A Framework for Developing Distributed Cooperative Decision Support Systems – Inception Phase

Alexandre Gachet
University of Fribourg, Switzerland
alexandre.gachet@unifr.ch

Abstract

This paper describes the inception phase *of the development process of a Framework for Developing Distributed Cooperative Decision Support Systems (DSSs). It analyzes the reasons why the broad use of DSSs has not occurred yet and makes propositions to improve this situation. It shows that, for the most part, modern distributed computing architectures could solve many of the presented issues.*
In the first section, this paper gives an overview of DSSs, based on definitions, history, taxonomies and DSS architectures. In the second section, it covers three categories of problems in the DSS area: human factors, conceptual factors *and* technical factors. *To finish, it proposes possible solutions to these problems using concepts borrowed from new distributed computing architectures.*

Keywords: cooperative DSS, model-driven DSS, data-driven DSS, distributed computing

Introduction

Definitions of Decision Support Systems

The concept of a *decision support system* (DSS) is extremely broad and its definitions vary depending on the author's point of view (Druzdzel and Flynn, 1999). It can take many different forms and can be used in many different ways (Alter, 1980). On the one hand, Finlay (1994) and others define a DSS broadly as "a computer-based system that aids the process of decision-making". In a more precise way, Turban (1995) defines it as "an interactive, flexible, and adaptable computer-based information system, especially developed for supporting the solution of a non-structured management problem for improved decision making. It utilizes data, provides an easy-to-use interface, and allows for the decision-maker's own insights." On the other hand, Schroff (1998) quotes Keen (1981) ("there can be no definition of Decision Support Systems, only of Decision Support") to claim that it is impossible to give a precise definition including all the facets of the DSS. But according to Power (1997), the term Decision Support System remains a useful and inclusive term for many types of information systems that support decision-making. He humorously adds that every time a computerized system is not an on-line transaction processing system (*OLTP*), someone will be tempted to call it a DSS…

For more information, we recommend reading Druzdzel and Flynn (1999), Power (2000), Sprague and Watson (1993), the first chapter of Power (2000a) and the first chapter of Silver (1991).

A Brief History of DSSs

In the absence of an all-inclusive definition, we will focus on the history of DSSs. Power (1999) remarks that, according to Keen and Stabell, the concept of Decision Support has evolved from two main areas of research: the theoretical studies of organizational decision making done at the Carnegie Institute of Technology during the late 1950s and early 1960s, and the technical work on interactive computer systems, mainly carried out at the Massachusetts Institute of Technology in the 1960s. It is considered that the concept of DSS became an area of research of its own in the middle of the 1970s, before gaining in intensity during the 1980s (Hättenschwiler, 1999). In the middle and late 1980s, Executive Information Systems (EIS), Group Decision Support Systems (GDSS) and Organizational Decision Support Systems (ODSS) evolved from the single user and model-oriented DSS. Beginning in about 1990, data warehousing

and One-Line Analytical Processing (OLAP) began broadening the realm of DSS.

It is clear that DSSs belong to an environment with multidisciplinary foundations, including (but not exclusively) database research, artificial intelligence, human-computer interaction, simulation methods, software engineering and telecommunications.

Taxonomy of DSSs

As for the definition, there is no all-inclusive taxonomy of DSSs either. Different authors propose different classifications. At the user-level, Hättenschwiler (1999) differentiates *passive, active* and *cooperative DSSs*. A *passive DSS* is a system that cannot bring out decision suggestions or solutions. An *active DSS* can bring out such decision suggestions or solutions. A *cooperative DSS* allows the decision-maker (or its advisor) to modify, complete, or refine the decision suggestions provided by the system, before sending them back to the system for validation. The system again improves, completes, and refines the suggestions of the decision-maker and gives them back to him for validation, etc.

At the conceptual level, Power (2000b) differentiates *Communication-Driven DSSs, Data-Driven DSSs, Document-Driven DSSs, Knowledge-Driven DSSs* and *Model-Driven DSSs*. A *Model-Driven DSS* emphasizes access to and manipulation of a statistical, financial, optimization or simulation model. Model-Driven DSSs use data and parameters provided by decision-makers to aid decision-makers in analyzing a situation, but they are not necessarily data intensive. A *Communication-Driven DSS* supports more than one person working on a shared task; examples include integrated tools like Microsoft's NetMeeting. *Data-Driven DSSs* or *Data-oriented DSSs* emphasize access to and manipulation of a time-series of internal company data and, sometimes, external data. *Document-Driven DSSs* manage, retrieve and manipulate unstructured information in a variety of electronic formats. Finally, *Knowledge-Driven DSS* provide specialized problem-solving expertise stored as facts, rules, procedures, or in similar structures.

At the technical level, Power (1997) differentiates *enterprise-wide DSS* and *desktop DSS*. *Enterprise-wide DSSs* are linked to large data warehouses and serve many managers in a company. *Desktop single-user DSSs* are small systems that reside on an individual manager's PC.

Other authors (Alter, Holsapple and Whinston, Donovan and Madnick, Hackathorn and Keen, Golden, Hevner and Power) propose different taxonomies that are less relevant in this paper. We recommend reading should read the first chapter of Power (2000a).

Architectures of a DSS

Once again, different authors identify different components in a DSS. Sage (1991) identifies three fundamental components of DSSs:

- Data-Base Management System (DBMS)
- Model-Base Management System (MBMS)
- Dialog Generation and Management System (DGMS)

According to Power (2000a), academics and practitioners have discussed building DSSs in terms of four major components:

- The user interface
- The database
- The model and analytical tools
- The DSS architecture and network

Finally, Hättenschwiler (1999) identifies five components of DSSs:

- *Users* with different roles or functions in the decision-making process (decision-maker, advisors, domain experts, system experts, data collectors)
- A specific and definable *decision context*
- A *target system* describing the majority of the preferences
- A *knowledge base* made of:
  - External data sources, knowledge databases, working databases, data warehouses and meta-databases
  - Mathematical models and methods
  - Procedures, inference and search engines
  - Administrative programs and reporting systems
- A *working environment* for the preparation, analysis and documentation of decision alternatives.

Situation

This paper analyzes the reasons why the broad use of DSSs has not occurred yet and makes propositions to improve this situation. The research that we are currently conducting elaborates on the works of Schroff (Schroff 1998, Hättenschwiler *et al*. 1998, and Hättenschwiler 1999) and his Object Manager Environment (OME).

On the user level, we will consider *cooperative DSSs*, in order to avoid the limitations of passive and active DSSs. On the conceptual level, we will consider both *model-driven DSSs* and *data-driven DSSs*, following assumptions used in OME. It is foreseen that we will also borrow a few ideas of *communication-driven DSSs* as well. On the technical level, we will explore new distributed paradigms and thus consider *enterprise-wide DSSs* in multi-tier architectures (also known as *inter-organizational* or *intra-organizational DSSs*).

In addition, the DSSs considered in this paper conform to the description proposed by Hättenschwiler (1999). DSSs are highly organized information systems, designed especially for an environment of decision with clear boundaries, and able to be developed continuously along with their environment. The DSSs do not make decisions themselves, but propose to the decision-makers analyses of the advantages and disadvantages of existing alternatives, feasibility and unfeasibility studies, as well as specific documentation of these alternatives. These DSSs are typically composed of the five components described at the end of the previous paragraph.

Problem definition

The field of DSSs is too vast to try to establish an exhaustive list of the reasons why these systems create rather low interest in practice. Nevertheless, we can divide the various factors of the problem into three main categories: *human factors*, *conceptual factors* and *technical factors*.

Human factors

In this paper, **human factors** cover the reasons why the people involved, users and decision-makers, subjectively oppose the computerized decision-making systems. This opposition is based mainly on the personal feelings of the actors towards the proposed data-processing environment.

Ghasemzadeh and Archer (2000) note that the decision-makers are not sufficiently implied in the process of finding a solution. In fact, according to Schroff (1998), it is rare that the decision-maker is even the immediate user of the DSS. The immediate users are usually decision assistants who interact directly with the system and play the role of interface between the DSS and the decision-maker. But according to Drucker (1954), the decision-maker must understand the basic method involved in making decisions. Without such understanding, he will either be unable to use the new tools at all, or he will overemphasize their contribution and see them as the key to problem solving. This can only result in the substitution of gadgets for thinking and of mechanics for judgment.

Even so, according to Ghasemzadeh and Archer (2000) and Power (1997), it is very difficult to explicitly formulate in advance all the preferences of the decision-maker. This phenomenon exerts a negative influence on the interest and the confidence of the decision-maker in the system used.

Moreover, the DSSs focus too often on information management and do not provide enough support to the users (Sauter, 1996). DSSs are designed as a substitute for the human choice process or an elaborate report generator.

Another human factor responsible for the disinterest towards traditional DSSs can be explained by the growing interest in "*end-user computing*" (Kreie *et al.*, 2000). The concept of "end-user computing" refers to people developing software applications for themselves or for others even though they are not trained MIS professionals. The study of Kreie *et al.* explains the growth of this tendency: on the one hand, the advances in information technology made microcomputer hardware relatively cheap, but quite powerful. On the other hand, traditional software packages such as spreadsheets (for example, Microsoft Excel) integrate now in an intuitive way the definition of reports, graphs and tables in one same application. Another factor explaining the growth of "end-user computing" is the fact that users are seldom involved in the *implementation* of the DSS. Santhanam *et al.* (2000) mention that user participation should be an important part of any new IS implementation strategy. Although Kreie *et al.* show that "end-user computing" remains of poor quality, the users prefer this independent working method, which increase their satisfaction and avoid communication problems and delays when dealing with the MIS department.

Finally, given that DSSs remain complex applications that are difficult to use for *non-specialists* (i.e. most of the decision-makers), though Sprague and Watson (1993) mentioned that a good DSS should be easy to use to support the interaction with non-technical users, it is understandable that they tend to keep their distance from these systems. This last issue will be discussed in detail when dealing with *technical factors*.

Conceptual factors

In this paper, **conceptual factors** cover the problems encountered by DSSs because of wrong or incomplete choices carried out during the design of systems, that is to say after the analysis, but before the implementation. These factors concern a lower level than the human (and subjective) factors, but do not relate yet to the purely technical considerations of the system.

Huber (1982) mentions that DSSs are helpful in tasks such as information retrieval, evaluation of alternatives and choice making, but are less helpful in earlier tasks such as problem exploration, information-needs-analysis and creative alternative generation. This increases the likelihood that decision-makers will then solve the wrong problem or choose an inappropriate or low-quality solution.

In addition, a fundamental task of the cooperative DSSs consists in modeling the environment, or context, of the problem to be solved. Modeling describes the process of decomposing and formalizing a problem (Druzdzel and Flynn, 1999). Hättenschwiler (1993) *et al.* (1998) define many conceptual factors precisely related to modeling in the DSSs: missing standards and basic concepts for modeling, missing user-friendly systems for modeling, missing support for the evolutionary process of modeling, missing support for the reuse of existing models. According to Sprague and Watson (1993, p.19), the model creation process must be flexible, with a strong modeling language and a set of building blocks, much like subroutines, which can be assembled to assist the modeling process. In other words, the natural evolution of the decision-making area, spread out between *planning* and *data warehousing* while passing by *operations research*, *expert systems*, *databases*, *worksheets*, *modeling environment*, *DSS generators* and the *office automation* tools, constantly neglected three conceptual factors specific to DSSs (Hättenschwiler 1999).

Firstly, the evolution neglected the fact that a DSS is based on *situations* (i.e. *facts* - or data - often creating an incomplete knowledge base) that are combined with assumptions, or *scenarios*. Data warehousing systems are shown to be very powerful in extracting the *facts* (past horizon) but propose few mechanisms for managing *scenarios* (uncertain future horizon). In the same way, *operations research*, based on mathematics, defines static systems looking for optimal solutions to fully structured problems (*pure problems*), seldom corresponding to the open and ill-structured environments (*impure problems*) of DSSs.

Secondly, the evolution neglected the fact that a DSS must propose to the decision-maker an environment of *unconstrained decidability*. At one end of the spectrum of the possibilities offered by a DSS, the decision-maker receives a batch of alternatives according to the *situation*, the goal - or *task* - to reach and the *exogenous decisions* of the decision-maker. At the other end of the spectrum, the decision-maker must have the ability to propose to the system his own alternative (possibly subjective) and receive from the DSS a study of the consequences of this alternative (*feasibility*). No current technique offers the decision-maker an environment of unconstrained decidability. Operations research seeks optimal solutions to fully defined problems that the decision-maker cannot modify without difficulty (*constrained decidability*), and data warehousing is not designed to evaluate the projected consequences of an alternative suggested by the decision-maker. The DSSs should also offer the process of constructing alternatives based on different types of decisions (Drudzel and Flynn, 1999).

Thirdly, the evolution neglects the manner of presenting the alternatives to the decision-maker (*reporting*). No precise answer was given for questions such as: how many alternatives must be offered to the decision-maker? How to avoid overwhelming the decision-maker with numbers? What should be presented first? How should the advantages and disadvantages be enumerated? Etc. However, these questions prove fundamental if we want to avoid problems related to certain human factors presented above.

Technical factors

In this paper, **technical factors** cover the problems encountered by DSSs related to purely software or hardware considerations. Thus, these factors are not directly connected to the high level concepts concerning decision-making, but rather with the constraints that data-processing structures impose on the implementation of these high level concepts.

Bhargava *et al.* (1999) mention that the complexity and long development time inherent in building decision support systems has thus far prevented their wide use. Building a DSS requires significant expertise in decision analysis, programming, and user interface design. A DSS may also be required to work in real time with other enterprise applications, further complicating the task.

Hättenschwiler *et al.* (1998) mention another technical factor, which attributes the disinterest for DSSs to

inflexible frameworks applied in building highly adaptable DSS (development costs too high, lack of reuse, monolithic architectures).

Ill-defined user interfaces to DSSs represent another technical factor, as systems with user interfaces that are cumbersome, unclear, or require unusual skills are rarely useful and accepted in practice (Druzdzel and Flynn, 1999). Effective user interfaces are especially important for systems that will be used directly by managers (Power, 2000a). Systems have to be first designed to provide all the DSS functions of interactive dialog, flexibility, and tools to examine alternatives (Santhanam *et al.*, 2000).

DSSs often must be supplemented with non-transactional, non-accounting data, some of which has not been computerized in the past (Sprague and Watson, 1993, p.18).

Much research, including that of Keen (1981), Liberatore and Titus (1983) or, more recently, Fjermestad and Hiltz (1998), show that many decisions are made by groups of people rather than by one isolated decision-maker. The more complex the organizations become, the less the decisions are taken by single individuals (Gannon, 1979). These observations caused the emergence of Group DSSs (GDSSs) and of Organizational DSSs (ODSSs), often based on the architecture of multiparticipant DSSs (MDSSs). Even though it is easy to find definitions of such high level architectures, there exists to our knowledge only few concrete implementations of GDSS, ODSS or MDSS taking advantage of the new possibilities offered by distributed computing, undoubtedly because these architectures are new (for example, the Java 2 platform, Enterprise Edition, the Jini technology and the JavaSpaces service). Consequently, the implementations of DSSs often produce centralized and static systems, which are poorly designed for multiparticipant use, or not designed for all of the time/place categories of the GroupWare map (that is, people can be separated in space, or separated in time, or both).

Moreover, current DSSs (including GDSSs and ODSSs) poorly support the internationalization of current distributed systems. Sauter (1999) observes that if DSSs are truly to facilitate decision making across cultures, then they must be sensitive to differences across cultures.

Finally, DSSs are complex systems often composed of heterogeneous subsystems (various databases, complex mathematical libraries, proprietary data, etc.) and are therefore difficult to integrate in only one productive system. This difficulty of integration–a recurring topic in modern computing–complicates the implementation of flexible, light and modular DSSs. On the contrary, the existing systems are often closed, thick and monolithic.

Proposition

Following Sprague and Watson (1993) ("DSSs are not developed according to traditional approaches but require a form of *iterative development* that allows them to evolve and change as the situation changes"), we strongly believe that it would be utopian to try to build a "one size fit all" Decision Support System. We will instead focus on the definition of a *framework* for developing Decision Support Systems. According to Buschmann *et al.* (1996), a *framework* is a partially complete software (sub-) system that is intended to be instantiated. It defines the architecture for a family of (sub-) systems and provides the basic building blocks to create them. It also defines the places where adaptations for specific functionality should be made. One of the aims of this framework is to bridge the gap between decision support theories and real-life DSSs. It should not be seen as a *DSS Generator* (Sprague and Watson, 1993), but rather as a more fundamental *DSS Tool*.

Propositions for solving human issues

Human factors, because of their very subjective nature, are the most difficult ones to deal with. Nevertheless, we propose in this paper some generic solutions that could help solving some of these human issues if they are used during the early stages of software development.

Recently, because of technological development, managers have become more enthusiastic about implementing innovative DSSs (Power, 2000a). *Distributed computing* is undoubtedly part of this technological development. Firstly, today's distributed architectures are tailored for open, highly interactive systems based on many subsystems. This definition describes DSSs well. Secondly, the Internet since 1996 has become a part of every business and person's life (Petrie, 1998). The main impacts of the Internet on various technology sectors are: accelerating deployment, enabling individuals to collaborate across great distances, simplifying user interfaces, and reducing training requirements. All of these impacts are related to certain human issues. Moreover, the Internet, which distributes its infrastructure worldwide, has enhanced the notion of *thin clients*. Thin clients have many advantages over *fat* (or *thick*) *clients*; they allow

corporations to distribute data and analytical tools to a much broader user community than was previously feasible. They define a clear distinction between data and operations. They represent applications easier to manage, more convenient and with an improved accessibility. According to Power (2000a), thin clients in the realm of DSSs naturally lead to the notion of *Web-based DSS*. In other words, *distributed computing* should make DSSs more familiar for decision-makers. They should provide them with well-known and intuitive user interfaces (e.g. web browsers, Java applications, etc.)

Furthermore, the notion of distributed computing associated with the notion of mobility leads to the notion of *field computing* (Hughes, 2000), which could leverage this new enthusiasm for innovative DSSs. Indeed, personal digital assistants (PDA) and hand-held computers (ideally Java-based) that easily fit in a suit-jacket pocket are now powerful enough to enhance the idea of distributed *and* mobile computing (Graham, 2000). PDA and hand-held computers have friendly user interfaces; with several options for connecting these devices directly to regular computers or remotely via wireless communications, PDA and hand-held computers are a low-cost option for deploying a distributed decision architecture to a work force with minimal computing skills. Geographical Information Systems (GIS), which are data-driven DSSs, already use field computing.

Web-driven and Java-based distributed computing should help solving some of the human factors laid out in the first section, i.e. multi-users decision spaces (decision-makers, advisors, system administrators, etc.), better support the user thanks to well-known and proven web technologies, renewed enthusiasm for distributed and innovative DSS in comparison to end-user computing.

Finally, the use of a framework for developing DSSs facilitates the implementation of new systems (using tried and true building blocks), and thus allowing users to participate actively in the development and implementation process. Indeed, the section dealing with *human factors* reminded us that user participation was an important part of any new IS implementation strategy. Adequate training during the *transition* phase of the development, so that users can operate the new system effectively, can also reduce users' opposition to a DSS.

Propositions for solving conceptual issues

In his approach to user oriented DSSs based on the Object Management Environment (OME), Schroff (1998) proposed many ideas to solve most of the conceptual factors presented above. By building our distributed framework for developing DSSs on top of OME, we will borrow many of those ideas.

Our distributed framework should improve and complete the modularity proposed by OME. In OME, modules are called *object managers*, and each one is aimed at solving one part of the general requirements (i.e. *system manager, data manager, scenario manager, task manager, evaluation manager,* and *representation manager*). New managers can then be added to solve new or specific conceptual factors. The novelty of our framework will reside in the fact that these modules will be more loosely coupled than in OME, leading to an even more flexible and open architecture. This will enhance the development of real *cooperative DSSs* able to dynamically react to the refinements proposed by the decision-maker or his advisors.

Schroff's Object Managers are themselves handled in a user-friendly interface especially designed to enhance rapid prototyping. Object Managers handle *Decision Support Objects (DSO)*, which are the building blocks (components) of the DSS. This object-oriented design enhances the reuse of existing components and the components of the DSS are presented to the decision-makers in an intuitive and structured way. Moreover, OME uses a strong modeling language developed at the University of Fribourg and called *LPL* (Huerlimann, 1998).

This cooperative, object-oriented architecture should help solving some of the conceptual factors laid out in the first section, i.e. modeling of situations using pertinent DSOs (facts, scenarios, etc), unconstrained decidability provided by the flexible and dynamic architecture and flexible reporting, as it would be just another component in our architecture.

For more information, we recommend reading Schroff (1998), Hättenschwiler (1999) and Hättenschwiler *et al.* (1998).

Propositions for solving technical issues

There are many compelling reasons for using distributed computing in order to solve the technical issues described above. At a fundamental level, distributed computing brings many advantages: enhanced performance, enhanced scalability, resource sharing, fault tolerance and availability, elegance, etc.

At a higher level, new paradigms in the distributed computing realm have recently become quite popular: *Enterprise Java Beans (EJB), Java Server Pages (JSP)* and *Java Servlets* are hot spots in today's distributed world (Flanagan *et al.*, 1999, Wilcox, 2000). But other new technologies–less *e-commerce*-driven–are also emerging, unveiling some features of tomorrow's distributed architectures. Sun Microsystems's *Jini* and its related *JavaSpaces* service or Microsoft's *Universal Plug and Play (UPnP)* are amongst these. These new forms of telecommunication technology enable work teams within organizations to interact better and enhance their business decisions. These technologies could help us to implement the proposed high-level solutions for solving technical issues using distributed computing and supporting the ideal time/place arrangement of the *GroupWare* map (i.e. *any* time, *any* place). In addition, they go far beyond traditional distributed DSS architectures (client/server, sharing, etc.) still presented by various authors. These new technologies often provide features like scalability, transactionality, fault-tolerance and security that are expensive and time-consuming to implement if they are designed from scratch.

Additionally, modern programming languages provide graphical libraries that allow the rapid development of high-quality user interfaces, using powerful *graphical builders*. These new tools help the developer concentrate on the desired GUI without losing too much time implementing it. These tools also make GUI more flexible, as it is easier for the developer to change part of it without having to change all of the subsequent code. They can prove even more powerful when combined with GUI development framework, such as the ROMC approach presented by Sprague and Carlson (1982).

Finally, new technologies related to application integration and inter-application communication (e.g. CORBA) make it easier to implement our distributed framework for developing DSSs.

These new technologies and tools should help in solving some of the technical factors laid out in the first section, i.e. the complexity and long development time of DSSs, the lack of flexibility of current frameworks, cumbersome user interfaces, the lack of internationalization and inter-application communication.

Future

The next phases of this project - *elaboration* and *construction* - are in progress at the University of Fribourg, Switzerland. It is foreseen that the framework will be implemented using the Java programming language and the Jini technology (with its JavaSpaces service) on a local area network (LAN). Jini can be used in conjunction with EJB components, servlets and Java Server Pages. For reasons of simplicity, issues like security will not be addressed in detail and are therefore left for other research projects.

Object persistency will be implemented using Java's built-in serialization mechanisms and/or the JavaSpaces service of Jini. Database connections will be handled using Java's JDBC and/or JDO APIs. Inter-objects communication will be implemented using the Jini technology and RMI. Low-level inter-applications communication will be implemented using Java IDL and CORBA. Graphical User Interfaces will be implemented using the Java Foundation Classes (Swing). It should also be possible to define a pure XML/HTML user interface. General-purpose configuration data will be formalized using the XML language and open source parsers.

As far as possible, mobile computing using PDA or handheld computers should be added to the core of the framework. Most likely, mobile computing will be covered during the *elaboration* phase.

References

Alter, S.L. (1980). *Decision Support Systems: Current Practice and Continuing Challenge*, Reading, MA, Addison-Wesley. (p. 71)

Buschmann F., Meunier R., Rohnert H., Sommerlad P., Stal M. (1996). *Pattern – Oriented Software Architecture: A System of Patterns*, John Wiley & Sons Ltd.

Drucker, P. (1954). *The Practice of Management*. New York: Harper and Brothers, p. 368

Druzdzel, M.J. and Flynn R. R. (1999). "Decision Support Systems", to appear in *Encyclopedia of Library and Information Science*, Allen Kent (ed.), Marcel Dekker, Inc.

Finlay, P. (1994). *Introducing Decision Support Systems*, Oxford: Blackwell.

Fjermestad J., Hiltz S.R. (1998). An assessment of group support systems experiment research: methodology and results. *Journal of Management Information Systems* 15 (3) (1998-1999) 7-149

Flanagan, D., Farley J., Crawford W., Magnusson K. (1999). *Java Enterprise in a Nutshell*, O'Reilly & Associates, Inc., Sebastopol, CA.

Gannon M.J. (1979). *Organizational Behavior: A Managerial and Organizational Perspective*, Boston: Little, Brown.

Graham, L.A. (2000). Life in the Fast Lane, in *GEOWorld*, July 2000, GeoTec Media, pp. 30-35

Ghasemzadeh F., Archer N.P. (2000). Project portfolio selection through decision support. *Decision Support Systems and Electronic Commerce*, July 2000

Hättenschwiler P. (1993). *Computer Assisted Top Down Modeling*, Modeling Tools for Decision Support, University of Fribourg. p. 101-131

Hättenschwiler P. (1999). *Neue Konzepte der Entscheidungs-unterstützung*, Working Paper 99-4, Institute of Informatics, University of Fribourg, March 1999

Hättenschwiler P., Moresino M., Schroff A. (1998). *Rapid Prototyping of Decision Support System*, conference proceedings of ICSC Symposium, Tenerife.

Huber G. P. (1982). *Group Decision Support Systems as Aids in the Use of Structured Group Management Techniques*, DSS-82, Conference Proceedings.

Huerlimann T. (1998). *LPL: A mathematical programming language*, Institute of Informatics, Working Paper, June 1998, Fribourg

Hughes, J.R. (2000). GIS Goes Mobile, in *GEOWorld*, June 2000, GeoTec Media, p. 8

Keen P.G.W. (1981). *Decision Support Systems: A Research Perspective*, in Decision Support Systems: Issues and Challenges, Pergamon Press

Kreie J. Cronan T.P., Pendley J., Renwick J.S. (2000). Applications development by end-users: can quality be improved?, *Decision Support Systems*, August 2000

Liberatore M.L., Titus G.J. (1983). The practice of management science in R&D project selection, *Management Science* 29 (1983), 962-974

Petrie, C. (1998). A Brief History, *IEEE Internet Computing*, IEEE Computer Society, November/December 1998, pp. 6-7

Power, D.J. (1997). What is a DSS? *The On-Line Executive Journal for Data-Intensive Decision Support*, October 21, 1997: Vol. 1, No. 3.

Power, D.J. (1999). *A Brief History of Decision Support Systems*. DSS Resources, World Wide Web, http://dss.cba.uni.edu/dss/dsshistory.html.

Power, D. J. (2000a). *Decision Support Systems: Concepts and Resources*. Cedar Falls, IA: DSSResources.COM, pre-publication PDF version, 2000, accessed on october 2000 at URL http://dssresources.com/dssbook/.

Power, D. J. (2000b). *Web-Based and Model-Driven Decision Support Systems: Concepts and Issues*. Prepared for AMCIS 2000, Americas Conference on Information Systems, Long Beach, California, August 10th - 13th, 2000, "Model-Driven and Web-Based Decision Support Systems" Mini Track.

Sage, A.P. (1991). *Decision Support Systems Engineering*, John Wiley & Sons, Inc. New York.

Santhanam R., Guimaraes T., George J. F. (2000). An empirical investigation of ODSS impact on individuals and organizations, *Decision Support Systems*, December 2000, pp. 51-72

Sauter, V.L. (1996). *Decision Support Systems: An Applied Managerial Approach*, John Wiley & Sons Ltd, Chichester, England.

Schroff A. (1998). *An Approach to User Oriented Decision Support Systems*, Inaugural-Dissertation Nr. 1208, Druckerei Horn, Bruchsal.

Silver M.S. (1991). *Systems that Support Decision Makers – Description and Analysis*, John Wiley & Sons Ltd, Chichester, England.

Sprague R., Carlson E. (1982). *Building Effective Decision Support Systems*, Englewood Cliffs: Prentice Hall.

Sprague R., Watson H. (1993). *Decision Support Systems – Putting Theory into Practice, 3rd Edition*, Englewood Cliffs: Prentice Hall.

Wilcox M. (2000). *Professional Java Server Programming, J2EE Edition*, Wrox Press Ltd., Birmingham.

Biography

Alexandre Gachet has a diploma in computer science from the University of Fribourg, Switzerland, where he is currently working on a Ph.D. He is working on the development of distributed decision support systems using new distributed technologies, namely Java and Jini.