 

But the structure of the organization also influences its adaptability (Kampfner, 1992, 1997). The compatibility of the information system with the adaptability of the organization can therefore be best achieved on the basis of an architecture that is compatible with the structure of the organization. Otherwise, not only the contribution of the information system to the adaptability of the organization would be in doubt, but also its ability to support the functions of the organization on the face of the environment currently faced by the system. Once the compatibility of the information system architecture with the structure of the organization has been proven, however, the information that the structure of the organization provides about its requirements for adaptability can be confidently used in the information system’s process and database design in a manner that contributes to the adaptability of the organization.

20.1 Flexibility of design in the ASM.

The design independence of the requirements specification developed with the ASM gives the information system designers a great deal of flexibility in the sense that any kind of design alternative can be considered. This flexibility applies also to the approach to design and implementation used. In this respect, an important advantage of the ASM is that it uses tools such as the OCSM and the LIPS/LIPN modeling formalisms to capture the relationship between information processing and organizational function. To preserve this relationship, however, a good understanding of the modeling tools used by the particular approach to design an implementation used.
In the case of structured design, for example, the designer can do modular design on the basis of the information flow diagrams of the LIPS specification, of course, taking into account the performance and user interface requirements and making reference when required to the hierarchy diagrams and the informational interaction diagrams of the information needs specification. The systems analyst/designer has the option of drawing data flow diagrams before developing the structure charts. Frequently, the data flow diagrams can be developed and used in connection with CASE tools.

In the case of object oriented design, use cases can be defined with reference to the information needs specification and the information system requirements specification of the ASM, that provide the necessary information about the organizational context. Once the use cases are defined, object oriented design tools such as UML can be used. The layered method of Coad and Yourdon can also be used in connection with the ASM in a similar manner. The main observation, in the case of object-oriented design is that the choice of objects and their associations can greatly benefit from the context provided by the information needs specification and the information system requirements specification of the ASM.

Frequently, already existing packages can be adopted as all or part of a design alternative. The flexibility provided by the design independence of the requirements specification of the ASM is useful in this case as well. In the case of enterprise resource planning (ERP) systems, for example, the information system requirements specification provides the necessary information for the verification of the validity of the design alternatives as they are synthesized. Together with the information needs specification, the information system requirements specification helps to determine the acceptability of the design alternatives. In the case of ERPs, one of the issues of concern, from the standpoint of acceptability is the contribution of the information system to the adaptability of the organization. The problem with ERPs, in this respect, is that they favor the integration of processes which greatly increases the efficiency of the functions. The issue is, however, that there is a potential problem of loss of adaptability due to the loss of subsystem independence that the integration causes.

References