InterNeg Research Papers INR 06/02

Revised version appeared in:

Decision Support Systems

2004, Vol. 38 No. 2, (pp. 283-303)

Decision Station: Situating Decision Support Systems

Rustam Vahidov and Gregory E. Kersten

J. Molson School of Business, Concordia University, Montreal, Canada H3G 1M8
{rvahidov, gregory}@jmsb.concordia.ca

Abstract

Internet facilitates access to data, information, and knowledge sources, but at the same time threatens to cognitively overload the decision makers. This necessitates the development of effective decision support tools to properly inform the decision process. So far the response from the field of decision support systems has not been adequate. Internet technologies require new type of decision support that provides tighter integration and higher degree of direct interaction with the problem domain. The central argument of this work is that in dynamic and highly complex electronic environments decision support systems should be situated in the problem environment. A generic architecture, the set of capabilities for our vision of a situated DSS is proposed, and the architecture is illustrated with a DSS for investment management.

Keywords: decision support systems, software agents, situated DSS, active DSS, DSS architecture

1. Introduction

During the past thirty years since the conception of DSSs the business environment has changed in several ways. The most significant changes include:

- Globalization of economy and the growing complexity of economic relationships;
- Flattening of organizations and growing employee empowerment;
- Increasing need for fast response in the dynamic competitive environment;
- Explosion of information accessible through electronic networks;
- Emergence and growth of electronic commerce; and
- Better-informed and empowered customers [38].

These changes contribute towards higher degree of complexity of the environment and, subsequently the number and difficulty of problems faced by the decision makers. It appears, therefore, that the need for decision support tools should be, if anything, increasing [47]. Contrary to these expectations we are not witnessing an adequate heightening of interest reflected by the limited number of recent scholarly publications on the subject of DSSs.

With the growing interconnectedness of the business environment the "problem-solving" characterization of DSSs needs to be expanded and integrated. The "isolated DSS" view hampers their usefulness for today's decision makers and is incompatible with such new information technologies as enterprise resource planning, supply chain management and customer relationship management. DSSs need to be seamlessly integrated to the firm's information environment. They also need to empower users by providing them with relevant information, informing decision-making process, responding to the emerging situation, and being capable of influencing the environment in the desired direction.

The main purpose of this paper is to lay a foundation for a new generation of decision support systems, the situated DSSs, which we refer to as a *decision station*, and propose a model and architecture for these systems. We view such DSSs as being active and closely linked to their respective problem environments. Section 2 presents the theoretical background for the proposed architecture of a situated DSS. It mainly builds upon the active DSS paradigm and research in software agents. Section 3 discusses the changing role of modern decision support, introduces the new components of situated DSS and argues in favor of software agents as means of building such components. The architecture of a decision station, its major components and their formal representation are discussed in Section 4. Types and examples of situated DSSs are discussed in Section 5. Section 6 presents an illustration of the proposed approach using investment DSS. Section 7 presents conclusions and future work.

2. Theoretical Background

Our framework for situated DSS stems primarily from the developments in two disciplines: decision support systems, and software agents. In this section we will briefly discuss the relevant aspects of these fields in order to properly introduce our vision. We will not elaborate much on the subject of classical models for DSSs as we assume the readers are familiar with the major contributions to the field made in the 70s and early 80s. Instead, we will focus on more recent developments.

2.1 Recent Developments

It has been stressed that the need for decision support in the age of the internet and e-business is now becoming even more critical than before [47, 49]. However, while DSSs were one of the most popular areas of research in information systems in the past, a closer look reveals that lately the interest in DSS appears to be declining [8]. This somewhat paradoxical observation calls for a closer analysis of requirements for decision support tools posed by the new dynamic environment.

From the very beginning the focus of DSS research and development has been on generic problem-solving activities, for example, Expert Choice (www.expertchoice.com) and Decision Explorer (www.banxia.com). The sphere where actual business operations or transactions took place was often seen as secondary concern for the adoption and implementation of DSS. While criticism of the "stand-alone" DSS approach and the need for closely linking DSS with business work processes have been voiced [2], this theme has not yet resulted in the introduction of new concepts, frameworks or architectures.

The requirement of the DSS connectedness to its environment builds upon the concept of an active DSS [1, 24, 42]. The advocates of active DSS point out the weakness of traditional support for being passive, where the user has to have full knowledge of the system's capabilities and must exercises initiative to perform decision related tasks. An active DSS need not be capable of undertaking all the tasks on its own and completely without the user's intervention. Ideally, a proactive DSS would establish a two-way interaction with both the user and environment, and would be capable of maintaining those links if any of the entities is active. Such a system would allow for the integration of decision-making and decision implementation activities.

2.2 Insights from the software agent research

During the last decade a broad category of software referred to as agents or bots has gained tremendous popularity. Although software agents defy precise definition, several characteristics are often cited; they are proactive, reactive, exhibit social ability, intelligence, and purpose [13]. One central theme in agent-based technologies is that agents are situated in their environment; they are capable of sensing the state of the environment (e.g. load in electrical power grids, presence of e-mail messages in one's mailbox), and of affecting the state (e.g. switching power lines, sorting e-mail messages).

Jennings [25] notes that "an agent is an encapsulated computer system that is situated in some environment and that is capable of flexible, autonomous action in that environment in order to meet its design objectives." The emphasis here is on the existence of direct links with the problem environment.

The difference between situating a system in the environment can be illustrated with the difference between closed and open systems in artificial intelligence (AI). The situated "view" in AI has been gaining popularity since the late eighties [53]. The distinguishing characteristic of this perspective, as opposed to traditional "rationalistic" perspective, is the focus on the system interaction with its environment [32]. If systems are to have cognitive abilities, they need to be situated because cognition cannot be abstracted away from the environment. The behavior of a system is the combined result of its purpose and its interaction with the environment [43].

An agent can be a vehicle capable of purposeful seeking and using information sources. It can, therefore, be used as an advanced sensor with encapsulated intelligence. Agents can gather information to promote user awareness [5] and support decision-making (e.g., use airline databases to support travel planning [41]). Agents embedded in a DSS can decide on information being monitored, gathered and filtered in geographically distributed environments [6]. A cognitive agent can be used to amplify human perception and cognition in critical environments [11]. In e-business agents have been used to obtain information and undertake tasks on behalf of their principals [17, 35, 45]. Agents are also used to maintain databases of vendors [62], and assess their reputation [64].

3. The Changing Role of Decision Support

3.1 Situating DSS

One result of the information revolution is a very high level of connectivity that pervades multiple aspects of managing and conducting business activities. In many aspects the network is becoming the actual place for conducting business [44]. We've been recently observing an increasing integration of information systems in business environments. E-business and enterprise resource planning systems are just a few examples of integration of previously disparate information systems. While they provide some DSS functionality, there is a need for the development of decision support tools which can be tightly integrated with the business environment. In particular, the phenomenon of "ubiquitous network" needs to remain within the focus of DSS research.

Effective linking of DSSs to their problem environments would enable improvement of strategic capabilities of organizations through timely response to the dynamically arising challenges and management of organizations "by wire", that is, combining high level decision-making with automation and IS support of various business operations [18]. The terms "cockpit of the business" and "cyberspace cockpit" [38] have been coined to signify the new requirements for computer-based support. Furthermore, we're witnessing an increased interest in real-time, more responsive breed of DSSs [58]. On the consumer side, the predictions are made about the emergence of a "new breed of consumer ... more selective, better informed, and with a range of powerful tools at his or her disposal" [38].

One possible reason behind the inherent constraint in expanding traditional DSS out of the detached problem realm and into the dynamic environment comprising all business interactions has been the wide adoption of Simon's "intelligence-design-choice" problem solving model [62]. Although Simon later extended the model with monitoring, DSS research remained primarily focused on the three-phase model. Other suggestions to use alternative models that incorporate implementation and monitoring were proposed but not accepted in the mainstream of decision support research and development [4, 50].

The difficulty in the design and implementation of active and connected DSSs was caused by the lack of connectivity among different information systems in an organization and between these systems and the organization's environment. The ubiquitous network and pervasive computing demand new approaches that will allow a DSS to become a part of the information infrastructure, through interaction with its environment, and leveraging its cognitive capabilities through its connectedness to the decision problem's environment. This would provide a DSS with means to sense the problem environment, offer decision support to a decision maker and act upon the environment to adequately respond to his/her needs that may undergo changes and refinement during the process. In other words, these systems will be the *situated* decision support systems.

The above discussion stresses recent trends towards the adoption of the connected and situated concepts. The relationships between AI, agents, and decision support systems are presented in Table 1. Within AI, expert systems have been proposed as means of mimicking expert decision making. Situating these systems in the problem environments led to the development of intelligent agents. Situating DSSs within problem environments, providing them with capabilities to seek and sense the relevant data, and giving them ability to change the environment can bring about the type of support that today's organizations need.

Table 1. Comparison of DSS, expert systems, and intelligent agents

	Decision support	Artificial intelligence
Traditional	DSS	Expert systems
Situated	Situated DSS	Intelligent agents

The goal of a situated, connected and active DSS is to provide all services necessary for decision-making and implementation. To reflect the comprehensive nature of such a system and also its integration with other systems and with the environment we call a *decision station* (DS). Thus, DS can be seen as a software component of a dedicated workstation. A DS is used to: (1) sense what's going on in the problem domains; (2) utilize traditional DSS facilities to inform decisions; (3) make choices; and (4) undertake implementation and monitoring activities.

3.2 Active and passive components

The key components for situating DSSs are sensors and effectors. In Table 2 two types of these components, as well as their capabilities and functions are presented. The examination of the basic capabilities of sensors and effectors reveals the fact that some of them are more "advanced" than the others. For example, the ability to search for new information sources (more generally, adaptive capability) is more advanced than simple connecting capability. This insight leads to a dichotomous distinction between the "active" and "passive" capabilities of sensors and effectors.

Table 2. Active vs. passive capabilities of sensors and effectors

	Sensors		**Effectors**	
	Capabilities	**Supported functions**	**Capabilities**	**Supported Functions**
Passive	Connecting	Importing data	Connecting	Exporting data, carrying out actions
	Transforming	Filtering, pre-processing, noise reduction, etc.	Transforming	Converting decisions into actions
	Alerting	Drawing user's attention, signaling to effectors	Querying	Requesting information or authorization from user or sensors
Active	Adapting	Search for new sources, attuning transformational and alert generation logic	Adapting	Identifying alternative destinations, Adjusting transformation and alerting logic, bidding tactics, etc.
	Planning	Determining order of actions, scheduling sensory (monitoring) and adapting actions	Planning	Determining order and scheduling of actions

Sensors have five capabilities that support their main functions. Some capabilities affect others, for example, the adapting capability can modify the connecting capability in order to reflect the identification of new information sources. The division of the basic capabilities into two categories is not completely "clear-cut" and is primarily determined by our perception of the level of the proactive nature present in the capabilities. To further clarify the distinction between the active and passive situating components let's consider an example of a DSS for supply chain management.

With the growth of B2B commerce the supply chains are expected to be more dynamic [9, 40]. Assume that decisions need to be made regarding which parts to purchase from which suppliers. Consider first the case with passive components. The (passive) sensors would be able to collect the prices and other relevant information on parts from the existing vendors by consulting their databases. The decision maker could use this information to run optimization models in DSS kernel and perform sensitivity analysis in order to come up with the desired decision. The decisional output then will be given to (passive) effectors that would inject necessary information into the order forms and submit the orders to the respective vendors (e.g. through the Web, e-mail, fax).

Consider now the case with active DSS components. The sensors could monitor the prices by the vendors on a regular basis and notify about any changes in price for the timely response (similar scenario has been described as an application of agent technologies to DSS in [20]). Moreover, the sensors would be able to look for and identify new potential vendors through consulting Yellow Pages, auctions or crawling the Web. The sensors could also consult the auctions to see in what range the prices for the parts are. The monitor in DSS kernel could decide whether to invoke DSS functions depending on the state of the environment. The user would make a decision using the tools of DSS, possibly including ranges instead of crisp values and pass the results to the effectors. It is possible that the effectors would be able to conduct automated bidding in an auction in order to make the best deal. Upon winning of the bid they would finalize the transaction and monitor the delivery of the parts by consulting other organizational IS.

3.3 Agents as decision station building blocks

As we have mentioned earlier, the DSS research has stressed the importance of active and high cognitive level DSS. In this paper we argue for a situated DSS. Ideally, a new type of DSS would be fully situated, active, and with advanced capabilities. These characteristics point towards employing agent technologies for building such systems. It is not surprising that most of the work reviewed by us relies on agent technologies, since agents have the characteristics that fit our purposes quite well. Let's review some of the important agent characteristics in light of the components of a DS.

Situatedness is the feature that helps sensors and effectors link to the respective problem environments. The target environment could be either virtual or physical one, or it could span across both. Autonomy and proactiveness are necessary for allowing the components of a system to take initiative and execute various related tasks in an automated fashion. Intelligence may be required for performing high-level cognitive tasks, e.g. negotiations. Social ability allows the situated DSS to communicate with other systems as well as the user; and reactivity enables monitoring the environment, or the user to signal and handle significant problem-related events.

The recent articles on the future directions of DSS research have stressed the importance of using agent technologies in DSS [7, 48]. Use of agents for building active DSS has been discussed earlier. Angehrn [1] proposed a vision where a decision maker is supported by a team of stimulus agents forming a virtual teamwork. Hess et al. [20], presents a generic architecture with DSS components (models, data, dialogue) being managed by agents. Stohr and Viswanathan [51] proposed the use of recommendation agents as the basis for DSS, and Vahidov and Elrod [61] proposed the use of critiquing agents within DSS.

An interesting model of an agent-based distributed DSS that comes close to our notion of a decision station is given in [21]. Here the three DSSs encapsulated in raw materials, production, and finished goods inventory agents have been described. The agents automate much of the work, enjoying limited authority in performing minor and routine tasks. They also collaborate with the decision makers when necessary. The agents obtain all necessary information from electronic sources, and their decisions are implemented electronically as well (i.e. submitting an order, communicating the results of a decision, etc.). Hence the systems are both active and situated.

A closely related research direction is reported by Levary et al. [29] who focuses on the supply chain management. The authors discuss their work at IBM Research on "sensing and responding enterprise" model. Coming from Operations Research perspective, the group proposed the use of additional "sensing and responding" layer sandwiched between the supply chain planning and supply chain execution layers in order to deal with emerging problems and opportunities. The sensing-and-responding capability was carried out by intelligent agents. This work strengthens our belief in the appropriateness of the notion of situated decision support.

4. Decision station

In this section we present the generic architecture of a DS followed by detailed capabilities and functions of its five components. In Section 5 we discuss several examples of DS functionalities in selected domains. An example of a DS is given in Section 6.

4.1 Generic architecture

Situating the DSS necessitates the addition of at least two key capabilities: (i) the capability to access the state of affairs, and (ii) the capability to change the state of affairs. The former is achieved with sensors and the latter with effectors. Sensors, effectors (together with the kernel), and active user interface comprise generic DS illustrated in Figure 1.

The *kernel* is composed of the DSS facilities in a traditional sense (i.e. database, models, and knowledge base) relevant to a problem domain. It includes an active component: the DSS *Manager*. The inclusion of the Manager reflects the view that the situated DSS needs to be active irrespectively of the presence of the user and capable of performing certain tasks autonomously, for example, contacting the user, preparing the system for interaction prior to the user's request, and even making decisions when the user cannot be contacted.

The Manager requires a knowledge base containing, among others, business rules, to be capable of performing some of the tasks autonomously. Since our primary focus in this paper is the links of a DS with the problem environment, a detailed description of functionality of the Manager is not discussed here.

Sensors capture the data relevant to problem domain from a variety of sources. The sensors however, should not be thought of only as mere means of capturing the data. They may incorporate more advanced functions as well, i.e. search for relevant sources, filtering and pre-processing of data, alerting generation and other useful features.

Effectors are the devices used by a decision station to send signals to the problem environment with the purpose of directly altering current state of affairs. Similarly to the sensors, the effectors are not necessarily simple vehicles of decision execution or communication, but may engage in different activities required to implement a decision (e.g. converting the decision into more detailed plans, optimizing the well structured aspects of a decision, determining sequence of actions, monitoring execution of a decision, and conducting negotiation in the course of implementing a decision). Note that the separation of sensors and effectors here is prompted by the distinct roles they play in carrying out the interactions with the problem domain. Implementation-wise these parts can be integrated into a single module.

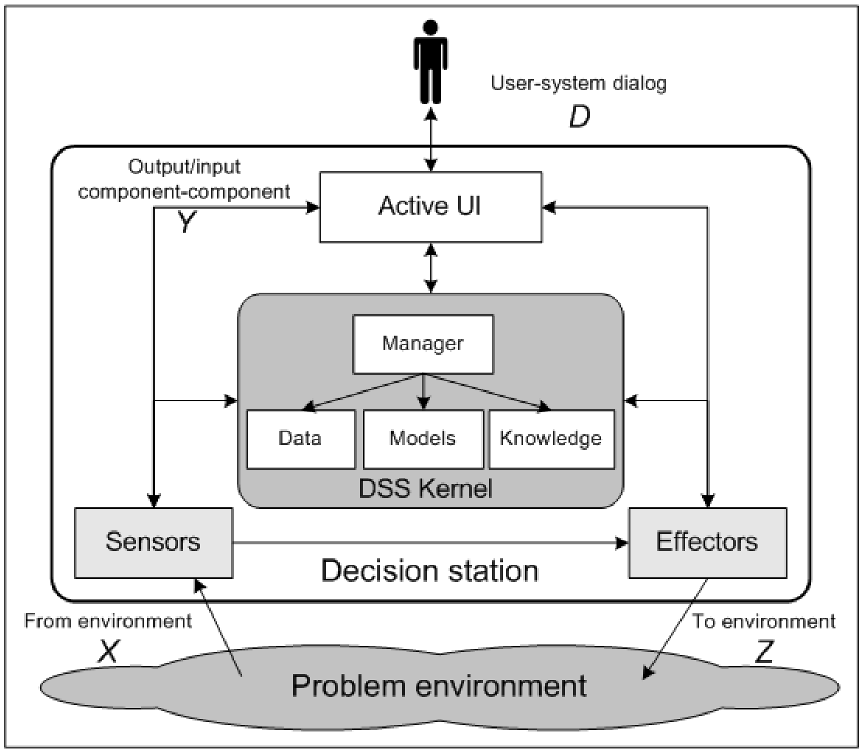


Figure 1. Generic architecture for a decision station

Active user interface has been included to signify new ideas and developments in facilitating human-machine dialogues. Such interfaces may incorporate synthetic characters, reside on wearable devices, and have learning capabilities. The functionalities of decision station components are discussed below.

The structure of the decision station can be described as:

$$DS = \{S, E, N, KN\}, \quad (1)$$

where: S indicates sensors, E – effectors, N – user interface, and KN – the DSS kernel. In the following sections each of these constructs is described in more detail.

In Figure 1 the information exchanged between the DS components and the user is indicated with D. X denotes information obtained from, and Z passed to the environment. Y denotes the information flow between the DS components. In the following sections the component's type is not indicated in its local inputs and outputs unless it is necessary to avoid ambiguity.

4.2 DSS sensors: assessing the state of affairs

Delivering new and relevant information is an important task in all phases of decision-making, in particular in the intelligence and monitoring phases. In a DS this task is performed by means of sensors, which are various tools used to access problem domain and assess its state of affairs. In a trivial case, the sensors import relevant information into DSS from the environment. More advanced sensors need capabilities for locating, filtering and transforming relevant information, and generating alerts. With the explosion of the information available in electronic form on the Internet the task of finding the right sources, filtering useful information and presenting it in a suitable way to the decision maker has become of the critical importance. Hence, it is not surprising to see the growth of smarter, more advanced tools used for these functions.

The sources of information can vary. These could be embedded in physical environments, various databases, or the Internet [12, 22, 46]. Depending on the definition of problem domain the DS sensors should have the means to access a variety of sources.

We propose the following set of sensor functions: accessing, filtering, conversion, and monitoring of information and generating alerts, should a critical situation arise. More advanced features including identification of new sources of information (e.g. through interacting with search engines, or crawling the web), and fine-tuning of some of the sensing functions (e.g. adjusting thresholds that trigger alert generation) could also be provided. This leads to the following model of a generic sensor:

$$S = \{C, G, K, R, O\}, \qquad (2)$$

where: C is the set of sensor capabilities, G is the set of goals, K is the knowledge-base of the sensor that can schedule and direct its use of capabilities, R denotes requests received from other parts of the system, and O denotes operations that a sensor performs; the concrete utilization of a single capability at some point in time. Henceforth it will be assumed that the "null" action, that is, "do nothing" is also a valid operation.

The sensor "reads" the state of the environment $\boldsymbol{x}$ ($\boldsymbol{x} \in X$) and transforms it to outputs $\boldsymbol{y}$ ($\boldsymbol{y} \in Y$), that is, $S : X \rightarrow Y$. The inputs and outputs are those of a sensor, not of the entire DS.

The five capabilities of sensor C considered here are:

$$C = \{ plan, conn, trans, alert, adapt\}. \qquad (3)$$

The key capability of an advanced sensor is the planning capability *plan* that decides on the order and time in which the other capabilities are invoked, which sources of information are sought and used, and how the sensor's outputs are directed. For example, depending on the volatility of the market the sensor may decide to monitor the developments more or less frequently. In a degenerate (trivial) case the planning capability is reduced to executing pre-specified fixed operations, such as reading certain piece of information at regular fixed intervals. This capability utilizes the sensor's knowledge base K. In order to enable planning capability, the sensor should exhibit a goal oriented behavior. Therefore, representation of goals is necessary.

$$\boldsymbol{o}_{t,T} = plan\ (K_{t,}, \boldsymbol{x}_t, \boldsymbol{g}_t\ , \boldsymbol{c}_t), (\boldsymbol{o}_{t,T} \in O, \boldsymbol{g}_t \in G, \boldsymbol{y}_t \in Y, c \in C),$$

where: c is a capability defined by (3), $\boldsymbol{o}_{t,T}$ is the sequence of operations at time t ($0 \leq t < T$) for the next T periods (T is the planning horizon):

$$\boldsymbol{o}_{t,\,T} = [o_{t+1}, o_{t+2}, \ldots, o_{t+T}]\ , \qquad (4)$$

and $\boldsymbol{x}_t$, $\boldsymbol{g}_t$, $\boldsymbol{c}_t$ are the inputs, goals and capabilities of a sensor respectively at time t.

Below we discuss capabilities possessed by DS components in a generic form; their concrete implementations depend on the problem domain. In Section 6 we provide examples from the investment domain.

Let's assume that the environment is described by a vector $\boldsymbol{x} \in X$. The connect capability *conn* of S allows direct access to the sources and information passing to other parts of the system without any modification, that is:

$$conn: \boldsymbol{y} = \boldsymbol{x}, (\boldsymbol{x} \in X, \boldsymbol{y} \in Y).$$

In general $Y \subset X$ because the sensor selects and passes only some environment descriptors.

Information filtering, conversion, noise reduction and other preprocessing done by the sensor are due to its transformational capability *trans*:

$$\boldsymbol{y} = trans\ (\boldsymbol{x}), (\boldsymbol{x} \in X, \boldsymbol{y} \in Y).$$

Note, that in this case the number of outputs (average ranking) could be different from the number of inputs (individual rankings).

The capability to generate signals that require immediate users' attention or actions of the other parts of the system is *alert*.

$$\boldsymbol{y}_a = alert\ (\boldsymbol{x}), (\boldsymbol{x} \in X, \boldsymbol{y}_a \in Y_a),$$

where Y_a, ($Y_a \subset Y$), is a set of alert messages.

Although *alert* is a specialized capability of *trans* it is separated here because of its impact on other parts of the system. While *conn* and *trans* produce output that is used when needed, *alert*'s outputs are used when they are produced. The sensor communicates directly with the user interface to alert the user, with the DSS kernel, or with the effectors (see Figure 1). The decision regarding the invocation of the *alert* capability and components to which the alerting signal should be directed can be done by the *plan* capability.

The capability of a sensor to seek different information and to modify its own capabilities is *adapt*. For example, we already mentioned a possibility of tuning a threshold for alert generation. Another example includes changing the *conn* capability by identifying and connecting to new sources of information. In general, the *adapt* capability modifies the capabilities of the sensor at time t based on τ previous periods described by input-output-capability triplets:

$$C_t = adapt\ \{(\boldsymbol{x}_{t-\tau}, \boldsymbol{y}_{t-\tau}, C_{t-\tau}), \ldots, (\boldsymbol{x}_{t-1}, \boldsymbol{y}_{t-1}, C_{t-1})\}, (1 < \tau \leq T). \tag{5}$$

The information gathering capabilities of a sensor could be invoked on a regular basis or in reply to the requests R from other parts of the system.

Figure 2 illustrates the sensor's capabilities and information flow between the environment and other DS components. To simplify the figure we did not include the capability *adapt* which affects all other capabilities and components. Information about its past states, which is used by the capabilities *plan* and *adapt*, is stored in its database (memory).

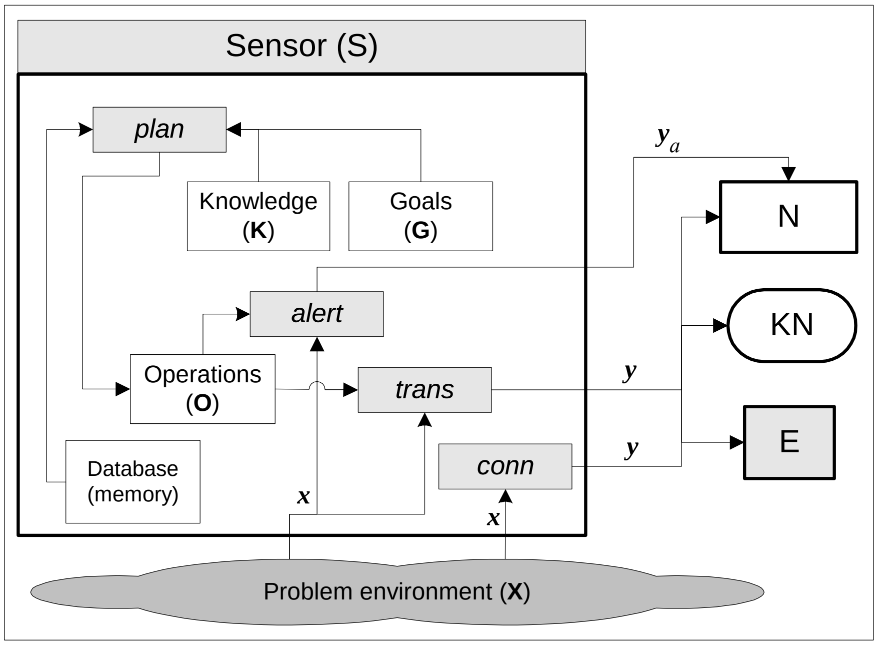


Figure 2. Architecture of a sensor

4.3 Effectors: changing the state of environment

The focus on the intelligence-design-choice trinity in DSS research largely led to the oversight of the implementation phase. With diminishing requirement of “switching media” when moving from decision-making to decision implementation, systems should enable implementation as well as monitoring of the results of decisions [38]. For example, an investor should be able to view the financial indicators and news, performance of his/her portfolio, obtain decision support in building/improving the portfolio, make buy/sell decisions, and implement those decisions through an online system. Moreover, since 98% of the computing power is embedded in different types of devices facilitating proactive computing with humans placed “out and above the loop” [56], the effectors can also reach beyond the virtual world and into the physical one.

Implementation primarily involves carrying out the decisions, but it may also entail planning and optimization activities, monitoring of execution, reviewing, and negotiating changes, if necessary. Conduct of these activities requires that the effectors have advanced capabilities. For example, an effector with the optimization capabilities [10] may finalize the details; find the best way of carrying out; and overview the execution of decisions. Another possibility is to delegate the authority and capability to conduct automated negotiations within a certain range of criteria to the effectors as part of the decision implementation process. For example, production decisions may require purchase of items from suppliers with whom effectors could negotiate the purchase terms [40].

In general, effectors can be described as:

$$E = \{C, K, G, O\},$$

Where C is the set of capabilities, K is the knowledge-base of the effector that schedules and directs its use of capabilities, G denotes the goals of the effector, and O is the set of operations that effector performs. We have included the following set of effector capabilities:

$C = \{plan, conn, trans, query, adapt\}$,

As in the case with sensors, the key capability of an effector is planning and it is used to establish the order in which other capabilities are invoked in order to achieve a given goal $\boldsymbol{g}$:

$$\boldsymbol{o}_{t,T} = plan\ (K, \boldsymbol{y}_t, \boldsymbol{g}_t\ , \boldsymbol{c}_t), (\ \boldsymbol{g}_t \in G, \boldsymbol{y}_t \in Y_{DS}, \boldsymbol{c}_t \in C), \qquad (6)$$

where the sequence of operations for the next T periods of time $\boldsymbol{o}_{t,\,T}$ is defined in (4), and $\boldsymbol{y}_t$, $\boldsymbol{g}_t$, $\boldsymbol{c}_t$ are respectively the inputs, goals and capabilities of a sensor at time t. Y_{DS} denotes the set of the inputs from other components of DS.

The connecting capability is the ability to directly export or write the values of decision variables from a sensor or the DSS into some action variable that attempts to change the state of affairs. Let Z denote the effector's outputs directed to the environment, then we have:

$$conn: \boldsymbol{z} = \boldsymbol{y}_{DS}, (\boldsymbol{z} \in Z, \boldsymbol{y}_{DS} \in Y_{DS}).$$

The output vector $\boldsymbol{z}$ contains actions, e.g., the sensor sends orders to an online store.

The transformational capability requires the formulation of action $\boldsymbol{z}$ ($\boldsymbol{z} \in Z$) in order to implement a decision generated by the DSS or a request sent from a sensor or the user (via active UI), e.g., informing prospective vendors about a request for proposals, or executing transaction:

$$\boldsymbol{z} = trans(\boldsymbol{y}), (\boldsymbol{z} \in Z, \boldsymbol{y} \in Y).$$

The effector may also need additional information in order to implement the decision. Its output Y that is directed to other components includes querying in order to obtain additional information necessary to undertake an action:

$$\boldsymbol{y}_q = query(\boldsymbol{y}_{DS}), (\boldsymbol{y}_q \in Y_E, \boldsymbol{y}_{DS} \in Y_{DS}, Y_E \cup Y_{DS} \subset Y),$$

The capability of an effector to change its own capabilities allows it to adapt over the time similar to sensors in (4). Effectors, use also their past actions, that is they take into account ι previous periods described by input-output-capability-action:

$$C_t = adapt\ \{(\boldsymbol{z}_{t-\tau}, \boldsymbol{y}_{t-\tau}, C_{t-\tau}), \ldots, (\boldsymbol{z}_{t-1}, \boldsymbol{y}_{t-1}, C_{t-1})\}, (1 < \tau \leq T).$$

In addition to the above capabilities, the effector may also have some capabilities of the sensor. The two perhaps most useful are: (1) the interactivity capability (inter) to respond to the environmental changes in an autonomous manner; and (2) the capability to generate alert signals (*alert*).

Figure 3 illustrates the effector's actions directed to the environment, capabilities and information flow between the effector and other DS components. To simplify the figure we did not include the capability *adapt* which affects all other capabilities and components. The information about its past states, which is used by the capabilities *plan* and *adapt*, is stored in its database (memory).

4.4 Active interfaces: towards advanced collaboration

An idealistic vision for the DSS to establish synergy between people and machines had not come to realization in the past. Because decision makers can easily be overwhelmed with the amount of information and multitude of modeling tools available through their computers the task of designing an effective, in addition to easy-to-use, user interface is no less important than before.

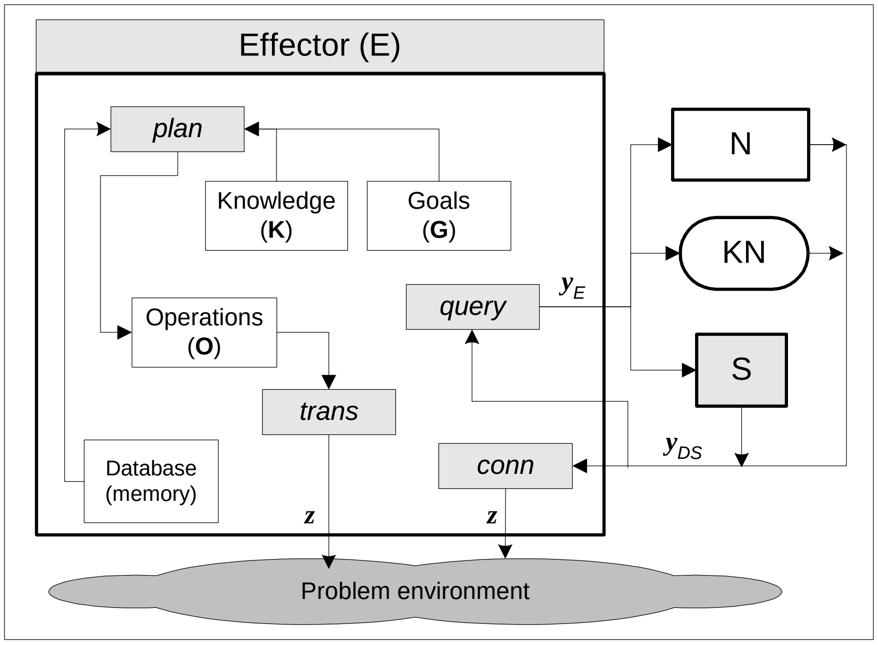


Figure 3. Architecture of an effector

With the growth of the web one can expect more user interfaces having web browser-like characteristics which are well known and easy to use. However, while basing user interface on WWW promotes standardization and familiarity level of users as well as places minimal requirements on their geographical location, a higher level of human-machine collaboration can be achieved through promoting more active interfaces. Such mixed-type interfaces could help alleviate information overload by combining direct manipulation with the automation [23]. Negroponte predicts that: “Future human-computer interface will be rooted in delegation, not the vernacular or direct manipulation” [39]. These interfaces populated with personal digital assistants and virtual secretaries will be able to perform certain tasks, including scheduling, filtering information, learning user preferences and many others in an (semi-) autonomous fashion [3, 23, 34].

To advance use of visual channels and psychologically comfort the user new directions in the design of virtual beings can be employed [27]. In [14] the use of animation in DSS user interfaces is presented. Empirical studies suggest that people prefer more natural-like personae [37]. There is some research in the direction of designing characters with advanced psychological dialogue skills [57]. The opponents, however, warn about the possible negative legal consequences of employing such beings, in particular in e-business transactions [19]. Other, more exotic directions, include, the virtual work environment including room elements and furniture [52]. In general, it is reasonable to expect a higher degree of human-machine integration through the use of such devices as wearable computers [63].

To set apart these new developments from the traditional notion of port-like user interfaces, we use the term “active interface”. We describe the active interfaces as follows:

$$N = \{C, V, K, G, O\},$$

where: C is the set of capabilities of the interface, V is the set of user characteristics, (user profile), K is the knowledge base of the interface, G denotes the goals that the interface may have (e.g. the maintenance of the accurate user profile), and O is the set of operations the interface performs.

The set of capabilities, in general, includes:

$$C = \{conn, trans, alert, query, adapt, plan, profile\},$$

The active interface N is an intermediary between the user and the system. Therefore, its communication is two-way and it is convenient to think of separate "languages" used to communicate with the system and with the user. As indicated in Figure 1, Y denotes messages used to communicate with other DS components, and D denotes dialog elements used in communication with the user. As before Y_{DS} denotes the set of inputs to the interface from other components; Y_N is the set of N outputs to the components. Further, D_U is the set of user's inputs, D_N is the set of outputs to the user, and $D = D_U \cup D_N$.

The capability to connect the user with the system means N passes information received from other components to the user, that is:

$$conn: \mathbf{y} = \mathbf{d}, (\mathbf{y} \in Y_{DS},\ \mathbf{d} \in D_N),$$

$$conn: \mathbf{d} = \mathbf{y}, (\mathbf{y} \in Y_N,\ \mathbf{d} \in D_U).$$

Interaction with the user may require transformation of inputs and undertaking actions that allow the inputs to be represented in a rich and easy to understand context. This may involve relating sensors', effectors' and the DSS' activities in a historical perspective, and their comparison with information obtained (via the sensors) from other systems. The transformation capability can be broken into a pair of capabilities used to transform (translate) messages from the user to the system and back to the user:

$$\mathbf{y} = trans(\mathbf{d}),\ (\mathbf{y} \in Y_N,\ \mathbf{d} \in D_U),$$

$$\mathbf{d} = trans(\mathbf{y}),\ (\mathbf{y} \in Y_{DS},\ \mathbf{d} \in D_N).$$

The interface may need additional information from other DS components and from the user. Its output directed to other components includes querying for additional information necessary in order for N to formulate output directed to the user:

$$\mathbf{y} = query(\mathbf{d}\ , \mathbf{y'}),\ (\mathbf{y} \in Y_N\ \mathbf{,}\ \mathbf{y'} \in Y_{DS},\ \mathbf{d} \in D_U).$$

Similarly, interface N queries the user when it receives ambiguous input or requires clarification to formulate requests directed to other DS components:

$$\mathbf{d} = query(\mathbf{d'}\ , \mathbf{y}),\ (\mathbf{y} \in Y_{DS}, \mathbf{d} \in D_N\ \mathbf{,}\ \mathbf{d'} \in D_U),$$

alert denotes the capability to generate alert signals directed to the user. If something goes wrong with implementation (e.g. product was not delivered), these signals can be used to prompt an immediate action and to draw user's attention:

$$\mathbf{d}_a = alert\ (\mathbf{y}),\ (\mathbf{y} \in Y_{DS},\ \mathbf{d}_a \in D_a\ , D_a \subset D_N).$$

Adapting capability allows the interface to change its own capabilities, e.g. to alter the mode of dialog:

$$C_t = adapt\ \{(\boldsymbol{y}_{,t-\tau}, \boldsymbol{d}_{,t-\tau}, C_{t-\tau}), \ldots, (\boldsymbol{y}_{t-1}, \boldsymbol{d}_{,t-1}, C_{t-1})\}, (1 < \tau \le T, \boldsymbol{y} \in Y_{DS}, \boldsymbol{d} \in D_N)$$

Planning is the capability of the interface to act and interact with a purpose. It allows to determine the order in which the other capabilities are invoked in order to achieve a given goal:

$$\boldsymbol{o}_{t,T} = plan\ (K, \boldsymbol{y}, \boldsymbol{d}, \boldsymbol{g}_t, \boldsymbol{c}_t), (\boldsymbol{g}_t \in G, \boldsymbol{y} \in Y_{DS}, \boldsymbol{d} \in D_U, \boldsymbol{c}_t \in C),$$

where the sequence of operations for the next T periods of time $\boldsymbol{o}_{t,T}$ is defined by (3).

The elicitation of user profiles from the dialogues with the user is done with profiling capability:

$$\mathbf{v}_t = profile\ ((\boldsymbol{d'}_{,t-\tau}, \boldsymbol{d}_{,t-\tau}), \ldots, (\boldsymbol{d'}_{t-1}, \boldsymbol{d}_{,t-1})), (1 < \tau \le T, \boldsymbol{v} \subseteq V, \boldsymbol{d'} \in D_N, \boldsymbol{d} \in D_U).$$

The profile is updated using information on past interactions with the user. For example, from the past dialogs the interface may infer that the user is a risk-taker. We are assuming that the user profile is implicitly present in all of the above formulas expressing different capabilities except for the basic connecting capability. We haven't explicitly included it to reduce the complexity of expressions.

4.5 DSS kernel

A number of representations for describing functionality of DSS kernel (i.e. traditional DSS) appeared in the past [50]. Since our primary interest here is in the relationship of DSS to its situating components, we will focus on these interactions. We describe DSS kernel as:

$$KN = \{C, G, K\},$$

where: C is the set of capabilities of a DSS, G is the goals of a kernel, K is the knowledge base that directs the use of it's capabilities.

The DSS has the following capabilities:

$$C = \{query, generate, suggest, adapt, plan\}.$$

query is the capability of producing results upon input which the DSS kernel obtains from other components, e.g. data retrieval, solution of linear programming model. The DSS may need additional information from other DS components in order to respond to the query:

$$\boldsymbol{y} = query(\boldsymbol{y'}), (\boldsymbol{y} \in Y_K, \boldsymbol{y'} \in Y_{DS}).$$

Here Y_K denotes DSS kernel outputs to other components.

generate denotes the capability to generate, based on the output from other components (Y_{DS}), decisional outputs directed to effectors (E):

$$\boldsymbol{y} = generate(\boldsymbol{y'}), (\boldsymbol{y} \in Y_E, \boldsymbol{y'} \in Y_{DS}).$$

where Y_E is the set of outputs directed to effectors

This capability also allows the DSS kernel to make autonomous decisions and react to inputs to the kernel within the pre-specified limits of its authority. It uses information obtained from sensors to generate actions to be implemented by effectors:

$$\boldsymbol{y} = generate(\boldsymbol{y'}), (\boldsymbol{y} \in Y_E, \boldsymbol{y'} \in Y_S),$$

where Y_S is the set of inputs from sensors.

suggest denotes the capability of DSS to proactively generate suggestions $\boldsymbol{y}$, ($\boldsymbol{y} \in Y_K$) directed to the user through the interface N, based on the current inputs from other components (Y), goals (G), and user profile (V):

$$\boldsymbol{y} = suggest(\boldsymbol{y'}, \boldsymbol{g}, \boldsymbol{v}), (\boldsymbol{y} \in Y_K, \boldsymbol{y'} \in Y, \boldsymbol{g} \in G, \boldsymbol{v} \in V).$$

adapt denotes the capability of a DSS kernel to change it’s own capabilities:

$$C_t = adapt\ \{(\boldsymbol{y}`_{t-\tau}, \boldsymbol{y}_{t-\tau}, C_{t-\tau}), \ldots, (\boldsymbol{y'}_{t-1}, \boldsymbol{y}_{t-1}, C_{t-1})\}, (\boldsymbol{y}_{.} \in YK, \boldsymbol{y}` \in Y_{DS}.).$$

The DSS, similarly as the effector (6) and other components, has planning capability *plan*, i.e. to determine when to invoke which capabilities:

$$\boldsymbol{o}_{t,T} = plan\ (K_{t,}, \boldsymbol{y}_t, \boldsymbol{g}_t, \boldsymbol{c}_t), (\boldsymbol{g}_t \in G, \boldsymbol{y}_t \in Y_{DS}, \ \boldsymbol{c}_t \in C).$$

5. Types of situated DSS

We are particularly interested here in the roles of sensors and effectors play in achieving tight integration between the environment and the DSS. As we already mentioned, both “situators” and mediators between the DSS and the environment can be implemented as one unit. For instance, the same software can implement viewing account balance and handling withdrawal. But it is not the case in general. For example, in e-commerce the sensors could be software agents used to review the prices of a certain category of products, while the effectors may be separate agents responsible for payment and delivery.

We have used the terms “problem domain” or “environment” to denote what the mediators interact with. These terms need some clarification. Consider for example, purchase of a good. From the buyer’s perspective, the task of carrying out a transaction and actual shipping of the product is not his/her task. The task of the buyer is to make a payment, and, monitor the progress of the delivery. Therefore, those elements of the information delivery and decision implementation that are essential to the conducting of the task, but wouldn’t be normally executed by the potential user of a system belong to the environment. Hence, the system that the user uses to purchase a good wouldn’t get involved in the details of packaging, and managing the shipment of a good. In particular, the implementation of a decision can trigger decision-making process by other entities in a chain-like fashion.

As we have discussed, both mediators can have a range of capabilities. We can distinguish trivial and advanced capabilities, where the trivial moderator would include connecting and basic transforming capabilities. The increasingly more advanced capabilities include transforming, alerting, interacting, adapting, and planning capabilities. On the other hand, the DSS may have both mediators handling all traffic from and into the environment on one hand, and part of the traffic on the other hand. In the former case we have a fully situated DSS, or a Decision Station, while the latter case can be called a ‘semi-situated’ DSS.

Consider, for example, a DS with trivial capabilities. In the simplest case one can think of an on-line banking system. Here the user can view balances on different accounts, do “what-if “ analysis to see how much amount he or she could transfer to a foreign currency account considering the exchange rates, and perform the transfer. The user actually accesses information, makes a decision, and implements a decision without having to change the media. A more elaborate extension of this example would be on-line trading. Many e-commerce systems fall into this category. Consider, for example purchasing an air ticket on-line. Here the users can specify queries, view alternatives, modify criteria, make a purchase, receive confirmation, and even obtain an electronic ticket.

In manufacturing, it has been stressed that IS and operations should be properly integrated [15]. The range of manufacturing systems including CAD/CAM/CAPP, CNC, AGV, AR/AS would fit well into such an integrated system. In such an integrated system a DSS with planning and analysis capabilities could write business decisions to the business database, which would be then used by CAD/CAM/CAPP systems as inputs to make design decisions, which, in turn will be entered into the engineering database, and later translated into actions for CNCs, robots, and other devices coming into contact with the physical world [29].

A semi-situated DSS with trivial mediators would let some of the traffic from or into the environment go through other channels, than those directly employed by DSS. Third-party comparison shopping tools (see e.g. [34]) and e-commerce sites that enable running queries by the buyers, but do not allow to actually execute transactions, fall into this category. This approach can be justified when a value of a particular good to be sold is considered rather high. For example, when GM and Ford were trying to make an on-line sale to the customer an option, Toyota decided to refer the customer to the dealer [33]. In another example, Berndt describes a customer web-based DSS that enables users to view important information about providers and surgeons. Based on this information, the customers can make health care decisions, but implementation goes beyond the system. In manufacturing Lockamy argued for an integrated performance measurement system that would capture operational measures (through sensors) and describe the situation at hand at a high level [31].

A semi-situated DSS with advanced mediators will have either sensors or effectors with the advanced capabilities. An example is a system that monitors remote environmental data and supports human perception and cognition through its capabilities including situation assessment, and alert generation [11]. Sycara et al. describe an agent-based architecture (“Retsina”) for decision support that has three layers of agents to capture data and perform different information tasks, e.g. security analysis [54, 55]. Seligman et al. describe a DSS with decision centric information monitoring, where the data that could influence decisions are closely monitored for changes. Hess et al. describe a system where agents monitor supplier prices, and if they detect a change, they invoke an optimization model to generate new solutions [20]. Kilvijarvi and Tuominen describe a DSS that also supports implementation and control phases [26].

Focusing on intangible investments problem, the DSS helps to coordinate and translate goals into plans, and supports users in implementing plans, distribution of plans through network, and preparing necessary documentation. Thus the DSS supports implementation, rather than automating it. Mangina et al. describe a multi-agent system for decision support through data interpretation. The system is used for condition monitoring on a gas turbine [36]. Guena and Ossowski propose architecture for distributed DSS and discuss three cases, including environmental monitoring, power lines management, and traffic management [16]. In these distributed systems the agents collected sensor data, generated alerts and recommendations to the operator. (Some of these systems may have had effectors as well, but it was not clear from the discussion).

The situated DSSs with advanced capabilities are probably the most interesting type of those discussed here. For example, a service at http://www2.activebuyersguide.mysimon.com/ lets the buyer choose the product category, identify important features, perform tradeoffs, and offer a range of products the website thinks are attractive to the buyer. When the buyer chooses one, he or she can do comparison shopping, and later go to the seller's site to complete the transaction. Bui and Lee describe architecture of a geographically distributed agent-based DSS, where agents monitor the environment, assess situations, and implement actions in a semi-autonomous manner [6]. The system is described in the crisis management context. Collins et al. describes a semi-autonomous system for awarding contracts [9]. In this agent-based system, an agent composes RFQs, and evaluates bids. The user can interfere and override the agent's recommendation.

6. Decision Station: An Illustrative Example

6.1 Architecture for an investment decision station

We present here an example of an agent-based investment Decision Station. This example can be viewed as an extension of the developed investment DSS prototype discussed in [59]. The problem is in determining the portfolio of securities, monitoring its performance, and making modifications to the portfolio if necessary. The system architecture is shown in Figure 4.

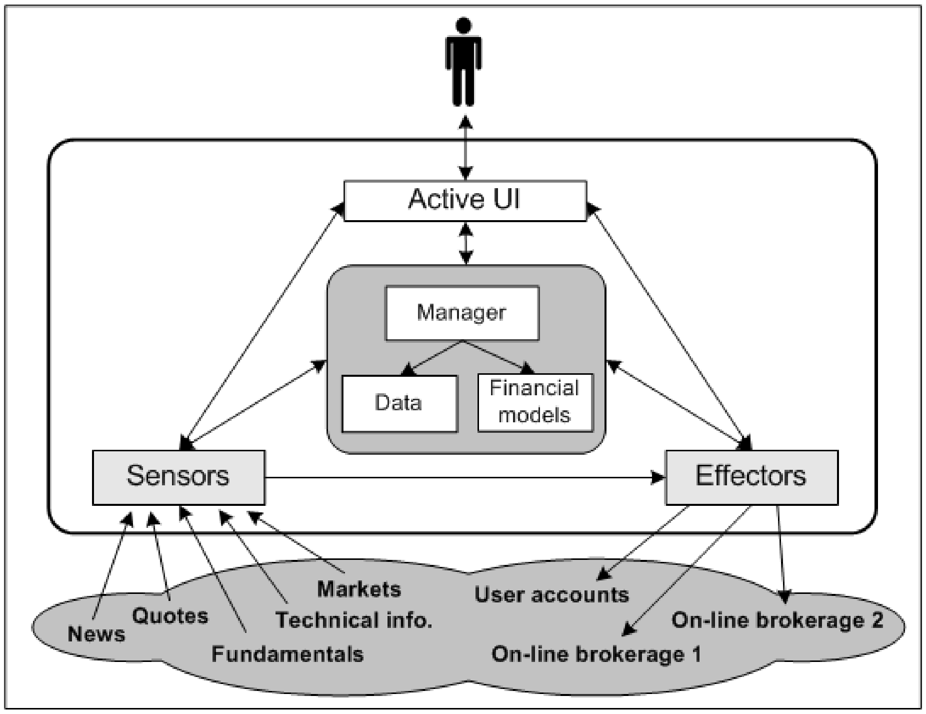


Figure 4. Decision Station for investment management

The sensors incorporate multiple agents that collect information from different sources. These include financial markets, historical information, analysts' opinions, news articles, and other relevant sources. The sensors monitoring the markets collect information about overall market and specific industry performance indicators (S&P 500, DJIA, etc.), performance of individual securities from the user portfolio, and the other securities on the "watch list". They can easily be configured to add new securities and alarms. They can do some basic transformations, e.g. calculating moving averages and other technical indicators. The sensors attached to the historical data retrieve historical security performance upon demand and calculate some basic indicators (e.g. historical return and standard deviation for a security). The sensors for news articles deliver the financial, economic, and political news that may have an impact on a market to the user in a timely fashion. Active sensors will be able to perform advanced tasks, e.g. based on the market volatility they could decide to monitor market more closely (planning capability) and adjust the levels at which the alert signals are generated (adaptive capability).

The effectors are the means of executing the user's investment decisions. These can be linked to various online brokerage firms as alternative outlets for the ordering. The choice of the firm can be made interactively with the user on the basis of fees charged, reputation of the firm, past experiences, and other factors. The effectors support different types of order and can monitor execution of an order to see whether it had actually gone through or not. Active effectors can take charge of re-evaluating and re-submitting an order if necessary. For example, in limit orders, it is possible that the order will never be executed, because the price surpassed the limit. The effectors can detect this situation, and re-adjust the order, if necessary, within certain limits. Table 3 describes the key capabilities of sensors and effectors embedded in the investment DS in more detail.

Table 3. Capabilities of sensors and effectors in an Investment Decision Station

Sensors		**Effectors**	
Capabilities	**Key functions**	**Capabilities**	**Key functions**
Connecting	Accessing stock quotes, market indicators (DJIA, S&P 500), news articles, firms' financial data, historical data.	Connecting	Placing buy/sell orders, transferring funds between accounts
Transforming	Calculating moving averages, portfolio performances, speed of change in stock prices, market indices, reconciling conflicting data, extracting keywords from news articles.	Transforming	Calculating total amounts to be paid, placing special orders using pre-specified rules.
Alerting	Signaling a sharp change in stock prices, market conditions, notifying the user about key variables (price, P/E ratio) reaching pre-specified targets, signaling breaking news.	Querying	Querying sensors about current prices to execute special orders, requesting additional information on order or seeking confirmation of decision from the user.
Adapting	Finding new sources of financial information, adjusting the thresholds for alert generation (e.g. in the case of sharp changes), assessing sources' credibility and reliability.	Adapting	Adjusting the rules for placing special orders, adjusting planning capabilities.
Planning	Deciding how frequently to read the stock, firm and market data, when to search for new sources, when to adjust alert thresholds.	Planning	Deciding when to query the sensors, when to execute orders.

The DSS kernel incorporates the financial models for estimating portfolio risk and return, knowledge and formulas for conducting fundamental and technical analysis, and others. The manager decides when to update the local information, keeps track of performance of the models, translates user decisions into buy/sell signals for the effectors and may even authorize minor buying/selling decisions without user involvement within specified limits. The active interface adapts to the user preferences using direct and indirect input from the user. It displays the stock performance indicators and news articles that fit the user profile and interests.

6.2 Sample scenario and a prototype

In order to provide a more concrete example of the investment DS we present a sample scenario and a prototype, and provide examples of sensor and effector capabilities outlined earlier in the context of the scenario. A UML activity diagram summarizing the working of a DS is presented in Figure 5.

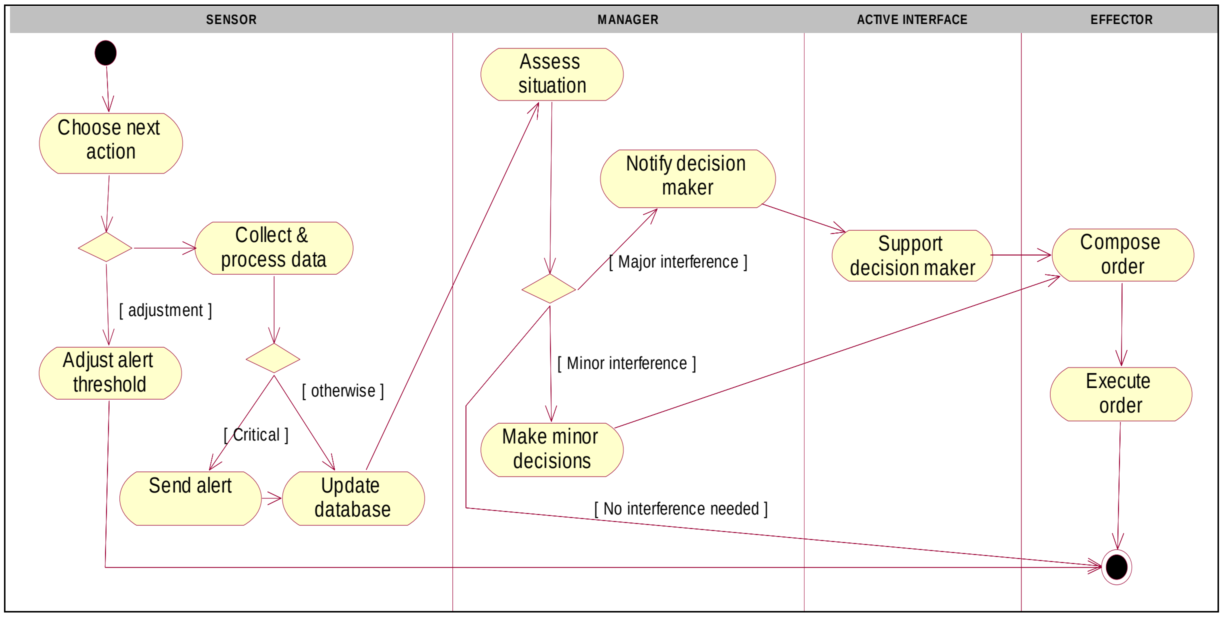


Figure 5. Investment DS activity diagram

The diagram outlines the operation of a DS for portfolio management. The scenario presented in Figure 5 begins with activities pertaining to collection of information about the portfolio and market performance. Then, the portfolio status is assessed in order to determine a need for an intervention. The intervention decision is either made automatically or in interaction with the user. If an intervention is made, the changes are implemented using the effector capabilities of composing and executing an order defined in the intervention. The operation of different DS components is detailed below.

The sensor connects to electronic information sources and imports the relevant data about the stock performance, market, and economy according to pre-defined rules. The sensor also performs some calculations including portfolio performance evaluation and changes in market indicators. Periodically, the sensor will adjust its parameters, most notably the threshold for alert generation. The capabilities of the sensor defined in (3) are specified here as follows:

plan = {“choose next action”};
conn = {“collect relevant data (SQL querying, XML, HTML parsing, etc.)”};

trans = {“process data (e.g. calculate return on portfolio)”};
alert = {“send notification of the sharp change (via e-mail)”};
adapt = {“adjust alert generation threshold”}.

For example, the *alert* capability is realized as:

$$alert = \begin{cases} \text{"}Sharp\ Change\text{"} & if\ \ \Delta Index > \theta \\ \varnothing & otherwise \end{cases},$$

where $\Delta Index$ is the change in a market index, such as DJIA, and θ – a pre-defined threshold value.

The *adapt* capability affects the above alerting capability by adjusting the threshold according to some function of the variance of the market index:

$$\theta_t = \theta_{t-1} + adj(\sigma^2(Index)),$$

where *adj*(.) is a function that calculates the magnitude of adjustment.

The sensors also update the database of the DS.

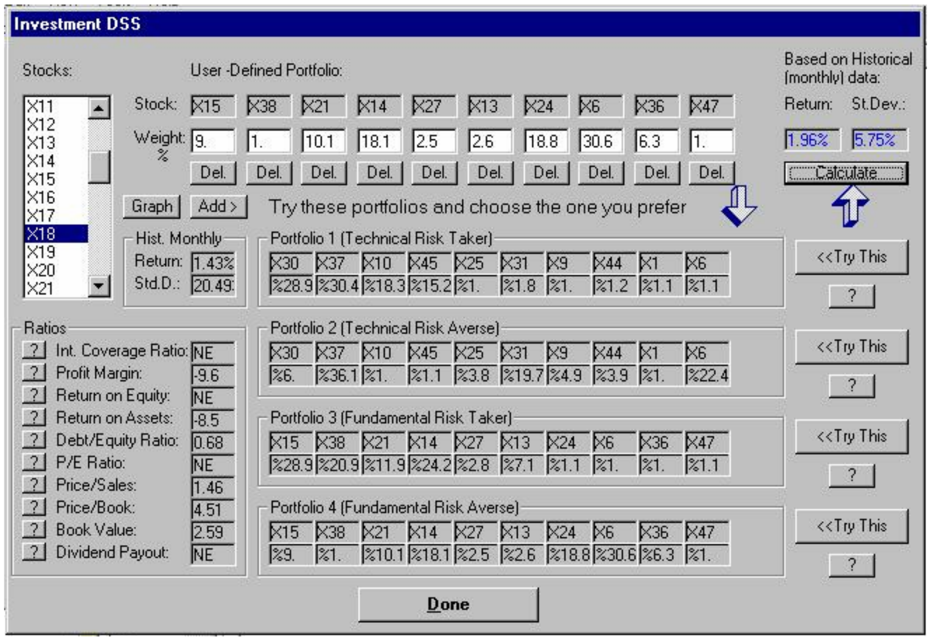
The DS manager (see Figure 4) performs a more detailed assessment of situation. A sensor’s alert prompts the DS manager to reassess the portfolio of securities using available financial models (e.g. fundamental and technical models). The manager’s authority to make modifications to the portfolio is limited. If an interference that exceeds the manager’s authority is required, the user is notified and an attempt to initiate the decision support session is made. The active interface supports the user’s decision making process, employing alternative portfolio generation and critiquing [60]. The resulting portfolio is passed to the effector.

The effector determines changes that have to be made to the current portfolio and fills out the necessary order forms electronically. The set of capabilities described earlier as *C* = {*plan*, *conn*, *trans*, *query*, *adapt*} in this example includes:

plan = {“choose time for submitting order”, “choose the time to resubmit order”};
conn = {“fill out order form”, “execute order”};
trans = {“calculate all data needed to submit an order”};
alert = {“send notification if the order couldn’t be submitted” (via e-mail)”}.

For example, having been given an order to purchase shares of a certain company for a specified amount the effector can use the *trans* capability to calculate the number of shares and fill out the order form: *number_of_shares = amount/price*. Subsequently, the effector executes the changes (submits the forms) and monitors their implementation.

In the DS prototype—see Figure 6 for a screenshot—the system’s agents generate four candidate portfolios proposed by agents called: “risky fundamental,” “risky technical,” “non-risky fundamental” and “non-risky technical.” The critiquing agents that are also part of active interfaces generate critiques of the analyzed portfolios based on user profile.



Stocks: X11 X12 X13 X14 X15 X16 X17 X18 X19 X20 X21

User -Defined Portfolio:

Stock:	X15	X38	X21	X14	X27	X13	X24	X6	X36	X47
Weight: %	9.	1.	10.1	18.1	2.5	2.6	18.8	30.6	6.3	1.

Based on Historical (monthly) data:

Return:	St.Dev.:
1.96%	5.75%

Try these portfolios and choose the one you prefer

Hist. Monthly

Return:	1.43%
Std.D.:	20.49

Ratios

Int. Coverage Ratio:	NE
Profit Margin:	-9.6
Return on Equity:	NE
Return on Assets:	-8.5
Debt/Equity Ratio:	0.68
P/E Ratio:	NE
Price/Sales:	1.46
Price/Book:	4.51
Book Value:	2.59
Dividend Payout:	NE

Portfolio 1 (Technical Risk Taker)

X30	X37	X10	X45	X25	X31	X9	X44	X1	X6
%28.9	%30.4	%18.3	%15.2	%1.	%1.8	%1.	%1.2	%1.1	%1.1

Portfolio 2 (Technical Risk Averse)

X30	X37	X10	X45	X25	X31	X9	X44	X1	X6
%6.	%36.1	%1.	%1.1	%3.8	%19.7	%4.9	%3.9	%1.	%22.4

Portfolio 3 (Fundamental Risk Taker)

X15	X38	X21	X14	X27	X13	X24	X6	X36	X47
%28.9	%20.9	%11.9	%24.2	%2.8	%7.1	%1.1	%1.	%1.	%1.1

Portfolio 4 (Fundamental Risk Averse)

X15	X38	X21	X14	X27	X13	X24	X6	X36	X47
%9.	%1.	%10.1	%18.1	%2.5	%2.6	%18.8	%30.6	%6.3	%1.

Figure 6. Screenshot of the prototype

The DS prototype uses data related to the software industry stocks (coded as X1 through X50) downloaded from the sources on the web to come up with recommendations. The data is processed using "transformation" capability to be used by the DSS models and to inform the decision making process. More in-depth elaboration of the prototype is given elsewhere [59].

The described decision station will inform the investor about the current situation, support his/her decision process, execute and monitor execution of the orders thus becoming an active situated system.

7. Discussion and conclusions

We have argued for a DSS that is situated in its environment, i.e. has the capabilities of directly sensing and acting upon it. The generic design for such a system, which we call a decision station, involves the use of sensing and effecting components as well as active user interfaces. One major concern that we haven't discussed so far is that of complexity of such a system. Complexity is discussed here from three different perspectives: complexity of using the system, computational complexity, and development complexity.

The complexity of use is concerned with both the system's ease of use and its effectiveness in the decision making process. This ultimately depends on the proper design of the active interface. One approach to effectively designing such interfaces has been elaborated in [60]. The idea is to facilitate effective decision making by using DSS–user intermediaries. Although we haven't directly measured the usefulness of the prototype mentioned above, the related measure of satisfaction elicited from the student subjects that used the system showed the value of 4.25 on a Likert scale from 1 to 7. While clearly more thorough and formal studies are needed, this measure provides some basis for our belief that the users will be able to make use of the system with adequately designed active interface.

The issue of computational complexity (and computational feasibility) arises because in the proposed architecture new components are added on top of the traditional DSS. Naturally, this would lead to the increased demand for the computational power. It is difficult to provide an accurate assessment of the computational complexity involved for all possible problem domains, since we are not proposing an algorithm, but a generic architecture. We note, however, that the use of agent technology as the basis for the DS implies the consideration of the agents' computational complexity. This issue is addressed in [30, p. 242], where it is shown that "the complexity of autonomy-oriented computation (agent-environment interactions) is adapted to the complexity of tasks (agent environments)." The tasks that an agent performs depend on its capabilities. The common capabilities that we have outlined in this manuscript include: connecting, transforming, planning, and adapting. In general, it would be reasonable to assume that the capabilities listed above are ordered by the increasing degree of computational complexity (assuming that the planning capability could also be adapted). Therefore, depending on the type of the problem domain (environment) the designer would include different types of capabilities in the system; the more active capabilities would require higher level of complexity. For a more formal treatment of the computational complexity of autonomous agents the reader is referred to [30].

Since the basic capability enabling the situatedness of DSS is that of connecting, we will briefly discuss it here. Ideally, both sensors and effectors have access to the databases/files in order to learn about the state of environment and to communicate decisions. Then connecting can be reduced to executing SQL statements. If this is not possible, ftp, telnet or WWW could be used. The data can be located in HTML documents and HTML parsing may be required in order to access it. Many tools exist to accomplish this task, e.g. the Network Query Language (http://www.nqltech.com/nql.asp) provides several flexible ways that enable agents to issue web queries. However, HTML parsing has obvious disadvantages, the major one being possible redesign of the webpages. While having maintenance personnel detect such changes and rewrite the connecting code may be an option (assuming that the webpage redesign is not a very frequent activity) the current trends towards adoption of XML may alleviate this problem considerably.

Finally, the issue of the complexity (feasibility) of development concerns the question of ease of development of DSs. Clearly, developing a situated DSS requires considerably more effort than developing a traditional DSS. However, the recent trend towards component-oriented software development is encouraging, since it enables and simplifies the development of more complex software. Moreover, the increased shift towards distributed architectures would enable use of distributed components. In this regard it is useful to mention the work reported in [28] on DSS intermediaries that suggests using CORBA in order to assemble DSS components.

In our opinion, revival of interest in DSS research depends on new frameworks and architectures that would extend the cognitive capacities of DSS to meet the complexities encountered by the decision makers. We hope that our work points in this direction. The work does not produce detailed specifications or comprehensive guidance on how to build a situated DSS. Instead, our objective is to provide a basis for DS, description of its components, and specification of their capabilities and functions. Much work is required. For example, incorporation of the implementation phase in the DS needs proper analysis and design of system's effectory capabilities with the utilization of findings from the management and problem solving literature. Furthermore, future work should be done on the nature of interaction between the components of a decision station and the kernel. Development of agent-based architectures for decision stations is another possibility to explore. Development of research prototypes and their evaluation would help to test the benefits and viability of the new concept.

An important research direction that could potentially arise from the outlined concept is its application to web-based customer decision support for e-commerce applications. In a recent paper

Silverman et al. [49] have emphasized the critical importance of providing adequate DSSs capabilities for shopping decisions. The paper distinguishes the three levels of such DSSs, including access-focused (basic search, browsing, etc.), transaction-focused (shopping support, guided choices, etc.) and relationship-focused levels. The latter one represents the highest level of support and expands the temporal and scope dimensions. This is an indication for a situated DSSs being a promising candidate for the conceptual foundation of providing such decision support in e-commerce applications.

References

[1] A. A. Angehrn, "Computers that Criticize You: Stimulus-Based Decision Support Systems," *Interfaces*, vol. 23, pp. 3-16, 1993.

[2] R. Balasubramaniam and M. Kannan, "Integrating Group Decision and Negotiation Support Systems with Work Processes," presented at Proc. of the 34th Hawaii International Conference on System Sciences, Hawaii, 2001.

[3] M. Bauer and D. Dengler, "TrIAs: Trainable Information Assistants for Cooperative Problem Solving," presented at Third International Conference on Autonomous Agents, Seattle, Washington, 1999.

[4] C. H. P. Brookes, "A Framework for DSS Development," in *Decision Support and Executive Information Systems*, P. Gray, Ed. Englewood Cliffs, NJ: Prentice Hall, 1994, pp. 27-44.

[5] J. Budzik, S. Bradshaw, X. Fu, and K. J. Hammond, "Supporting on-line resource discovery in the context of ongoing tasks with proactive software assistants," *International Journal of Human-Computer Studies*, vol. 56, pp. 47-74, 2002.

[6] T. Bui and J. Lee, "An Agent-Based Framework for Building Decision Support Systems," *Decision Support Systems*, vol. 25, pp. 225-237, 1999.

[7] C. Carlsson and E. Turban, "DSS: Directions for the Next Decade," *Decision Support Systems*, vol. 33, pp. 105-110, 2002.

[8] E. Claver, R. Gonzales, and J. Llopis, "An Analysis of Research in Information Systems (1981-1997)," *Information & Management*, vol. 37, pp. 181-195, 2000.

[9] J. Collins, C. Bilot, and M. Gini, "Mixed-Initiative Decision Support in Agent-Based Automated Contracting," presented at Proc. Of Fourth International Conference on Autonomous Agents, Barcelona, Catalonia, 2000.

[10] D. G. Conway and G. J. Koehler, "Interface agents: Caveat mercator in electronic commerce," *Decision Support Systems [Decis Support Syst]*, vol. 27, pp. 355-366, 2000.

[11] S. Das and D. Grecu, "COGENT: Cognitive Agent to Amplify Human Perception and Cognition," presented at Proc. Of Fourth International Conference on Autonomous Agents, Barcelona, Catalonia, 2000.

[12] O. Etzioni and D. Weld, "A Softbot-Based Interface to the Internet," in *Readings in Agents*, M. N. Huhns and M. P. Singh, Eds., 1997, pp. 77-81.

[13] S. Franklin and A. Graesser, "Is it an Agent, or Just a Program?: A Taxonomy for Autonomous Agents," in *Intelligent Agents III: Agent Theories, Architectures, and Languages*, J. P. Muller, M. J. Wooldridge, and N. R. Jennings, Eds. Berlin: Springer Verlag, 1997, pp. 21-36.

[14] C. Gonzalez and G. M. Kasper, "Animation in User Interfaces Designed for Decision Support Systems," in *Emerging Information Technologies: Improving Decisions, Cooperation, and Infrastructure*, K. E. Kendall, Ed. Thousand Oaks: Sage Publications, 1999, pp. 45-74.

[15] V. Grover and M. K. Malhotra, "A Framework for Examining Interface between Operations and Information Systems," *Decision Sciences*, vol. 30, pp. 901-920, 1999.

[16] J. Guena and S. Ossowski, "Distributed Models for Decision Support," in *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, G. Weiss, Ed. Cambridge, MA: The MIT Press, 1999, pp. 459-504.

[17] R. Guttman, A. Moukas, and P. Maes, "Agent-mediated Electronic Commerce: A Survey.," *Knowledge Engineering Review*, vol. 13, 1998.

[18] S. Haeckel and R. Nolan, "Managing by Wire," *Harvard Business REview*, vol. Sept.-Oct., pp. 122-132, 1993.

[19] C. E. Heckman and J. O. Wobbrock, "Put Your Best Face Forward: Antropomorphic Agents, E-commerce Consumers, and the Law," presented at Proc. Of Fourth International Conference on Autonomous Agents, Barcelona, Catalonia, 2000.

[20] T. J. Hess, L. P. Rees, and T. R. Rakes, "Using Autonomous Software Agents to Create Next Generation of Decision Support Systems," *Decision Sciences*, vol. 31, pp. 1-31, 2000.

[21] A. Hinkkanen, R. Kalakota, P. Saengcharoenrat, and A. B. Whinston, "Distributed Decision Support Systems for Real-Time Suppply Chain Management Using Agent Technologies," in *Readings in Electronic Commerce*, R. Kalakota and A. Whinston, Eds. Reading, MA: Addison Wesley Longman, 1997, pp. 275-292.

[22] T. A. Horan, "The Paradox of Place," *Communications of ACM*, vol. 44, pp. 58-60, 2001.

[23] E. Horvitz, "Principles of Mixed-Initiative User Interfaces," presented at Human Factors in Computing Systems, CHI 99, 1999.

[24] M. T. Jelassi, K. Williams, and C. S. Fidler, "The Emerging Role of DSS: From Passive to Active," *Decision Support Systems*, vol. 3, pp. 299-307, 1987.

[25] N. R. Jennings, "On agent-based software engineering," *Artificial Intelligence*, vol. 117, pp. 277-296, 2000.

[26] H. Kilvijarvi and M. Tuominen, "Computer Based Intelligence, Design, Choice, Implementation, and Control of Intangible Investments Projects," presented at Proc. of the 32nd Hawaii International Conference on System Sciences, Hawaii, 1999.

[27] A. Kraft, S. Pitsch, and V. Michael, "Agent-driven Online Business in Virtual Communities," presented at Proceedings of the 33rd Hawaii International Conference on System Sciences, Hawaii, 2000.

[28] K. R. Lang and A. B. Whinston, "A design of a DSS intermediary for electronic markets," *Decision Support Systems*, vol. 25, pp. 193-214, 1999.

[29] R. R. Levary, "Computer-Integrated Manufacturing: A Complex Information System," in *Manufacturing Decision Support Systems*, *Engineering Series*, H. R. Parsaei, S. Kolli, and T. R. Hanley, Eds. New York: Chapman & Hall, 1997, pp. 281-291.

[30] J. Liu, *Autonomous agents and multi-agent systems: explorations in learning, self-organization and adaptive computation*. Singapore: World Scientific Printers, 2001.

[31] A. I. Lockamy and J. F. I. Cox, "Linking strategies to actions: integrated performance measurement systems for competitive advantage," in *Manufacturing Decision Support Systems*, *Engineering Series*, H. R. Parsaei, S. Kolli, and T. R. Hanley, Eds. New York: Chapman & Hall, 1997, pp. 41-54.

[32] C. Lueg and R. Pfeifer, "Cognition, Situatedness, and Situated Design," presented at Second International Conference on Cognitive Technology, Aizu, Japan, 1997.

[33] B. K. Lundegaard, "E-Commerce (A Special Report): Selling Strategies --- Changing Lanes: Toyota hopes to avoid the potholes that have plagued its competitors' online efforts," *Wall Street Journal*, 2001.

[34] P. Maes, "Modeling Adaptive Autonomous Agents," in *Artificial Life: An Overview*, C. G. Langton, Ed. Cambridge, MA: MIT Press, 1995, pp. 135-162.

[35] P. Maes, R. H. Guttman, and A. G. Moukas, "Agents that Buy and Sell," *Communications of the ACM*, vol. 42, pp. 81-87, 1999.

[36] E. E. Mangina, S. D. J. McArthur, and J. R. McDonald, "COMMAS (COndition Monitoring Multi-Agent System)," *Autonomous Agents and Multi-Agent Systems*, vol. 4, pp. 279-282, 2001.

[37] H. McBreen, P. Shade, M. Jack, and P. Wyard, "Experimental Assessment of the Effectiveness of Synthetic Personae for Multi-Modal E-Retail Applications," presented at Proc. Of Fourth International Conference on Autonomous Agents, Barcelona, Catalonia, 2000.

[38] J. McDonald and J. Tobin, "Customer Empowerment in the Digital Economy," in *Blueprint to the digital economy: creating wealth in the era of e-business*, A. Lowy and D. Ticoll, Eds. New York: McGraw-Hill, 1998, pp. 202-220.

[39] N. Negroponte, "Agents: From Direct Manipulation to Delegation," in *Software agents*, Jeffrey M. Bradshaw ed, 1997, pp. 57-66.

[40] M. E. Nissen, "Supply Chain Process and Agent Design for E-Commerce," presented at 33rd Hawaii International Conference on System Sciences, 2000.

[41] J. Nunes-Suarez, D. O'Sullivan, H. Brouchoud, P. Cros, C. Moore, and C. Byrne, "Experiences in the Use of FIPA Agent Technologies for the Development of a Personal Travel Application," presented at Proc. Of Fourth International Conference on Autonomous Agents, Barcelona, Catalonia, 2000.

[42] S. A. Raghavan, "JANUS: A Paradigm for Active Decision Support," *Decision Support Systems*, vol. 7, pp. 379-395, 1991.

[43] S. J. Rosenschein and L. P. Kaelbling, "A Situated View of Representation and Control," *Artificial Intelligence*, vol. 73, pp. 149--173, 1995.

[44] J. Roth, "The Network is the Business," in *Blueprint to the digital economy: creating wealth in the era of e-business*, A. Lowy and D. Ticoll, Eds. New York: McGraw-Hill, 1998, pp. 283-297.

[45] A. Scarl, C. Bauer, and M. Kaukal, "Commercial Scenarios of Digital Agent Deployment," *Journal of Electronic Commerce Research*, vol. 1, 2000.

[46] L. Seligman, P. Lehner, K. Smith, C. Elsaesser, and D. Mattox, "Decision-centric information monitoring," *Journal of Intelligent Information Systems [J Intell Inform Syst]*, vol. 14, pp. 29-50, 2000.

[47] M. Shaw, J., D. Gardner, M., and H. Thomas, "Research Opportunities in Electronic Commerce," *Decision Support Systems*, vol. 21, pp. 149-156, 1997.

[48] J. P. Shim, M. Warkentin, J. F. Courtney, D. J. Power, R. Sharda, and C. Carlsson, "Past, present, and future of decision support technology," *Decision Support Systems*, vol. 33, pp. 111-126, 2002.

[49] B. G. Silverman, M. Bachann, and K. Al-Akharas, "Implications of buyer decision theory for design of e-commerce websites," *International Journal of Human-Computer Studies*, vol. 55, pp. 815-844, 2001.

[50] R. H. J. Sprague and E. D. Carlson, *Building Effective Decision Support Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1982.

[51] E. A. Stohr and S. Viswanathan, "Recommendation Systems: Decision Support for the Information Economy," in *Emerging Information Technologies*, K. E. Kendall, Ed.: SAGE Publications, 1999, pp. 21-44.

[52] N. A. Streitz, J. Geibler, T. Holmer, S. Konomi, C. Muller-Tomfelde, W. Reischl, P. Rexroth, P. Seitz, and R. Steinmetz, "i-LAND: An Interactive Landscape for CReativity and Innovation," presented at Human Factors in Computing Systems, CHI 99, 1999.

[53] L. Suchman, *Plans and Situated Actions: The Problem of Human-Machine Communication*. Cambridge, MA: Cambridge University Press, 1987.

[54] K. P. Sycara, K. Decker, and D. Zeng, "Intelligent Agents in Portfolio Management," in *Agent Technology: Foundations, Applications, and Markets*: Springer verlag, 1997, pp. 267-282.

[55] K. P. Sycara and D. Zeng, "Multi-Agent Integration of Information Gathering and Decision Support," presented at European Conference on Artificial Intelligence, 1996.

[56] D. Tennenhouse, "Proactive Computing," *Communications of the ACM*, vol. 43, pp. 43-50, 2000.

[57] K. Thorisson, R., "Real-Time Decision Making in Multimodal Face-to-Face Communication," presented at Second International Conference on Autonomous Agents, Minneapolos/St.Paul, MN, 1998.

[58] C.-C. Tseng and P. J. Gmytrasiewicz, "A real time decision support system for portfolio management," presented at Thirty-Fifth Annual Hawaii International Conference on System Sciences, Big Island, Hawaii, 2002.

[59] R. Vahidov, "A Framework for Multi-Agent DSS," in *Decision Sciences*. Atlanta, GA: Georgia State University, 2000.

[60] R. Vahidov, "Intermediating User - DSS Interaction with Autonomous Agents," presented at Decision Sciences Institute Annual Meeting, San Diego, CA, 2002.

[61] R. Vahidov and R. Elrod, "Incorporating Critique and Argumentation in DSS," *Decision Support Systems*, vol. 26, pp. 249-258, 1999.

[62] X. F. Wang, S. C. Zhang, P. K. Khosla, and H. Kiliccote, "Anytime Algorithm for Agent-mediated Merchant Information Gathering," presented at Proc. Of Fourth International Conference on Autonomous Agents, Barcelona, Catalonia, 2000.

[63] C. K. West, *Techno-Human Mesh: The Growing Power of Information Technologies*. Westport, Connecticut: Quorum Books, 2001.

[64] G. Zacharia, A. Moukas, and P. Maes, "Collaborative reputation mechanisms for electronic marketplaces," *Decision Support Systems*, vol. 29, pp. 371-388, 2000.