A Proposal for an Open DSS Protocol

Before Decision Support Systems can be shared via the Web, a common protocol for deploying them must be agreed upon.

The Web was designed to be a “pool of human knowledge, which would allow collaborators to share their ideas and all aspects of a common project” [2]. This goal for the Web makes it an ideal distribution system for the Decision Support Systems (DSS) of the future. A DSS is an object class that includes software applications, mathematical subprograms or model solvers designed to interact with humans to facilitate decision-making. The determination as to whether an application is a DSS is in the hands of the application developer. However, the definition of a DSS is user-based; that is, repeated discretionary use by end users represents the final authority on the value of a DSS. This definition includes expert systems as DSS, should those expert system developers choose to deploy the systems on the Web in a protocol-compliant format.

At present the Internet provides access to hundreds of gigabytes each of software, documents, sounds, images, and many other types of information [6]. It is only a matter of time before vendors begin deploying DSS on the Web on this type of scale. For DSS to be efficiently discovered on the Web, a protocol that allows DSS information to be found and transmitted must be established. In addition, a mechanism to provide a consistent organized view of DSS information is necessary. A DSS resource discovery system will need to identify a resource, collect information about it from several sources, and convert the representation to a format that can be indexed for efficient searching [6]. The purpose of this article is to propose a protocol suite that will facilitate the discovery of DSS on the Web by allowing Web pages containing DSS to be easily distinguished from other Web pages. In addition, the protocol provides a common format for describing DSS, such that autonomous intelligent search agents can identify DSS that meet specific user-defined requirements.
Background

Three recent articles have examined how DSS could be deployed on the Web. The first two, by Bhargava et al. [4, 5], present DecisionNet, a prototype of a brokered system that facilitates transactions between providers and consumers of decision technologies. Under this system, all DSS developers must submit their DSS for inclusion in the DecisionNet system and all DSS users must register in order to use DSS from their system. This approach greatly simplifies the deployment of DSS on the Web. First, registered DecisionNet subscribers do not have to download the DSS they want to use; instead they access the DSS remotely and run them on the DSS provider’s platform. This allows users to utilize a DSS even if they do not have the hardware or software necessary to run it. In addition, specialized search agents or browsers are not necessary, since DSS search is only necessary within the index of registered DSS.

The other approach for DSS deployment is an open system that would allow DSS to be distributed on individual Web pages, as other types of data are currently being offered. Goul et al. [10] propose a set of requirements for a protocol suite that will allow the deployment of Open DSS on the Web. A protocol based on these requirements would utilize specialized Web search agents (robots, spiders, wanderers, and so forth) to provide automated intelligent discovery of DSS pertaining to a specific decision-making or problem-solving situation.

The open protocol contributes to the disintermediation of the Web. It allows individual users or automated intelligent search agents to find any DSS compliant with the protocol, not only those posted with an individual broker. However, the open system has no mechanism to control what is put on the Internet and portrayed as a DSS [10]. It is possible that the eventual system for deploying DSS on the Web will combine the two approaches currently being proposed. Brokers could be used to provide access to high-quality, tested DSS while other DSS could be provided at individual Web sites. The protocol described in this article is capable of supporting both types of DSS deployments.

Open DSS Protocol Requirements

The Open DSS protocol requires DSS builders to supply information that defines the purpose of a DSS, the computing environment, the data inputs, the data outputs, and other information about their DSS. This information would be added by the DSS builder at the time of creation of the DSS and can be used to aid end user DSS discovery. Robots, wanderers, and spiders will be used to identify addresses and build an index of compliant DSS. This index can then be searched by end users or end-user agents to identify DSS which meet a set of end-user stipulated search parameters.

Goul et al. [10] proposed a set of six requirements that serve as the basis for the Open DSS protocol developed in this article. These requirements are:

1. Automated, intelligent DSS search agents should provide end users with improved discovery of existing DSS that pertain to a particular decision-making or problem-solving situation [10].
2. The decomposition of Open DSS standards should be along the lines of the client/server computing model, and should be consistent with prevailing notions of directed network search. Two clients are required, one for the DSS end user community, and one for DSS builders. A server process is required to act on behalf of an end user client process to discover relevant DSS [10].
3. The Open DSS Protocol Suite must go beyond technical architectural considerations in order to address human behaviors. Accountability guidelines for both DSS builders and end users are required. To the extent possible, the Open DSS Protocol Suite must adhere to emerging de facto standards for Web robots, spiders, and wanderers (including the exclusion standards) [10].
4. DSS built to be shared both within and across organizations will require the inclusion of computer-based training materials. DSS builders also need end user feedback in order to improve DSS and to guide future DSS research. This requires a suitable interface that asks the user for permission to set up a feedback relationship, based on the end user’s email address [10].
5. It is possible that the solution to a given problem-solving situation will require a new DSS to be constructed from a collection of component DSS. Thus, a model integration requirement for the Open DSS protocol suite is necessary. Model integration research is currently developing techniques that use formal logic, to provide the capability to use outputs from different models as inputs to other models automatically. Since model integration is an emerging area of promising research, further research will be needed to finalize implementation details. However, it should be possible to expand the information requirements for DSS builders once the model integration information needs are defined [10].
6. Emerging research in the DSS area will undoubtedly require the addition of implemen-
tation details that add more intelligence to automated Web DSS search agents. In such cases, there should be “not yet defined” portions of protocol suite details, and those portions should serve as stimuli for needed research in DSS [10].

These requirements provide a basis for the development of an Open DSS protocol.

The Open DSS Protocol

The previous section defined the six requirements that are key to creating an Open DSS protocol. In order to meet these requirements, a set of preliminary protocol specifications is proposed. The Open DSS Protocol is a general protocol that provides facilitated access to DSS utilizing the existing Internet application layer protocols HTTP and HTML [3, 12], and consists of two layers. The first layer in the Open DSS protocol is the Metainformation Layer. It indicates the Web site contains a DSS and includes all of the information necessary to completely explain the DSS. The second layer is the Transaction Processing Layer. This layer is responsible for any transactions that are necessary before the software will be made available to the client. The detailed requirements for each of these layers are discussed in the following sections.

The Metainformation Layer.

When objects are transferred over the Internet, information about them (“metainformation”) is transferred in HTTP headers. The Open DSS protocol utilizes a set of specialized headers to provide basic information about the DSS to the automated intelligent search agents. The robots, wanderers, and spiders will traverse the Web requesting entity-header information only (using the HEAD command) to determine whether the Web site contains a DSS. Since, by convention, unrecognized HTTP headers and parameters are ignored, other search agents can also access DSS Web sites without being affected by the specialized DSS headers.

The header information provided by DSS providers must be in a consistent format so that the automated DSS search agents can index them correctly. The first item in the header should indicate that the site contains a DSS. This would be accomplished by a CONTENT-TYPE metainformation label:

<META NAME="CONTENT-TYPE" CONTENT="dss/html">

In addition to specifying the content_type, every DSS will be required to have a title, a list of keywords, and a description.

The remaining metainformation will define the functionality of the DSS being offered, the user-site requirements and other information necessary to evaluate the DSS. The metainformation related to DSS functionality was selected based upon model management research. The goal of this research has been to develop techniques to select or construct appropriate models to be run so as to provide the appropriate answer [7]. Researchers have conducted research on the storage, representation, utilization and manipulation of models. To date there has been no universally
agreed-upon method for representing and specifying DSS models. However, at a minimum, a DSS representation scheme should include descriptions of the stimuli (inputs) and responses (outputs), state (data structures), and procedures (control structures) [1].

The user site requirements should include information on the hardware requirements (computing platform), software requirements (operating system or application needs), and any specific user skills required to use the DSS. Finally, the metainformation must contain all other information necessary to purchase and download the DSS. This would include information on the DSS’s cost, its references, related DSS, and vendor information. A list of the variables that should be used to define the specification information is shown in Figure 1.

Not all of these metainformation variables are necessary to define a given DSS. The individual variables can be defined in any order and unknown variables will be ignored by the search agents. Metainformation variables are defined as follows:

\[\text{<META NAME="Variable_type1" CONTENT= "value1, value2, ... valueN"}>\]
\[\text{<META NAME="Variable_type2" CONTENT= "value1"}>\]

The metainformation variables should be defined in a standard HTML document [12]. HTTP places no limits on the number of extension headers that can be defined in the metainformation header, thus the header information requirements can easily be expanded as the Open DSS protocol evolves.

**The Transaction Layer.** A transaction layer is required in the Open DSS protocol to define the standard information that will be exchanged when an end user decides to purchase or download a DSS. This information will help DSS developers to obtain feedback to improve their DSS and to guide future DSS research. The transaction layer will include log-in and registration templates, which DSS providers use to gather data about customers. When registering, users will be asked to enter their names, email addresses, phone numbers and location in the world, along with information on the planned use for the DSS and the expected frequency of use. This would allow DSS builders to contact individual DSS users to obtain feedback on product performance and to provide users with DSS update information. Figure 2 shows a standard user registration form.

In addition to the standardized registration form, many DSS developers will have additional transaction processing needs. Many DSS available on the Web would likely be offered for a fee. Including some type of billing services in the transaction layer would allow payment capture, invoicing, and activity tracking. On the Web, the users traditionally specify their payment preference at the time of purchase, through a monthly billing or by credit card. Credit card processing would require the following features:

- Communicating with established credit card acquirers;
- Handling authorizations, captures, sales, and card validity checks; and
- Utilizing a leased line or secure Internet connection for transmission of credit card transactions.

Currently, there are several commercial products available that could provide the transaction services necessary for DSS providers. One such product, Netscape Publishing System, provides these features and also supports the creation and maintenance of the Web site where the DSS would be offered [11].

**DSS Search**

Once DSS are deployed in a specific format, end users require a mechanism for discovering them. It is proposed that DSS deployed on the Web be indexed utilizing autonomous intelligent search agents such as robots, spiders, and wanderers. In the Open DSS protocol, these Web search agents would act as federated facilitators. In a federated system, agents do not communicate directly with each other. Instead, the agents communicate only with system programs called facilitators or mediators [9]. This communication consists of the agents’ needs, abilities, appli-
cation level information, and various requests. Under the Open DSS protocol, the intelligent search agents will continuously explore the structure of the Web by examining the header information located in HTML Web pages. When the agent encounters a DSS, it stores the header information and the DSS location in the DSS index.

End users would be allowed to specify the desired DSS attributes to another search agent and then initiate a search against the index. Depending on the sophistication of the search agent, either the search agent or the end users will then evaluate the header information and retrieve the full specifications for the “best” DSS. Finally, end users will examine this specification information and select the DSS that best meet their needs.

The advantage of this approach is that it minimizes the impact the search agent has on the Web sites it visits. A single agent is responsible for examining and downloading header information only, minimizing the amount of information it requires Web sites to provide. End users need only search the DSS index and specific candidate DSS sites to find appropriate DSS tools. The consideration of the potential impact of autonomous agents is one of the important requirements for building ethical Web agents [8].

Future DSS search agents may have the intelligence necessary to select candidate DSS from the index based on the header information and then return to the appropriate Web site to retrieve the full specification information. These agents should be designed such that they could then examine the specification information and rank the DSS based on how well they meet the users’ needs. In addition, future intelligent agents should be able to select a set of DSS that could be integrated to solve large problems. These intelligent search agents can be considered DSS themselves, because they will aid end users in the overall decision-making process.

Example

Consider a program that implements a set of forecasting models. This type of program is capable of performing forecasting calculations using any type of time series data. To set up this program so that it could be accessed using the Open DSS protocol, the DSS builder would need to create an HTML document containing appropriate meta-information and user registration. An example of an appropriate HTML document for this application is shown in Figure 3.

Once this DSS Web page is placed on the Web, end users will be able to discover it using an autonomous DSS search agent. The data entry form to browse the Web would allow the user to specify keywords, hardware types, and cost. Depending on the implementation, users may enter their own input (as in keywords) or they may choose from preselected entries (representation). Such data entry would be converted by the browser to standards within the protocol suite. For this example, the following steps would be performed:

- The end user requests the keyword—forecasting, and the hardware type—PC.
With the deployment of DSS on the Web, many fundamental tenets of the theory of DSS are in need of review. Those tenets will likely be extended. For example, the engine that will be used to search for DSS has been described as a DSS! This extension implies that the rich history of empirical approaches to the study of DSS is relevant to the nascent area of Web search. Ongoing work in this area will next require an examination of the sufficiency and completeness of the proposed protocol for representing commercial DSS, development of prototype search engines, and empirical examination of the efficacy of those systems.

For additional Open DSS Protocol information, see www.public.asu.edu/~dgregg/dssprotocol.

References

Dawn G. Gregg (dawn.gregg@asu.edu) is a Ph.D. candidate in the School of Accountancy and Information Management at Arizona State University. Michael Goul (mike.goul@asu.edu) is an associate professor at Arizona State University.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.