Bulletin of Environment, Pharmacology and Life Sciences
Bull. Env. Pharmacol. Life Sci., Vol 3 (Spl issue II) 2014: 16-28

Online ISSN 2277-1808
Journal's URL:http://www.bepls.com
CODEN: BEPLAD
Global Impact Factor 0.533
Universal Impact Factor 0.9804

Evolution and Development of decision support systems

Mohsen Ghaffarzadeh,Mehran Behartash,Mehrdad Azimi,Sanam Eyvazzade
Email: ghaffarzadeh.1361@gmail.com

ABSTRACT
Decision support systems need to "evolve" over time for many reasons, including changing user needs, technologies and problem understanding. This paper investigates what constitutes DSS evolution. The paper takes the view that DSS evolution means that changes occur in all aspects of those systems, including hardware, databases, user interface, applications and knowledge. This paper summarizes and extends some of the literature on evolution. In addition, the paper also summarizes some approaches designed to help manage DSS evolution, including both prediction and facilitation of evolution.
Keywords: *Decision Support System, Evolution*

INTRODUCTION

Apparently, Courbon, Grajew and Tolovi (1978) were the first to use the notion of "evolution" in decision support systems (DSS). Soon after that, Keen (1980) elaborated on key aspects related to evolution in DSS. That research was most concerned with the notion that DSS evolve over time: the development methodology of DSS is an evolutionary one. In a closely related set of developments, Lehman et al. (1983) appear to have been the first to use the term "evolution" in conjunction with generic computer software. In particular, Lehman (1998) labeled software development and maintenance, as software "evolution." He described software change and enhancement as "unending," suggesting that evolution also is unending.

Scope

DSS as a bundle of hardware, data and knowledge, user interface and software application -change and evolve over time. As a result, the purpose of this paper is to investigate the notion of DSS evolution and DSS characteristics and component evolution. Previous literature has primarily been concerned about the notions that DSS evolve and that methodologies of DSS development consider that evolution. In addition, there has been some concern as to why DSS evolve. However, there has been limited research according to how DSS actually change and evolve over time. Accordingly, we review the previous literatureon DSS evolution, according to its individual components and provide specificity for DSS evolution through those components changing over time. In addition, we extend the notion of evolution to a more proactive perspective, aimed at management of evolution, where we try to predict and facilitate evolution as part of DSS management, rather than just passive evolution.

The scope of the paper is to investigate evolution of DSS, in general, and in its components, specifically. For some DSS components there is an extensive evolution literature, for example, databaseschema. However, for others there is a more limited literature, e.g., evolution of different knowledge representations. Because of the broad reaching and extensive nature of this topic, we provide additional discussion on knowledge evolution, including knowledge artifacts, such as taxonomies.

This Paper

This chapter proceeds as follows. Section 2 discusses key issues associated with evolution and how it relates to DSS, including such issues as what is DSS evolution, what are some sources of evolution and the extent to which backward compatibility is an important issue in DSS evolution. Section 3 provides a review of some of the previous literature that deals with DSS evolution, analyzing each major component of a DSS for previous discussions on evolution. Section 4 focuses on knowledge evolution, while section 5 drills down on how to manage that knowledge evolution by facilitating and predicting knowledge evolution. Section 6 provides a brief summary of the paper and the contributions.

DSS Evolution

The purpose of this section is to lay out key issues in DSS evolution, including defining what we mean when we talk about DSS evolution.

What is Evolution?

Before we talk about DSS evolution, what do we mean by "evolution?" Typically, definitions suggest a gradual change in whatever is evolving, generally as it moves to a different state. For example, definitions include,

- "A process in which something passes by degrees to a different stage (especially a more advanced or mature stage)" or
- "A gradual process in which something changes into a different and usually more complex or better form." (http://www.thefreedictionary.com/evolution)

What is DSS Evolution?

In terms of DSS, what does this mean? DSS evolution relates to

- changing DSS features or components over time,
- changing technology on which the system is used,
- getting more efficient algorithms over time,
- evolving knowledge in the system over time,
- changing users and user preferences over time.

If DSS evolve, that raises additional questions. For example,

- Is evolution a formal process or is evolution something that just happens?
- How can we tell if evolution has occurred?
- How do we measure the extent of evolution?
- Can we predict aspects of evolution?
- Can we facilitate evolution, perhaps increasing the speed of evolution?

Evolution vs. Revolution

In general, it will be assumed that evolution is different than a revolution. Revolution is a more dramatic change than the change associated with evolution. For example, revolution has been defined as "A sudden or momentous change in a situation" (http://www.thefreedictionary.com/revolution)

As an example of "revolution," in the 1980's there was a rapid growth of so called "expert systems" or "knowledge-based systems" or "rule-based systems." Although expert systems were oftentimes aimed at supporting decisions, rule-based knowledge historically was not generally viewed a part of what was included in decision support systems. Expert systems and their rule-bases provided an alternative way to capture decision supporting activity.

However, over time, the revolution of expert systems and other technologies, has become an evolution point for DSS, rather than a revolution point. Rule-based systems have become integrated with other technologies and problem solving approaches. A review of papers published in *Decision Support Systems* yields many papers that employ rule-based or knowledge-based approaches. Now DSS that employ rule-based technologies are often referred to as "Intelligent" DSS if they wish to distinguish them from other DSS or increasingly even just DSS because the intelligence is integrated into the system.

Why Evolution?

Why do DSS evolve? Although Lehman (1998) refers to "development and maintenance," that provides limited insight into why DSS evolve. The purpose of this section is to summarize some of those rationales, as to why DSS need to evolve.

There are a number of reasons for the evolution of a DSS over time. First, user preferences may change gradually over time. Accordingly, the DSS must respond to those changes by allowing the user to either make preference changes or by monitoring system use and facilitating those changes.

Second, specific user needs may change, and there can be a number of rationales driving that change. It is well-known that requirements for system have a tendency to change as the user(s) sees the system and better understands how it is going to work. As they better understand the system, users have a better understanding as to how it can meet their needs.

Third, the specific user may change. As users are promoted or assigned other job activities within an organization, the actual system users may change (Arnott 2004). Different users are likely to have different requirements.

Fourth, the set of users for which the system is designed may change. When so-called "executive information systems" (EIS) emerged, they were aimed at executives. However, this led to a setting where executives and non-executives had differential information and support. Further, those that worked for the executives were affected by the use of the systems. As a result, the non - executives wanted access to the system. In addition, by spreading the costs over a larger base of users, firms could drive down the per

user cost. Accordingly, it was not long before a larger base of users was granted EIS access. Different users have different capabilities. As the user set changed, the system capabilities needed to change.
Fifth, there can be errors in the system and those errors need to be fixed as part of a normal maintenance process. As errors are fixed the system will change, and the user's view of the system will change, likely precipitating additional evolution. This link is often overlooked in evolutionary analysis.
Sixth, available technology may change. Technology is constantly changing. Those changes can have a substantial impact on delivery of a DSS. For example, before the Internet and the web browser, developing a user interface was a large portion of any DSS task. Now, developers simply build the systems to employ a browser interface. As another example, Erwin and Snelling (2001) explore the evolution toward a grid computing environment. Although we are unaware of any grid computing DSS, it does illustrate the change in computing environments over time. Keen (1980) felt this was a key cause of evolution.
Seventh, understanding of the problem may change. A DSS can be built to solve one problem, but as the problem becomes better understood, the scope of the problem may change. For example, the problem may start out as a warehouse location problem, but turn into a broader problem, more oriented at the entire supply chain. This has been referred to as a "cognitive" change (e.g., Arnott 2004).
Eighth, as noted by Keen (1980) the problem being solved tends to move from simple to complex. As those complexities are introduced the system will need to change to address those complexities as part of the application.
Ninth, decisions may evolve from being decisions made by an isolated individual to a group decision. Shakun (1991) suggested that the same evolutionary methodology suggested by Keen (1980) be used to facilitate development of group DSS.
Tenth, as noted by Arnott (2004) internal organization structure may change. For example, there may be downsizing, outsourcing, division restructuring, and other changes.
Eleventh, ideally the DSS is designed in concert with an organization's strategy in order to create value for the organization. As a result, if that strategy changes, then the DSS also will need to change. As organizations and strategies change and evolve, supporting systems will need to do the same.

Twelfth, the "world" in which the decision is being made may change. A new competitor or a change in resource availability can have major affects on the DSS model and the decisions that need to be made. Angehrn and Jelassi (1994) indicated that environmental changes were a major cause of the need for DSS to evolve. Arnott (2004) noted that external events such as changes to industry structure and government regulations also can lead to the need for evolution.

Accordingly, there are a large number of reasons as to how and why a DSS is likely to change and evolve. Evolution is not limited to one such factor, but may include multiple factors. Further, as with much evolution, time plays an important role, allowing evolution to take place in a dynamic environment.

2.5 Is Evolution "Backward Compatible"?

In general, evolution does not mean that what was previous processed historically will continue to be processed in the same manner, i.e., backward compatibility. DSS artifacts, such as taxonomies and knowledge bases, in general are not backward compatible. They evolve to solve the problem as the user, problem and capabilities change. Further, as noted above, users may change, so that even users are not backward compatible. As a result, typically, a problem of interest at time t, is not necessarily of interest at time t+1, and not necessarily solvable with the same DSS, to derive the same solution.

2.6Predicting vs. Facilitating Evolution

There are at least two distinct features associated with DSS evolution, beyond knowing that it will occur: prediction of what the system will evolve to and facilitation of that evolution. Predicting evolution has to do with trying to understand what future state or states the current system will evolve towards. Facilitation has to do with trying to understand how to best push or change the system to a new future state.As a result, facilitation may include identification of a particular future state for the system to evolve to.In any case, both and prediction and facilitation are critical to the overall management of the DSS. Both play an important role in managing change in the DSS and managing the DSS. These topics are discussed further later in the paper.

2.7 Is Evolution a Formal Process?

In machines we can formally plan for evolution of the overall system and particular system aspects. As a result, in that setting, evolution is a formal process, typically with particular observable end states or

intermediary states. Generally, the formal aspect of evolution is associated with the formal evolution of individual components, e.g., database evolution, aimed at planned end states.

However, evolution is not necessarily a "formal" process. In classic DSS evolution (e.g., Keen 1980) there is an "air" of informal evolution. In that setting evolution can result from informal or non planned changes that occur over time. In this later approach, evolution can take on emergent properties.

Emergent Properties

Emergent system properties are those properties that result from using the system and from system components interacting with each other. Individual components would not exhibit the behavior, instead it is with the interaction of the components that the behavior arises. Because of the interactions, behavior evolves. In general, emergent properties are not always predictable. Instead, they are a function of the interaction and evolution of the components. Emergent properties are likely to be less predictable, and possibly less visible, than other system properties.

How can we tell if Evolution has occurred?

If evolution is an informal process, determining the extent of evolution may become an important issue. Evolution can occur in the systems or its users and the processes that they use. In many ways, the key question is simply, "Is the system different than it used to be?" However, the extent and type of evolution will vary by DSS component, whether it happens to be knowledge used in the system, solution approaches used by the system or even technology. Further, emergent properties may be more difficult to find and measure and predict, a priori. So-called "network effects" are a classic emergent property of integrating multiple computers on the same network.

The extent, to which a DSS has changed, might be assessed by using a standard set of inputs. In a deterministic system, associated with the standard inputs would be a related set of standard outputs. Unfortunately, being able to use this approach assumes that there is backward compatibility in the evolution of the DSS.

As a result, determination of the extent of evolution typically will depend on change in DSS artifacts. Those artifacts can either be an explicit part of the system or ancillary to the system. Artifacts embedded in the DSS might include database schema or taxonomies. Comparing artifacts at particular points in time will allow us to gauge the extent of evolution. Since the analysis is based on particular artifacts, much of the analysis is likely to be artifact specific.Ramesh and Sengupta (1995)discussed one such ancillary artifact. They suggest using multimedia to capture historical information about a system's evolution. Using multimedia a history of system usage could be used to provide snapshots of evolution. For example, a video of a design session could be used to understand how and why a system has evolved.

Scope of "Evolution"

The term "evolution" has received many different uses over time in the decision support literature. In particular, when it comes to the term "evolution," DSS have found many uses that are not directly related to "DSS evolution." Accordingly, an important issue is what "evolution" is not included in this paper. The scope of the paper does NOT include

- "industry evolution" (Schuler 2001), where DSS is used to analyze the evolution of a particular industry.
- "managing software evolution" (Berzins 1998), where a DSS is used to provide support for software prototypes.
- "product evolution" (e.g., Tiwana and Ramesh 2001 or Kojo et al. 2003), where the DSS is designed to facilitate the evolution of a product.
- specific "problem evolution" (e.g., Balbo and Pinson 2005) where knowledge about how a problem can evolve and how to respond to it is addressed as a DSS. However, the problem that a DSS solves may well evolve and change as needs change.
- "evolution of strategic customer relationship management practices" (Pan et al. 2006). However, ultimately, the DSS may change to reflect changes in best practices that occur over time.
- using DSS as a basis and tool to predict the evolution of other topics, e.g., predicting new patients (Riano and Prado 2001). Instead the focus is on the DSS and using tools (which could include a DSS) to predict and facilitate change in the DSS, in general.

Accordingly, the focus of the paper is on DSS evolution, and the evolution of what are historically taken to be some of its components.

Previous Research: Evolution of DSS

There has been a limited amount of research on "evolution" of DSS, mostly aimed at individual characteristics or components of DSS. Those components include the technology on which the DSS is based, database, user interface, application and knowledge built into the system. Further, that research

has really focused on changing systems but not understanding the process or predicting or facilitating DSS evolution. The research is summarized in the following table:

Table 1
Summary of Research

DSS Evolve	Courbon et al. (1978) and Keen (1980)
DSS Development Methodology	Keen (1980)
Technologies Evolution	Gibson and Nolan (1974)
Database Evolution	Banerjee (1987), Chen et al. (1995)
User Interface Evolution	Integrated into Operating Systems
Application	Rao and Turoff (2000)
Knowledge	
Processes	Zhuge (2005)
Taxonomy Knowledge	O'Leary (2004, 2007)
Ontology Knowledge	Haase and Stojanovic (2005)
Association Rules	Golani and Etizion (1999)
Discovered Knowledge	Yoon and Kerschberg (1993)

DSS Development Methodology: An Historical Perspective

The fact that DSS evolve and including that evolution in the development methodology associated with DSS has been apparent from the beginning of DSS. Historically, Keen (1980) was most concerned with the notion that system actually evolved, and what seemed to cause that evolution. Evolution was built into the notion of how DSS developed over time. When Keen (1980) discussed the concept of evolution, he directly indicated that the final system can be developed only through a process of "learning and evolution" and that the DSS evolves in response to learning. Keen (1980) felt that evolution occurred only because of interaction between user and designer, learning, personalized use, or the evolution of new functions. He also indicated that the system evolves in response to evolving needs. Keen indicated that "program structures and programming methods" need to facilitate evolution. Keen indicated that evolution means adding "new commands" that probably would be translated as adding new capabilities, but did not provide further guidance as to how to actually evolve DSS. Keen did suggest that there is a need to study the evolution of data and data structures and that data-based DSS do not evolve as easily as model-based DSS. Keen also saw evolution as technology-based, since he noted that evolution can be blocked by the inability to obtain additional technology. Finally, he noted that complex systems evolve from simple systems.

As a result, historically, the focus of evolution has been on the fact that DSS evolve, and not on what particular aspects of DSS evolves or how evolution should be managed or predicted. A brief history of some DSS evolution issues is provided by Arnott (2004). For example, Keen and Gambino (1983) suggested that DSS adaptation occurs at the sub-task level. This view focuses on the decision problem rather than other factors. Stabell (1983) indicated that DSS evolution should occur with the tension between the descriptive and prescriptive views of the decision. However, this ignores evolution due to factors other than the model of the decision. Sage (1991) developed a seven stage requirements analysis approach for DSS, and indicated that requirements determination are likely to be a driver of evolution. Accordingly, this approach focuses on the role of requirements and a focus on the decision. Silver (1991) also focused on the importance of the decision in evolution, considering how DSS affect decision making processes, and the role of the DSS in guiding system use. Accordingly, Silver focused more on the user, rather than other aspects of the DSS.

DSS Technology Evolution

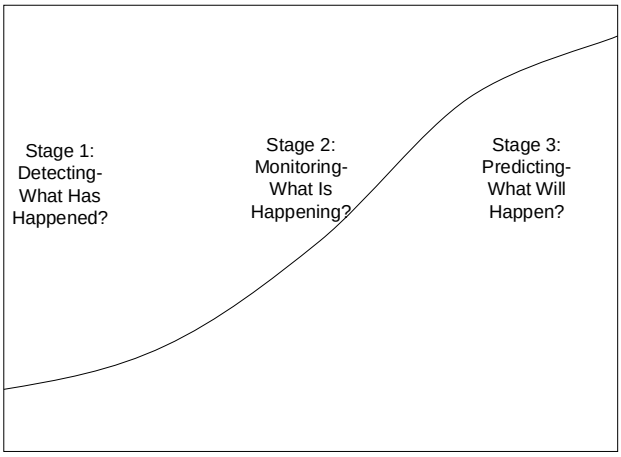
DSS technology also has undergone substantial evolution over the years and will continue to do so. This evolution has not gone unnoticed, although it may not have been referred to as evolution.

In particular, there has been some research focus on how IT systems have changed. In an era before package software, enterprise software and even before formal notions of DSS, Gibson and Nolan (1974) suggested that computer based applications follow a common growth cycle over time. Although it is arguable as to the specific applicability of the discussion in that paper, the overall notion was one (although the terminology was not used) where application type evolved over time, starting with cost cutting accounting applications, moving to functional applications and a focus on control, to database applications where the user could query the database.

This type of evolution could be expanded and generalized to DSS evolution over time. A number of models could be developed. For example, a focus on decisions is likely to first be concerned with what has happened and structuring the problem. This may take any of several forms, including a database query,

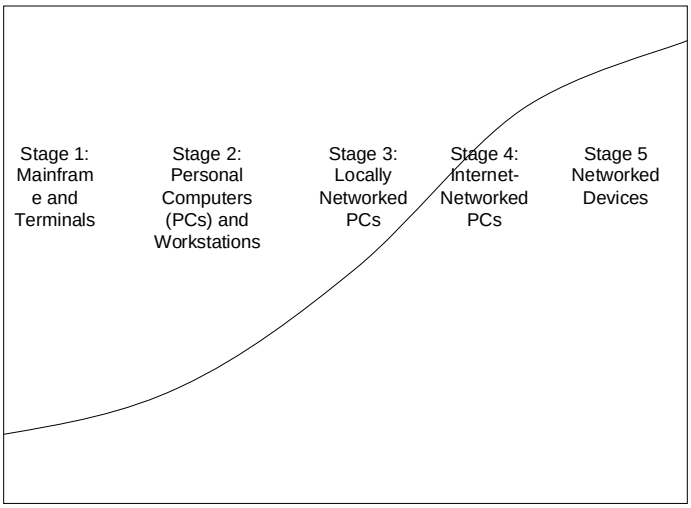
where the user tries to understand what the problem is and what is the source of the problem. As they begin to understand the nature of the problem, the system and what they do is likely to change. Once there is a strong concept as to what the problem is and what data is necessary to understand it, a second step is likely to focus on monitoring the particular problem area. Reports may be created as part of managing the problem area. Finally, in the third step, rather than just monitoring and reacting to problems, users are likely to want to anticipate problems, forecasting data to facilitate that prediction process. This approach is summarized in figure 1, as part of decision understanding.

Decision Understanding Stages



Further, the technology associated with DSS systems have gone from dumb terminals linked to main frames, to stand alone work stations and personal computers to locally networked computers to computers networked across the Internet. Figure 2 generalizes and extends the Nolan (1993) view of the three eras of IT organizational learning, by expanding the networked stage to local and Internet-networked PCs. As emerging technologies, such as grid computing (Erwin and Schelling 2001) increase in importance, they may be integrated into our view of DSS. Further, this view can be extended beyond PCs to alternative technologies, such as mobile computing devices, including mobile phones. Stage 6 is likely to be an environment of wearable and embedded computing.

Figure 2
DSS Hardware Evolution



Recently, McAfee (2006) continued the "stages" research investigating the emerging three worlds of information technology. The evolutionary flow associated with information technology moved from function to network to enterprise, with a corresponding focus on discrete tasks, tasks that interact and business process-based information technology, respectively.

DSS can be a part of any of those stages. Perhaps DSS fit best as functional information technology, because they often assist with the execution of particular tasks. DSS also may fit as network IT (information technology), particularly if multiple users have access to it, say over the Internet. For example, emerging technologies such as WIKIs may fit into this view of DSS. Enterprise IT is IT that specifies business processes. Emerging technologies such as "business process management," connect DSS and so-called enterprise IT.

The focus of this paper is on evolution, so a complete analysis of all of the technologies influencing DSS is outside of scope and probably impossible anyway. However, technologies that already have an impact on DSS are many and wide ranging, including mobile technologies such as Palm and telephones. In addition, other technologies, such as data warehouses and artificial intelligence also have had a substantial impact.

Database Evolution

One of the key components of a DSS is likely to be a database. In the same sense that a DSS evolves, a database needs to evolve. If decision making needs change then the data supporting those needs also is likely to change. Accordingly, DSS databases need to evolve (e.g., Banerjee et al. 1987). Two key concepts associated with database evolution are database schema and metadata.

There is a substantial literature on database schema evolution (e.g., Roddick 1995 and Liu et al. 1994) and in the context of enterprises (Chen et al. (1995). Loosely, schema evolution refers to the ability of a database schema to change without loosing information. Schema evolution has been necessary in those settings where there is a need to retain data under schema definitions that have been changed. Research has been done relating to both relational and object oriented data.

In general, schema evolution is guided by a database administrator. Further, schema evolutions ideally are closely symmetric to the previous schema so that data under the previous schema can be viewed under the current schema. In addition, evolution should be reversible so that erroneous changes can be reversed. However, database evolution does not imply a complete historical support for the particular schema.

Unlike database evolution, DSS evolution may not allow even partial support for previous capabilities. Evolution in DSS is much more similar to evolution in biology. For example, DSS evolution may push the DSS to supporting entirely new questions, at the expense of supporting previous questions. Further, evolution to new technologies may not be fully backward compatible.

Sen (2004) traces the history of metadata, as the evolution of the concept of metadata. However, only recently has there been limited research on evolution of metadata, per se, particularly the prediction and facilitation of metadata. One such case study (Loasby 2006) traces metadata evolution at the BBC. Much of the recent focus on metadata occurs because of the focus on so-called service oriented architectures (Gabriel 2005). Because DSS can be developed as stand alone applications, they may evolve their own metadata, which could cause substantial difficulties with integration with enterprise data interfaces or other data sources.

3.4 User Interface

DSS user interfaces have evolved over time. Early DSS spent substantial development time and resources on the user interface. However, general user interface work at Xerox Parc, translated into generally applicable user interfaces. For example, as the Windows operating system and software, has evolved from DOS (disk operating system), to Windows 3.1, to Windows 95, Windows 98, Windows 2000, and Windows XP or even Windows Vista, the user interface increasingly is built into the operating system capabilities, and basic software capabilities. As another example, Microsoft's Excel provides the ability to generate pie charts, bar charts and many other capabilities. Further, even enterprise software companies like SAP are integrating their software with Microsoft Office applications (Ferguson 2006) in order to make it easier for user and to more fully leverage user interface capabilities. Accordingly, fewer and fewer “special” user interface capabilities are needed for DSS developed for those environments.

As DSS migrate to mobile computing environments, user interfaces will continue to evolve. For example, mobile computing environments, such as phones or other devices provide smaller screens and have different keyboard and other human computer interfaces.

3.5 Application Evolution

DSS evolution includes the particular application and software for which the system is designed. One of the few and best examples of application evolution in the literature is Rao and Turoff’s (2000) analysis of the evolution of medical decision making DSS. An extensive analysis of the problem domain for which the

DSS was constructed, is presented. Through the analysis of a number of systems the authors trace the evolution of a number of key system medical support characteristics, including
Nature of support (clinical vs. diagnosis)
Information nature (detailed and complex vs. not detailed and not complex)
Information currency and stability (highly current vs. highly non current, and dynamic vs. static)
Reasoning (abstraction vs. causal)
Medical group interaction (high vs. low)
Nature of tools (qualitative vs. quantitative)
In addition, the systems are analyzed according to collaborating multiple decision making features and supporting group DSS feature. For example, respectively,

Temporal Representation	Customizable Temporal Markers
Prior Research/ Factual Reference	Supports Hypertext
Knowledge Base Interaction	Supports Rule-based Knowledge
Multiple criteria decision making (MCDM)	Supports MCDM tools
Individual decision making styles	Encourage individual participation (e.g., pen names, conference areas)

Knowledge Evolution

Increasingly, DSS have knowledge embedded within them and they use knowledge-based artifacts. As a result, as DSS evolve the underlying intelligence on which it is based and the knowledge artifacts also need to change. Accordingly, knowledge evolution is a critical part of DSS evolution. The remainder of the paper focuses on that knowledge evolution.

Decision support systems are likely to use many different kinds of knowledge, each with their own unique characteristics. Because there are many forms of knowledge representation, we can not discuss each of the different kinds of knowledge. However, we will discuss a few different types of knowledge to illustrate some of the evolutionary issues associated with knowledge evolution, including process knowledge, taxonomy knowledge, ontology knowledge, associative knowledge and general discovered knowledge.

Process Knowledge

There has been interest in the evolution of process knowledge. For example, Pan et al. (2006) discuss the evolution of customer relationship management process knowledge. Further, Zhuge (2005) discusses the evolution of the flow of knowledge through an enterprise.

Formalization of process knowledge in computer-based systems can take many forms. One example of a formalization of process information is that of RosettaNet and their "Partner Interface Processes."

> RosettaNet Partner Interface Processes (PIPs) are specialized system-to-system XML-based dialogs that define business processes between trading partners. Each PIP specification includes a business document with the vocabulary, and a business process with the choreography of the message dialog. (http://xml.coverpages.org/rosettaNet.html)
>
> PIPs are organized into seven Clusters, or groups of core business processes, that represent the backbone of the trading network. Each Cluster is broken down into Segments -- cross-enterprise processes involving more than one type of trading partner. Within each Segment are individual PIPs. (http://xml.coverpages.org/rosettaNet.html)

PIPs are formally categorized in a life cycle with a number of different potential states that enable us to formally ascertain where they are in their evolution:

- "On Hold"
- "In Production"
- "Waiting Validation "
- "In Validation"
- "Obsolete"
- "Versioned"

In addition, the PIPs are each attributed a version number , e.g., "V01.00.00" so that evolution and change of the knowledge over time can be captured and versions controlled. As corporate processes change over time, PIPs also can change. Further, PIPs can move from one state and back to another. A PIP that is versioned still can become obsolete and replaced.

Similar models of knowledge versions can be used with other types of knowledge. Still other forms of process knowledge might be captured, indexed and stored using taxonomies.

Taxonomy Knowledge

Another frequently used form of knowledge representation is the taxonomy. For example, the APQC (1996) and Arthur Andersen Taxonomy is used to capture, index and store knowledge about processes. Apparently based on the taxonomy framework, information about business process is categorized to a number of different "levels" within the taxonomy. Within each of the categories, knowledge such as case studies and key performance indicators are stored for users. O'Leary (2004 and 2007) documents empirically the evolution of the APQC taxonomy into different artifact versions available over time from different sources.

Ontology Knowledge

Recently, ontologies have gathered substantial attention, perhaps more than any other knowledge representation. Ontologies also provide a key basis on which to index, capture and search knowledge. The evolutionary change of knowledge has been categorized as an impediment to ontologies by O'Leary (1997). As a result, it is not surprising to argue that ontologies evolve over time. There is a growing literature concerned with evolving ontologies (e.g., Haase and Stojanovic 2005) and ensuring that those ontologies are consistent (e.g., Haase et al. 2005) even in environments where the ontology is shared (Xuan et al. 2006).

Although ontologies are closely related to database schema, Noy and and Klein (2004) argue that there are important differences. However, they use the previous research on schema evolution to provide insight into ontology evolution. Ultimately, the set of change operations that they develop is different than database schema evolution.

Association Rules and Evolution in Time

Association rules provide relationships between different objects in an available database, e.g., height and weight, or smoking and illnesses. Golani and Etizion (1999) and Koundourakis and Theodoulidis (2002) explore how such rules can evolve over time. For example, changes in tax law will lead to changes in the rules to evolve from the old rules to the new rules. Similarly, as individuals' age, rules will need to change to reflect the changes associated with getting older. In order to understand and capture that evolution requires periodically refitting rules to data gathered over time. Issues related to the association rules, such as the confidence level can be tracked as part of the evolution process of the association rules.

Discovered Knowledge

Since the early 1990's knowledge discovery has been an important research area. Discovered data is dependent on the particular data analyzed. Discovered knowledge changes over time, as more or different data is analyzed. As a result, discovered data is not necessarily backward compatible. Instead as the data on which the knowledge is based changes, the knowledge is also likely to change. This is not unexpected. As an example of research in this area, Yoon and Kerschberg (1993) present a framework for evolving knowledge as the data in the database evolves.

5. Predicting and Facilitating Knowledge Evolution

Given that evolution will occur, one concern of this paper is with trying to predict and facilitate that evolution in order to manage the DSS. The same issues of prediction and facilitating technology evolution occur with each aspect of DSS. For example, we would like to be able to predict and facilitate the technology evolution of DSS or the knowledge evolution. As a result, some of the same approaches used here can be applied to other components also.

Predicting Knowledge Evolution: An Empirical Approach

We see that knowledge and other DSS components evolve, which raises the question, "Can we predict evolution or the effects of evolution?" One of the primary assumptions in O'Leary (2004 and 2006) is that we can predict how taxonomy knowledge will evolve. As a result, O'Leary (2004 and 2006) focused on knowledge in the form of taxonomies and thus with taxonomy evolution. A number of different approaches were employed to try to empirically analyze taxonomy evolution change, with the goal of predicting the evolution.

O'Leary (2006) gathered various kinds of quantitative information about the taxonomy changes associated with different taxonomy artifacts. A number of empirical approaches were used to analyze that data. For example, entropy was used to measure complexity, and it was found that as the taxonomy evolved, it evolved to greater complexity and entropy. As a result, taxonomy changes would be expected to result in greater complexity. Accordingly, using various empirical relationships, future states of the taxonomy might be predicted from past relationships.

Predicting and Facilitating Knowledge Evolution Using Genetic Algorithms

Rather than taking an indirect approach of predicting evolution, perhaps it is possible to directlypredict and facilitate evolution. Genetic algorithms (GA's) mimic the evolutionary approach in nature of natural selection. As a result, conceptually we could use GAs to mimic DSS evolution across its different components or even treating the components as a whole for evolution.

GA's keep a number of solutions, typically as strings, that can be used to create new solutions. Based on those strings, GA's use a number of operators as a basis of simulating evolutionary factors, including mutation and crossover (also called recombination). The resulting children then can become a part of the population used to create new solutions, that evolve toward different configurations.
How might this be done with respect to knowledge embedded in a DSS? One approach is to focus on a particular form of knowledge representation. O'Leary (2007) found that entropy captured taxonomy change information. As a result, entropy might be used to measure the quality or fitness of different proposed solutions. Further, O'Leary (2007) also found that taxonomy categories experienced a number of different operations over time as taxonomies evolved. Taxonomy categories were

- aggregated
- disaggregated from other categories
- eliminated
- added

These taxonomy category operations could be embedded within the context of a genetic algorithm approach to evolving taxonomies. There a number of ways of capturing these category operations. An example approach could be as follows. Let a two level taxonomy, with one primary level ("I") and three sub items ("A", "B" and "C"), be represented by a sequence (x_1 x_2 x_3 x_4 x_5 x_6 x_7)where (x_1 x_2 x_3 x_4) = (1,1,1,1,0,0,0) means that the primary item and the three sub items are included. (x_5 x_6 x_7) could then refer to aggregations I-A and I-B, I-B and I-C, and I-A and I-C. The original taxonomy would then be (1,1,1,1,0,0,0). Elimination of item I-C would result in the replacement of the "1" with a "0" (1,1,1,0,0,0,0).Aggregation of I-A and I-B would yield (1,0,0,1,1,0,0). Given these sequences we could use a genetic algorithm approach to creating and choosing new sequences.

Koza (1997) provides a generic approach that potentially could be used to facilitate evolution in the following three steps:

1. Generate an initial population of taxonomy elements, for example, two taxonomies
x_1 x_2 x_3 x_4 x_5 $x_6$$x_7$
y_1 y_2 y_3 y_4 y_5 y_6 y_7

2. Iteratively perform sub-steps using the following rules:

Reproduction (maintain the same strings)
x_1 x_2 x_3 x_4 x_5 x_6 x_7
y_1 y_2 y_3 y_4 y_5 y_6 y_7

Cross Over (create new strings)
x_1 x_2 x_3 x_4 x_5 | y_6 y_7
y_1 y_2 y_3 y_4 y_5 | x_6 x_7

Mutation (with small probability change an item)
x_1 x_2 x_3 y_4 x_5 x_6 x_7

3. Using some approach to determine which constructed strings are most appropriate, choose and keep cycling or finish. "Fitness" measures such as entropy might be used to provide such a measure or it could be based on user specified design requirements.

Although this approach clearly needs additional development and empirical testing, it does suggest that genetic algorithms could be used to predict or facilitate knowledge evolution by providing a potential evolved taxonomy.

Facilitating Knowledge and Preference Evolution using Intelligent Agents

User preferences shift over time as the user better understands the problem, the system and the two in context. Enembreck and Barthes (2003) and others have proposed using intelligent agents to facilitate a continuous adaptation to user preferences, e.g., in web environments. In this approach, intelligent agents can monitor the use of the system to better understand user preferences. Agents can gather data about how users use or don't use knowledge, or about user preferences.

A similar approach can be used to understand the need to evolve and to facilitate DSS evolution. Agents can be used to explicitly gather information from users. For example, periodically, agents could be used to query users as to whether or not the system was meeting their needs, and what changes would be helpful. This analysis could leverage specific information that the users have about the system and potential changes.

Agents also could be used implicitly to facilitate evolution. For example, agents could watch and keep track of which features and capabilities are and are not being used. If a feature or capability is being used extensively, then that can indicate that is an area of the system that could be extended. If a feature is not being used or is only being used on occasion then either the user is not that aware of that feature or capability, that feature is not necessary or that feature needs improvement and change. As a result, agents could be used to signal a need for evolution.

Predicting and Facilitating DSS Evolution Using Delphi Method

The Delphi Method is designed to try to predict the future. It does so by asking multiple experts what they think will happen in the future. As the approach is used, an "understanding" of the future is attained, and then fed back to the same experts to see if any other perceived or expected changes are found, based on the feedback.The Delphi Approach also can be useful in facilitating an understanding of what is likely to change and how to facilitate that change.

The Delphi Method has been used previously in conjunction with knowledge-based systems. Roth and Wood (1990) used the Delphi approach to help elicit knowledge for a knowledge base. They found that it was an important tool for generating additional knowledge over and above that available to a single expert. Multiple experts generated many more ideas, suggesting that it was important to involve multiple experts in knowledge acquisition.

In terms of DSS evolution, the Delphi Method could be used to capture expert or user assessments as to which system features or capabilities could be changed and how that revised system could look. Experts and users could be asked what knowledge they expected to be different or relevant and what knowledge was missing from the existing system. The Delphi Method could also be used to anticipate which technologies could be integrated into a DSS in order to extend and evolve the system.

Summary and Contributions

This paper has investigated DSS evolution and how to manage that evolution through predicting and facilitating evolution.

This paper investigated notions of evolution in DSS. We defined evolution, and characteristics of evolution, such as backward compatibility, measuring evolution, and why we would expect DSS to evolve. In addition, we investigated issues of backward compatibility, determining if evolution has occurred and whether evolution is always a formal process.

The previous research on DSS evolution also was investigated. Research on each component was summarized and in some cases extended. Based on the analysis of that literature, the weakest point in the literature seems to be evolutionary studies of particular application types and in the evolution of knowledge. As a result, the paper investigates in more detail, knowledge evolution as it might be construed in a DSS. The paper also investigated issues such as the prediction of evolution, e.g., the knowledge in a DSS, and trying to facilitate evolution of a DSS, using e.g., Delphi and genetic algorithm approaches.

This paper extended the notion of evolution to DSS as a bundle and as a set of components. Further, this paper has brought together and extended some of the research on DSS evolution. In addition, the paper noted that evolution could occur actively or passively. In the case of active evolution, we can predict and facilitate evolution as part of the management of evolution. In the case of passive evolution, DSS evolution still happens. In either case, DSS will have emergent behavior, but our primary means of tracking evolution will be through artifacts, because of the general lack of backward compatibility.

As noted in this paper, in some cases, there has been substantial research in the analysis of the evolution of particular components of DSS, e.g., databases, technologies, ontologies, etc. However, there has been limited research addressing how DSS are affected by evolution.

This paper has provided some tools and methods that can be used to predict evolution of different components, e.g., knowledge. The Delphi Method was suggested as a means to predict evolution of all different types of components, including technology on which the DSS is based. Genetic Algorithms also were approached as a basis to try and evolve DSS. Intelligent agents were seen as a tool that can gather data or knowledge about the user and use it as a means of understanding how the parameters in a DSS or even components might need to change to accommodate the user.

REFERENCES

1. APQC's International Benchmarking Clearinghouse and Arthur Andersen and Co., "Process Classification Framework," 1996
2. A.A. Angehrn and T. Jelassi, "DSS research and practice in persepective," *Decision Support Systems*, Volume 12, Volume 4-5, November 1994, pp. 267-275.
3. Arnott, "Decision Support Systems Evolution: Framework, Case Study and Research Agenda," *European Journal of Information Systems*, Volume 13, 2004, pp. 247-259
4. F. Balbo and S. Pinson, "Dynamic Modeling of a Disturbance in a Multi – Agent System for Traffic Regulation," *Decision Support Systems*, Volume 41, 2005, pp. 131-146.
5. Banerjee, J., Kim, W., Kim, H., and Korth, H. (1987), Semantics andImplementation of Schema Evolution in Object-Oriented Databases. *Proceedings of the ACM/SIGMOD Annual Conference on Management of Data*, San Francisco, California, 311-322.
6. V. Berzins, "Merging Changes to Software Specifications," M. Broy and B. Rompe (eds.), *Lecture Notes in Computer Science*, 1526, pp. 121-131, Springer Verlag, Berlin Heidelberg, 1998.
7. JL Chen and D. McLeod, D. E. O'Leary "Schema Evolution for Object-Based Accounting Database Systems*," Expert Systems with Applications*, Volume 9, Number 4, 1995, pp. 491-502.
8. J-C Courbon, J. Grajew and J. Tolovi, "Design and Implementation of Interactive Decision Support Systems: An Evolutionary Approach," Technical Report, Institute d'Administration des Enterprises, Grenoble, France, 1978.
9. F. Enemback and J.P. Barthes, "Agents for Collaborative Filtering," in M. Klusch et al., CIA 2003, Lecture *Notes in Artificial Intelligence*, 2782, pp. 184 – 191, Springer Verlag, Berlin Heidelberg, 2003.
10. D. Erwin and D. Snelling, "Unicore: A Grid Computing Environment," *Lecture Notes in Computer Science*, R. Sakellariou et al. (eds.), #2150, 2001, Springer – Verlag, Berlin Heidelberg, 2001.
11. R. Ferguson, "Microsoft, SAP unveil Duet 1.5," *eWeek*, November 27, 2006, p. 33.
12. J. Gabriel, "Metadata evolution management in your SOA," http://webservices.sys-con.com/read/47670.htm, 2005.
13. C. Gibson and R. Nolan, "Managing the Four Stages of EDP Growth," *Harvard Business Review*, January – February 1974, pp. 76 – 87.
14. M. Golani and O. Etizion, "Temporal Active Rules," R. Pinter and S. Tsur (Eds.) *Lecture Notes in Computer Science* #1649, 1999, pp. 159 – 172.
15. P. Haase and L. Stojanvic, "Consistent Evolution of OWL Ontologies," Springer *Lecture Notes in Computer Science* 3532, pp. 182-197, 2005.
16. P. Haase, F. van Harmelen, Z. Huang, H. Stuckenschmidt, Y. Sure, "A Framework for Handling Inconsistency in Changing Ontologies," in Y. Gil, ISWC, *Lecture Notes in Computer Science* 3729, pp. 353-367, 2005.
17. P. Keen, "Adaptive Design for Decision Support Systems," *ACM SIGMIS Database* Volume 12, Issue 1-2, Fall 1980, pp 15-25.
18. P. Keen and T.J. Gambino, "Building Decision Support Systems: The Mythical Man-Month Revisited," in J.L. Bennett, *Building Decision Support Systems*, Addison Wesley, Reading Massachusetts, pp. 133-172, 1983.
19. T. Kojo, T. Mannisto and T. Soininen, "Towards Intelligent Support for Managing Evolution of Configurable Software Product Families," in B. Westfechtel, A. van der Hock (EDS): SCM 2001/ 2003, *Lecture Notes in Computer Science*, 2649, PP. 86-101, 2003, Springer – Verlag, Berlin Heidelberg 2003.
20. G. Koundourakis and B. Theodoulidis, "Association Rules & Evolution in Time," in I. P. Vlahavas and C.D. Spyropoulos, SETN 2002, *Lecture Notes in Artificial Intelligence* 2308, Springer – Verlag, 2002.
21. J. Koza, "Genetic Programming," *Encyclopedia of Computer Science and Technology*, 1997 http://citeseer.ist.psu.edu/212034.html
22. M. M. Lehman, "Software's Future," *IEEE Software*, January – February 1998, pp. 40-44.
23. M.M. Lehman, V. Stenning, W.M. Turski, "Another Design of Software Design Methodology," ICST DoC Research Report, June 1983.
24. C. Liu, S. Chang and P. Chrysanthis, "Database Evolution using Ever Diagrams," Advanced User Interfaces, http://citeseer.ist.psu.edu/liu94database.html, 1994.
25. A. K. Loasby, "Changing Approaches to Metadata at BBC.co.uk: From Chaos to Control and then Letting Go Again," http://www.asis.org/Bulletin/Oct-06/loasby.html, 2006.McAfee, "Mastering Three Worlds of Information Technology," *Harvard Business Review*, 2006, pp. 1-10.
26. R. Nolan, "The Stages Theory: A Framework for IT Learning and Organizational Adoption," 9-193-141, Harvard Business School, 1993.
27. N. Noy and M. Klein, "Ontology Evolution: Not the Same as Schema Evolution," *Knowledge and Information Systems*, July 2004, Volume 6, Number 4, pp. 428 – 440.
28. D.E. O'Leary, "Impediments in the Use of Explicit Ontologies for KBS Development," *International Journal of Human Computer Studies*, Volume 46, 1997, pp. 327-337.
29. D. E. O'Leary, "On the Evolution of a Taxonomy for Best Practices: A Comparative Analysis of Two Organizations," versions of this paper were presented at *MIKDS*, September 2004 and International
30. *Conference on Creativity and Innovation in Decision Making and Decision Support CIDMD2006*.
31. D. E. O'Leary, "Empirical Analysis of the Evolution of a Taxonomy of Best Practices," *Decision Support Systems*, Forthcoming 2007.
32. S-L. Pan, C-W Tan and E. Lim, "Customer Relationship Management (CRM) in e-Government: A Relational Perspective," *Decision Support Systems*, Volume 42, 2006, pp. 237-250.

33. B. Ramesh and K. Sengupta, "Multimedia in Design Rationale Decision Support System," *Decision Support Systems*, Volume 15, 1995, pp. 188-196.
34. G. Rao and M. Turoff, "A Hypermedia based Group Decision Support System to Support Collaborative Medical Decision Making," *Decision Support Systems*, Volume 31, 2000, pp. 187 – 216.
35. D. Riano and S. Prado, "Improving HISYS1 with a Decision Support System," in Artificial Intelligence Medicine, 8th Conference on AI in Medicine in Europe, AIME 2001, Cascais, Portugal, July 1-4, 2001, Proceedings, p. 140
36. J. F. Roddick, "A Survey of Schema Versioning Issues for Database Systems," *Information and Software Technology*, Volume 37, Number 7, 1995, pp. 383-393.
37. R. Roth and W. Wood, "A Delphi Approach to Acquiring Knowledge from Single and Multiple Experts," Trends and Direction in Expert Systems, *Proceedings of the 1990 ACM SIGBDP conference on Trends and directions in expert systems*, Orlando, Florida, Pages: 301 – 324, 1990
38. A.P. Sage, Decision Support Engineering, John Wiley & Sons, New York, 1991.
39. R. Schuler, "Analytic and experimentally derived estimates of market power in deregulated electricity systems: policy implications for the management and institutional evolution of the industry," *Decision Support Systems*, Volume 30, 2001, pp. 341-355.
40. A. Sen, "Metadata Management: Past, Present and Future," *Decision Support Systems*, Volume 37, 2004, pp. 151-173.
41. M. F. Shakun, "Airline Buyout: Evolutionary Systems Design and Problem Restructuring in Group Decision and Negotiation," *Management Science*, Volume 37, Number 10, pp. 1291-1303, 1991.
42. M.S. Silver, Systems that Support Decision Makers: Description and Analysis, Wiley, Surrey, UK, 1991.
43. H. Simon, *The Science of Management Decision*, Harper Brothers, New York, 1960.
44. C.R. Stabell, "A Decision Oriented Approach to Building DSS," in Building Decision Support Systems, edited by J. L. Bennett, Addison-Wesley, Reading Massachusetts, pp. 221-260, 1983.
45. A,Tiwana and B. Ramesh, "A Design Knowledge Management System to Support Collaborative Product Evolution," *Decision Support Systems*, Volume 31, Number 2, June 2001, pp. 241 – 262.
46. D. N. Xuan, L. Bellatreche and G. Pierra, "A Versioning Management Model for Ontology-based Data Warehouses," Springer Verlag, *Lecture Notes in Computer Science*, 4081, pp. 195-206, 2006.
47. J. Yoon and L. Kerschberg, "A Framework for Knowledge Discovery and Evolution in Databases," *IEEE Transactions on Knowledge and Data Engineering*, Volume 5, Number 6, December 1993.
48. H. Zhuge, "Knowledge Network Planning and Simulation," *Decision Support Systems*, 2005.