14. The Architecture of the Information System

14.1 Three levels of information systems design

We view an information system as a means of enhancing the ability of a complex system such as an organization to perform its functions. Thus, in our view a computer-based information system is essentially a software system designed to fulfill the information needs of a particular set of organizational functions. Clearly, the requirements of an information system capable of providing effective support to a particular set of organizational functions must be based on the information needs of these functions. This is actually the way in which the requirements of an information system are determined in the ASM. Recall that a specification of information needs describes the information flows that need to be maintained in order for a particular set of organizational functions to be performed appropriately. Relevant characteristics of the organizational units that realize the organizational functions in question must of course also be included in a specification of information needs. An information system that provides effective support to a particular set of organizational functions, on the other hand, must contribute to maintain these information flows so these functions achieve their goals. The requirements of such an information system must define the characteristics that it must have in order to contribute to maintaining these flows in an appropriate manner. This clearly implies the compatibility of an information system meeting this kind of requirements with the structure and dynamics of the functions being supported and with the adaptability of the organization. The goal of the synthetic approach is to design a software system that meets this kind of information system requirements and, consequently, provides effective function support.

An information system design can be described at three main levels: Information system architecture design, process and database design, and detailed design. The information system architecture design level describes an information system in terms of its main functional components. In a banking application, for example, one functional component of the software system may support customer transactions such as deposits, withdrawals, cashing checks, and cash advances with credit cards. Other functional components would deal with the updating of customer accounts, processing customer payments for loans and mortgages, etc., while a third functional component might process the bank’s payroll, and accounting and finance applications. Each functional component of a software system does its job with the help of one or more, typically
many, computer programs. From the standpoint of information processing, however, each computer program included in each functional component of the information system incorporates a particular subset of the IPFs included in the specification of logical requirements.

The information system architecture level is essential from the standpoint of developing information systems that will effectively support a particular set of organizational functions. In fact, some aspects of the design principle, especially those that address the problem of structural compatibility, are most relevant to architecture design. Recall that to provide effective support the information system must have an architecture that is compatible with the structure of the organization. For this reason, although the synthesis process is primarily concerned with producing valid design alternatives, it must also make sure that these design alternatives are compatible with the structure of the organization as well. We will review these aspects of the synthesis process in more detail below. The importance of architecture design is consistent with software engineering practice. Shaw and Garlan (1996) suggest that software architecture is the first step in producing a software design. They distinguish among three design levels: architecture, code design, and executable design (See also Pfleeger, 2001). From the standpoint of information systems design, however, we will keep in mind the three levels of design mentioned earlier.

The process and database design level describes the structure of the data, the decomposition of the functional components of the system into computer programs, and the organization of computer programs into modules. Each module performs a specific function within the program; it communicates with other modules through a well-defined interface. The structure of a computer program is determined by the way its modules relate to each other. Typically, this relationship consists of a hierarchy in which lower-level modules perform sub-functions of higher-level ones. Associated with each program structure is the function assigned to each module and the interfaces between modules. These structural characteristics of programs are typically associated with criteria for the evaluation of software quality such as the strength or cohesion of individual program modules, and the tightness (or looseness) of the coupling between them. The module logic further describes the internal structure of modules. This includes the data structures used by the modules and the way their code is organized.

The design of the database should be done in coordination with process design. In the next section we discuss some design issues that arise at the information system architecture level that are relevant to database design. At this level, design decisions concerning data communications and network design, and design decisions concerning the centralization versus distribution of data and processing is especially relevant.

The detailed design level addresses the code design at a lower level. This level is similar to the executable code level suggested by Shaw and Garlan for software architecture (1996). It is concerned with memory allocation, data formats, bit patterns, etc., and it is clearly dependent on the computer architectures used.
Process design is important to software quality. The way in which the different functions performed by a computer program are assigned to specific modules, for example, is an important determinant of software quality. The way a module’s code is organized is another important factor. Software characteristics such as maintainability, understandability, efficiency, algorithm complexity, and portability are also dependent to varying degrees on the way modules are coded.

It should be mentioned here that, in addition to the development of new software systems, information system designers consider with increasing frequency the use, customization, and integration of already developed software packages and systems as an integral part of the information system design options.

Although the characteristics of software systems and the evaluation criteria mentioned above are important for the development of high-quality software systems, the primary goal of the synthesis process in the ASM is to build computer-based information systems that effectively support specific organizational functions. This is the reason why, in the synthetic approach to design, meeting the requirements of the information system is essential. Ideally, the information system selected for implementation would, in addition to meeting its logical, performance, and user interface requirements, achieve specific software quality objectives of the kind mentioned above.

14.2 The information system architecture level

Myers (1976) defines system architecture as the process of partitioning a large software system into smaller pieces, and considers the components of the system's architecture as levels of abstraction that help reduce the complexity of a system. According to Pressman, an important purpose of the system architecture is to represent the relationship between system elements (Pressman, 1992). By the architecture of a software system we mean the way it is organized into major functional components. These components correspond to high-level functions of the software system and represent major aspects of the support that the information system provides to the organization.

We consider the architecture of an information system as a defining feature of design alternatives that is determined by its basic functional components and by the way they relate to each other. In a banking system, for example, the information system that supports the operation of a branch may be divided into functional components, including one that supports daily transactions and another that supports the updating of customer accounts. The system architecture of the branch operations information system describes the composition of its high-level subsystems, including in this case daily transactions and customer accounts, in terms of the computer programs composing each of these components or subsystems and how they relate to each other.

The architecture of an information system design alternative is determined by its main functional components and their interrelationships. Identifying the main functional components of the software system consists essentially of determining the computational
contents of the computer-based information system. An important characteristic of the architecture of an information system is the particular way in which the IPFs that will be performed by the computer-based information system are assigned to its functional components. This determines, for each design alternative, which IPFs are going to be performed by the computer-based system and which will be performed outside its boundaries. By partitioning this way the set of IPFs considered in the logical requirements (that is, the IPFs underlying the organizational functions to be supported by the information system that are relevant to the application) the boundaries of the computer-based system are also defined.

Data communications and computer networks. A proposed scheme of distribution of data and information processing also determines the type of computer networks required and the need for data communications in the design alternative adopting such scheme. Knowledge about the spatial distribution of users and producers of information is clearly essential, in this respect. The processing throughput requirements associated with each information processing node, or element, in a distributed processing scheme, provides a basis for determining the capacity of the processor assigned to a particular information processing element, and the amount of secondary storage required. Each proposed scheme of distribution of data, together with the decision as to whether the database will be partitioned, or replicated, will in turn determine the corresponding data communications requirements. In particular, it determines the capacity and speed of each data communications line. The tradeoffs associated with these kinds of decisions should be carefully determined so that their impact can be evaluated in the broader context of the acceptability of a design alternative with respect to the information system requirements specification as a whole.

14.3 Some approaches to information system architecture design

Most current approaches to network design, information system architecture design and, more generally, information systems development (see Burch’s SDLC [2], for example) do not take into account structural and dynamic aspects of the parent organization. In these notes I will refer to some important determinants of network and computer architecture design that stem from structural and dynamic features of the organization. Notice that network design, and more generally, the decisions concerning the centralization and distribution of data and information processing depend strongly on organizational issues. The degree of centralization or distribution of data, for example, heavily influences network design, and the type of information processing favored in a given organization. The degree of centralization or decentralization of data and information processing depends also on a number of factors including the philosophy of the organization’s management, the type of organization, and the nature of the functions being supported. A number of issues concerning specific aspects of the centralization/distribution problem are discussed below. However, none of this aspects or factors tells by itself the whole story. Consequently, a final decision must not be the reached before all relevant factors are considered. The complexity of the tradeoffs involved in the various design decisions makes apparent the need of guiding principles.
for the design, and broad enough criteria for the evaluation of alternatives. In this respect, the function-support concept provides a broad enough framework, within which the design principle established in previous notes provides the necessary guidance [8]. Some of the factors affecting the centralization/distribution decision will be discussed first, then the role of adaptability theory and some of the implications of the design principle will be briefly discussed.

According to Powers et al. (1990), although the detailed work of network design is typically performed by communications software and hardware specialists, the project team must provide the support people with the necessary inputs concerning business requirements that affect network design. This input includes such things as an outline of the geographic distribution of sites, the systems functions to be supported at each site, descriptions of types of processing by function and site, response time required for each function, data elements used at each site, and existing network capabilities.

Centralization versus distribution. One basic reason for the centralization or distribution of system capabilities is, of course, the organization's management philosophies. The more centralized the outlook of a given management group, the more likely it is that a centralized system will be implemented (Powers et al., 1990). The nature of the organizational functions to be supported by the information system is another important factor affecting the degree of distribution or centralization of the information system capabilities. The nature of the organizational functions imposes requirements for the centralization of both, processing and data.

From the standpoint of data, questions focus on the location of data, level of detail of data at each location, timeliness of data, and the level of data redundancies. An important consideration concerning decentralization is the cost of data communications. Such costs are reduced if data are located at each site, or where needed for processing. This approach introduces redundancy in the case of shared data. In an extreme case, this would involve replicating all files at all locations. However, this kind of measure would call for specific procedures to maintain, distribute and protect those resources. In this respect, a possible approach is for maintenance to occur at a single site. In this case, procedures are necessary to provide periodic distribution of data to all sites. However, depending on the volume of data and the capacity of the communication network, the transfer of data can be expensive and time-consuming.

From the point of view of processing, centralization or distribution follows functional lines and the handling of data. Separate decisions are usually made for data capture, data entry, mainframe processing, storage, and output. These decisions typically conform to corporate policies, or positions, while attempting meet user needs or desires. A general principle is to incorporate data capture and, if possible, entry functions within user source transactions. Following this principle, an increasing number of systems assign data capture and entry responsibilities to the organization's customers themselves, making them part of the processing cycle. An example of this practice can be seen in automatic teller machines installed by some banks. Decisions concerning the decentralization of the processing of transactions depend on the reliability of the
information system functions and the level of performance required from the system. In this respect, according to Powers et al. (1990), there are no hard-and-fast-rules, just a vast number of available solutions.

Another important aspect is control. In this respect, a basic principle is that it is easier and safer to apply controls at a single location and have a single entry point for all transactions. Operating controls deal with access (entry into and use of the system), authorization of the use of systems functions and data, and verification of successful completion of the processing. Systemic controls, on the other hand, aimed at ensuring that the functional compatibility and data compatibility are maintained throughout the system.

A basic tradeoff found with the centralization/distribution issue is the concentration of computing power through centralization. This type of concentration, however, increases dangers of mechanical, electronic, or accidental disruption of service. An advantage of centralization is the ease with which resources may be controlled. At the same time, centralized facilities are considerably more vulnerable than systems distributed in multiple sites.

**Distributing Processing.** A basic working guideline for distributing information-processing capabilities, as well as for distributing the database, is to minimize communication. W. Leigh (1987) discusses the interrelationship of data and processing, from the perspective of resource sharing, in terms of two basic design objectives.

i) Locate data at the sites that update them most often.

ii) Distribute resources so as to minimize communication.

Leigh describes two types of relationships among processing units: Vertical and horizontal. The horizontal relationship refers to peer-to-peer relationships between units, as in a ring network. The vertical distribution, on the other hand, relies on hierarchical, master-to-slave relationships. Leigh (1987) also considers basically three application types: resource sharing, distributed computation, and remote communication. In the latter, distributed communication networks are actually implemented in many small- and large-scale systems. The others, however, are still only in a formative stage.

**Client/Server systems.** Paul Renaud defines client/server technology as a paradigm, or model, for the interaction between concurrently executing software processes (Renaud, 1993). According to Renaud, client processes send requests to a server process, which responds with results for those requests. Server processes, on the other hand, provide services to their clients, usually by way of specific processing that only they can do. The client process, freed from the complexity and overhead of processing the transaction, is able to perform other useful work. The interaction between the client and server processes is a cooperative, transactional exchange in which the client is proactive and the server is reactive (Renaud, 1993).
Client/server computing has had a rapid growth due to powerful technical and economic forces. Technical trends helping this growth include faster and smaller components, standardization of interfaces, improvements in distributed communications, increasingly interactive user interfaces, and a growing use of enabling technologies. Among the economic forces favoring client/server computing, Renaud mentions relative computational cost advantages, capital cost saving opportunities, commodity cost advantages from standardization and smaller processing elements, reduced opportunity costs from being able to develop new systems faster, and improvements in productivity.

Renaud describes a vision of enterprise-wide computing, based on the client/server computing paradigm, that (according to Renaud) maximizes the productivity of its users. From a function-support perspective, it should be clear that any solution to the problem of distributing data and information processing must be sensitive to the actual information needs of the organization.

14.4 The key role of architecture design

The ASM follows a synthetic approach to the design information systems. In the synthetic approach, the design process, as mentioned above, consists of the synthesis and evaluation of design alternatives. Only those design alternatives that meet the logical, performance, and user interface requirements of the information system and contribute positively to the adaptability of the organization are considered acceptable. The synthetic approach, however, allows for the synthesis of various acceptable design alternatives. This further allows for the selection, for implementation, of a design alternative that in addition to being acceptable achieves specific design objectives and meets specific software quality criteria. The synthetic approach takes advantage of the design flexibility that the design independence of the information system requirements specification obtained in the ASM provides. This flexibility is needed in order to synthesize design alternatives capable of providing effective support to organizational functions. In particular, this flexibility facilitates the use of a top-down design strategy in which architecture design, which is done first, takes a pivotal role. The top-down approach favored by this design strategy is in fact consistent with software engineering practice (see for example Shaw and Garlan, 1996). Essentially, our design strategy aims at improving the effectiveness of the design process by facilitating the detection of problems with the acceptability of design alternatives as early as possible.

In order to be acceptable, a design alternative must be valid, that is, it must satisfy the logical requirements of the information system. Verifying the validity of the design alternatives ensures that the computations that will be performed by the information system are specified correctly. The top-down design strategy requires for the architecture of the information system to be designed first, and its validity verified. The design principle mentioned earlier provides a framework for the integration of all the design aspects into an information system architecture capable of providing effective function support. The design process starts with the synthesis of design alternatives. Starting with the synthesis process the guidelines provided by the design principle should be carefully
followed so as to ensure the compatibility with the functions and the contribution to adaptability of each design alternative synthesized.

The compatibility with the structure of the organization and with its adaptability are especially important from the point of view of information system architecture design. But although the main concern of the synthesis process is with the validity of the design alternatives considered, their acceptability should also be taken into account. Otherwise, there would probably be too many valid alternatives that would have to be discarded later because of their lack of acceptability. Let us discuss in more detail the issues of compatibility with the organization and its adaptability in the context of architecture design and the synthesis process.

14.5 The Architecture of Function-Supporting Information Systems

Let us illustrate the issue of structural compatibility in the context of the synthesis of design alternatives for the customer order processing system of ABC Manufacturing. What we need, from the point of view of information system architecture design, is to determine the main functional components of the software system and how they should relate to each other. Especially important from the standpoint of designing an information system architecture that is compatible with the structure of the organization is to establish a relationship between the functional components of the software system that is consistent with the way in which the organizational functions they support relate to each other. The reason is that the structure of a system provides a basis for its dynamics including, of course, the dynamics of information processing, that is, how information is processed and communicated throughout the system. The architecture of the information system provides the basis on which the information it handles is processed and communicated throughout the organization. Since the functional components of the information system support specific functions of the organization, the way in which these components relate to each other must be consistent with the relationship between these functions. For example, in the case of ABC Manufacturing, any proposed information system component that verifies customer orders must relate to the component that process the approved orders in the same way in which the corresponding organizational functions of ABC Manufacturing relate to each other. The fact that the orders are received by the various sales branches but that once they are approved they all undergo a similar order fulfillment process is an aspect of the relationship between functions that is relevant to the design of the information system architecture. In this case, structural aspects of the function such as the fact that the sales function is organized into five branches, the physical location of the branches, together with the way in which the shipping, the inventory, the purchasing and the manufacturing are organized, relate to and communicate with each other, together determine how the information needed by these functions must flow through the organizational units implementing them.

Recall that in the ASM the information needs analysis captures the relationship between organizational functions, a key structural feature of the organization captured by
the OCSM and expressed in the form of hierarchy diagrams. On the basis of these structural features, the information needs analysis captures also the relationship between information and organizational functions and expresses it in the form of informational interaction diagrams. The Information system requirements analysis, on the basis provided by the information needs specification, proceeds with the determination of these requirements which are, in fact, the characteristics that the information system must meet in order to fulfill the information needs of the functions, in other words, to provide effective function support.

Figure 14.1 Hierarchy diagram of the Marketing unit. The Sales unit, the Order Preparation and Verification Unit, the order Processing unit, and some control units are shown. The Controls and the Reports-to relations are explicitly indicated. The Subsystem relation is shown graphically by placing subsystems inside their parent systems.
Figure 14.2 Informational interaction diagram for the order fulfillment system of ABC Manufacturing. The interactions with other units such as the Manufacturing unit are not shown.
Figure 14.3 Logical requirements of the organizational functions to be supported by the customer order and inventory system of ABC Manufacturing.

Particularly important for the synthesis process are the logical requirements of the information system that, in the ASM, are determined with the help of the LIPS/LIPN formalism (Kampfner, 1985). The information flow diagram of Figure 14.3 depicts the logical requirements of the organizational functions to be supported by the customer order processing system of ABC Manufacturing. Of course, more detailed information about these requirements should be included in the supporting documentation. The architecture of a design alternative of this system, based on a client/server approach to computer system architecture, is shown in Figure 14.4. Notice how the synthesis process assigns IPFs to each component of the computer-based information system. This determines a number of design tradeoffs that have to be assessed on the basis of the performance and user interface requirements and their impact on the adaptability of the organization when the design alternatives are evaluated for acceptability.

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. The synthesis process, although based primarily on the logical requirements of the information system, takes into account the performance and the user interface requirements of the information system as well. 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, the verification of acceptability does take place as a separate evaluation process. An important reason for this is that it is indeed more economical and efficient to direct most of the effort needed to verify the acceptability of design alternatives to those that have already been found valid. It is important to notice, however, that, as implied by the top-down design strategy recommended here, it is convenient to determine the validity of a design as early as possible as it is synthesized, so that the designer’s attention can be directed toward the consideration of its acceptability. The idea is to focus first on those aspects of design that are more critical from the standpoint of validity. This facilitates the early detection of schemes of process organization that are not consistent with the logical requirements of the information system. Because of the impact it has on the rest of the design, information system architecture design is especially important. 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. The process of evaluation for acceptability, that requires the consideration of the performance and user interface requirements as well as the consideration of the impact of the information system on the adaptability of the organization, can be carried out more effectively when only the acceptability of individual modules needs to be verified.

As it is the case with the synthesis of design alternatives, the design principle provides also the necessary guidance for the evaluation of their acceptability. The evaluation of the acceptability of a design alternative starts in full when its validity has been determined. In order to determine the validity of a design alternative, however, the processing to be done by each component of the information system needs to have 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. A detailed design, especially as it relates to process design, is therefore needed in order to evaluate properly the validity of a design alternative. But the more detailed the specification of the information processing is, the more 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.

REFERENCES


Figure 14.4 Schematic view of Design Alternative 1 of the customer order and inventory system of ABC Manufacturing.