Chapter 4

Designing and Developing Decision Support Systems

INTRODUCTION

In the Decision Support Systems (DSS) literature, experts prescribe a variety of approaches or methodologies for designing and developing DSS. Everyone does not, however, agree on what methodology works best for building different types of DSS. If managers and DSS analysts understand the various methods, they can make more informed and better choices when building or buying a specific DSS.

In general, what is called a "decision-oriented approach" seems best for Decision Support Systems projects. After reviewing design and development issues, decision-oriented diagnosis, and feasibility studies, this chapter reviews three alternative approaches for developing a DSS. Because the scope of DSS is expanding, and because development tools are changing rapidly, the perceived advantages of the three alternative development approaches have become somewhat controversial. For example, a highly structured life-cycle development approach has recently become popular with some consultants for developing enterprise-wide DSS. The advantages and disadvantages of each development approach are discussed. The final sections of this chapter discuss outsourcing DSS, project management, and the various participants on a DSS design and development team.

OVERVIEW OF DESIGN AND DEVELOPMENT ISSUES

How does one plan and implement a new DSS? What does it mean to design a DSS? How does one develop a DSS? Who develops a new DSS? When should a company build a DSS and when should a company buy a DSS package? Both managers and Management Information Systems (MIS) professionals need to explore these questions. A company does not receive any

advantage from a great idea for a DSS until the new system is built and successfully implemented.

Many Information Systems (IS) professionals develop, modify, and customize software to support decision making. They work in diverse business and organization settings and in specialized DSS software companies. DSS software vendors sell a wide array of products and provide DSS development services. For example, Comshare (www.comshare.com) and Cognos (www.cognos.com) both market business intelligence and management planning and control products.

Design and development is an important topic because DSS serve many different functions and are quite diverse in terms of the software used for their development. Choosing an appropriate approach or methodology for building DSS has been a popular and controversial topic in the Information Systems (IS) literature. Many consulting firms focus on using what they claim is the most effective development methodology. We can define a methodology as an organized set of practices and procedures used by developers. Despite many differences in methodologies and terminology, the prescriptions in the IS literature have generally followed three different conceptual paths.

One group of MIS and DSS experts develop their recommendations for building DSS in the context of the traditional systems analysis and design literature (cf., Thierauff, 1982). A second group has prescribed and explained an iterative, prototyping, or "quick-hit" approach for designing and developing DSS (cf., Sprague and Carlson, 1982). Some authors refer to both types of approaches without explaining clearly the advantages and disadvantages or contingencies favoring a specific approach or some combination of approaches. A third approach to building DSS, called end-user development, is to let managers develop their own personal DSS. In general, the DSS prescriptive literature on design and development is based on personal experiences, case studies, the general IS development literature, and a wide variety of DSS "war stories" from developers. Very little empirical research has been conducted on design and development methodologies.

Because of design and development problems, some highly innovative and potentially useful DSS have been failures. The problem often is that the DSS are designed and developed from the perspective of the programmer and developer rather than from that of the manager and user. Sequences of commands or icons may be obvious to the programmer, but may be totally unknown and puzzling to the DSS user. From a prescriptive standpoint, effective DSS need to be user-oriented. The key issue is what design and development process and which procedures can increase the likelihood that a usable and effective DSS will be created and built.

Building DSS is often very expensive. So, it is important to investigate alternative design and development approaches. We want to choose an approach that increases the chances the DSS will be used and will accomplish its purpose. We need to remember DSS are designed and developed to help people make better and more effective decisions than they could without computerized assistance. Building any type of DSS is difficult because people vary so much in terms of their personalities, their knowledge, their ability, their preferences, the

jobs they hold, and the decisions they need to make. Also, DSS must often meet a diverse set of requirements. This wide variety of differing requirements has led to the design and development of a wide variety of DSS capabilities and systems.

The following discussion separates out the diagnostic design and feasibility portion of an overall systems development process. The phrase systems development life cycle (SDLC) is the most commonly encountered term used to describe the steps in a traditional systems development methodology. SDLC is also sometimes known as the applications development life cycle approach and involves three steps: (1) initiation and diagnosis, (2) acquisition (build or buy), and (3) introduction of the new system.

As mentioned above, the two commonly prescribed alternatives to the SDLC development approach are a prototyping approach and end-user development of DSS. In both of these approaches, a portion of the DSS is quickly constructed, then tested, improved, and expanded. Prototyping is similar to a related approach called rapid application development (RAD).

DECISION-ORIENTED DIAGNOSIS

Increasing decision-making effectiveness through changes in how decisions are made should be the major objective for any DSS project (cf., Stabell 1983). Stabell proposes a decision-oriented design approach for DSS. He argues that predesign description and diagnosis of decision making is the key to securing a decision-oriented approach to DSS development.

The diagnosis of current decision making and the specification of changes in decision processes are the activities that provide the key input to the design of the DSS. Diagnosis is the identification of problems or opportunities for improvement in current decision behavior. Diagnosis involves determining how decisions are currently made, specifying how decisions should be made, and understanding why decisions are not made as they should be. A specification of changes in decision processes involves choosing what specific improvements in decision behavior are to be achieved. These statements of improvements provide the objectives for the DSS development.

According to Stabell, diagnosis of problems in a decision process involves completing the following three activities:

1. Collecting data on current decision-making using techniques such as interviews, observations, questionnaires, and historical records;
2. Establishing a coherent description of the current decision process; and
3. Specifying a norm for how decisions should be made.

These activities are interdependent and provide feedback for the DSS analyst. In many DSS development projects it is not feasible to perform a full-scale diagnosis of decision making. A shortened study is often necessary due to cost considerations, limited access to managers, or other organizational constraints. As a consequence, DSS analysts should develop the ability to produce a diagnosis after only limited analysis of a decision situation.

A related diagnostic activity is conducting a Decision Process Audit. In general, it can be very useful to audit operational and managerial decision processes. An audit can be a first step in identifying opportunities to redesign business processes and include new decision aids and DSS in business processes. In some situations an audit can suggest changes in decision technologies that can improve performance and reduce costs. When an audit is completed, the central questions should be how can we do better and what changes should have the highest priority. Table 4.1 identifies five steps in a Decision Process Audit.

	Decision Process Audit Plan
Step 1.	Define the decisions, decision processes and related business processes that will be audited. Define the authority of the auditor, purpose of the audit, scope of the audit, timing of the audit, and resources required to perform the audit. Identify a primary contact.
Step 2.	Examine the formal design of the process. Diagram the process using Data Flow Diagrams and specify participants, data, criteria, etc.
Step 3.	Examine the actual use of the decision process. Observe the process. Interview decision makers and collect data. Is the process implemented and used as intended?
Step 4.	Assess performance of the actual decision process. What works? Can cycle time be reduced? Are decisions appropriate? Timely? Cost effective? Is the process producing value in meeting business objectives? If not, why?
Step 5.	Reporting and recommendations. Summarize steps 1-4 in a written report. Discuss what is working well and what needs to be improved. Develop recommendations for improving the process. Hold an exit meeting with decision makers.

Table 4.1. A Decision Process Audit Plan.

Creating a Data Flow Diagram (DFD) is an important step in a Decision Process Audit. A DFD graphically depicts a business process and the flow of data through the process. To create the DFD, the DSS analyst decomposes the process that is being investigated into small steps, actions or events. The steps may occur sequentially or in parallel. A DFD is particularly useful for enhancing communication between a DSS analyst and managers.

An audit should also focus on identifying what is assumed by decision makers in a decision situation and on what is defined by decision makers as the range of available remedial actions. Identifying assumptions and actions is especially appropriate if building a model-driven DSS is a possibility. Assessing the performance of the actual decision process is an important task in the audit. One needs to determine what tasks are effective. Can cycle time be reduced? Are decisions appropriate? Timely? and Cost effective?

Some processes like business planning need improved data access and analysis to increase business intelligence. Rockart (1979) identified an approach for defining decision-making data needs that is appropriate for data-driven DSS and especially Executive Information Systems (EIS). Rockart's Critical Success Factors (CSF) Design Method focuses on individual managers and on each manager's current hard and soft information needs. A CSF analysis can be beneficial in identifying "the limited number of areas in which results, if they are satisfactory, will insure successful competitive performance for the organization." If organizational goals were to be attained, then these key areas of activity—usually three to six factors—would need careful and consistent attention from management.

Good diagnosis is difficult, but DSS diagnosis involves skills that can be developed and sharpened. Both managers and MIS staff need to work on completing the diagnosis task. Does diagnosis always provide sufficient information for specifying a DSS? In most cases, the diagnosis does provide sufficient information for specifying several alternative designs. DSS design usually involves a number of difficult trade-offs. The first trade-off is whether the DSS should support both the existing process and a prescribed new process. There is also a trade-off in the extent of the capabilities of the DSS and the scope of the process the DSS is designed to support. In most cases, the initial version of a DSS focuses on either extensive capabilities for a narrow scope process or on a few capabilities for a broad scope process.

PREPARE A FEASIBILITY STUDY

Diagnosis of decision making should be followed by additional initiation and diagnostic activities and preparation of a feasibility study of the technical and economic prospects related to developing a DSS. This study should occur prior to actually committing resources to developing a proposed DSS. What should be included in a DSS feasibility study? This is a common question. An outline for an extensive feasibility study report is included in Table 4.2. The outline has 15 sections on topics like DSS scope and target users, anticipated DSS impacts, benefits, risks and mitigating factors. Shorter, less comprehensive studies and reports are usually prepared for small scope DSS projects.

A Decision Support System feasibility study examines a proposed project's consequences and impacts. A feasibility study is summarized in a formal report or document. The study addresses issues including the project's benefits, costs, effectiveness, alternatives considered, analysis of alternatives, opinions of potential users, and other factors. This feasibility analysis is a way of exploring the factors and risks affecting the potential for successful development and implementation of a DSS. Large-scale information systems development efforts typically include a feasibility study as a major checkpoint providing critical information about whether it is possible to develop a system, given the project's goals and constraints. This report should be framed to offer important information about the range of issues likely to affect success and therefore should be considered in decisions about whether and how to move forward with a DSS development effort.

I. **EXECUTIVE SUMMARY**
- A. Key Business Needs
- B. Issues
- C. Solutions
- D. Benefits and Costs
- E. Critical Success Factors
- F. Project Management

II. **INTRODUCTION**
- A. Background and Definitions
- B. Key Questions
  1. Site Readiness: To what extent is the company ready for and interested in implementing a new Decision Support System? See Appendix II.
  2. Technical Feasibility: Is it possible to develop or adapt software to perform the proposed types of analyses.
  3. Financial Feasibility: What are the projected costs of implementing the DSS, and do potential benefits justify these costs?
- C. Feasibility Study Approach

III. **BACKGROUND NEEDS AND ASSESSMENT**
- A. Goals
- B. Constraints
- C. Related Projects
- D. Business Decision Support Needs
- E. Decision Support Diagnosis

IV. **OBJECTIVES**

V. **DSS SCOPE AND TARGET USERS**
- A. Scope and Decision Process Definition
- B. Scope Recommendation
- C. Scope Issues

VI. **ANTICIPATED DSS IMPACTS**

VII. **PROPOSED SOLUTION**
- A. System Integration Issues
- B. Major Functions Provided
- C. Technology Tools/Infrastructure Used
- D. New Organizational Structure and Processes

VIII. **MAJOR ALTERNATIVES**
IX. **CONFORMITY WITH CURRENT IS/IT PLAN**
X. **PROJECT MANAGEMENT AND ORGANIZATION ISSUES**
XI. **ESTIMATED TIME FRAME AND WORKPLAN**
XII. **INCREMENTAL COSTS AND BENEFITS**
XIII. **RISKS AND MITIGATING FACTORS**
XIV. **DRAFT CONCEPTUAL DESIGN**

Table 4.2 DSS Feasibility Study Outline.

After completing a feasibility study, a decision is often made between purchasing an application package and in-house development. In general, packaged DSS applications are quite versatile and are usually less expensive to implement than in-house development. Packaged solutions are also often faster to implement. In addition, a packaged DSS may reduce political problems if a DSS project fails. The problems associated with purchasing a packaged DSS should not however be ignored. A package may not "fit" the needs that have been identified, and competitors can also purchase a package. Using a packaged application is less likely to create a competitive advantage. Customizing a packaged application can sometimes overcome these problems and limitations.

CHOOSE A DEVELOPMENT APPROACH

As noted in the overview, three approaches to DSS development are discussed in the IS and DSS literature and are used by practitioners. The approaches or methodologies have been called by a variety of names. Essentially, we begin by focusing on decisions and decision processes in the decision-oriented design steps; then, a project manager or an end-user implements a more or less structured development methodology.

Figure 4.1 shows a recommended process hierarchy for DSS design and development. The process begins with a decision-oriented diagnosis and feasibility analysis and then moves to in-house or outsourced development of the proposed DSS using one of three development approaches. Let's examine these alternative approaches.

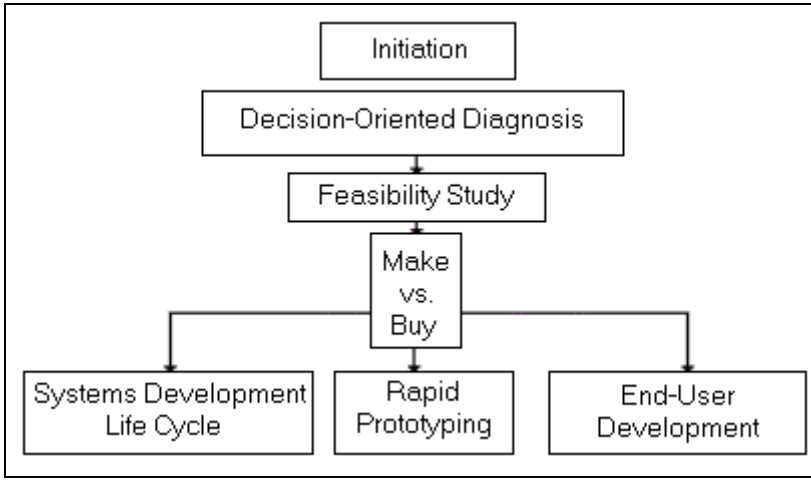

Figure 4.1. A DSS Design and Development Hierarchy.

Systems Development Life Cycle Approach

The systems development life cycle (SDLC) approach is based on a series of formal steps, including these seven steps:

1. Confirm user requirements;
2. Systems analysis;
3. System design;
4. Programming;
5. Testing;
6. Implementation; and
7. Use and Evaluation.

Although different versions of SDLC vary in the precise number of steps and in the detailed definitions of those steps, those listed above illustrate the approach. Decision-oriented design begins to address user requirements, but, in SDLC, user requirements need to be defined in great detail.

This formal SDLC approach is sometimes called the "waterfall" model because of the sequential flow from one step to another. Each formal step concludes with preparation of a written progress report that must be reviewed and approved. Reviewers include both prospective users of the system and developers. For example, in Step 5, prospective users verify that the documented functions and capabilities and the user interface meet their needs. Developers verify that the system's internal interfaces are consistently defined and meet all technical requirements.

When the SDLC approach was first formalized in the mid-1970s, it provided structure and discipline to system developers. It was soon adopted widely for developing large-scale TPS. SDLC is especially common when a formal contractual relationship exists between the developers of an application system and its eventual users because it provides written evidence that can be used to arbitrate any disputes.

The development of a large, shared, enterprise-wide DSS is often an undertaking of great complexity. Organizational decision processes are complex, and computerizing systems to support them can increase that complexity. Using a methodology like SDLC provides one way in which business organizations can systematically approach the development of an enterprise-wide DSS.

When the SDLC approach is used, then project plans must be carefully prepared. When developing requirements, it is best to start by determining the needs of all potential users. Then, analysts should identify the outputs that would fulfill those needs. Technical requirements should follow logical requirements, and constraints must be identified for all DSS system components. These requirements must be documented carefully and reviewed by the targeted users.

Several alternatives may exist for meeting the needs identified during the requirements and design steps. Each of these alternatives should be carefully reviewed and the best one chosen. Another choice to be made concerns the "make or buy" decision. If in-house development is not chosen, a request-for-proposal (RFP) may be required. During the design stage, technical processes

must be managed, people and procedures prepared, and an installation plan developed.

In many situations a full-scale SDLC approach is too rigid for building DSS, especially those DSS whose requirements are changing rapidly. User requirements, agreed upon at the first stage of the process, are rigidly specified with SDLC. Any significant change restarts the entire development cycle, as subsequent requirements documents are based on the agreed-upon user needs. Changes are therefore often expensive; in fact, SDLC limits change in a DSS rather than accommodating it.

Rapid Prototyping

All of the different versions of rapid prototyping accommodate and even encourage changes in the requirements of a proposed Decision Support System. A typical prototyping methodology usually includes five steps:

1. Identify user requirements.
2. Develop and test a first iteration DSS prototype.
3. Create the next iteration DSS prototype.
4. Test the DSS prototype and return to step 3 if needed.
5. Pilot testing, phased or full-scale implementation.

The prototyping development approach evolved in response to perceived deficiencies and limitations of the SDLC approach. In a prototyping development approach, DSS analysts sit down with potential users and develop requirements. These requirements are specified in general terms and should evolve from the decision-oriented diagnosis and design. The analyst then develops a prototype of a system that appears to meet the requirements. DSS analysts use tools such as Database Management Systems and DSS application generators that support rapid development. Analysts focus on capabilities rather than on resolving problems. A prototype may not resolve how to access a real database, what "help" screens are needed, or define other capabilities that require extensive development time. The prototype is something that users can try out, react to, comment on, and eventually approve with confidence that it meets their needs. Missing features are added later, once users are satisfied with the way the prototype works. Rapid Application Development (RAD) specifies incremental development with constant feedback from potential users. The objective of RAD is to keep projects focused on delivering value and to keep clear and open lines of communication. In most situations, oral and written communication is not adequate for specification of computer systems. RAD overcomes the limitations of language by minimizing the time between concept and actual prototype implementation.

Once approved, a prototype can be expanded in the development environment, or the prototype can be used as a specification for a DSS, developed in a language like Java, C, or C++. When a prototype is reprogrammed, the prototype serves as a detailed specification that is turned into an operational system. The best prototype development approach is to have the

actual prototype evolve directly into the finished product. In this approach the prototype is attached to a database and features are added to it, but it remains written in the high-level tools originally used for prototype development.

Compared with the SDLC approach, prototyping seems to improve user-developer communication. It introduces deliberate flexibility and responsiveness into the development process. Change is no longer something to be avoided; it is built into the process and encouraged. The system that is developed is more likely to meet user needs than is a system developed through SDLC.

Prototyping can extend the development schedule if it is improperly used. Managers and developers are often tempted to "tinker" with a DSS and make changes that do not really improve the usability of the finished product. If managers and developers want to build a useful system and meet project deadlines, then they must manage and control systems development efforts.

End-User DSS Development

End-user development of DSS puts the responsibility for building and maintaining a DSS on the manager who builds it. Powerful end-user software is available to managers, and many managers have the ability—and feel the need—to develop their own desktop DSS. Managers frequently use spreadsheets, like Microsoft Excel and Lotus 1-2-3, as DSS development tools. Using a spreadsheet package, they can analyze an issue like the impact of different budget options. Following the analysis, managers select the alternative that best meets their department's needs. Also, managers can develop tools to help them conduct market analyses and make projections and forecasts at their desktop.

The major advantage of encouraging end-user DSS development is that the person who wants computer support will be involved in creating it. The manager/builder controls the situation and the solution that is developed. End-user DSS development can also sometimes result in faster development and cost savings.

End-user DSS development of complex DSS is much less desirable. Managers are paid to manage, *not* to develop DSS. At some point DSS specialists can do the work much better and much faster. Also, managers are not trained to test systems, create documentation, provide for back-up and data security, and design sophisticated user interfaces. DSS analysts should help managers develop more complex end-user Decision Support projects. DSS analysts can help the manager build, document, and test the application. Managers need to emphasize the content of the DSS and not become overly involved with extensive DSS development projects.

End-user DSS development is a controversial topic. IS staffs have many concerns including:

1. End-users may select an inappropriate software product or hardware platform as a development environment.
2. The end-user may have limited expertise in the use of the product, and the IT group may have limited resources to support end-user development.

3. Errors during end-user DSS development are frequent. Even experienced developers can make errors, and end-users are likely to overlook the need for checking formulas and auditing the DSS they have developed.
4. Unnecessary databases are sometimes developed by end-users for their DSS. Redundant databases can contain outdated and inaccurate data.
5. A major quality issue involves testing and limited documentation. End-users often perform only limited testing of DSS they develop, and they have limited experience in documenting applications.
6. End-user databases may be poorly constructed and difficult to maintain.
7. End-users rarely follow a systematic development process. Some needs and requirements may be overlooked.
8. An end-user developer may leave a company; the DSS then becomes difficult to support.
9. An end-user-developed application may not work when many concurrent users are trying to access its capabilities.

If an organization's MIS group gets actively involved in supporting end-user DSS development, many of the above problems can be minimized, reduced, or eliminated. Packages used for end-user development can be standardized; end-users can be trained in the use of selected packages; support staff can act as consultants and reviewers; a central databases can be maintained for use with end-user applications; and documentation can be encouraged by MIS staff.

One approach is to create an information center. An information center can provide support for end-users and the director of the information center may be able to manage end-user computing. Services that an information center might provide include: software training; user support, including answering specific development questions; installation assistance and advice about new systems; and setting standards for documentation, software application, and distribution of applications. Choosing either SDLC or a prototyping development approach requires selection of a project manager. Let us now examine DSS project management issues.

DSS PROJECT MANAGEMENT

Moving from an informal exploration of a suggestion or desire for a DSS to a formal project is an important step. An executive sponsor should push to have a project manager assigned to the project. The initial tasks of the project manager include diagnosis, a feasibility study, and a definition of the objectives and scope of the proposed project. Once these steps are done, the executive sponsor needs to choose to continue the project or postpone any further work on it. Depending upon the scope of the DSS project, an executive sponsor may be able to directly fund the project, or funding may be budgeted as part of business and information systems planning. The larger the scope of the proposed project, the more important it is to solicit widespread agreement and sponsorship of it. The objectives of a large-scope DSS project must be strategically motivated, should have strong executive support, and must meet a business need. Large-scope projects may benefit from having co-project managers: a business and a

technical manager. If co-managers are designated, clear authority and responsibility guidelines should be established.

Once a project is approved, then a methodology and project plan needs to be developed and a project team should be assembled. If the project will be outsourced, then a process needs to be developed for creating a request for proposals and then evaluating the proposals submitted. If the development will occur in-house, development tools and technical issues need to be resolved. (The feasibility analysis should have determined if the project could be completed in-house.)

User requirements need to be specified in some detail. For large projects, the DSS architecture must be specified and any changes or additions to the Information Systems and Information Technology (IS/IT) infrastructure must be planned. Once these crucial preliminaries are completed, then systems design or prototyping can occur. The project tasks will not be completed in a simple, linear sequence, and the project manager must actively manage the project. Whenever possible, the project manager and, in some cases, a co-project manager from the business area most affected should consult and work with other potential users. The project manager must keep the executive sponsor informed. If problems are occurring or might occur, the sponsor needs to be alerted.

The project manager should identify tasks that must be completed, resources that are needed, and project deliverables. Deliverables are especially important for monitoring the progress of the project. Milestones or important project events are also often identified to help nontechnical managers monitor a project. The Chief Information Officer (CIO) of a firm and one or more business managers usually monitor the progress of a large-scope or high-visibility DSS project. Managers expect results from DSS projects. Understanding and meeting the expectations of managers who will use a DSS is the most important and most difficult part of a DSS project manager's job.

The project manager defines project plans and manages the daily activities associated with the project. She also coordinates project resources, the project budget, status reporting, changes in requirements and tasks, relations with vendors, and relations with sponsors, skeptics, and MIS staff. A DSS project manager may come from information systems or from a functional department and needs strong technical skills, outstanding people skills, assertiveness, and knowledge of the business.

Outsourcing

Outsourcing involves contracting with outside consultants, software houses, or service bureaus to perform systems analysis, programming, or other DSS development activities. The outsourcer should be evaluated as a long-term asset and as a source of ongoing value to the company. Time and resources need to be dedicated to managing the relationship and maximizing its value. The customer needs a project manager to manage the outsourcing relationship. The intent should be to keep the relationship for as long as it brings value to the customer. Over time, new technology alliances may be required as technology

and organizations change. Therefore, a customer should strive for long-term relationships and work to align the outsourcer's motivation with company goals by developing appropriate incentives and penalties.

Outsourcing DSS projects has a number of risks. First, a company relinquishes control of an important capability to an outside organization. Second, contracts for DSS services may be long term and may lock a company into a particular service provider. Finally, a reliance on external sources for new systems development can lead to low technical knowledge among in-house MIS staff.

Some of the benefits of outsourcing include potentially lower cost development, access to expertise about new technologies, and "freeing up" company resources for other projects. The high risks, however, often lead to in-house DSS development rather than to outsourcing. When does outsourcing seem to work? Outsourcing can be successful when a company needs to turn around DSS activities quickly and when a company's MIS staff seems unable to build innovative DSS in-house. In some companies this situation exists today for Web-based DSS.

DSS PROJECT PARTICIPANTS

A complex DSS, built using either an SDLC or a prototyping approach, requires a team approach to development. Once the system is developed, a group may also need to maintain the system. Some large-scale DSS are built with teams of two or three people, or with a larger group of ten or more. Members of DSS teams are drawn from many areas in an organization, including the IS group.

Any DSS development project requires a mix of complementary skills. Usually, one does not find all of the needed skills in one person. So, in most situations, it is necessary to assemble the right mix of contributors for a DSS project team. The key DSS development roles identified by Sprague (1980), O'Neil et al. (1997), and others are listed below in order of increasing technical expertise. Figure 4.2 summarizes the various roles. A given individual may be assigned more than one role.

Project Manager. This is a business or technical manager who can organize and manage the resources needed to complete the DSS project.

Executive Sponsor or Project Champion. This is a senior manager who has access to other senior executives and has the influence to help resolve major resource and political problems. The sponsor is occasionally actively involved in the development tasks.

Potential DSS User(s). This is a person who makes decisions that a proposed DSS will support. Users are often nontechnical people in functional areas of a business such as marketing and finance.

DSS Analyst. This is the MIS specialist who acts as an intermediary or liaison between users and DSS developers. A DSS analyst may make the decisions about the software tools to use, the hardware platforms to use, the models and/or databases to incorporate into the DSS, and how they will be integrated with each other. This is generally a person with a great deal of

experience who understands both the business problem and the available technologies. A DSS analyst often works gathering requirements, analyzing solutions, writing specifications, maintaining product information as well as assisting in training and documentation support. A DSS analyst often works with users to define and document system requirements for Decision Support Systems. A DSS analyst may help redesign business processes to better use a computerized Decision Support System. This person sometimes also assumes the role of project manager.

Technical Support Staff. A number of MIS professionals are involved as technical support staff including data warehouse architects, network specialists, application architects, operations researchers, and developers. A data modeler and data quality analyst are often involved in building data-driven DSS. The data quality analyst is concerned with data integration, metadata, and data scrubbing. A database administrator is an integral part of a development team for a data-driven DSS project. Data administrators, systems administrators, and networking specialists are often consulted on DSS projects and may join a development team for some projects.

DSS Toolsmith. This is a specialist with the tools and technologies that will be used in the construction of the DSS and the packages that will be combined to create the DSS. He or she is an expert on these tools and packages, and their effective use. This is the person who creates underlying capabilities and integrates existing packages into one overall system and carries out custom programming that contributes directly to DSS functionality. His or her responsibility begins with the packages that will comprise part of the DSS and ends with completion of a specific DSS.

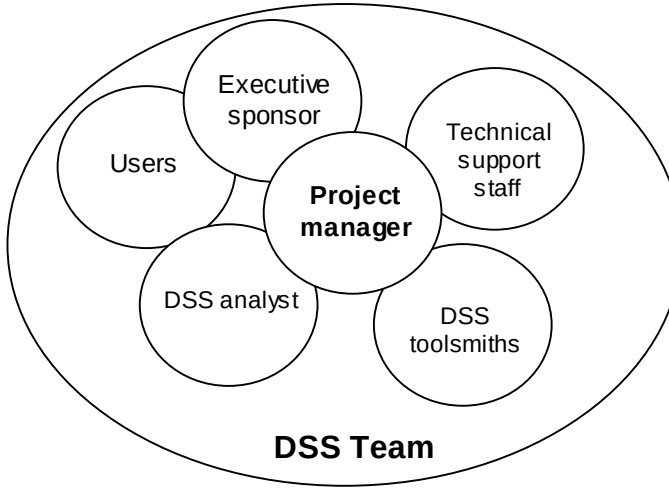

Figure 4.2. Participants on a DSS Development Team.

The composition of a DSS team may change over the development cycle, so the project manager and DSS builder need to provide direction and motivation for the DSS team. Also, the executive sponsor needs to maintain an active commitment to the project. Losing a project sponsor can harm and even doom a DSS project.

CONCLUSIONS AND COMMENTARY

In 1985 Jack Hogue and Hugh Watson surveyed managers in organizations with DSS. Each participant was an active DSS user. Two-thirds of the organizations had built their DSS using an evolutionary, prototyping approach and the remaining organizations had used more of an SDLC approach. It appeared that if the DSS supported managers throughout the company, or if it required company-wide data, then the SDLC approach was used. The evolutionary approach was used for smaller-scale systems where a DSS development tool was available. Nine of the eighteen companies used DSS generators to develop their systems. This finding is probably descriptive of current practice.

When managers could specify information requirements in advance, then the SDLC approach was more likely to be used. Hogue and Watson also found that when IS specialists developed the DSS, then SDLC steps were more likely to be followed. Senior managers reported they were most involved in the idea, information requirements, and acceptance steps associated with building a DSS. Middle managers reported they were somewhat involved in all of the steps involved in building the DSS they were using. When prototyping and evolutionary design was used, managers reported more involvement in the design and development process. The IS group was usually involved in building the DSS, but staff from an IS department were rarely in a leadership role. Potential users of the DSS usually assumed the leadership role.

The DSS design and development approach that is used for a new DSS project should depend on the amount of data needed and its sources, the number of planned users, any models and analytical tools used, and the amount of anticipated use. Many small, specialized DSS are built quickly using end-user development or rapid prototyping. Large, enterprise-wide DSS are built using sophisticated tools and systematic and structured systems analysis and development approaches. Creating enterprise-wide DSS environments remains a complex and evolutionary task. An enterprise-wide DSS inevitably becomes a major part of a company's overall information systems infrastructure. Despite the significant development differences created by the scope and purpose of a DSS, all DSS have similar technical components and share a common purpose—supporting decision making.

A number of authors suggest that the perceived usefulness and the perceived ease of use of an IS or DSS is a major determinant of its use. MIS managers can influence both the perceived usefulness and the perceived ease of use of a new system by using a participative development process. MIS staff need to establish a meaningful "social exchange" with potential users, and DSS developers must be responsive to user requests, questions, and needs.

More research is needed on the effectiveness of approaches for designing and developing DSS. But in general, MIS professionals should use a decision-oriented design process and then use either a rapid prototyping or SDLC process. End-user DSS can be satisfactory and inexpensive, and MIS staff should collaborate to support such development rather than discourage it when it seems appropriate. Rapid prototyping is useful in building many types of DSS, but SDLC has a role in developing complex, networked, enterprise-wide, data-driven DSS. DSS analysts and managers need to be familiar with all of the approaches for building DSS.

One can state some generalizations about Design and Development of DSS. First, when a project idea is proposed, the focus should be on description and diagnosis of decision making and an analysis of the decision and the processes involved. This approach is called decision-oriented diagnosis (cf., Stabell, 1983).

Second, following diagnosis, one should conduct a feasibility study and, in many situations, prepare a feasibility report. Third, if the project seems feasible, then managers and IS staff need to decide to build or buy the proposed DSS. In many situations, a solution will be customized for the DSS.

Fourth, in general, model-driven and knowledge-driven DSS are built using rapid prototyping. Data-driven and document-driven DSS are built using rapid prototyping or an SDLC approach depending on the complexity and scale of the system. Communications-driven DSS are usually purchased and installed on company computers.

Finally, managers need to develop a comprehensive understanding of how to design and develop various types of DSS. Ultimately, senior managers are responsible for ensuring that DSS projects support business goals and provide benefits to an organization. The appropriateness of the design and development process coupled with activities to train users in the operation of a new DSS determine if business goals will be met and if benefits will be realized.