Using Objects to Communicate Legal Information

Edward P. Richards, J.D., M.P.H. and Barry B. Bounds, M.B.A.

Abstract
Modern commercial litigation poses three communication problems. The first is the simple problem of coordinating the activities of attorneys who are geographically separated. The second is communicating unambiguously through time in order to manage a case over a natural history of many years. The third, and most fundamental, is the problem of rigorously communicating information between members of a profession and its clients in such a manner as to allow the evolution of knowledge.

Our work is focused on the third problem, that of developing a formal knowledge representation system for communicating information about legal events. This system is based on the use of the object metaphor embodied in Smalltalk-80. Once a knowledge representation system has been developed, it will be iteratively refined to better approximate the real world issues in communicating legal information.

This project is part of a joint research effort between the National Center for Preventive Law at the University of Denver College of Law, and the COPIC Insurance Company. This joint research has developed a litigation events database that is the necessary precursor to building and validating a litigation event knowledge representation schema.

Introduction
Modern commercial litigation involves complex activities that must be coordinated through time and space. These activities are carried out by teams of lawyers and corporate personnel in an ad hoc fashion because law has no formally defined knowledge representation system. This makes effective intra-team and intra-corporate communication impossible. The objective of the research is to develop a prototype knowledge representation system for legal events.

This prototype system will be object-based. The object metaphor has been chosen for three reasons: 1) it appears to approximate constructs used in legal problem solving; 2) it allows rapid, incremental development; and 3) it facilitates the development of user-oriented visual interfaces. A key objective of the prototype is to facilitate communication between legal team members and corporate personnel. This will be achieved through the use of generalized objects that encapsulate legal information, but may be viewed and queried without the use of legal constructs.

The Representation of Expert Knowledge
A uniform system for the representation of expert knowledge is necessary for the evolution of that expert knowledge. This is best illustrated by comparing legal and medical services. Physicians and attorneys both provide professional services that are knowledge intensive and require the exercise of judgment under conditions of uncertainty. Medicine, because it is based on a scientific paradigm, has a standardized knowledge representation system. This system is based on the representation system for biological knowledge, with extensions such as a uniform system for representing medical diagnoses.

The International Classification of Diseases, Revision 9, (ICD-9) is used by physicians throughout the world. The ICD-9 was developed to allow physicians to communicate about the spread and treatment of the diseases. Physicians in any part of the world can benefit from the cumulative medical knowledge base because they can match their patient's illness to the literature on how that illness has been managed by other physicians. This standardized knowledge base also allows different physicians to coordinate their care of an individual patient so that the patient may benefit from the skills of several experts.

While there are indexing systems for legal reference materials, legal scholars and practitioners have not recognized the central role of knowledge representation in the evolution of a knowledge-based profession. In contrast to the ICD-9, the prevalent system for representing legal diagnoses is so general that it does not allow the unambiguous classification of legal problems. The lack of a proper knowledge representation system is a circular problem: it makes it appear that legal problems are so diverse that it is impossible to categorize them.

Attorneys are fond of dismissing attempts to regularize legal nomenclature and practices as hopelessly naive. The predominant belief is that law is too complex and idiosyncratic to be systematized. If biologists had followed this same rational, Linnaeus would never have named his first bug because the specter of 500,000 species of beetles would prove that insects are too complex to be categorized. The solution is to iteratively develop classification systems through time. It is only in the context of a highly evolved and detailed knowledge representation system that it is possible to see the similarities in individual cases that allow the classification of seemingly uncountable individuals.

The Representation of Expert Actions
Part of the knowledge base of a profession is the representation of its procedures: the actions that professionals take that manifest their decisions. For example, experts must collect information on which to base their decisions. When the diagnostic decision is made, the expert then recommends or carries out a series of interventions that manifest the expert decision. A knowledge representation system must include these procedural steps if it is to effectively encapsulate a
profession. The medical knowledge representation system is being extended to include medical procedures. This system, called current procedure terminology (CPT), is being adopted at the insistence of the insurance industry. CPT coding is being used to rationalize the reimbursement system and to facilitate health services research. Combined with diagnostic information from ICD-9 coding, standard procedure codes allow the evaluation of the cost effectiveness of different medical practice patterns.

There is not even a rudimentary nomenclature for describing the actions that attorneys perform. Legal clients are in the same position as medical insurers were before CPT coding. They are being asked to pay large bills without a clear understanding of what was done. Moreover, the lack of a knowledge representation system for legal diagnoses prevents legal clients from communicating with other clients about the cost of their legal services. Clients are not alone in being unable to communicate about legal procedures. Lawyers also suffer because it is difficult to track and manage legal work that cannot be unambiguously described.

The Research Objective

This project differs from traditional attempts to build legal expert systems. Existing legal expert systems are based on top-down approaches that depend on developing general rules to describe legal decision making. These systems have used ad hoc approaches to knowledge representation, making them dependent on both specific laws and facts, and on users who cut their interests to fit the constraints of the expert system. As a result, these systems are highly domain-specific and impossible to generalize to real world problems. In contrast, our approach is bottom-up. We have chosen to treat the lawyer's mind as a black box and concentrate on the actions the lawyer performs. It is implicitly assumed that legal reasoning is sufficiently deterministic that a rigorous classification and analysis of legal actions will provide insight into legal thinking.

The Choice of Domains

As with all artificial intelligence research, the choice of domain is critical to shaping this project. The ideal domain is narrow and well defined, but generalizable to the universe of related problems. (The dearth of commercial AI projects suggests that such idealized domains do not exist.) We have selected the narrow domain of litigation management as our development area.

Why Focus on Litigation?

Corporations spend billions of dollars each year on the legal costs of litigation. They lose substantialy more in judgments, settlements, and lost business opportunities. Corporations fear litigation because it is done in public, the outcome is frequently unpredictable, it ties up corporate assets for years, and the costs are uncontrolled. These factors should make corporations interested in litigation management tools. Litigation management is also attractive for negative reasons: litigation poses such an immediate threat that it displaces corporate resources from less "bottom-line" oriented AI projects. If AI-based techniques can demonstrate their value in litigation management, then it will be much easier for the corporation to extend them to other legal situations.

Litigation management is also intellectually interesting for domain-specific concerns. Litigation is rule driven. The rules of civil procedure both the timing and nature of litigation events. While each state has its own rules of civil procedure, they are generally patterned after the federal rules of civil procedure. Litigation event objects should be generalizable across legal jurisdictional boundaries. This allows the litigation management system to be a common language between lawyers working in different jurisdictions. This is especially valuable for corporations who must litigate similar problems in different states.

What is Litigation Management?

While all large law firms, corporations, and insurance companies practice litigation management, it is done in an ad hoc fashion. In particular, it is not done in a manner that addresses the communication needs of litigation that is spatially and temporally distributed. Even when specific litigation facts are accessible from a central computer, these facts are often difficult to interpret because they are stripped of their procedural context. With a properly structured knowledge representation system for litigation information, litigation facts will be embedded in the larger context of strategic information about both its individual lawsuit and the larger universe of the entity's overall litigation.

Rationalization of Individual Case Decisions

In individual case-oriented decision making, each legal decision is made without reference to the impact of this decision on other cases or on the corporation's general strategic plan. In rationalized decision making, decisions in individual cases are made with reference to their effects on other cases and on long range legal objectives.

Iterative Refinement of Strategy

Iterative refinement is the converse of individual case rationalization. In iterative refinement, the results of legal actions taken in individual cases are captured, encapsulated, and used to refine or restructure the overall corporate legal plan. The refinement or restructure of the Smalltalk-80 image will consist of building broad categories of classes where instances of the objects will have flexible protocols to dynamically interact with other objects within the corporate legal plan modeled environment.

Capture of Management Information

Much of the effort in modern litigation is consumed in discovery proceedings. Adverse litigants extensively examine the corporation's files for information to support their allegations, while the corporation seeks to support its contentions by obtaining information from its opponents. This information is usually buried with the rest of the file when the case is closed. A legal information management system would survey information produced in lawsuits and incorporate it as appropriate into the corporate management information system.
Case Specific Information

Litigation produces detailed information about specific events that is otherwise not economically feasible to obtain. For example, in the process of defending an insured, an insurance company's attorneys will gather detailed information about the insured. This has little relevance in commodity insurance such as automobile insurance, but is extremely valuable in high risk underwriting such as medical malpractice insurance.

In these high risk areas, an insured has a high probability of being sued during his tenure with a given company. If the insurance company can capture information years before a new claim occurs, it can develop a detailed profile of a substantial subset of insurands. The details available in the profiles will be encapsulated and placed into classes with the appropriate development of protocols to implement the model.

Generalizable information

Legal claims provide invaluable warnings about corporate practices or product defects. In addition to information that is unique to the facts in issue, litigation frequently deals with issues such as corporate management practices. The opponent will dissect these procedures through legal discovery procedures, such as: oral depositions, motions for the production of documents, and written interrogatories.

Irrespective of the outcome of the litigation, the opposing litigant will have forced a more thorough review of the procedures in issue than will have been conducted during the drafting of the procedures by the company. This information should be captured as early in the process as possible. By the time a corporation loses a lawsuit, the products or products in issues will have been accumulating liability for several years. The corporation can use this information as part of its ongoing review of management procedures and to update the AI model.

Why Smalltalk?

Why Smalltalk, or more specifically... Why Object Oriented Programming? The use of objects to model the real world is derived from our observation that "Humans process and categorize their environment using entities." Humans interact with the environment and "learn" that there exists entities which have both properties, attributes, and functions. Human observations are encapsulated into integrated packages which are in essence, objects.

The interactions of properties and functions on one entity to another entity, from a human perspective, map well to the interactions of objects to objects provided by Smalltalk. Smalltalk's rich environment allows for objects to exchange messages and to interact with another object's methods. Entities in the human world, as a state of being, have attributes and properties. In Smalltalk, the objects have memory and remember their respective states pertaining to their attributes and properties.

The use of Smalltalk facilitates the modeling of the real world for legal expert systems because in concentrates on the Concept of the problem, not the Construction of the problem. The use of Object Oriented Programming will cause the user to be more concerned with what are the object's interactions and how observed interactions can be simulated and modeled instead of concentrating upon how to implement a particular data structure or protocol.

The Concepts are stressed, not the Constructs. This argument is why Object Oriented Programming is a move in the right direction for better systems. An example of this issue is, users do not need to know how stacks are constructed by the language, but need to understand the Concept of the stack's LIFO characteristics to use the tool effectively.

The system is being developed upon a IBM Model 70, operating at 20 MHz with 10 megabytes of RAM, a hard drive with 120 megabytes and a VGA 8514 display. The software development environment is ParcPlace's Smalltalk-80 Version 2.5. Additional Smalltalk-80 application software packages includes Xerox's Humble, Xerox's Analyst and Knowledge Systems's Pluggable gauges.

Objects

An Object is the basic computational atom in a pure object-oriented programming language such as Smalltalk-80. Objects may be purely computational entities without real world analogues, or they may be meat-level representations of actual physical objects.

An object is self-contained. It bundles (encapsulates) both data and procedures (methods). The object's private memory contents can only be accessed by procedures which are present in that object's interface. In the Smalltalk-80 development environment, objects contain all representations of information.

The act of creating a new object is accomplished by sending a message to an existing object telling it to produce a clone (instance) of itself. These newly created instances will respond to messages as defined in the progenitor class descriptions.

Abstraction

An abstraction in the Smalltalk-80 environment is the encompassing of the total idea. The metaphors which best describe the concept of abstraction are an "overview" or "executive summary" (i.e. overview of the entire model). This is the starting point to which the development of classes and the objects for modeling will begin. The abstraction may contain essential details, but the hope is for more generalization without the burden of the specific details and the expectation is for a concise description explaining the overall functionality and purpose.

Classes

The collection of objects which respond similarly to the same messages, represent their private memory similarly, define protocols similarly and have similar class hierarchy parentage with regard to their immediate superclass are considered to be members of the same class. These associations could be loosely considered as elements of a set where each element is related. Distinct classes are themselves grouped together into form categories of similar behavior classes.

A grouping of classes into specific related categories is not system enforced but is by convention. The classes are arranged into a hierarchy. The parent of a class is called the superclass and progeny subsequently spawned by a class are called instances of the class and are referred to as subclasses.
The subclasses are more specific and/or refined than their superclass. The subclass inherits protocols from the superclass and whenever messages are received to which the subclass does not have a specific method, the message by convention, unless overridden, will be passed up the hierarchy to allow the superclasses to respond, thus reducing the need to replicate code.

Inheritance
Inheritance is where an object gains the ability to respond to additional messages. Additional capabilities are gained by the object's ability to access methods encapsulated by it's ancestors. The object's ancestor's methods are made available to the object by direct lineage as described by the hierarchy tree. These additional methods are gained by the progeny for free, that is they automatically can respond to messages which their predecessors were able to understand without additional coding. Methods which could be inherited from it's ancestors can be selectively redefined within the object's own methods and the inheritance capabilities of it's spawn would therefor inherit the new capabilities.

Messages
The manipulation of the information contained by an object is accomplished by a message exchange. Messages are objects. All messages and receivers of messages are objects. Messages are causes of actions, which are used by the receiving object to implement a method or a method which it has inherited from it's superclass as defined by the object's hierarchy. The receiving object will pass unfamiliar messages upward to it's superclass if it doesn't understand a message. By proceeding up the tree towards the root to find an ancestor which can respond to the message, the object inherits the capabilities of its superclass. When the appropriate method is involved, the method will respond with it's own message. The message is also an object, and such response is passed back along the tree path invoking additional methods, if appropriate, thus acting recursively within the class's hierarchy.

Encapsulation
Encapsulation is the fractionalization of the overview or abstract into components and is similar to the divide and conquer metaphors used in the top-down software design methodology. Encapsulation is the act of packaging the data and functionality into objects. At this level of software design, the procedures and functions are created to apply elbow grease to the objects. The compartmentalization of the abstract into objects provides protection to the object's private memory components and becomes the interface for object message response. The act of encapsulation produces the collection of methods to which an object can respond. The encapsulation can be likened to creating subsets of methods from a superset of available methods within the object's class and can be designed to modify and/or isolate certain methods to meet specific software requirements.

Methods
Methods are the receivers of messages and are the Smalltalk-80 way to make functions or procedures. Methods are designed to be implemented when appropriate messages are received, and their responses are predefined within the method of the object's class and may trigger or cause the cascading of additional message sends to other objects.

Polymorphism
Polymorphism is the ability of disjoint objects to receive the same message and respond according to their capabilities. For example, an attorney could create a message to query objects about the work that needed to be done on certain data. Each object would respond to this message in a way that is appropriate to its own data.

Multiple Inheritance
Multiple inheritance means that a class inherits properties from two or more classes that are not directly related as superclass or subclass of each other. Multiple inheritance allows objects to become instances of more than one particular class. This relationship allows the object to access methods from both classes, but restrictions can be applied from other instances of single class objects, particularly pertaining to another single class object's private memory.

The Litigation Management System - Implementation
The litigation management system is written in Smalltalk-80 version 2.5 with embedded Humble expert system inference engines. Smalltalk-80 provides the basic development framework objects for the system and Humble provides the objects inference engines.

The Humble expert system consists of a knowledge base which contains a rules base and a facts base. The rules base consists of static rules pertaining to the conditions and inferences upon which conclusions can be reached when using the facts base. The facts base contains information which is dynamic knowledge. The dynamic knowledge is the knowledge which represents facts that are "learned." The "learned" portion of the knowledge base can be changed as a result of conclusions previously reached. The rules base and the facts base together determine how conclusions are deduced and where actions will be directed within the system. The rules and facts bases of Humble use the typical If Then logic approach with numeric certainties being returned for evaluation to deduce a conclusion.

The first inference engine will assist the attorney with the classification of the raw data. The engine will guide the user to enter the data into the appropriate object when necessary. The concept of a forward chaining inference engine will be utilized by this portion of the system to determine how to classify the raw data into a predefined hierarchical structure.

The second inference engine will assist the user in designing meaningful and structured queries about the objects in the database. Examples of system supported queries include: What are the important dates to be considered; What critical missions need to be done today; What are the critical path(s); What options do the users have, if any; What historical inferences are possible; and What projections could be possible. The concept of a forward chaining inference engine is being proposed to implement this second objective of the expert system.

The inference engines are designed to allow the user to access the databases objects with queries that are meaning-
ful. Objects can retain information and generate appropriate responses to any messages. User messages can be sent to an object and objects can respond accordingly. If spurious messages are received by an object, they can be trapped, analyzed and countered with offers of help. The ability to structure meaningful queries on the database of objects, is maintained by the system.

The system will utilize existing objects in Smalltalk-80 and Humble, such as Dictionaries, Collections, Date, Text, Browsers, and the ever present Model View Controller triad. The main Portfolio object belongs to the class Dictionary, and was chosen because of its ability to hash and provide quick access. The Portfolio object will contain Case objects which are made up of the various component objects. The Portfolio object will be the focus of the management type objects for determining the What and Who needs of the second inference engine.

The Case object will be an ordered-collection object and composed of various objects which are component objects. The component objects are exemplified by objects such as, Claim Number (used as an unique number for hashing), Service Date, Amendment Date, Pleadings, Interrogatories, Depositions, Parties, Claim summary, and, Special orders. Component objects will know their respective limitations and operating parameters. Date objects and their descendants, for instance, will know statutes of limitations and appropriate response dates as required by the judicial system. Inheritance and Multiple Inheritance, present in Smalltalk, allow for spawned objects to derive capabilities similar to those of their ancestors, without the duplication of code. The descendent objects also will encapsulate unique methods and properties to accommodate limitations critical to each specific type.

Summary objects with text and their descendants will respond to edit messages by opening Browsers. Browsers allow for the user to both peruse and edit an object's contents. If an object's contents are altered, during the editing session, the object will notify any dependent objects a change has occurred, so updates can be initiated if required.

The design and use of component Case objects allows for the possibility of the future development of a Relational Database schema. The current implementation will utilize a hierarchal Database schema. The polymorphism of Smalltalk allows for greater flexibility in queries because generic querying methods are not dependent upon the data structure of an object. The system will provide capabilities to furnish to the user, multiple views of disparate data.

Query methods operating on Portfolio objects, which contain Case objects, can routinely obtain grouping of all Case objects with critical thresholds. Case objects, which contain the Date objects, have been designed, through the use of inheritance, multiple inheritance and polymorphism, to present common object message to method interfaces. Query methods are designed to return messages of ordered collection objects.

For example, a management object's method might require a return message to contain the Date type object's matching of appropriate responses and/or matching specific sets of scheduled events.

Responses can be tailored to be specific to the object's class. Such responses could consist of return messages that inform the user that a formal reply is required or needs to be filed by either party, the status of any previously required filing, or alert the user that a critical date has passed. The objects also would know the appropriate responses for action as required or expected within their scope of the judicial system.

For example, an Interrogatories object will know that it must have a response within thirty days. A Deposition object would know that a deposition date needs to be set, parties must be contacted, and can also generate form letters to summon the parties.

Conclusions

Law presents a unique problem in computer based communications. Lawyers and their clients cannot unambiguously communicate legal information because law lacks a standard knowledge representation system. This complicates the management of litigation. More critically, it prevents the step-wise evolution of legal knowledge. We have found the object-oriented metaphor to be useful in modeling legal events. Lawyers, more so than professionals such as physicians who practice under science based paradigms, do not separate data and procedures in their analysis. Computational objects have an intrinsic parallel to legal objects because of their encapsulation of data and methods.

Law presents a rich set of communication problems. This work focuses on litigation because it is conducted under uniform rules throughout most of the United States. Litigation is also interesting as a domain for research in computer-based communications because the current lack of a language for communicating about legal events costs businesses billions of dollars each year. It is hoped that a successful object-oriented litigation management system will pave the way for broader research into legal knowledge representation systems, independent categorization, and tracking systems for litigation events.

Bibliography


