Visualizing Epistemic Structures of Interrogative Domain Models

By

Tracey D. Hughes

Submitted in Partial Fulfillment of the Requirements

for the Degree of

Master of Computing and Information Systems

Youngstown State University

May 2008

Visualizing Epistemic Structures of Interrogative Domain Models

Tracey D. Hughes

I hereby release this thesis to the public. I understand that this thesis will be made available from the OhioLINK ETD Center and the Maag Library Circulation Desk for public access. I also authorize the University or other individuals to make copies of this thesis as needed for scholarly research.

Signature:

Tracey D. Hughes Student Date

Approvals:

Dr. Alina Lazar, Thesis Advisor Date

Dr. John Sullins, Committee Member Date

Dr. Yong Zhang, Committee Member Date

Peter J. Kasvinsky, Dean of School of Graduate Studies & Research Date

Abstract

In this paper, we explore the concept of epistemic visualization in interrogative domains. Epistemic visualization is the process and result of developing visual models that capture the structure, content, justification and acquisition of knowledge obtained by a software agent in a knowledge-based system. The knowledge is the foundation in which the agent can respond to queries against a corpus containing questions and answers. The visualizations are therefore used to examine the quality of the software agent's knowledge. The visual models will include justification and commitment artifacts as well as knowledge acquisition flow. The visualization will demarcate the apriori and posteriori knowledge. The knowledge of the software agent is stored in epistemic structures which are knowledge representation schemes that supports the basic concepts of knowledge as defined by the tripartite analysis of knowledge. Epistemic visualization is used to analyze the quality of the knowledge of a software agent in an interrogative domain. For our purpose, interrogative domains are hearings, trials, interrogations, personality test or any document source in which the primary content is questions and answers pairs. In this paper, we introduce the Epistemic Structure Es that captures the agent's knowledge and the visualization of that epistemic structure using common visualization techniques.

Acknowledgements

I would like to acknowledge Dr. Alina Lazar, Dr. John Sullins, Dr. Yong Zhang, Dr. Eugene Santos and the Open Source Movement without which this thesis would not have been possible.

Table of Contents

Table of Contents

Table of Figures

Table of Tables

1 Introduction

There are 100s of 1000s archived public documents in question and answer form accessible via the internet. These interrogative-based documents include interviews, surveys, hearings and trials with a wide range of topics and interest to the public. When a user desires to examine the content of these documents, what is commonly accessible to the average user are browsers capable of rendering and searching the semi-structured documents that have been encoded using HTML or XML. The user may search the documents for particular topics using keywords or key phrases. The browser's search features will display the location of the keyword or key phrase in the document. We propose the development of an epistemic agent that directly answers a natural language query. The epistemic agent will base its response on the deductive propositional knowledge discovered from the document and stored in Epistemic Structures. The epistemic structures are knowledge representation schemes that supports the basic concept of knowledge as defined by the tripartite analysis of knowledge. Epistemic logic is used to provide the structuring and relational basis of the epistemic structure. The epistemic visualization will be visual models of the knowledge stored in these structures for the purpose of examining the quality and content of this knowledge. The visual models will include justification and commitment artifacts as well as knowledge acquisition flow. The visualization will demarcate the apriori and posteriori knowledge. Epistemic visualization is used to analyze the quality of the knowledge of a software agent in an interrogative domain. For our purpose, interrogative domains are hearings, trials, interrogations, personality test or any document source in which the primary content is questions and answers pairs. In this paper, we introduce the Epistemic Structure E_S that captures the agent's knowledge and visualizing the epistemic structure using common visualization techniques.

SECTION 1: Visualization Fields

Visualization can be defined as the transformation of a data source to visual representations for the purpose of gaining insight and understanding of that data source [1]. The visualization will enable the user to observe and perceive the hidden features of the source data. A visual representation of the source data appeals to our innate ability to identify patterns and extract information. Visualization is used to communicate and identify patterns in large amounts of data, information or knowledge. According to Colin Ware [2]:

> *"the power of visualization comes from the fact that it is possible to have a far more complex concept structure represented externally in a visual display than can be held in visual and verbal working memories."*

Visualizations can be used to support and extend our cognitive systems by supplying a medium to exploit and examine existing mental images. It takes advantage of spatial and visual cognitive abilities we have to reduce the effort required to process complex data, information and knowledge. When mapping the data parameters to graphical elements that produce images, undetectable patterns and relationships may be reveal. These patterns may have remained hidden when presented as lists or tables. Visualizations can be used as tools of thought, extentions to cognitive processes and the exploration of knowledge spaces.

Table 1 Visualization Fields, their goal, source data and techniques..

Visualization Field	***Goal***	***Source Data***	***Common Techniques***
Program/software Visualization	Understanding and effective use of code	Program code	Histograms, scatterplots, graphs, trees, etc.
Information Extraction Visualization	Pattern Discovery and Trends	Documents collections containing natural language	Circle graphs Concept Association graphs
Knowledge Visualization	Communicate and transfer knowledge between 2 more people	Highly structured documents, data, information, etc.	Concept maps

There are several fields of visualization but for the purpose of this paper, we are limiting our discussion to these few:

- Program/software visualization
- Information extraction visualization
- Knowledge visualization

These fields and others can be characterized by their goal, their source data and the visualization techniques commonly used as listed in Table 1.

Software visualization is the use of typography, graphic design, animation and cinematography along with computer graphics technology to visualize program code, data and control flow to facilitate in the human understanding and effective use of the software systems [3]. It is used to visualize the internal constructs, processes, structure and functionality of software. The visualizations can be based on executing programs in order to examine the data produced and make performance evalutions. It can be based on static code for the purpose of evaluating the architecture of the software. Software visualizations can be produced from software metrics or other artifacts created by the software. Software metrics are concerned with the measurement of the software product and the development process [4]. Some metrics are lines of code, function points, cyclomatic complexity and control and information flow.

Artifacts can represent the state of the data, objects as they are modified by an algorithm or a program or the contents of data structures at various points of execution. Artifacts can also represent the state of the whole system (where key aspects of the system are determined and marked) or the environment. Figure 1 shows the towers of Hanoi at various states.

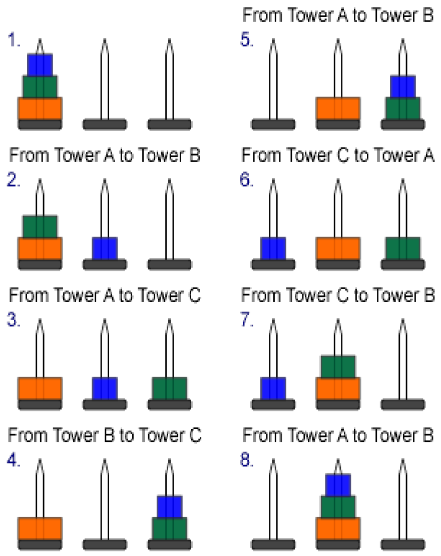

Figure 1 Towers of Hanoi at various states.

This field is composed of multiple sub-fields each having their focus. In the Figure 2, is a software visualization taxonomy [3]. We see that software visualization is divided into algorithm and program visualizations. Sometimes sofware and program visualization have been used synonymously. A distinction is made here in order to focus on the layers of abstractions in software, one layer being the program (lower layer) and the other being the algorithm (a higher level of abstraction) [3]. Algorithm visualization can be static or dynamic. A simple flowchart would be an example of a static visualization where an animation showing the steps of a bubble sort would be an example of dynamic visualization. The purpose would be to show how the algorithm works and performs. Program visualization is the use of graphical elements to visually represent the dynamic and static aspects of a program [5]. PV can be used to visualize the internal structures and states of a program. The programmer creates a model of the software to be developed by using program design tools such as flowchartings and UML [22]., etc. during the SDLC

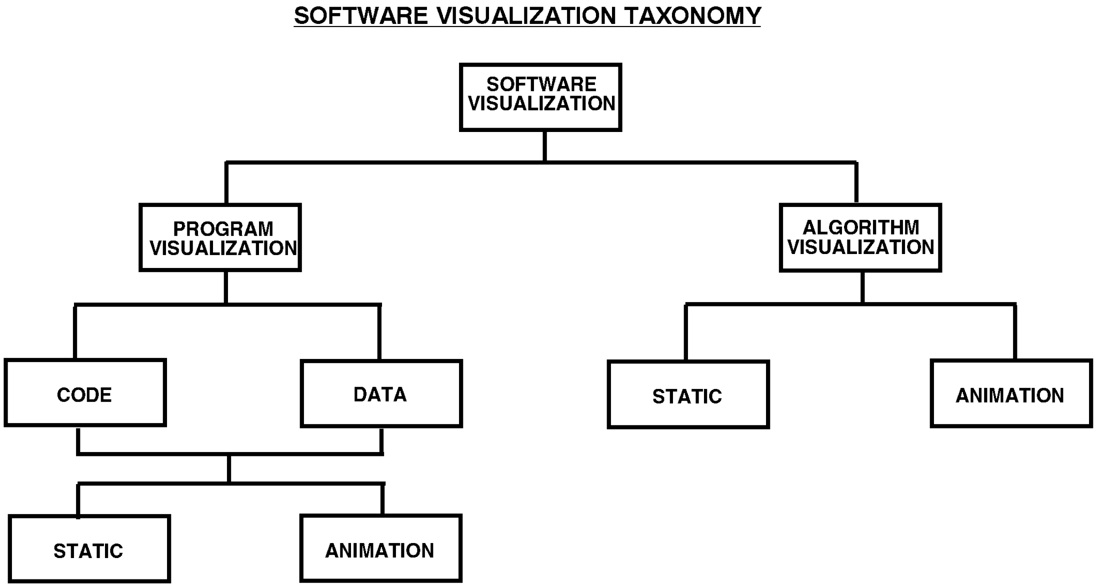

Figure 2 The Sofware Visualization Taxonomy.

(Software Development Life Cycle). This is a type of static program visualization. Those design models are transformed to a different type of static model by using software development tools such as editors and compilers. The source code is associated with the objects or types it is modeling. This can also be visualized by using UML produced by such visualization packages such as Umbrello. Umbrello shows objects and their relationships for programs written in C++. The static version of the software is transformed to a dynamic version once it has been compiled or interpreted and then executed. The run-time version of the software can then be compared to the design model in order for a programmer to understand or debug the system. Sofware visualizations systems such as debuggers, code tracers and optimizers can be used by the developer.

There are other types of program visualization systems for non-developers who are interested in the performance of the software such as TPM (Transparent Prolog Machine) [6]. It was orginally conceived to debug a program written in Prolog but eventually was used to understand declarative semantic and the rationale behind a particular response to a query. Figure 3 shows a diagram that captures TPM's response to the query:

? older(john,sue) meaning "Is John older than Sue?"

yes

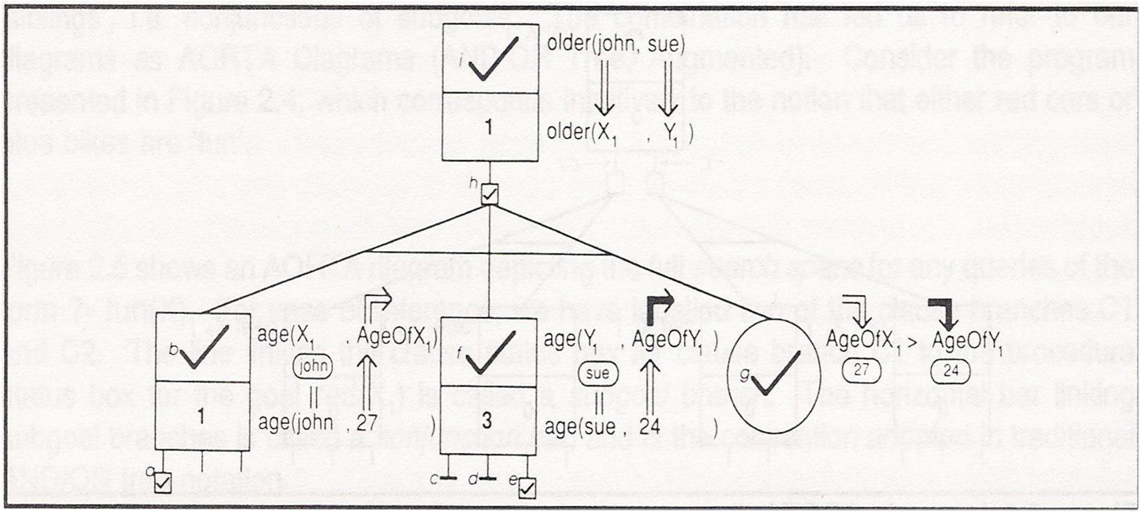

Figure 3 Trace of TPM's response to a query.

The visualization shows how the TPM arrived to the response, yes.

Information extraction is any process that extracts structured information from an unstructured corpus. The goal of information visualization is to gain insight into large amounts of data by using visual applications. From the unstrucutred or semi-structured text, the data that is found, explicitly stated or implied. Then the data is placed in a database or data tables. Information extraction uses natural language processing techniques to identify and then extract the desired information. Once the desired information is extracted and stored, it is presented to the user as a visualization that shows distributions, frequent sets, and association patterns [7]. The visualization can also be static dynamic, animated or interactive. Information extractions are visualized by using visualization techniques such as circle graphs, line and association graphs and hyperbolic trees to show associations.

Knowledge visualization creates visual models of knowledge. Knowledge is created from information and by the application and exchange between individuals [8]. According to [9] "knowledge is information, which has been cognitively processed and integrated into an existing human knowledge structure." It is the goal of KV to use one or more visual models to "improve the transfer of knowledge among people and to improve the creation of knowledge in groups" [10] by providing new insights and enhancing visual communications. The knowledge is created from information and data and by using visual representations, patterns and relationships are revealed.

Where there are several aspects of domain knowlege as defined by cognitive science (conceptual, episodic, analogical, procedural), KV focuses on conceptual knowledge, that is a "propositional representation of abstract concepts and their semantic relation" [9].

Table 2 The Date Model Information Visualization Pipeline.

Data Stages	***Description***
Value	Raw or original data.
Analytical Abstraction	Meta-data, data about data or information.

Data Stages	***Description***
Visualization Abstraction	Information that is visualizable using a visualization techniques or software.
View	The graphic representation of the mapping.
Data Transformations	***Description***
Data Transformation	Raw or orginal data is transformed to meta-data.
Visualization Transformation	Meta-data is reduces further to data that will be visualized.
Visual Mapping Transformation	Visualizable data is mapped to graphic elements.

1.1 The Visualization Pipeline

The visualization process can be characterized as a stream of transformations [1]. The data source is transformed from one data form to another until it can be finally mapped to graphic elements. This describes a pipeline in which the input of the pipeline is the data source and the output of the pipeline is a visual representation of the data. The Data State Model is an information visualization pipeline that focuses on the transformation of data and the processing operations [11]. The process is decomposed into four data stages, three data transformations and four types of "within stage" operators. The four distinct data stages are:

- Value
- Analytical Abstraction
- Visualization Abstraction
- View

For each data stage there are operators. These operators may perform additional processing on the data but they do not change the underlying data structures. Some operators can create new types of data sets where other operators create filtered versions of the data. There are four types of "within stage" operators. Each corresponds to a data stage: within Value, within Analytical Abstraction, within Visualization Abstraction, and within View.

The three data transformation operators are:

- Data Transformation,
- Visualization Transformation
- Visual Mapping Transformation

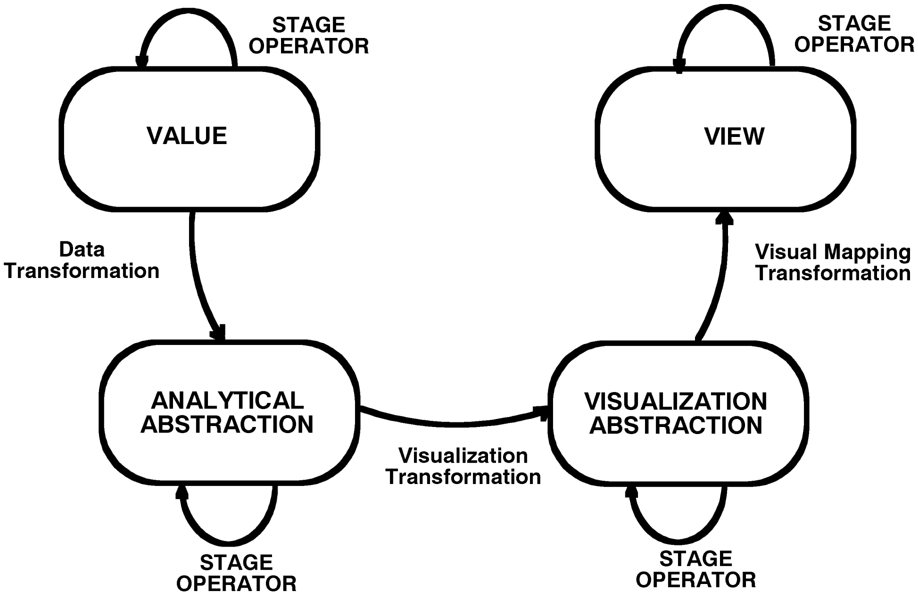

Figure 4 The Data Model as a UML Statechart

Table 2 lists the data stages and transformation with a brief description. Figure 4 shows the Data State Model as a UML statechart. For each state there are the "within stage" operators that can be used to modify or create more data sets.

This pipeline can be applied to the problem of visualizing trend information [12]. Trend information is a summarization of temporal statistical data. The Value or source data is multiple newspaper articles and blogs. Targeted information is extracted from these sources and then the information is visualized as graphs. What is extracted is trend information such as temporal expressions and statistical values. Using natural language processing techniques [12], objects such as the names of companies, products, etc. are identified. Statistical information is marked. Trend analyzing is performed to identify the hidden trends. These "trends" are stored in tables and converted to comma separated data It can then be read into a spreadsheet or other graphing software then visualized as a graph. The visualization pipeline for trend summarization is depicted in Figure 5.

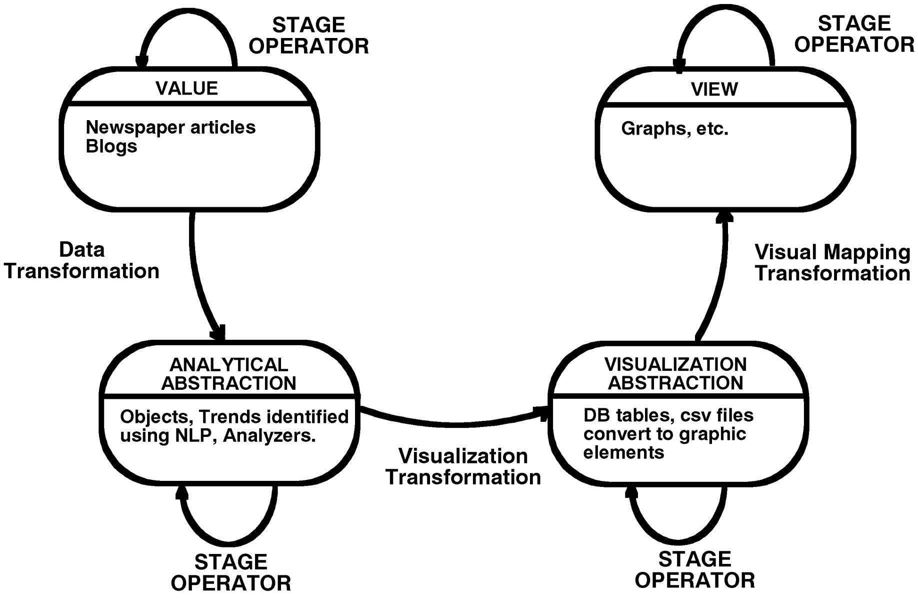
STAGE OPERATOR
VALUE
Newspaper articles
Blogs
STAGE OPERATOR
VIEW
Graphs, etc.
Data Transformation
Visual Mapping Transformation
ANALYTICAL ABSTRACTION
Objects, Trends identified using NLP, Analyzers.
VISUALIZATION ABSTRACTION
DB tables, csv files convert to graphic elements
Visualization Transformation
STAGE OPERATOR
STAGE OPERATOR

Figure 5 Trend Information Visualization Pipeline

Although the Data State Model was created for Information visualization, it is general enough to be used to describe the pipeline for other visualization processes.

Software visualization is the transformation of some aspect of program code (Value) to a graphic representation (View). For example, a visualization of numbers being sorted by a heap sort program can be visualized. The pipeline will transform the program code (Value) to a graphic representation (View). What is visualized is the content of the array at various points of code execution. The numbers are stored in an array. The contents of the array is extracted from the program and may be written to a file or stored in a table. The meta-data is then transformed to a representation that will be easliy mapped to a graphic element. This is the Visualization Transformation step and the Visualization Abstraction stage. The elements of the vector is then mapped to locations on the virtual canvas. This will be the graphic representation of the heap. In these two examples, most of the work is done during Data Transformations and the Analytical Abstraction stages. But this is not always the case for all visualization processes.

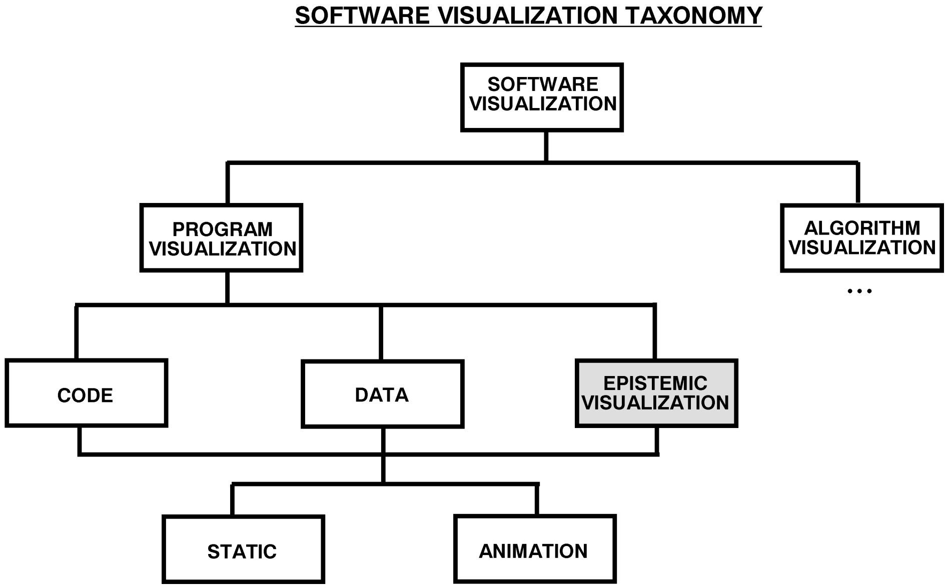

Figure 6 Software Visualization Taxonomy including Epistemic Visualization

1.2 Epistemic Visualization

Epistemic visualization creates visual models of propositional knowledge of a software agent for the purpose of analyzing the quality and content of its knowledge using conventional visualization techniques. In determining the quality of the knowledge, we are including content, justification, commitment and acquistition properties. Epistemic visualization has some similarites to the three types of visualization previously discussed. EV is a software visualization. It visualizes Epistemic Structures (E_S) that reside in software. Similar to software visualization, artifacts of the E_S is extracted and transformed to meta-data then mapped to graphic representations. One of the purposes of program visualization is to analyze the performance of a program. As with the TPM, the visualization could be used to analyze the rationale of the response. TPM's visualization traces its responses to queries. EV is used to analyze the rationale of the agent's responses to queries. Figure 7 is the taxomomy of software visualization with EV added. Epistemic visualization would reside comfortably as its own branch of program visualization considering it can visualize both code and data.

It is similar to information extraction as far as Value or data source and some aspects of data transformation. With Epistemic visualization's data source also include text documents that contain natural language. In the example discussed, HTML documents of newpaper articles and blogs of domain specific topics were used. For EV, we focus on

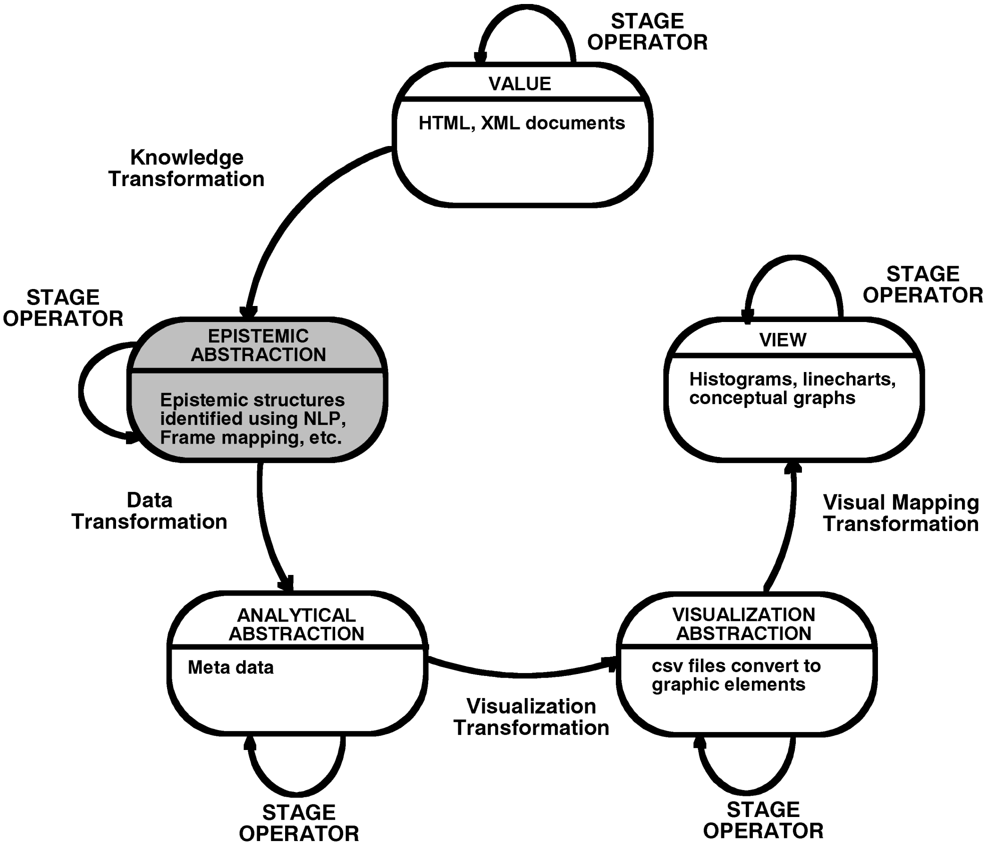
STAGE OPERATOR
VALUE
HTML, XML documents
Knowledge Transformation
STAGE OPERATOR
EPISTEMIC ABSTRACTION
Epistemic structures identified using NLP, Frame mapping, etc.
STAGE OPERATOR
VIEW
Histograms, linecharts, conceptual graphs
Data Transformation
Visual Mapping Transformation
ANALYTICAL ABSTRACTION
Meta data
VISUALIZATION ABSTRACTION
csv files convert to graphic elements
Visualization Transformation
STAGE OPERATOR
STAGE OPERATOR

Figure 7 The Data Model is altered to include Knowledge Transformation and Epistemic Abstraction.

HTML or XML documents that contain interrogatives or questions and answers pairs from trials and hearings. When transforming data, those tags could be useful in helping to identify certain objects in the documents (especially XML tags) but for our process, all tags are filtered. During Data Transformation and at the Analytical Visualization stage, natural language processing techniques [12] are used to help identify objects and attributes and to transform Q & A pairs to propositions.

Although EV share some similarities to information extraction during Data Transformation, here is where there is a departure. Epistemic visualization visualizing Epistemic Structures [13] (noted as E_s). The E_s is not simple data, it is a knowledge representation scheme that contains propositonal knowledge. The HTML is transformed to this structure. In the case of EV, the Data State Model would have to altered with an additional transformation and stage. In Figure 8, the pipeline has a Knowledge Transformation and an Epistemic Abstraction Stage. The Knowledge transformation and the Epistemic Abstraction is the process of transforming question and answer pairs to deductive propositional knowledge.

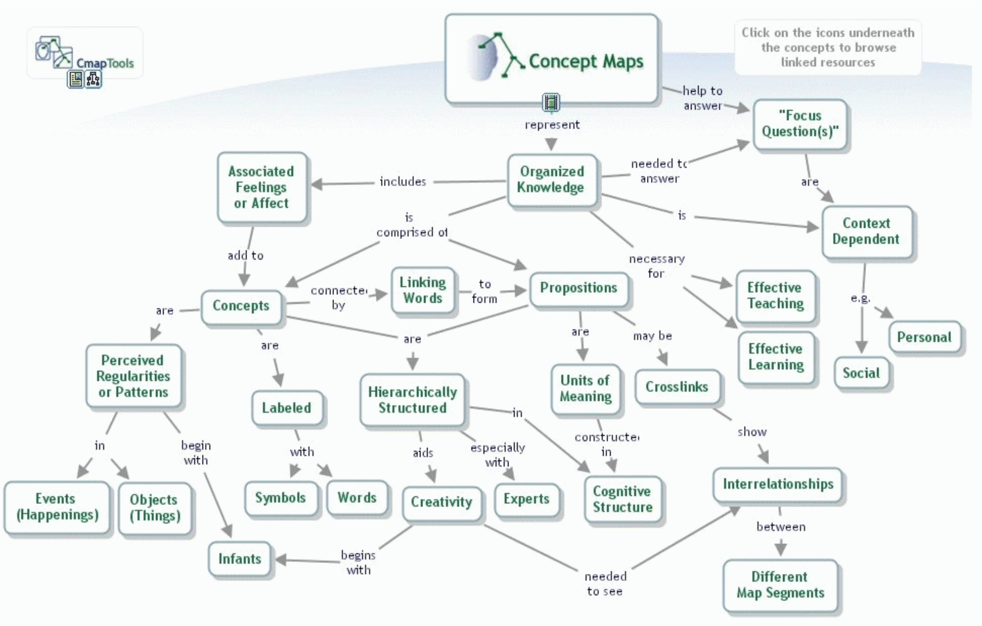

Figure 8 A Concept map on concept maps[cmap.ihmc.us].

1.3 Why not knowledge visualization?

Although the purpose of EV and KV visualizes knowledge, KV visualizes conceptual knowledge where EV is primarily concerned with propositional knowledge. The conceptual knowledge is graphically represented as conceptual maps predominantly. The conceptual map is based on the conceptual graph [14]. The conceptual graph is logic-based notation based on semantic networks and the diagramming of logical expressions. With conceptual maps, each node is a concept and an edge describes the connection between the concepts. Figure 8 shows a conceptual map of the on conceptual maps. The purpose of the conceptual map is to show the knowledge domain, the concepts within that domain and how they are connected. These nodes and edges form propositional statements not knowledge. With EV, knowledge is defined epistemologically by the

Tripartite Analysis of Knowledge i.e. knowledge as justified true belief (JTB). This is represented in an E_s. The formalism of the E_s relies on epistemic logic as well as our theory of an agent.

Epistemic Visualization is a type of program visualization as it visualizes software constructs namely the knowlegde of a software agent. It also shares some similarites with Information Extraction in that they both utilize the same type of source data and use natural language processing during the transformation from the HTML to data or knowledge (as in the case of EV). Epistemic Visualization is as an intersection of Information Extraction, Software Visualization and Epistemic Logic/Epistemology.

SECTION 2: Our Epistemological Foundations

Epistemology is the study of knowledge [15]. It is a branch of philosohy that is concerned with the nature and acquistion of knowledge. Epistemology addresses:

- Necessary and sufficient conditions of knowledge
- Types of knowledge
- Non-monotonic reasoning
- Sources of knowledge

Here we will briefly discuss each of these areas focusing on the aspects that are the most relevant to this paper.

2.1 Necessary and sufficient conditions of knowledge

Epistemology attempts to specify the necessary and sufficent conditions for it to be stated that a person *knows* something. According to the "Theory of Causality", conditions that are logically necessary for the occurence of an event (here the event is a person knows) means that the event could not occur in the absence of the condition. But that condition alone could not bring about the event, it is necessary but not sufficient. A sufficent condition for an event to happen is the circumstance in whose presence the event must occur.

The basic Tripartite Analysis of Knowledge not only provides the traditional definition that states that knowledge is justified , true, belief (JTB). It also specifies the necessary conditions for knowledge. So for the event to occur:

A knows **P** *iff*

these are the necessary and sufficient conditions:

P is true
A believes that **P** is true
A is justified in believing that **P** is True

Truth, belief and justification are huge concepts and difficult to define. Here we are focused on a functional description of these concepts as used in Epistemology in order to proceed in our discussion of ES and visualization. So we briefly describing these concepts within that context in Table 3.

Table 3 Brief description of JTB

Conditions for Knowledge	***Brief Description***
Justification	Adequate indication [16]
True	Conditions are met that establishes truth [16]
Belief	Commitment [17]

The Tripartite Analysis of Knowledge dates back to two of the most significant pillars of philosophy, Plato and Immanuel Kant. It continues to be an ongoing discussion in the field of Epistemology. This is the definition of knowledge in which the ES is based. We propose that **A** can be a software agent.

The Tripartite Analysis can be represented using discrete structures like First Order Logic (FOL), graph and set theory. FOL, set and graph theory provide the software structures and notions used to construct ES and perform operations on its behalf.

2.2 Types of knowledge

Epistemology defines several types of knowledge which includes:

- conceptual
- procedural
- propositional
- inductive/deductive

Conceptual knowledge is the knowledge created by "semantic memory" based on the interconnections of concepts. It is the knowledge of how concepts with a particular domain are related [18] . Conceptual knowledge is also called structural knowledge [18]. As mentioned earlier, this type of knowlege is what is represented in conceptual maps and visualized by knowledge visualization techniques.

Procedural knowledge is "know how". Where propositional knowledge is "know what" or the knowledge that something is true [16].

A true proposition can be concluded inductively or deductively. An inductively derived proposition is one that does not ensure that it is true. It can based on the generalizations of individual instances. A conclusion is drawn and a generalization is made. For example:

It rained today,
it rained on Tuesday.
It rained last week.
Inductive proposition: It will rain tomorrow.

Deductively derived propositions are based on entailment or logical implication (by applying the rules of logic it must follow). By deduction, conclusions are reached about an instance based on a generalization. This is the opposite of induction.

Rain occurs when the necessary and sufficent conditions in the atmosphere are present.
Those conditions are present today.
Deductive proposition: It will rain today.

We propose our software agent has propositional deductive knowledge.

2.3 Non-monotonic reasoning

Non-monotonic reasoning is concerned with changing beliefs based on the introduction of new knowledge. The truth of propositions are contingent based on what is accepted as currently true but reserve the right to retract them when confronted with additional propositions. When new propositions are introduced, any propositions on which the proposition in question was based will have to be re-examined. This is called the *frame problem*. The knowledge of the software agent is non-monotonic.

2.4 Source of Justification

The source of justification is concerned with the matter of 'where does knowledge come from?' and 'from where do we obtain justification to what we believe?'. We use the commonly accepted assumptions that knowledge and justifications can come from any source. Epistemology defines sources as having to be reliable. Examples of sources of justifications are memory, testimony and reason. *A priori* and *posteriori* are types of justification based on reasoning. Table 4 gives brief descriptions of both.

Table 4 A priori and posteriori descriptions.

Sources of Justification based on reasoning	***Description***
A priori	**S** believes **P** not because of experience but because **P** is true by defintion and its truth can be discovered by reason alone.
posteriori	**S** believes **P** because **S** has discovered the truth of **P** through experience although **P** is true by definition and experience was not necessary.

Our software agent's justifications for a proposition will be based on reason.

2.5 A Software Agent's Knowledge

Here we want to define the concept of an agent as "... an entity that acts upon the environment it inhabits. Agents are not merely observers of their environment, nor are they passive receipients of actions performed by other entities. Rather Agents are the active, purposeful originators of actions " [19]. In this paper we are concerned with the epistemology of *software agents* in a very narrow domain, specifically an interrogative domain, and the visualization of that knowledge. The knowledge of the software agent resides in a knowledge representation scheme called an Epistemic Structure.

SECTION 3: The Epistemic Strucutres

We developed an *Epistemic Structure* noted as E_s for the purpose of storing the knowledge of an epistemic agent. Our conjecture is any interrogative document has an epistemic representation.

The knowledge is constructed by first converting question and answer pairs stored in our document. The agent extracts the Q & A and converts them to propositions. For example, from the corpus of US vs. Osama Bin Laden Trial (2001), we have the question stated on day 21 of the trial:

Question: Agent Miranda, did you have any involvement in the investigation of the embassy bombings prior to visiting Mr. El Hage and his wife April Ray?

Answer: No, I don't believe I had any involvement.

The agent will produce the proposition:

Agent Miranda does not believe he/she had any involvement in the investigation of the embassy bombings prior to visiting Mr. El Hage and his wife April Ray.

3.1. Building the Epistemic Structure: Justified True Belief

We can build Epistemic Structures from the transcripts of the thousands of U.S. congressional hearings and trials that have been archived in electronic form. These transcipts have been stored as semi-structured text. They contain the Q & A that were given during the course of a public hearing or a federal trial. Our interrogative agent's propositional knowledge consists of those questions, their corresponding answers, and the propositions that are entailed by combining the questions with their corresponding answers. We describe the transformance of Q & A pairs to propositions as function θ.

Let t = {set of Q & A pairs from the transcript}

Let ρ = {set of propositions and Q & A pairs that the agent knows}

then we have:

$\theta: t \rightarrow \rho$

Where θ is an injective mapping from t to ρ and $|\rho| > |t|$. The transform function θ serves to build the agent's knowledge, in our case the epistemic structure E_s :

Let $\mathbf{E}_S$ be the structure:

$\mathbf{E}_S = < G_1, G_2, J_S, V_c, F>$

G_1 is a graph (V,E). V is a nonempty set of apriori propositions, models, and Q & A pairs. E is a nonempty set of relations between the elements of V.

G_2 is a graph(V,E) . V is a set of posteriori propositions and E is a set of relations between the elements of V.

G_2 will be populated with propositions inferred from quering the system. If the agent cannot respond to a query, meaning the answer is not stored in G_1 then the answer must be inferred from a proposition in G_1 or G_2 (once it has been populated). New inferred propositions as a result of a query will be stored in G_2 .

The propositions stored in G_1 and G_2 contains a priori and posteriori justifications or challenges, respectively. These propositions are created as result rules of language and deductive implication.

The software agent's beliefs are measured by what *A* (software agent) accepts as true. A accepts φ (proposition 1) as true as long as *A* is unaware of any ψ (proposition 2) that challenges φ. *A* is aware of V_c in $\mathbf{E}_s$. V_c represents the vector of commitment of the agent for $G_1 \cup G_2$ propositions. If φ has not been challenged then its entry in V_c will contain a 2 for φ otherwise the entry will contain a 1 for φ. The initial entry will be 2 until φ has been challenged.

Each proposition in G_1 is initially considered unchallenged and therefore contingently true. Once a proposition has been challenged, the original proposition is considered tainted. For each of the tainted propositions, the agent will acknowledge the existence of the proposition but the agent does not have any commitment to it (value 1). The user can still query the agent in regard to any tainted propositions. The agent can return to the user those propositions that it considers tainted. Tainted (t with value 1) and untainted (s with value 2) propositions are both element of the set of justifications:

$t \in J_S$, $s \in J_S$

An interrogative agent's knowledge may consist of multiple domains where each domain, *d*, is represented by an Epistemic Structure:

Let $d = \{ E_{s1}, E_{s2}, E_{s3} \ldots E_{sn} \}$

Where *d* is a set of E_s representing a particular domain, or a collection of domains

Then we have:

$$K_s = \bigcup_{i=1}^{N} d$$

where K_s is the total knowledge space of the agent and K_s is a set union of the domains.

3.2. The Pipeline Revisited

The Q&A entailed propositions and the building of a conceptual graph model of the transcript are parts of our five step process that transforms the semi-structure text into its epistemic representation. In Section II, we discussed our Epistemic Visualization pipeline. Our five step process is situated as "Knowledge Transformation" and the "Knowledge Abstraction" stage of our pipeline:

Step 1:

Using the HTML/XML and other structured rules of the transcript, convert the entire transcript into a tagged corpus.

Step 2:

Create a MTS (Model Theoretic Semantic) of the corpus capturing them both as predicates and frames.

Step 3:

Convert the Q&A pairs to propositions and assign there commitment value of 2 to each.

Step 4:

Construct frames from the propositions generated in Step 2.

Step 5:

Using the structures created in Steps 2, 3, 4 instantiate the structure that represents the knowledge space of the transcript/trial.

We can give these steps simpler names in order to make the representation of the pipeline simpler:

Step 1: Filter the corpus

Step 2: Create MTS

Step 3: Convert to Propositions

Step 4: Construct Frames

Step 5: Build Structure

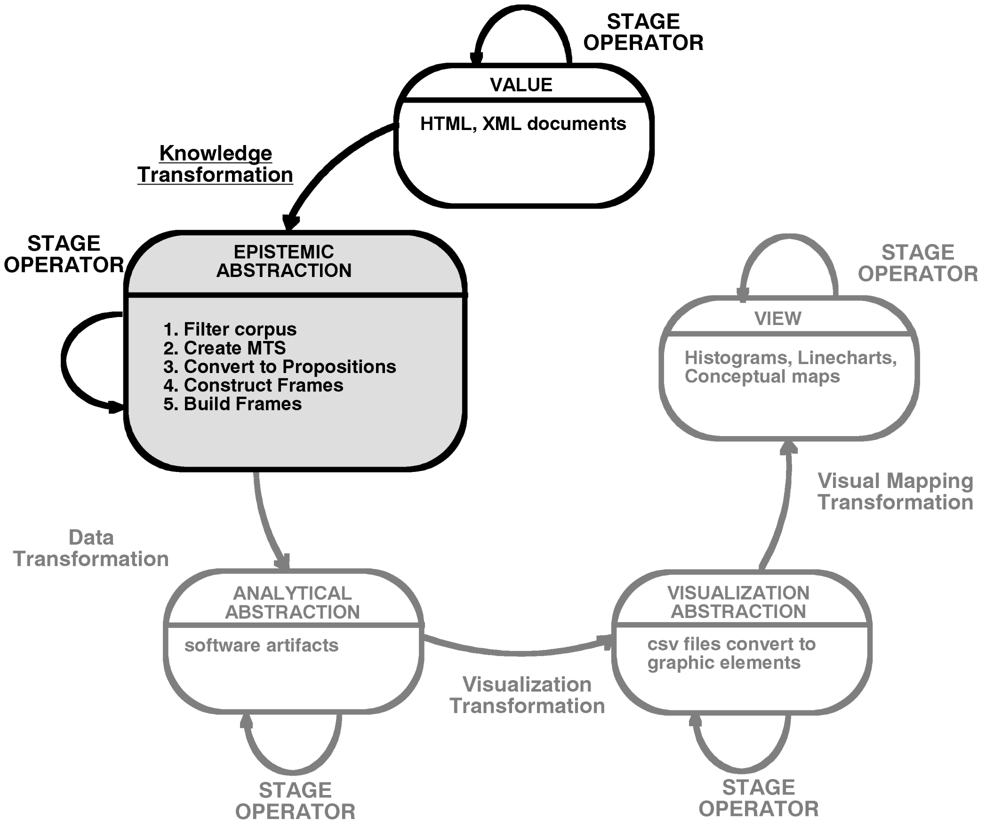

Figure 9 Our EV pipeline with more details for our Epistemic Abstraction stage.

For a more detailed treatment of these steps in building of the Epistemic Structure, see [13].

3.3 The NOFAQS Project

The process of converting interrogative documents to Epistemic Structures has been implemented in the NOFAQS system. NOFAQS is an experimental system currently under development at Ctest Laboratories. NOFAQS is designed to answer queries concerning the content of an interrogative document. NOFAQS uses rational agents and

Natural Language Processing (NLP) to aid users during the process of analyzing a corpus. The corpus was obtained from the Internet as a collection of Hypertext Markup Language (HTML) files consisting of:

- 76 File formatted in standard HTML (one for each day of the trial)
- 25,979 Questions
- 25,930 Answers
- 461,938 Lines of domain relevant text
- 1,854,242 Domain relevant words

This was the trial of the UNITED STATES OF AMERICA vs. USAMA BIN LADEN, et al., February 2001 for the first bombing of the Trade Center in 1993. For a detailed discussion on this process, see [19]. The Epistemic Structure built from this corpus will be used as the basis of our Epistemic Visualization techniques in the next section.

SECTION 4: Epistemic Visualization Techniques

The software artifacts we are using to generate our visual models are quantitative information. The visualization techniques we used are classic graphing formats used for information extraction and text mining such as histograms, linecharts and conceptual graphs. Although these are classic techniques, they have proven to be useful in elucidating patterns and associations of numeric and categorical data [7]. These may be the simplest techniques but they are very powerful for analyzing and communicating statistical information [20]. For visualizing Epistemic Structures, they are useful as visual models for our software artifacts.

Here are the classic visualization techniques used for Epistemic visualization currently:

Histograms/Barcharts

These are good for displaying proportions and distributions.

Linegraphs

These are good for comparisons of the results of different sets of queries, same queries on different sets, different sets under multiple constraints.

Conceptual graphs [14]

A knowledge representation language that has a notation and a graphic representation. The graphic representation uses concept nodes to represent entities and attributes, states and events. Relation nodes are used to show the relation between concept nodes. Knowledge visualization uses a concept map (based on the conceptual graph)

4.1 Visualizing the Knowledge Space

The view of the E_s can be the state of the whole K_s or of a single proposition. When visualizing the whole K_s , certain properties have been selected to represent the state of the whole K_s. These properties will elucidate some aspect of the Epistemic Structure as it relates to:

- How well is the K_s justified
- How much knowledge is inferred

By using the conceptual graphs, a single proposition can be examined for its justifications and challenges. A conceptual graph consists of concept nodes, relation nodes and arches.

Concepts nodes represent entities, attributes, states and events. Relation nodes show how the concepts are interconnected. Concepts are linked by conceptual relations to form a graph. The box will represent the concept and the circle is the relation. Concept relations may have any number of archs. If there are n >= 3 arcs the arc can be numbered with arrows pointing towards the circle and numbered 1 to n - 1.

Concept graphs are finite connected and bipartite (contians two different types of nodes). Every arc links one type of node to another types of nodes. If a relation has n arcs then it is considered n-adic, monadic means 1 adic, 2-adic is dyadic, 3-adic is triadic.

All visualizations will be a snapshot of the K_s at a particular time t in its existence. As the K_s is modified by the addition of new propositions inferred and other propostions are tainted, this will change the state of the K_s. It may be desireable to use the K_s at some previous state.

4.1.1 Q & A Category Distribution

In trials, many of the questions can be classified as yes/no but they can be further categorized to determine the nature and purpose of the question. Revealing the purpose of the questions provides more of an analytical description of the content of the interrogative document. Conceptual categorization of questions is used to analyze a question by decomposing it into two descriptive components:

- question concept
- conceptual category

[21]. For example, for the question:

Question: Did Odeh indicate whether or not he knew where the American Embassy was?

The question concept of this question would be:

Odeh indicate whether or not he knew where the American Embassy was.

The conceptual question category would be "verification". The purpose of the question is to verify whether *Odeh indicated whether or not he knew where the American Embassy was*. This type of categorization of a question helps in recognizing the conceptual differences between questions.

There are many ways to categorize questions. We use the Wendy Lehnert's 13 conceptual question categories. The 13 categories are listed in Table 4 along with a brief description and an sample classification question from the USA. VS. USAMA BIN LADEN transcript.

Table 5 13 Conceptual Question Categories with an Example and Description.

13 Conceptual Question Categories	**Examples**	**Brief Description**
Casual Antecedent	And can you tell us why you decided then to tell them about the money you stole?	About states or events that have in some way caused the concept question.
Goal Orientation	Why did the Sudanese government give al Qaeda the Khartoum Tannery or a portion of it?	About the motive and goals behind an action.
Enablement	How did you learn this?	Concept question is enabled by unknown acts or states.
Casual consequent	What happened to the videotape on November 1, 2000 that should have been	About the question concept having caused an unknown concept or causal chain.

13 Conceptual Question Categories	**Examples**	**Brief Description**
	played and recorded on 10 South?	
Verification	Did you get copies of the preliminary instructions?	About the truth of an event.
Disjunctive	Was it the same items on both days or just one day?	About the truth of an event but has multiple question concepts instead of one.
Instrumental/Procedural	How is it that you are able to recognize that as a part of an Atlas?	About totally or partially unknown instrumentality.
Concept Completion	What kind of work specifically do you do with them?	About specifying a particular event with missing components or completion of an event.
Expectational	Why isn't the witness present?	About an act that did not occur.
Judgemental	What can we do Your Honor?	About soliciting a judgement on the part of the listener.
Quantification	How much of the Khartoum Tannery did al Qaeda buy?	About amounts.
Feature Specification	What did he look like?	About the property of a given person or thing.
Request	Would you please explain to the jury what a motion for severance from codefendants means.	Ask the listener to perform an act.

We used a histogram to visualize the categorization of the questions and answers from a simulated trial for the purposes of this visualization.

Figure 10 shows a histograms of the categorization of 27,087 questions. The highest frequency of questions are:

- Casual Consequent
- Enablement
- Goal Orientation
- Casual Antecent

The least frequent questions are:

- Request
- Judgemental
- Expectation

This visualization gives you insight on the nature of the questions and therefore answers

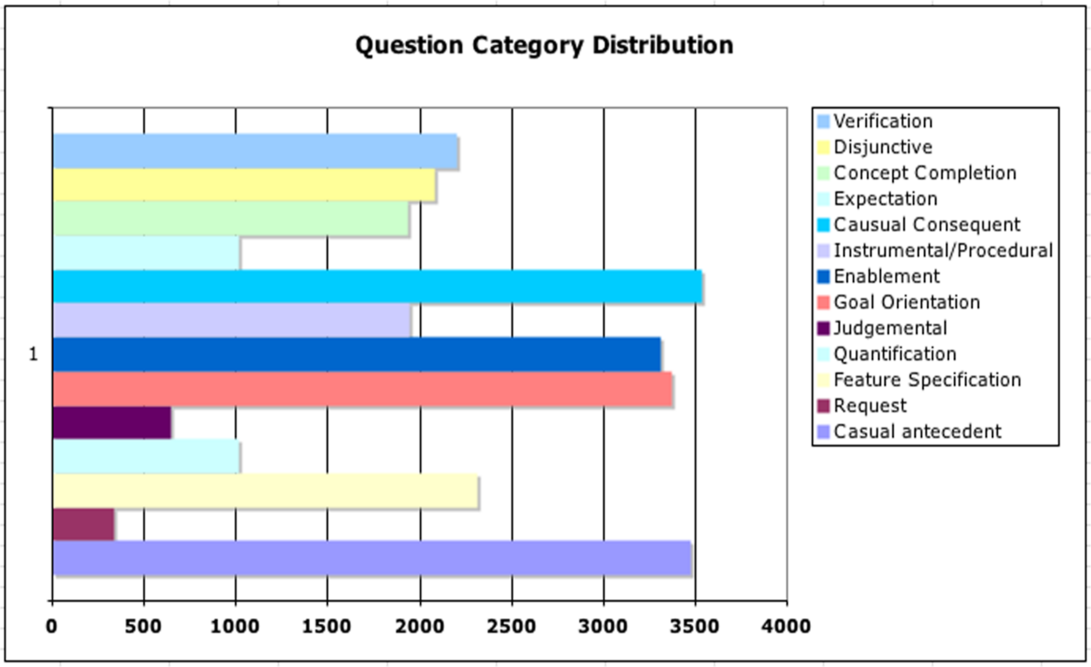

Figure 10 Question Category Distribution

of this sample interrogative domain. If this was a trial, it would be expected that there would be a high frequency of goal oriented questions and a low of judgemental questions. Judgemental questions are normally asked of experts in that context. Responses would be based on the knowledge of the witness. Unless the trial was a battle of expert-based testimony, this would have a low frequency. On the otherhand question about chains of events, motives behind actions are the types of questions used to build a case. These would be used a justifications of actions.

The frequency of certains types of questions would all be contigent on the nature of the case. As stated, if the case is based on technical evidence, then there would be higher frequencies of judgemental or request (asking for demonstrations) type questions. If the case is built of the testimony of witnesses to a crime, their may be a high frequency of feature specifications questions.

So as far as determining the quality of the K_s of the software agent, if the questions of the appropriate category has a high frequency (based on the type of trial) then one could conclude that the responses of the agent will be comprehensive, sound and substantial due to well examined witnesses and the production of a high quality of knowledge. If those appropriate questions have a low frequency then the knowledge of the agent would be suspect, and the responses returned would reflect it.

4.1.2 State of Interrogative Agent's K_s at Time t

The State of the Interrogative Agent's K_s refers to the state of its knowledge at time t. The state of the K_s is defined by these numbers of attributes:

- Propositions in G1
- Propositions in G2
- Queries at time t
- Justifications (Js)
- Propositions used as challenges
- Propositions with a commitment value of 1 (challenged)
- Propositions with a commitment value of 2 (non-challenged)

Time t is marked by queries to the interrogative agent. Whenever questions are asked of the agent, the state of the K_s is modified. A check of the system at time t will note the differences in the changes in the attributes' values. A modification of the state of the K_s may affect how the agent answers future queries or if past queries will require a different response. A response that requires the Interrogative Agent (IA) to make an inference, may result in a proposition that challenges current propositions in G_1 or G_2. Inferred propositions may also increase the number of non-challenged propositions. But the inferred proposition may also be immediately challenged by existing propositions. Changing the commitment value of a single proposition from non-challenged (2) to challenged (1) will call into question all propositions on which that proposition is a justification.

Here are the properties of the attributes, domain and ranges.

I. Number of propositions in G_1

Propositions in G_1 are entailed from the Q & A pairs extracted from the HTML document. So:

$| G_1 |$ = Number of Q & A pairs

There may be some information loss due the Q & A pairs that could not be converted due to formatting problems. The number of propositions in $| G_1|$ are constant although J_s, V_c and challenges may be added to the frame

II. Number of propositions in G_2

Propositions are not added to **G_2** until the Interrogative Agent (IA) is queried. As a consequence of a response, the IA may have to infer a new proposition maybe even multiple propositions. These new propositions are added to **G_2**. No inferences may be necessary if the query is answerable by existing propositions in G_1. So:

$| G_2 | <= | Q |$ or $| Q | <= G_2$

G_2 may increase as new queries are put to the IA.

III. Number of Queries

The number of queries to the system at some time t is accumalative. So:

let Q be the set of queries to system

$$| Q_t | < | Q_{t+1} |$$

IV. Number of propositions used as Justifications of G_1 and G_2

There are 0 to many justifications for a single proposition in $\mathbf{G_1}$ **or** $\mathbf{G_2}$. A single proposition can be used by different referents or sources (this would be the interviewee, witnesses, etc) as a justification for different propositions. So:

$$| J | <= | G_1 \cup G_2 | <= | J |$$

It is important to use a bag rather than a set in order to keep the multiple instances. A point of interest may be to visualize how many times a propositions are used as a justifications for other propositions.

V. Number of propositions used as challenges of G_1 and G_2

There may be no challenges to many challenges for a single proposition in G_1 in G_2. A single challenge can be used multiple times by different propositions. So:

$| C | <= | G_1 \cup G_2 | <= | C |$

Similar to justifications, we use a bag due to the multiple instances. A point of interest may be to visualize how many times a propositions is used as a challenge.

VI. Number of propositions with commitment value of 1

This is the number of propostions that have been challenged. A proposition may have none or multiple challenges. Once a proposition has been challenged, its commitment value is reduced from 2 (the default for unchallenged propositions) to 1. So:

$| V_1 | <= | C |$

VII. Number of propositions with commitment value of 1

Propositions have a default commitment value of 2 until they have been challenged. A challenge to a proposition may be inferred as a result of a query. Once a proposition has

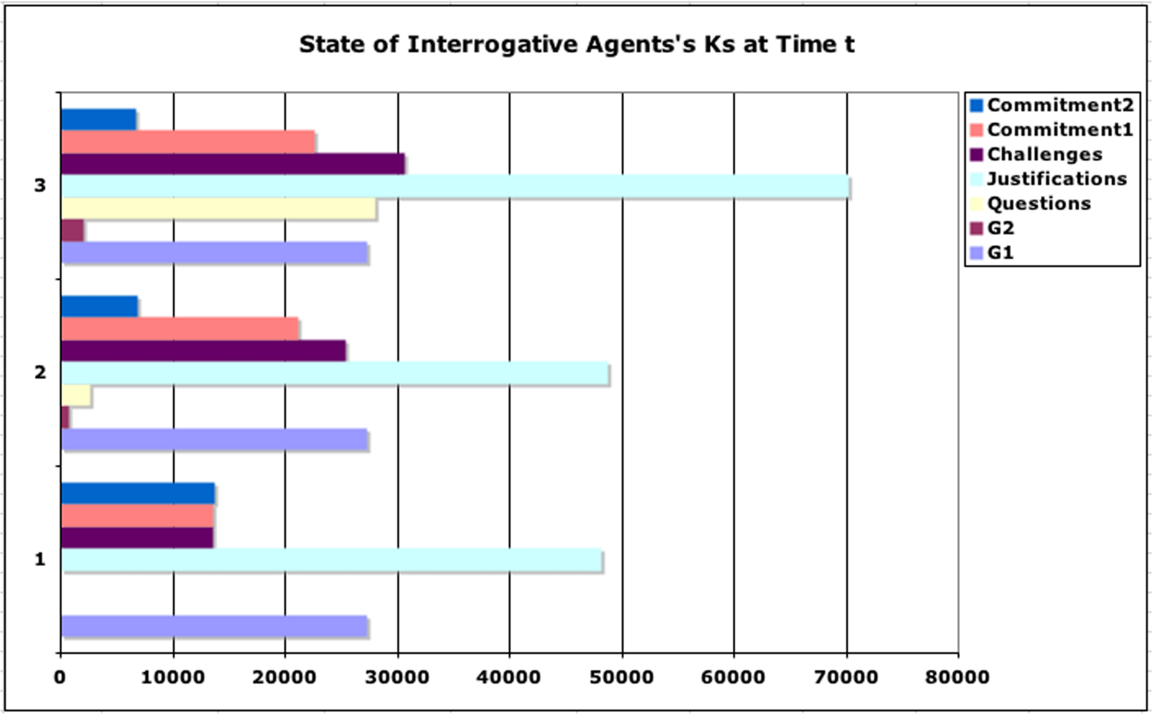

Figure 11 State Interrogative Agent's K_s at time t

been challenged, its commitment value is reduced from 2 (the default for unchallenged propositions) to 1. Once reduced and cannot be raised. So:

$$| V_2 | = | G_1 \cup G_2 | - | V_1 |$$

4.1.3 Visualization of the State of the Interrogative Agent's K_s at time t

Figure 1 is the visualization of the State of Interrogative Agent's $\mathbf{K}_s$ at Time t using a histogram. Here we can see the $\mathbf{K}_s$ at three different states. Initially the $\mathbf{K}_s$ at t = 0 where there has been no queries but to the agent. At that time:

$| Q_S | = 0$ and $| G_2 | = 0$

Also we see that number of propositions with value of 1 and value of 2 are almost equal and the number of justifications are quite large. This is the case because propositions can be used multiple times as stated. But 1/2 of the propositions at this point has already been challenged. This indicates a very weak case by both (if this is the $\mathbf{K}_s$ of a trial). Much of the testimony (propositions explicitly or implicitly given in testimony) has already been contradicted. But there is a chance, by inferenced non-challenged propositions, the $\mathbf{K}_s$ can be redeemed.

The state at time t + 1, we see that a few queries have been asked and a few inferred propositions have been added to G_2. The number of justifications has only increased slightly but the number of challenges has almost doubled and the number of propositions that are challenged has increased by 1/3.

At time t + 2 there is a problem. The number of justifications has increased by a 1/3 based on only a few inferred propositions in G_2. This means propositions are used multiple times. This may indicate many of the propositions are too dependent on each other. The number of propositions that are unchallenged is now neglible. The number of challenged propositions are almost equal to the number of propositions in G_1 and G_2. The quality of the agent's knowledge is non-existent.

4.1.4 Visualizing of Propositions

Here we will discuss visualizing the justifications of propositions.

The State of Propositions of the Interrogative Agent's K_s at Time t visualization is a linechart of these attributes for propositions in G_1 or G_2. For this visualization we want the ratios of:

- the number of propositions that have been challenged to the number of propositions in G_1 and G_2
- the number of propositions that have not been challenged to the number of propositions in G_1 and G_2
- the number of propositions that have been inferred (G_2) to the number of propositions in G_1 and G_2
- the number of propositions entailed from Q & A pairs (G_1) to the number of propositions in G_1 and G_2

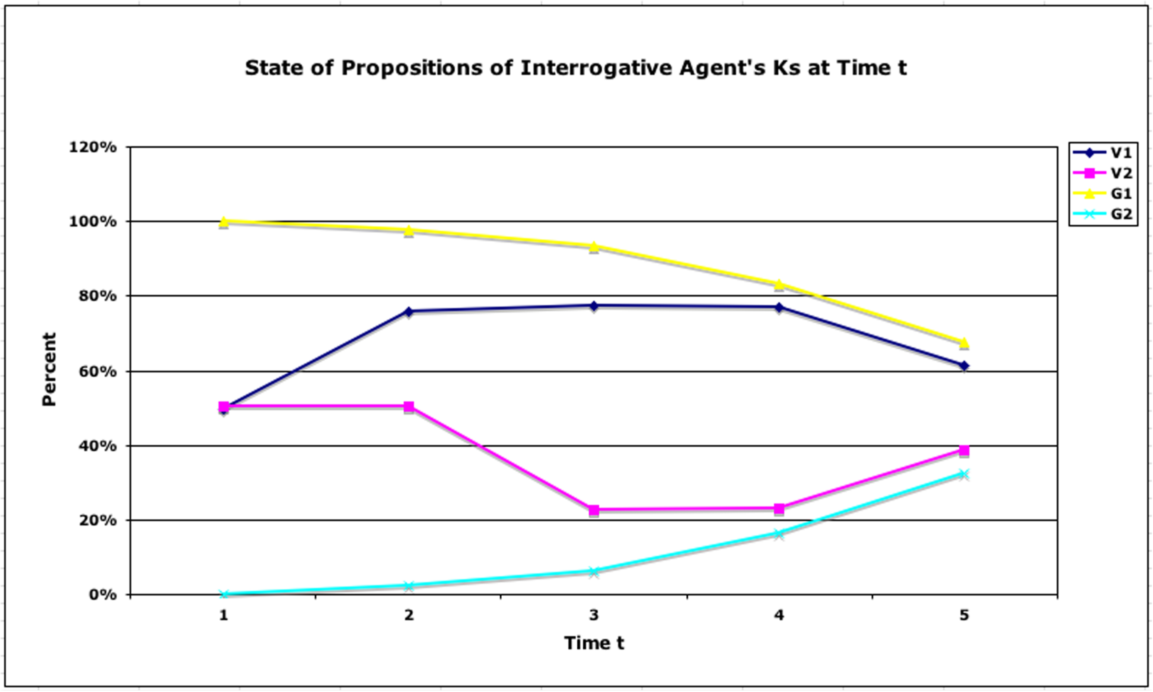

Figure 12 State of Propositions of the Interrogative Agent's K_s at Time t.

Figure 12 is the visualization of the State of Propositions of the Interrogative Agent's K_s at Time t. This linechart shows the ratios listed above. Comparing the series G_1 (yellow) to the series G_2 (cyan), at time t, the ratios of $|G_1|$ to $|G_1 \cup G_2|$ is 100% because their are no propositions in G_2 where the ratio of $|G_2|$ to $|G_1 \cup G_2|$ is a 0%. At time t + 3 there are propositions in G_2 and at time t + 5 the ratio of inferred propositions $|G_2|$ to $|G_1 \cup G_2|$ has risen to 32% where ratio of $|G_1|$ to $|G_1 \cup G_2|$ fell to 68%. The size of G_1 does not increase over time. The rate of the decrease will depend on the number of newly inferred propositions. At some time in the future $|G_2|$ may overtake $|G_1|$. What will that imply?

We can also compare the ratio | V_1 | to | $G_1 \cup G_2$ | and | V_2 | to | $G_1 \cup G_2$ |. At time t + 3 the number of non-challenged propositions dramatically drops but starts to rise again as more inferred propositions are introduced at time t + 5 and the number of challenged propositions decrease. The new inferred propositions have to increase the number of V_2 by default. They have not been challenged as of time t + 5.

A conceptual graph can be used to show the justifications and challenges of a specific propositions along with its referents. Figure 13 shows an example of the use of a conceptual graph to represent a proposition. The grayed box marked
"PROP A" is the proposition of interest. The relation nodes have these following meaning:

REF - referent or source of the proposition
CHAL - the challenge to the proposition
JUST - the justification of the proposition

The concept nodes are:
PROP X - is a labeling for a particular proposition
PERSON X - is the labeling for a person who is the source of the proposition

Here, PROP F and PROP D both have the same referent PERSON 5.

Conceptual graph are very useful to show the justifications, challenges and their referents. The problem with conceptual graphs is when their is a large number of justification and/or challenges it becomes difficult to visualize. Also, due to sizing and placement, the actual proposition or the name of the person is not shown.

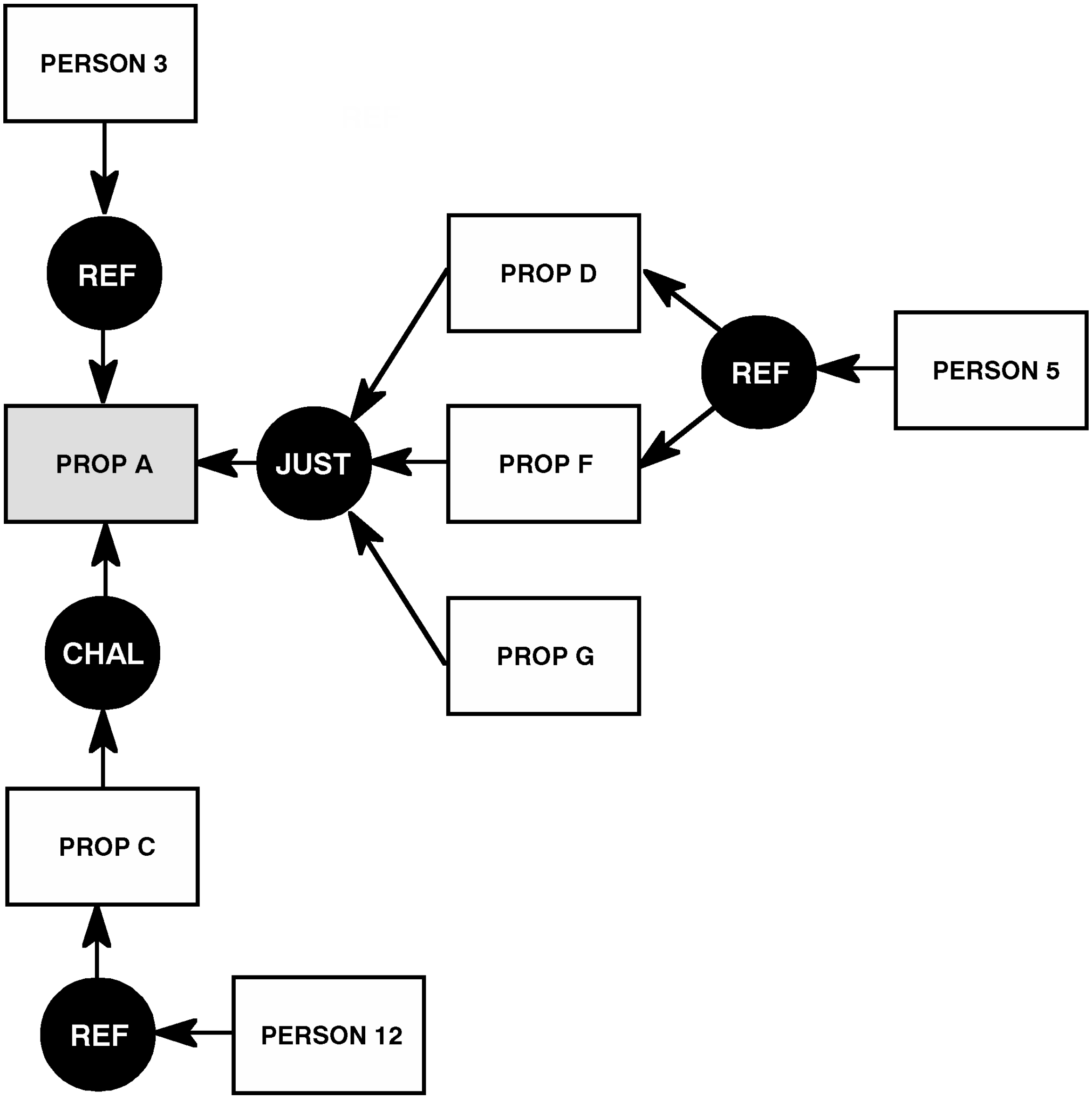

Figure 13 Conceptual Graph of a proposition, its justifications and challenges and referents.

Discusssion

Here we have shown that classic visualization techniques can be used to visualize Epistemic Structures. Epistemic Structures represents the knowledge of an interrogative domain. In visualizing this structure, we were able to examine the quality of that knowledge space. In using histograms to compare the $\mathbf{K}_s$ of the interrogative agent at different times, we were able to see the how the content and structure of the $\mathbf{K}_s$ evolves over time as new inferred propositions were added. Those inferred propositions contributed to increasing the number of non-challenged propositions possibly but widely contributed to interconnection of the propositions. This introduces the frame problem. A web of connected propositions causes the fabric of the $\mathbf{K}_s$ to become very fragile. If one of those propositions become challenged then all of the propositions in which it is connected becomes tainted. If the $\mathbf{K}_s$ is saved from a previous time, it is possible to reinstate a $\mathbf{K}_s$ from a previous state of less connectivity.

This describes one instance in which the quality of the $\mathbf{K}_s$ becomes critical. The other instance is when the challenged propositions overtake the number of non-challenged propositions. In the case of a trial, if most of the testimony of witnesses has been challenged, there is no case. In a hearing, if the testimony of an expert has been challenged there is reason to question the veracity of that expect. To what degree is completely subjective.

Conceptual graphs shows the justification, challenges and referents of a single proposition. This is useful in determining the quality of a single proposition. How well was it supported) by the number of justifications and sources). A propositions with a low number of justification or a high number of justification that have the same source is also point of importance.

In any of these cases, if state of the knowledge space of the agent is of high quality, meaning high number of non-challenged propositions and a high number of justifications reflects the consistency of the quality of knowledge contained in the document. If the knowledge space of the agent is a low quality, high number of challenged propositions, highly connected propositions, this also reflects the knowledge in the document.

Conclusion

The visualization of the statistical artifacts of the Epistemic Structure helped in the analysis of the knowledge space of our interrogative agent. The use of common visualization techniques revealed highlighted associations that would have been difficult to determine using other techniques. Epistemic Structures and Visualization captured the knowledge (as defined by epistemology) of a software agent for an interrogative domain. We used the classic techniques and was able to make very interesting observations. Ultimately the goals was to examine and query the content of a interrogative domain document. We presented techniques that would allow a user to examine not only the explicit content of the domain but the implicit content. Search browser do have this capability.

The visualization field has 1000s of techniques. It is our desire to use more sophisticated techniques. We can also build a software visualization system that would allow a user to examine the K_S from multiple layers, from the whole $\mathbf{K_s}$ drilling down to single propositions. We would also pursue techniques to visualize the knowledge acquistion flow. This would be visualizing the work of the search agent.

Future Work

All of the visualizations pertain to propositions. The Model Theoretic Semantic identifies all of the objects in the interrogative document. So we would like to create domains of all these objects namely defendants, witnesses, places, etc. For each object, there are challenged and nonchallenged propositions. There will also be justications for them. The knowledge space of these objects can also be visualized.

In the book, "*On Aesthetics in Science*" , Arthur I. Miller, in the chapter on "Visualization Lost and Regained: The Genesis of the Quantum Theory in the Period 1913-1927" he writes:

There is a domain of thinking where distinctions between conceptions in art and science become meaningless. For here is manifest the efficacy of visual thinking, and a criterion for selection between alternatives that resists reduction to logic and is best referred to aesthetics.

REFERENCE MATERIAL

[1] C. Hansen and C. Johnson Eds., "The Visualization Handbook",Amsterdam, Elsevier, 2005.

[2] Colin Ware, " Information Visualization: Perception for Design." San Francisco, CA:Morgan Kaufman, 2000.

[3] J. Stasko, J. Domingue, et. al, "Software Visualization: Programming as a Multimedia Experience", London, MIT Press, 1998.

[4] Everald E. Mills, "Software Metrics: SEI Curriculum Module SEI-CM-12-1.1" Carnegie Mellon University Software Engineering Institute, December 1998.

[5] P. Eades and K. Zhang, "Software Visualization", Singapore, World Scientific Publishing Co., 1996.

[6] M. Eisenstadt, M. Brayshaw and J. Paine, "The Transparent Prolog Machine: Visualizing Prolog", Oxford, England. Intellect Books, 1991.

[7] R. Feldman and J. Sangler, "The Text Mining Handbook", Cambridge, NY.Cambridge University Press, 2007.

[8] G. B. Judelman, "Knowledge Visualization: Problems and Principles for Mapping Knowledge Space", University of Lübeck, Germany, 2004.

[9] T. Keller and S. Tergan Eds., "Visualizing Knowledge and Information", Berlin, Heildelberg, Springer-Verlag, 2005.

[10] R. A. Burkhard, "Towards a Framework and a Model for Knowledge Visualization: Synergies Between Information and Knowledge Visualization", Berlin, Heildelberg, Springer-Verlag, 2005.

[11] E. H. Chi, "Taxonomy of Visualization Techniques using the Data State Reference Model', Proceedings of the IEEE Symposium of Information Visualization INFOVIS 2000, page 69.

[12] H. Nanba, N. Okuda and M. Okumura," Extraction and Visualization of Trend Information from Newspaper Articles and Blogs", Proceedings of NTCIR-6 Workshop Meeting, May 15-18, 2007 Tokyo, Japan.

[13] C. Hughes, "Epistemic Structures for Interrogative Domains", 2008.

[14] J. F. Sowa, "Semantics of Conceptual Graphs," *Proceeding of the 17th Annual*

Meeting of the Association for Computation Linguistics, pp. 39-44, 1979.

[15] M. Schlick, General Theory of Knowledge, LaSalle Illinois, Open Court,1925.

[16] P. K. Moser, D.H. Mulder and J.D. Trout, "The Theory of Knowledge", New York, Oxford University Press, 1998.

[17] N. Rescher " Epistemic Logic A Survey of the Logic of Knowledge" University of Pittsburgh Press, 2005

[18] D. Jonassen, K. Beissner and M. Yacci, "Structural Knowledge Techniques for Representing, Conveying, and Acquiring Structural Knowledge", New Jersey, Lawrence Erlbaum Associates Inc., 1993.

[19] M. Wooldridge , "Reasoning About Rational Agents" The MIT Press, 2000.

[20] E.R. Tufte, "The Visual Display of Quantitative Information 2nd Edition", Cheshire, Connecticut, Graphics Press LLC, 2001.

[21] W.G. Lehnert, "The Process of Question Answering",New Jersey, Lawrence Erlbaum Associates, Inc. Publishers, 1978.

[22] G. Booch, J. Rumbaugh, I. Jacobson, "The Unified Modeling Language User Guide", Boston, MA.