M.A. Jeusfeld, T.X. Bui

Interoperable decision support system components on the Internet

Proc. 5th Workshop on Information Technologies and Systems (WITS'95),

Amsterdam, December 9-10, 1995

also appeared as Aachener Informatik-Berichte 96-17, 1996.

Decision Support Components on the Internet[1]

Manfred A. Jeusfeld (*), Tung X. Bui (+)

*) RWTH Aachen, Informatik V, Ahornstr. 55, 52056 Aachen, Germ
jeusfeld@informatik.rwth-aachen.de

+) Hong Kong University of Science &Technology, ISMT,
Clear Water Bay Rd., KLN, Hong Kong, tbui@usthk.ust.hk

Abstract. Operational software like online transaction processing s
the lion's share of business information systems. Though many D
Systems (DSS) were developed, they failed to become mainstream
reason is lack of knowledge of availability, applicability, and
this paper, we propose to exploit the vast resource of the Int
second chance. Contrasting to other approaches, our strategy i
users and developers of DSS's cooperate in weaving a web of d
components. Repositories serve for ensuring application-centere
for searching. The method is scaleable in number of processo
domains, and platforms. A script language for combining DSS c
introduced based on observations from a real life example of
company.

[1] Most of the work reported here was done while the authors were with the Hong Kong University of Science & Technology, Department of Information and Systems Management, Clear Water Bay Rd., KLN., Hong Kong.

1. Introduction

According to a recent survey by PCWeek (1994), the deploymer Support Systems (DSS) across organizations is rather limited. The reasons that explain this poor distribution of DSS:

- Difficulty in matching problems to appropriate DSS: Decisi experience difficulties in finding available analytical models ap problems.
- Up-to-date DSS are not easily accessible: A comprehensive model-base is har maintain as models and tools are being continuously updated or i
- DSS are often too application-specific: Models have different assumptions ab input/output formats; some of these assumptions are implicit, i declared format.

Recently, with the explosion of Internet use, many academic started making their DSS components available to potentially intere: Bhargava et al. [95;1] cite three cite three factors contributing that, due to the currently limited size of the market for decision s is a market potential for DSS deployment on the World Wide Web (WWW). important, DSS availability on the WWW could eliminate, or at least problems related to the issue of “re-inventing the wheel” and version management. Goul al. [95] view DSS deployment on the Internet as an innovative means opportunities for using newly developed DSS, (ii) validate DSS, i feedback on their usefulness and performance, and (iii) promote wi DSS via training.

From an operational point of view, the goal is to match the der of DSS with much lower overhead costs than those of a more conventic distribution. Ideally, a decision maker should be able to com components in the same way that he/she builds and uses functions without having to be concerned with installation of the components environment. An Internet-based usable system must provide answers questions:

- How can a complex decision problem be planned using distributed as an Internet-based approach to model integration? Our answer i with a graphical and a textual representation.
- How can users access appropriate DSS components dispersed in unlimited Internet search space? Here we use the idea of Uniform (URL) known from the World Wide Web. Data, scripts, models, and are all indentifiable by their URL.
- How can data objects be transformed into the format expected by We propose a version of the uniform data exchange OEM which information with the complex DSS data. This minimizes the hands sender and receiver of DSS data and allows to store a data item i originator.

- How can the Internet traffic be minimized when data/script obj nodes? Here, the simplicity of our script language will allow optimizations.

Our approach is characterized by a lack of central control. DS models, scripts, programs) are created and maintained in an distribu the sum of all available DSS components may not have a consistent m Therefore, we propose repository systems mediating between DSS pro These repository systems define which components may be combined an to browse in an application-specific search space of components.

A Decision Support Assistant (DSA) was proposed by [Felter 8 networks (LAN). It provides an intelligent assistant that helps a d DSS components from a model base. Felter studied interference o invocations and noticed the limited processing time resources. In ou of the DSS components is more complex than that in a LAN. The main degree of distribution. In Felter's approach, all DSS components central DSA. In ours, a federated environment makes a central contr and undesirable. In that regard, we depart from the 'brokers-agent [Bhargava et al. 95;2]. Potential DSS users do not need to buy the make use of distributed computational resources on the Internet.

To allow distributed management of all resources (DSS script ar present two design principles: A uniform data representation as exchanging information between heterogeneous systems; and uniform naming for identifying input/output data as well as the DSS components that pa process. In addition, we advocate the implementation of repositor DSS users with the knowledge to design distributed decision support installation of DSS components across the Internet. As a whole, our to the framework of the "enterprise information bus" by [Ba et al. requirement of having a central agent.

A Corporate-Wide Financial Application Example. To illustrate the approach set in this study, a case study adapted from real-life application International (RI) is a multinational company with its headquart America. With annual sales of Can$ 1.5 billion, RI has its operati variety of products offered, RI seeks to constantly monitor its mar with its geographically distributed sales, RI information system distributed with major computing hubs located in Montreal, Canada France, as well as various regional computing centers, e.g., in Frib

As part of its decision support and executive information developed a financial planning system. The purpose of the system is establish

- yearly and 5-year budgeting
- short term and long term strategic planning
- cash flow management
- routine and ad hoc data analysis and reporting

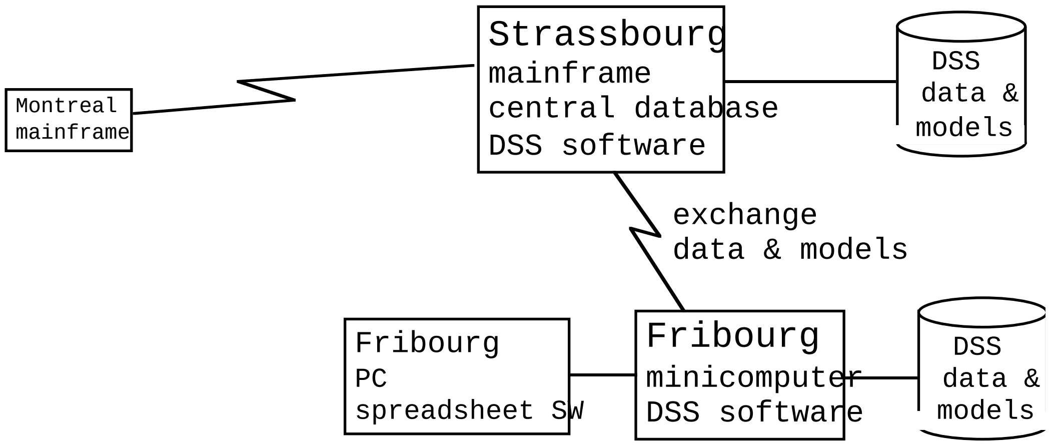
Montreal mainframe
Strassbourg mainframe central database DSS software
DSS data & models
exchange data & models
Fribourg PC spreadsheet SW
Fribourg minicomputer DSS software
DSS data & models

Fig. 1: Architecture of an existing federated DSS

Figure 1 shows an excerpt from the current federated DSS archit
European headquarters in Strassbourg maintains a financial data
company's European operation. Regional centers, like Fribourg in Sw
to the headquarters data from their operation but also require info
decision making. Some financial data and models are kept locally at
Both the headquarters and the regional computer centers provide D
evaluating the financial models on the financial data. Spreadsheet
personal computers serve for visualizing results of the analysis.
federated DSS architecture has been implemented by leased data comm
dedicated communication protocols. The location of the financial d
determined in group meetings between the headquarters staff and t
staff.

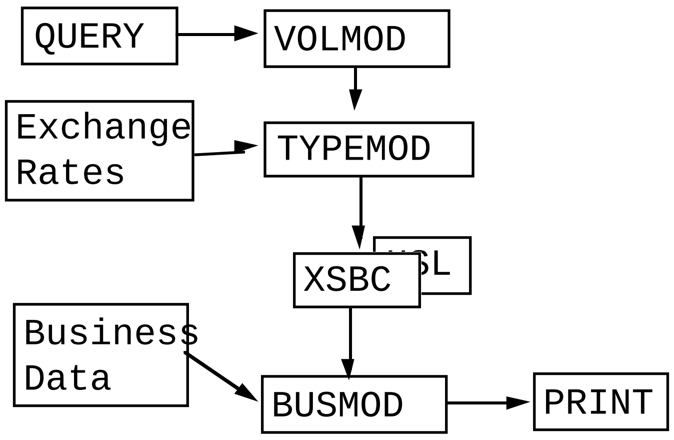

Fig. 2: A script for financial analysis (from RI)

Scripts for computing an analysis were specified by semi-f
diagrams (see Fig. 2) and then implemented by a 4GL language. The
hand side represent financial data, the boxes in the middle are fin
how to extract relevant information from the data, and finally the
how the output is treated. RI also has examples of scripts with lo
analysis for all month of a fiscal year.

The example shows that knowledge about the DSS is spread in heterog
(data, models, scripts). Interoperability, i.e. the ability to p
without information loss, is ensured by using the same software
dedicated transformation routines (wrappers). Scripts have a Janus
they can be viewed as a high level programming language, on the oth
semantic relationship between data and models.

2. Overview of the Proposed Federated System

We use the setting of RI as a running example for our method c
components on the Internet. Our goal is to provide a scaleable meth
and processors may be located anywhere on the network. Users (decisi
DSS software providers, etc.) of the proposed federated architect
exchange information after an agreement on location, representation
the information has been reached. The users may be in different dep
company but may also be from different cooperating companies.

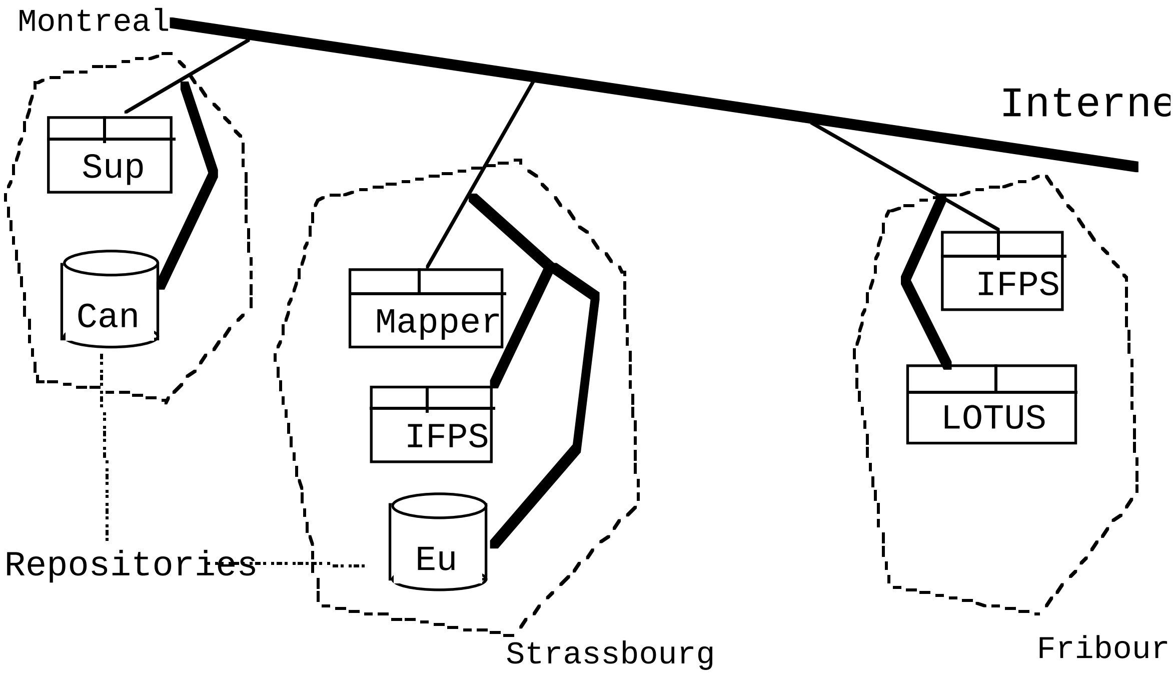

Fig. 3 Overall architecture of the proposed federated DSS

Figure 3 shows a simplified view of a network of DSS processo
systems (labeled ‘Can’ and ‘Eu’ in the Figure) connected by the Inte
thick lines). The rectangular nodes represent DSS processors, i.e.,
information relevant for some decision. The cylindrical nodes (her
examples of repository systems. The repositories constitute the fou
system in that they serve as the main information resource to su
composition of Internet-based DSS components. The repositories can b

standard database application program, or by knowledge-based system: a specific application domain or purpose (e.g., repository of f repository for statistical procedures, etc.).

Each repository creates its own name space, i.e., the set of for script and data objects. Specialized name spaces are justified different groups of decision makers are interested in different DS: and more importantly, distributed repositories create a higher deg script and data objects are known only to those users who have acces

One of our goals is to help DSS providers promote the access systems. The overall system can be continually updated by installing

1. A user (system administrator or even decision maker) browses a for DSS processors suitable to her problem definition, and select
2. The repository is queried for the installation procedure of processor(s).
3. The installation procedure is downloaded to the local system and the features of the procedure conforms to the local system config
4. At the end of a successful installation, script objects for cal DSS processor have to be inserted in the local repository (unifo types, etc.).

As more and more copies of the same DSS package are spread ov redundancy can be exploited for optimizing the load balance of the A DSS provider 'publishes' a new DSS component by inserting it: repository system known to him. This can include a test installatio DSS provider to help potential users test the component(s).

3. Interoperability Techniques

Let’s consider the following generic scenario. A DSS componen data object O1 which is later used by DSS component P2 as input. T are autonomously developed and the data exchange via O1 may not hav the time P1 and P2 were designed. The traditional approach is to s database for later processing (e.g., [Sprague 1989]). For this pur database must be known in advance and is relatively fixed. In our as script objects are assumed to be distributed and no common type s

Remember two principles for distributed processing: first, da into a transferable exchange format similar to IDL [Nestor et al. Resource Locators (URL) [Barners-Lee 95] can be used to identify installed on the Internet.

Uniform Data Representation. We regard any data as a string received and interpreted by the DSS component which processes it. external knowledge about the kind of data, and is therefore not environments where data and script objects evolve without central co

There are several similar approaches from the areas of dis heterogeneous databases, and distributed artificial intelligence co data exchange. We adopt the OEM format [Papakonstantinou et al. 95] the type information within each data object, i.e., the data struct the object representation. Objects are represented as terms (n, t, v the object, 't' is its type and 'v' is its value. The value is eithe term. As an example, consider the definition of a vector of currenc building principle of OEM objects are triplets '(instance-name, t component identifies the object, the second is the name of the type last is the value. OEM objects may be arbitrarily nested. Nested ob by a path expression, e.g., 'Selling-Data-1995.Q-2' identifies the 948 (in thousand). Such a representation makes a data object ind systems of programming languages: it simply carries its type defini that reads such a term can extract the type information and match system.

```
(Selling-Data-1995, Selling-Vector,
     (    (Q-1, K-CAN$, 1033),
          (Q-2, K-CAN$, 948),
          (Q-3, K-CAN$, 1018),
          (Q-4, K-CAN$, 316)   ))
```

Fig. 4: OEM object of type matrix

Uniform Naming. The input of DSS scripts are data objects that one or more (remote) locations. Therefore, both data objects and s identifiable across system boundaries. Uniform resource locator Consortium 95] serve this purpose. The general format is

protocol://node-address/path

The component 'protocol' specifies the communication protocol data or to activate a remote process. We assume the widely-used p following. The 'node-address' uniquely identifies the computer sys requested data or script object. Finally, the 'path' identifies t space of the local computer system. For example, the URL

http://strassbourg.rubber.fr/finance/
Selling-Data-1995.oem

uniquely identifies location of the data object Selling-Data-1995 in the file system o computer 'strassbourg' of organization 'rubber' in France. Via the is technically accessible from any node in the network in which 'st We assume that any persistent data object , i.e., a data object processing time of the script objects that produced or consumed it, with extension 'oem' indicating that its content is in OEM format.

A call to a DSS processor is also be named via URLs. The for procedure call with input parameters referenced by their URLs. For e

```
http://fribourg.rubber.ch/bin/IFPS?call=http://
strassbourg.rubber.fr/scripts/Earning-Forecast-1995.oem
```

invokes the DSS processor IFPS (actually a financial analysis pac
evaluate on the input object Earning-Forecast-1995.oem located i
specified in the HTTP protocol, such an URL names the result of th
exploited in the definition of script objects below.

Figure 5 shows a script object where all objects reside on d
Internet. The script object Earning-Forecast-1995.oem encodes the ca
processor IFPS at Fribourg, and the arguments of the call which ar
objects or calls of component scripts. Argument arg-1 of Earnin
financial model BUSMOD, arg-2 is Selling-Data-1995 from Strassbourg
result of a nested call to another DSS processor IFPS at Strassbour

```
(Earning-Forecast-1995, po-call,
   ((po-node,url,http://fribourg.rubber.ch/bin/IFPS),
   (arg-1,url,
    http://fribourg.rubber.ch/local/BUSMOD.ifps),
   (arg-2,url,
    http://strassbourg.rubber.fr/finance/Selling-Data-
1995.oem),
   (arg-3,po-call,
     ((po-node,url,http://strassbourg.rubber.fr/bin/IFPS),
     (arg-
1,url,http://strassbourg.rubber.fr/local/XSBC.ifps),
     (arg-2,url,
      http://strassbourg.rubber.fr/local/Exchange-Rate-
1995.oem))
)))
```

Fig. 5 OEM representation of a script object

The evaluation of a script object is explained in section 5.
sufficient to note that the script object encodes a data flow diagr
6. Also note that a script object is in OEM format like the data ob
be the output of a compiler script. Furthermore, it can be class
presented in section 4.

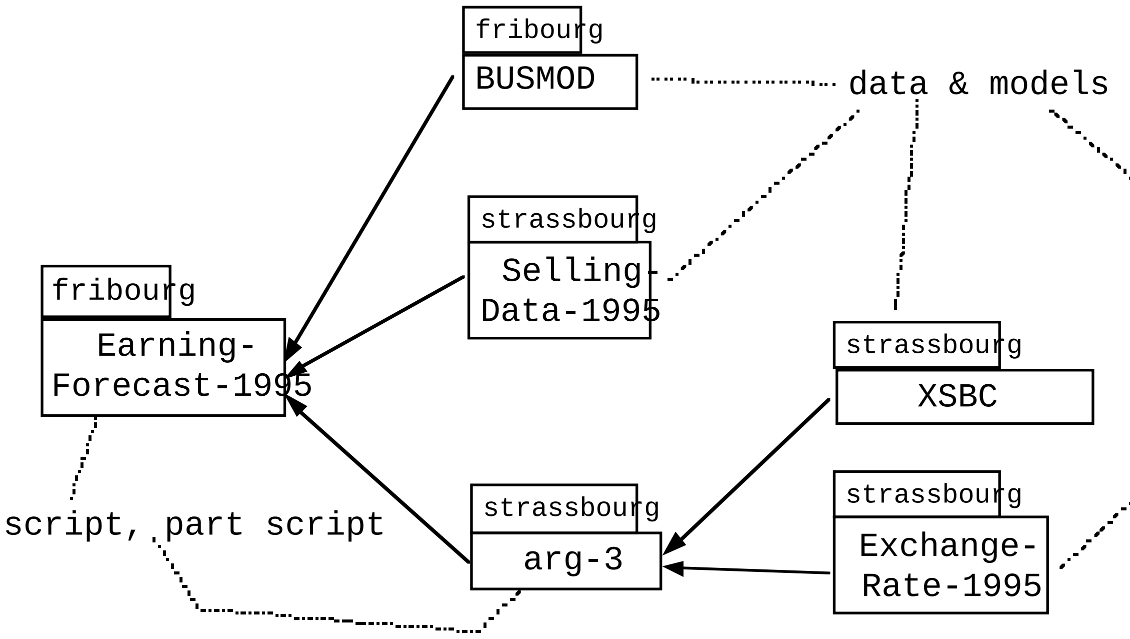

Fig. 5 Diagram denoting a distributed DSS script

The URL conventions are sufficient to invoke a remote DSS comp
or remote input objects and to receive the result in as a single
Legacy DSS components can be integrated by a transducer [Geneseret
which mediates between the network and the DSS component. The trans
call in the above syntax, transforms the (input) data into the inte
legacy DSS component and then invokes it.

The proposed approach is well-suited for DSS components w
interaction, i.e., those that are linked with other DSS components.
interface should not impose principal obstacles (i.e., at the cli
framework). An elegant way is to install a specialized DSS proces
system of the user. This processor accepts requests from the remot
the URL call mechanism shown above. Locally, it gathers the input
appropriate input device.

4. The Repository

Uniform data representation and naming described above pro
framework for communication between distributed DSS components.
repository which stores conceptual information about data objects a
and answers queries based on classification hierarchies.

Persistent Data Objects in the Repository. Each data object is in OEM format. Th
type can be extracted from its representation and be used for
repository. We use the statement 'OBJECT x IN T' to express that
instance (element) of type 'T'. The type is represented as a term o
and base types (given by their name). For example, the most specifi

1995 is Selling-Vector(K-CAN$,K-CAN$,K-CAN$,K-CAN$). Consequently, object is represented in the repository as:

```
     OBJECT http://strassbourg.rubber.fr/finance/
                   Selling-Data-1990.oem
     IN Selling-Vector(K-CAN$,K-CAN$,K-CAN$,K-CAN$)
```

Membership of the data object to more generic types Vector(Int,Int,Int,Int) can be derived provided Int is a more gene We express such generalization in the repository by a statement Further classifications can be obtained when additional attributes a provided. Such attributes are highly domain-specific, especially wh from different organizations. Therefore, it is difficult to prescri used. Book-keeping features like 'creator', 'creation-date', 'vali reasonable first choices. Semantic information explaining the meanin problem class and applicable models is more useful for data classi common terminology among the users of the repository (which is beyon paper). In our running example, the attributes could be as follows:

```
     K-CAN$ ISA Int
     ...
     OBJECT http://strassbourg.rubber.fr/finance/
                   Selling-Data-1995.oem
     IN Selling-Vector(K-CAN$,K-CAN$,K-CAN$,K-CAN$)
     WITH ATTRIBUTE
          creator: “Francois Nome”;
          creation-date: “4-Nov-1995”;
          valid-thru: “31-Dec-1995”;
          remark-1: “entries correspond to quarterly selling
                      of Rubber-Europe”;
     END
```

DSS-Scripts are special data objects that describe a computation problem. In the example below, the script Earning-Forecast-1995 i arguments of type IFPS-Model (a plain text) or Selling-Vector, resp expected to be of type matrix(Int). Note that the input and output s searching for data objects matching a script’s signature and conformance.

```
OBJECT http://strassbourg.rubber.fr/scripts/Earning-
Forecast-1995.oem
IN DSS-Script
WITH ATTRIBUTE
     arg-1: IFPS-Model;
     arg-2: Selling-Vector({Int});
     arg-3: Exchange-Vector({pair(Real,Real)})
     result: matrix(Int)
     preferred-processor:
http://fribourg.rubber.ch/bin/IFPS
END
```

The processors of scripts (attribute preferred-processor in the ex
software packages installed on some computer in the network. It m
same DSS package is installed on several nodes. Then, there is a ch
strategy of the script. In the repository, we store a statement like

```
OBJECT http://fribourg.rubber.ch/bin/IFPS
EQUIVALENT-TO
http://strassbourg.rubber.fr/software/bin/IFPS
```

to represent this fact. It is used for the run-time scheduling of th

Some scripts may have the only task to filter, transform or
objects. They constitute no special case. For example, the result o
may be transformed into a spreadsheet format and then displaye
spreadsheet package.

5. Implementation Steps

To implement the proposed approach, two principle steps are r
DSS components have to be connected to the Internet, and second , t
in the repository for identification.

Connecting DSS Components to the Network. Our approach relies
protocols and interfaces. Therefore, setting up a running system d
amount of programming. First, the DSS processors must be accessible
This can be achieved by installing a so-called HTTPD server [WWW-C
the local node. The server has the basic function of accepting cal
returning some result (in our case data objects). It can be config
with the right URL to the local DSS component. Second, a local DSS
able to retrieve the arguments of a call (as encoded in the script
remote HTTPD servers. The standard-compliant library 'libwww' for
available from [WWW-Consortium 95]. Figure 7 shows a schematic v
component linked to the WWW.

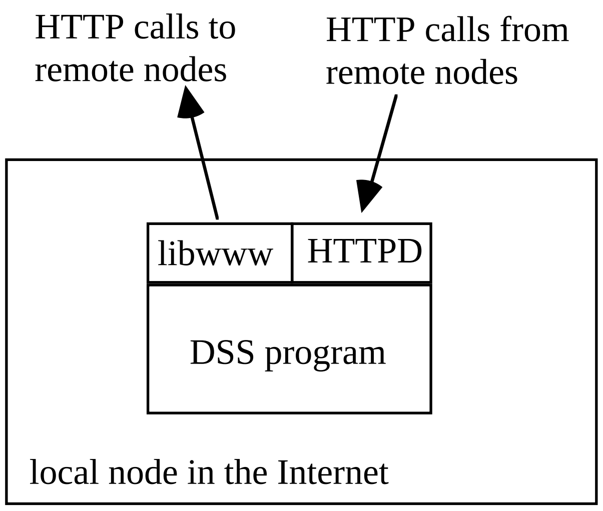
HTTP calls to remote nodes
HTTP calls from remote nodes
libwww
HTTPD
DSS program
local node in the Internet

Fig. 3Schematic architecture of a DSS processor linked to WWW

This architecture allows a user to activate a DSS processor from
certain script object. As third and final requirement for the conne
the result of a call to a script are expected to be in OEM forma
processor must be able to read/write such a format and transfer the
internal format. The effort for this function is minimal once the O
script is invoked by submitting a call like

```
     http://fribourg.rubber.ch
           /bin/IFPS?call=http://strassbourg.rubber.fr
           /scripts/Earning-Forecast-1995.oem
```

to the HTTPD server of a suitable processor. The DSS processor the
object (here Earning-Forecast-1995.oem) through its libwww inte
descriptions in the script object which are of type 'po-call', for
stored on a local file. e.g. http:// fribourg.rubber.ch/temp/arg-3
DSS processor can issue the corresponding call, in our running examp

```
     http://strassbourg.rubber.fr/software/bin/IFPS?
            call=http://fribourg.rubber.ch/temp/arg-3.oem
```

in the same manner it was called itself. The local DSS processor
deviate from the preferred processor specified in the script and c
processor, e.g. the IFPS package at the local site. This scheduli
control and is guaranteed not to influence the result of the comp
DSS processors must be equivalent. Finding equivalent processors is
(local or remote) repository.

Searching the Repository. The main functions of the repository sy
properties of data and script objects as shown above, and to fa
information about data and script objects. Three search strategies a

Simple search for scripts. Assume a decision maker has a problem encoded by
input data object and wants to find a suitable collection of DSS co
can be determined by matching the type of the data object with the
of the DSS scripts stored in the repository. The search may be limit
features of the DSS script. The search is efficient, more exactly,
of objects in the repository. Indexes on the input argument feat
average case performance to constant time.

Chain search for scripts. Assume that the repository contains some data tra
scripts P1,..,Pk such that the input argument of P1 matches the in
output for Pi (i=1,...k-1) matches the input of Pi+1. Then, suita
obtained by searching such chains and then finding a DSS scripts who
output of Pk. This search delivers more alternative solutions th
depending on the number of data transformation scripts known to t
search is linear in the number of matches between input and output a
components in the repository. In the 'worst' case (i.e., every
transformed to every output type), it is quadratic in the number of
quality of the search, and subsequently, the problem solution depen
of the set of transformation scripts

Interactive script object generation. This is both the most usefu
search. It assumes that the decision maker has more than one dat
definition and that some goal objects have to be computed by a comp
scripts. Theoretically, all combinations of chains, each starting w
objects, are applicable. Indeed, the number of combinations explod
input data objects and available scripts increases. The problem of
object is similar to scheduling problems. Since the repository has
about the available script objects (only input/output types plus s
the goal of the decision maker, we propose an interactive approach
script object. The user incrementally builds the script starting fr
Simple and chain search are used as auxiliary functions which offer
choice of possible component scripts operating on a selected data o
object or output of a script object). After validation the generate
in a repository for later use and re-use (then being component of ot

Implementation Properties. The script objects encode functional e
objects (OEM format). The functions are denoted by the URLs of the
the arguments are either functional expressions or URLs of data ob
following computational model for DSS processors:

Persistent data objects are never updated after their cr

That means that the OEM term referred to by an URL is always the sa
analogous to the treatment of variables in functional and logic pro

is different from the destructive updates allowed in imperative pr
The result of a script object only depends on its input objects. As
of the same DSS processor will have the same result if and only if
persistent data objects.

The computational model allows application of several heuristi
of script objects.

- Lazy evaluation lets arguments be evaluated only when actuall
  heuristic reduces the overall load on the network. Its implemen
  because it affects the internal routines of the DSS components.
- Caching materializes results of sub-expressions (parts of script
  type 'po-call' for speeding up subsequent calls of the same sub
  makes most sense for script objects having only persistent data
  their result never changes.
- Multi-processing allows arguments of a script object which are of typ
  called in parallel. This strategy is the opposite to lazy evalu
  response time provided the calls go to distant idle nodes in the r
  for this strategy is that the parallel calls do not interfere wi
  achieved by forbidding updates to persistent data objects.
- Load balancing can be enforced to optimize network traffic. Calls t
  (named by their URL in the script object) can be replaced by call
  objects located elsewhere. Load balancing can be incorporated
  (compile-time approach) or during evaluation (run-time approa
  processors may be installed redundantly at any time on any node
  overall system is scaleable to the workload. There is no need of
  heuristics can be applied with the (limited) knowledge of the loca

Script objects are hierarchical, i.e., they do not contain loc
script object will terminate if the evaluation of its components (e
objects) terminate. We conclude the following property:

> The evaluation of script objects will always terminate under
> strategy (e.g., top-down, depth-first).

This property allows DSS users not familiar with programming langua
without fearing to consume too much resources of the network. In
consumption can be estimated from a script object definition. The t
the sum of the processing times of the component scripts in the s
linear in the size of the script object. The processing times of tl
processors are in general indefinite. Response time, i.e., the time
object and receiving its result, depends on the degree of multi-pr
response time is the processing time divided by the number of dist
the processing. The actual response time also depends on data de
network bandwidth. Network costs (measured in the amount of dat
between distinct nodes) is also linear in the size of the script ob

only as many transfers occur as there are DSS processors referred
Network costs are also linear in the size of the input data objects
any call to a DSS processor is linear in the size of its input
assumption for DSS components since their purpose is to reduce de
large data sets to a small number of choices. Network costs can b
installing any DSS processor referred to in the script object on
Internet. This is an extreme case of load balancing whose expenses
local resources.

Integrity and Control On first sight, the distributed approach on the In
danger of uncontrolled access to models which where never meant t
reader should however note, that the repositories are specifically
space of meaningful data and scripts objects for a certain group o
case, a repository can just cover the objects on a local computer
hand, even small organizations have worldwide connections and will
for global decision support. The repository for such an organizati
restricted to cover those distributed objects important for its busi

As seen for the application of electronic data interchange (
information processing requires a clear understanding of the bus
participating organizations. We assume that this understanding prior to has
the exchange of models and software. The 'global' DSS, i.e. the sum
objects on the Internet, can never assumed to be consistent. Instea
which consistent subsets (maintained in the decentralized reposito
This global resource will increase in value when more and more d
scripts are published in the repositories. They are in fact
interoperability.

6. Final Remarks

A combination of three elements describe our approach. Uniform
and data objects allows the programs and data to be located anywhe
uniform data format, OEM, allows data exchange independent from th
language of the DSS components. Finally, decentralized repositories
distributed scripts and the evolution of the overall system by
semantic information.

The proposed approach relaxes the necessity of having centr
overall system. While the lack of a central clearinghouse creates
problem formulation, book-keeping issues, our approach empowers an
of decision makers (users) and DSS providers to participate in the
system. It also provides more flexibility in terms of DSS prolifera
Utilizing widely accepted protocols for data exchange keeps the i
reasonably low. Also, the effects of low bandwidths of existing I
reduced by duplicating DSS processors on the nodes (computers) whe
and a large number of nodes can be exploited via multi-processing.

Future efforts should concentrate on more sophisticated searc repositories and quality control of retrieved DSS scripts. Two othe cooperation among several decision makers and guidance in long-term The approach presented in this paper has only marginal support f makers cooperating in a decision script. To extend our approach to has to embed the individual script objects into complex script or p & Jarke 92]. Then, loops for scripts are likely to become an iss investigating such limited forms of such loop concepts.

References

[Ba et al. 95] S. Ba, Kalakota, A.B. Whinston, "Executable Documents DSS", Proc. 3rd Intl. Conf. on DSS Hong Kong, 22-23 June 1995, pp

[Barners-Lee 95] T. Barners-Lee, Uniform Resource Locators, Document http://www.w3.org/hypertext/WWW/Addressing/URL/URL_TOC.html, 199

[Bhargava et al. 95;1] H. Bhargava, A. King, D. McQuay, "DecisionNet for Modeling and Decision Support over the World Wide Web", Proc. 3rd Intl. Conf on DSS Hong Kong, 22-23 June 1995, pp.499-505.

[Bhargava et al. 95;2] H. Bhargava, R. Krishnan, D. Kaplan, "On Gene WWW-based Network of Decision Support Services", Proc. 3rd Intl. Hong Kong, 22-23 June 1995, pp. 507-515.

[Ellis & Jarke 92] C. A. Ellis, M. Jarke (eds.), Distributed Coopera Information Systems, Proc. 3rd Intl. Workshop on Intelligent & Co Information Systems, Report AIB-92-18, RWTH Aachen, Germany, 1992

[Felter 89] R. Felter, Decision Support Assistant, Deutscher Universitätsverlag, Wiesbaden, Germany, 1989.

[Genesereth & Ketchpel 94] M.R. Genesereth, S. P. Ketchpel, "Softwar Communications of the ACM, 37, 7, 1994, pp. 48-53 & 147.

[Goul et al. 95] M. Goul, A. Philippakis, M. Kiang, D. Fernandes, B. Client/Server Open-DSS Protocol Suite for Automating DSS Deployme World Wide Web", Proc. 3rd Intl. Conf. on DSS Hong Kong, 22-23 Ju 517-528.

[Nestor et al. 90] J.R. Nestor, J.M. Newcomer, P. Gianni, D.L. Stone and its Implementation, Prentice Hall, 1990.

[Papakonstantinou et al. 95] Y. Papakonstantinou, H. Garcia-Molina, exchange across heterogeneous information sources", Proc. 11th In Engineering, Taipei, Taiwan, 6-10 March 1995, pp. 251-260.

[Rasmussen et al. 91] J. Rasmussen, B. Brehmer, J. Leplat, Distribut Cognitive Models for Cooperative Work, John Wiley & Sons, 1991.

[WWW-Consortium 95] WWW-Consortium, CERN Httpd, Document http://www.w3.org/hypertext/WWW/Daemon/Status.html, 1995.