RC 20034 (4/12/95)
Computer Science/Mathematics

IBM Research Report

People Oriented Software Technology, and Its Use in Environmental Reporting

Terry Krueger, George Kurian, Anil Nari, Gustaf Neumann, Ulrich Neumerkel, Stefan Nusser, Peter Reintjes, Andrew Taylor, Daphne Tzoar and Adrian Walker

IBM Research Division
T.J. Watson Research Center
Yorktown Heights, New York

IBM Research Division
Almaden · T.J. Watson · Tokyo · Zurich

People Oriented Software Technology, and its Use in Environmental Reporting

Teresita Krueger, George Kurian, Anil Nair, Gustaf Neumann, Ulrich Neumerkel, Stefan Nusser, Peter Reintjes, Andrew Taylor, Daphne Tzoar, and Adrian Walker

All correspondence to: Adrian Walker
adrian@watson.ibm.com

Keywords: People oriented, specification, Syllog, English application authoring, accountability, surety, knowledge interchange, environmental reporting

Abstract

We describe a software technology that is "people oriented", in the sense that it allows us to:

- specify a task as English syllogisms, together with tables of facts,
- run the specification consisting of English syllogisms directly,
- ask questions in English,
- get hypertexted English explanations of answers,
- automatically fill in business forms, and
- to automatically generate database queries and updates.

In our approach, English words take their meaning from their context, rather than from a separately maintained dictionary and grammar. This makes it easy to write down knowledge with specialized words and phrases, such as "Environmental Protection Agency Form R", and then to run the knowledge directly. The knowledge in a specification is directly compiled and interpreted according to a formal theory of highly declarative knowledge. This eliminates the troublesome and expensive gap that often arises between a specification of a task and a program that is supposed to do the task, by eliminating the program. It is not necessary to know about the theory in order to write and to run specifications.

The technology is used to automatically fill in report forms about chemical usage that are submitted to the U.S. Environmental Protection Agency. One such form has over 300 entries per chemical reported, and there are significant penalties for incorrect entries, both for an organization and a private individual who signs the form. Our technology allows us to click on a form entry to see a step by step explanatory audit trail, showing how government regulations, plus engineering expertise, and data about chemicals, were used to automatically make the entry. Other uses of the technology include the mining of medical databases, business case justification, enterprise modelling, and experiments in knowledge based document routing within an organization.

1 Introduction

We describe some generic software technology that allows us to specify a task for a computer, and then do the task, in a way that is "people oriented". The technology allows a nonprogrammer to write down, as English syllogisms and as tables of facts, the knowledge needed to do a task, and then to directly run the knowledge as though it were a program. In our approach, English words take their meaning from their context, rather than from a separately maintained dictionary and grammar. This makes it easy to write down knowledge with specialized words and phrases, such as "Cupric sulfate, anhydrous", or "Environmental Protection Agency Form R", and then to run the knowledge directly. This flexibility is purchased by following a few simple guidelines when writing a specification.

Syllogisms and facts in a specification are interpreted in a highly declarative way that is based on a formal model theory of declarative knowledge [ABW88,Wal93], allowing the author of the specification to concentrate on writing down just the knowledge needed for a task, rather than programming a procedure to do the task. The knowledge in a specification is directly compiled and interpreted according to the formal theory. This eliminates the troublesome and expensive gap that often arises between a specification of a task and a program that is supposed to do the task, by eliminating the program. It is not necessary to know about the theory in order to write and to run specifications.

Once a task has been specified, we can:

- run the specification consisting of English syllogisms and facts directly,
- ask questions in English,
- get hypertexted English explanations of answers,
- automatically fill in business forms, and

- automatically generate and run database queries and updates.

We describe how the technology is used to automatically fill in report forms about chemical usage that are submitted to the U.S. Environmental Protection Agency. There are many such forms. Other uses of the technology include the mining of medical databases, business case justification, enterprise modelling, and experiments in knowledge based document routing within an organization.

The next Section outlines the technology, and Section 3 describes how it is used in environmental reporting. Section 4 outlines the implementation of the technology, and Section 5 lists our conclusions and directions for further work.

2 Outline of the Technology

As in the case of classical Object Orientation (see e.g. [KRO93]), some key aspects of our technology are

- encapsulation,
- reuse,
- picturing familiar items on the screen
- the use of large databases, and
- inheritance of properties.

However, we take a different approach. In our implemented technology, we

- encapsulate and
- reuse

knowledge that a programmer has about the efficiency and termination of programs. This knowledge is generic. We do not yet try to encapsulate knowledge about real world tasks. Rather, we provide a technology in which these tasks can be specified in English, and in which the resulting specifications can be used directly as programs, optionally over databases. The

- familiar items that we picture on the screen

are English sentences (some of which are hypertext linked), data tables, and business forms. To

- use large databases

we automatically generate and execute SQL [DaD93] queries over relational databases. We provide a flexible way to specify

- inheritance of properties.

For example, this syllogism

some-area is within some-region
some-attribute in that-region has value some-value
the value of that-attribute in that-area is not directly given in its description

---

that-attribute in that-area has value that-value

specifies a kind of inheritance with a local override. In the syllogism, "some-area", "some-region" and so on, are place holders, or variables, that are filled in consistently with actual values (such as "New Jersey" and "East Coast"), when the syllogism is used. The syllogism says that, if the three premises above the line are true, then so is the conclusion.

Our approach provides *explanations* of the results of inheritance and override [Wal91]. Indeed, it provides English explanations of all results, and even of what would be needed to get a result that is not forthcoming. These explanations are automatically hypertexted.

Encapsulation and reuse of programming knowledge is done by raising logic programming, see e.g. [WMSW90], to a higher declarative level, based on a theory of declarative knowledge [ABW88, Wal93]. (While the theory, and the inference engine based on the theory, are highly technical, it is not necessary to know about them in order to specify tasks for our system.)

Support for this approach is built into a generic system called Syllog. (Pronounced as in "syllogism", but with a hard "g".) The system executes task specifications that translate to the class "stratified datalog with recursion and negation as failure" (SDRN), plus certain kinds of numeric specifications. By way of example, multiple inheritance with override translates into SDRN, while critical path scheduling translates into the class of numeric specifications that Syllog executes directly. Conventional programs can be added to a Syllog specification by "procedural attachment". Successive versions of Syllog increase the kinds of specifications that are supported without the need for such attachments.

We believe that current software engineering approaches to task modeling and programming can leave an expensive and error-prone gap to be filled between the specification and the program. In our approach, the specification *is* the application program. We believe that the approach can significantly reduce software costs while improving life cycle flexibility and quality.

3 Using the Technology in Environmental Reporting

In Section 2, we outlined our technology. In this Section, we describe how our technology is actually used in a complex real world task, namely to prepare reports on chemical use and storage that organizations in the United States must send to the Federal and State Environmental Protection Agencies.

3.1 The Environmental Reporting Task

The Environmental Protection Agency (EPA) of the United States government requires all organizations over a certain size to report the chemicals that they use. Dangerous chemicals must typically be reported even if used in small quantities. Less dangerous chemicals must be reported if the amount stored or used is over a certain threshold. Different forms of a chemical (e.g. solid, powder, solution), and different chemicals, have different thresholds. The amount of each pure constituent of a mixture of chemicals must be reported. The hazards associated with each chemical, such as fire, sudden release of pressure, and reactivity, are part of the report. EPA standard codes must be used for items such as amounts, temperature, pressure, type of storage container, and the location of chemicals within an organization.

Most reports must be made on forms whose layout is specified by the EPA, or on the electronic equivalents of such forms. One such form, EPA Form R, has 9 pages and over 300 entries, for *each* chemical reported. The EPA instructions for filling in the form [EPA94] are approximately 120 pages long. A manufacturing organization may have hundreds, or even thousands, of potentially reportable chemicals. The forms must be filled in according to EPA regulations, and the organization submitting the forms is subject to audits to make sure that the regulations have been followed. An environmental engineer normally has expertise about the regulations and about chemicals, and he or she applies this expertise in filling in the forms, and in keeping notes for audit purposes. Technical errors on the form expose the organization to significant financial penalties. In addition to the penalties that apply to the organization, an executive who signs a form containing errors may also be *personally* exposed to fines, and, in extreme cases, a prison sentence.

3.2 Reasons for using People Oriented Technology for the Task

There are several ways in which conventional software engineering can fail to close the gap between a task description and a program that is supposed to do the task. For the reasons just described, it is particularly important in real environmental reporting to close the possible gap between the EPA regulations and any procedures or software that may be used to help in filling in the forms. The gap can arise because someone has

- misinterpreted the EPA regulations,
- placed a correctly computed entry in the wrong slot in a form,
- made a programming mistake, or
- someone has misunderstood the layout or meaning of the organization's data about chemicals.

Even if none of these happen, it may be hard to reconstruct how a form entry was arrived at in the event the organization is audited.

We have used our people oriented Syllog technology to fill in and submit to the EPA several kinds of environmental report forms, including Form R. The technology tends to close the gap between the task description and a program in the following ways.

- We write down the EPA regulations, together with the expertise that an environmental engineer uses to interpret the regulations, as executable English syllogisms. (In the case of Form R, the syllogisms are about one quarter the length of the EPA manual containing the regulations.)
- We *generically* fill in a form, by "drag and drop" of place holders, such as some-chemical or some-year, from sentences in the syllogisms into slots in the form. Thus, we link the knowledge about the EPA regulations to the slots in the form just once, and then we reuse the links (with no further effort) to fill in as many forms as are needed.
- As we write the specification, we run it and we ask for explanations of the answers that we get. Also, when the specification is completed, and is used to fill in a form, we ask for explanations of the entries on the form, and we examine these explanations step by step. This allows us to check, more easily than with conventional programming, for mistakes, and for misunderstandings about the layout or meaning of the organization's data about chemicals.

In addition to direct use, our system is a useful tutorial tool. We have found that people who are not experts in EPA regulations, in chemical engineering, in programming, or in the layout or meaning of the organization's data, can usefully run a specification to understand the regulations, expertise and facts that are used to fill in an EPA form.

3.3 Using the Technology to Fill in EPA Form R

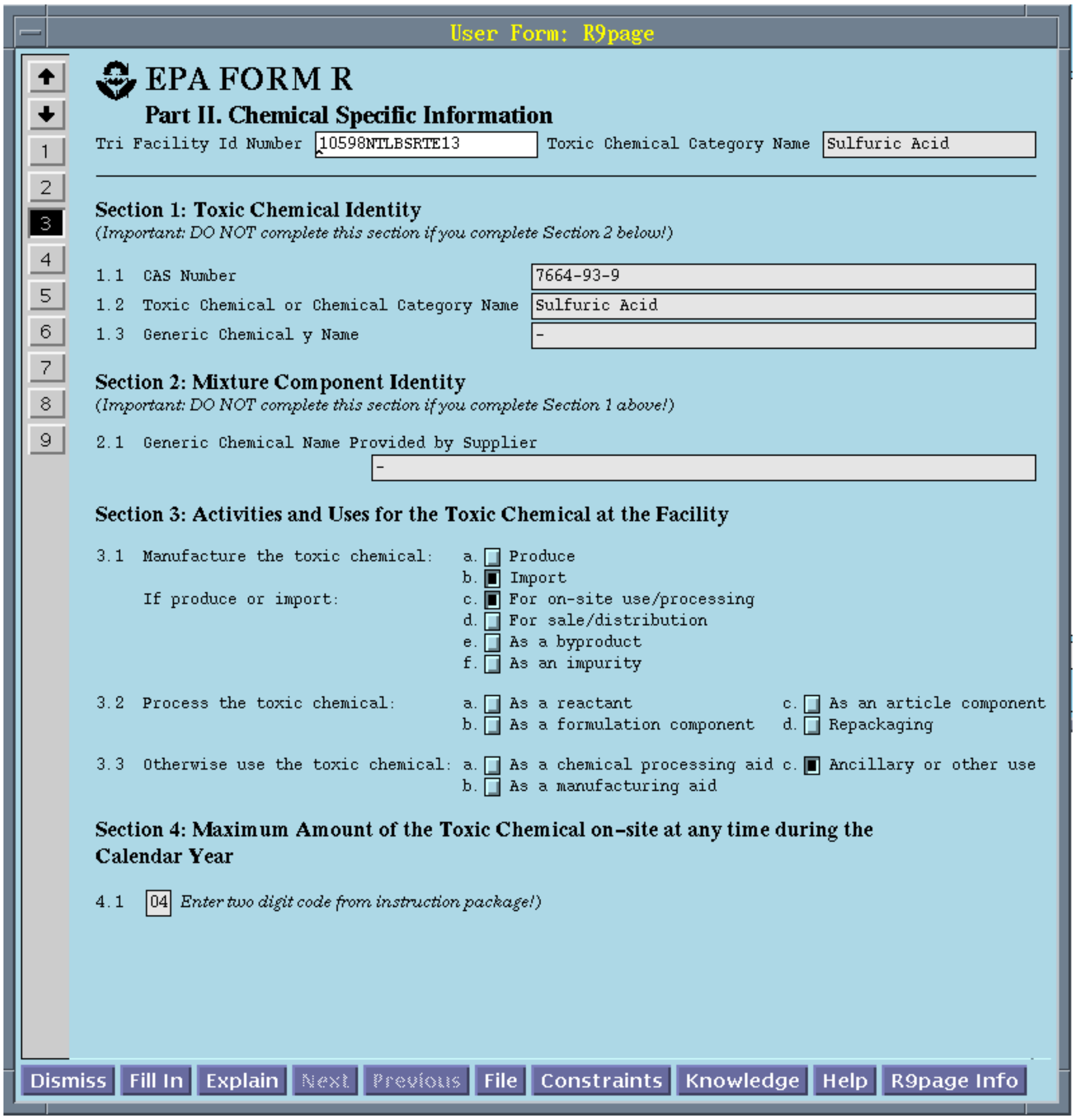
EPA Form R is a 9 page form that must be filed each year, for each chemical for which more than a certain amount has been used or stored. (Form R is also known as the SARA 313 form.) Pages 1 and 2 of the form ask for information about the organization, such as: address, map coordinates, 24-hour emergency contact information.

When completed, page 3 of our onscreen version of the form is as shown in figure 1 on page 8. The page shows that Sulfuric Acid, which has a "CAS" number 7664-93-9, was imported, and used for on-site processing. The maximum amount on-site at any time during the year was coded, according to the EPA regulations, as 04.

Pages 4 to 8 of Form R ask about amounts released to the environment on-site, discharges to publicly owned water treatment works, transfers to offsite treatment locations, on-site waste treatment methods and their efficiency, and on-site energy recovery and recycling. Page 9 asks for some of these items to be compared to their amounts in the previous year, and for planned amounts for the next two years.

On an IBM RS6000 model 350 workstation, filling in the 9 page EPA Form R for one chemical, using small amounts of data, takes approximately 1 minute if a relational database is not used, and about 3 minutes if the tables of data are stored in the DB2/6000 database management system. (We expect a crossover as the data tables increase in size, since the overhead of using the DBMS should be balanced by faster data processing.) The completed form can be printed by clicking on the File button and then selecting from menus.

A person responsible for signing the form might reasonably ask why Sulfuric Acid has to be reported, and about the meaning of the code 04, on page 3 of the form, for the maximum amount on-site. To see a step by step explanation, we can click on the 04 entry in the form, then on the Explain button at the foot of the form. The first step of the resulting hypertext explanation is as shown at the top of page 9.



User Form: R9page

EPA FORM R

Part II. Chemical Specific Information

Tri Facility Id Number 10598NTLBSRTE13 Toxic Chemical Category Name Sulfuric Acid

Section 1: Toxic Chemical Identity

(Important: DO NOT complete this section if you complete Section 2 below!)

1.1 CAS Number 7664-93-9

1.2 Toxic Chemical or Chemical Category Name Sulfuric Acid

1.3 Generic Chemical y Name -

Section 2: Mixture Component Identity

(Important: DO NOT complete this section if you complete Section 1 above!)

2.1 Generic Chemical Name Provided by Supplier -

Section 3: Activities and Uses for the Toxic Chemical at the Facility

3.1 Manufacture the toxic chemical: a. Produce; b. Import

If produce or import: c. For on-site use/processing; d. For sale/distribution; e. As a byproduct; f. As an impurity

3.2 Process the toxic chemical: a. As a reactant; b. As a formulation component; c. As an article component; d. Repackaging

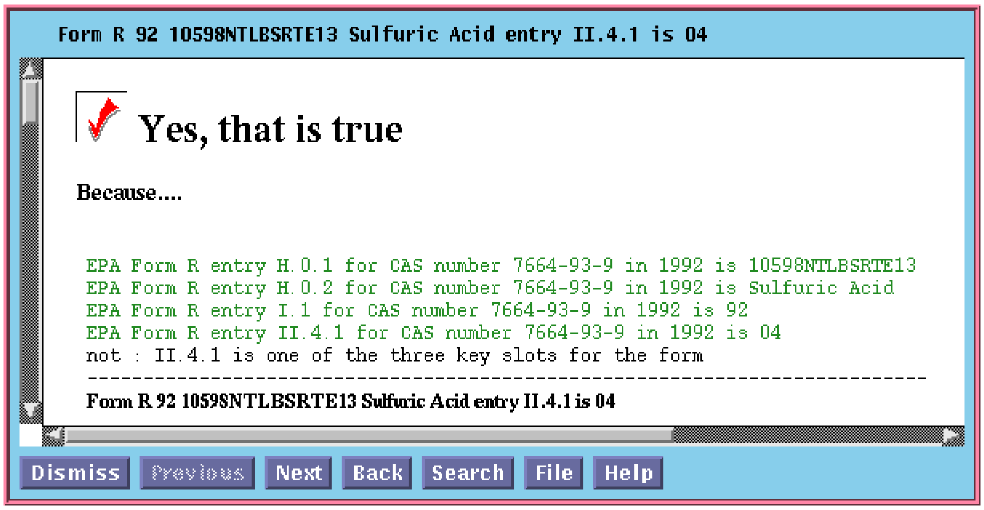
3.3 Otherwise use the toxic chemical: a. As a chemical processing aid; b. As a manufacturing aid; c. Ancillary or other use

Section 4: Maximum Amount of the Toxic Chemical on-site at any time during the Calendar Year

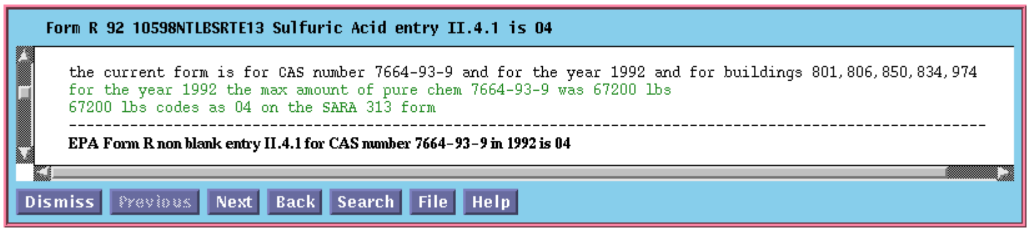
4.1 04 *Enter two digit code from instruction package!)*

Dismiss | Fill In | Explain | Next | Previous | File | Constraints | Knowledge | Help | R9page Info

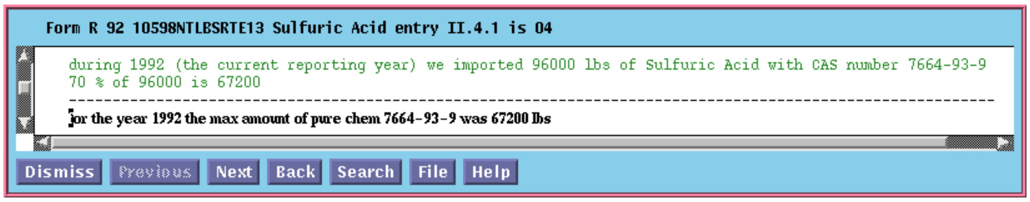
Figure 1: Page 3 of the completed form



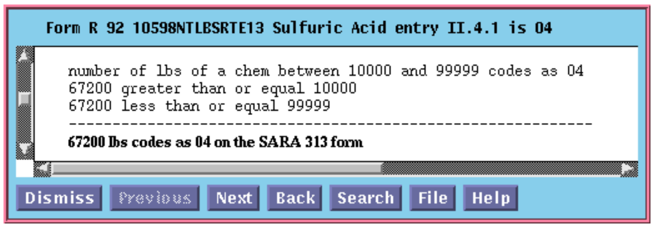
This step consists of 5 premises followed by the conclusion that the entry in slot 4.1 in Part II of the form is 04. On the screen, the first 4 premises are shown in green, indicating that further explanation is available for each of them. To go into more detail in an explanation, we can either just scroll downwards, or we can click on a premise shown in green to make a hypertext jump directly to see how the premise is justified. So, clicking on the 4th line of the explanation gets us to the step



If we click on the second line of this step we see



that we have estimated the maximum amount on site as 70% of the amount we imported. Going into further detail in the explanation would show that we were importing quaterly, and using up the chemical continuously during the year. However, we might reasonably be concerned about the 70% estimate, depending on how the final code 04 on the form is arrived at. To see this, we can hypertext jump to



which tells us that we are comfortably near the middle of the range of 10,000 to 99,999 lbs that the EPA requires us to code as 04.

Other parts of the hypertexted explanation allow us to check when we accepted deliveries of the chemical, and that we had a reportable amount of it according to the EPA rules.

3.4 Writing the Specification to Fill in EPA Form R

We mentioned that the EPA regulations for Form R [EPA94] consist of about 120 pages of English text and tables, and that our English executable specification is about one quarter of that length. Environmental engineering expertise is needed to interpret the regulations so as to fill in the form correctly. Ideally, an environmental engineer, who need not know any programming, writes down his or her expertise directly as a specification, and checks the specification by running each part of it in Syllog while writing it. This is possible for simple environmental reporting tasks, such as filling in the three page New York State Hazardous Waste form known as tp-550. In writing the specification for the much more complex Form R, an environmental engineer worked with a person experienced in writing specifications that run in Syllog. (This was necessary in part because the underlying Syllog system and its online help were themselves under development at the time.)

The main steps in writing the specification were:

- extraction of tables of data from the EPA regulations, for example

number of lbs of a chem between this-min and this-max codes as this-code
===

10000000	*999999999999*	*07*
1000000	*9999999*	*06*
100000	*999999*	*05*
10000	*99999*	*04*
1000	*9999*	*03*
100	*999*	*02*
0	*99*	*01*

The table says that a number of lbs of a chemical between 100 and 999 codes as 02, and so on.

- gathering tables of data about the site facility, for example

our facility engaged in source reduction this-type activities in this-year
===

Modified equipment layout piping	*1992*
Other	*1992*

- understanding the EPA regulations and writing them as syllogisms, for example

the current form is for CAS number some-CAS and for the year some-year and for buildings some-names
some-chemical , with CAS number some-CAS , is reportable under SARA 313
during that-year we imported some-number lbs of that-chemical with CAS number that-CAS
SARA 313 report threshold is some-threshold lbs of that-chemical
that-number greater than that-threshold

---

in that-year we imported that-number lbs of that-chemical , with CAS number that-CAS , so a SARA 313 report is required

Here, "CAS number" and "SARA 313" are jargon phrases. They take their meanings directly from the way they are used in the syllogism. In many other technologies, one would have to define the meanings in separate dictionary.

- noting EPA regulations about Form R itself, for example

EPA Form R this-slot should be filled in with this-entry rather than left blank

==	
II.6.2.1.A.2	*NA*
II.6.2.1.A.3	*NA*
....	...
II.8.9	*NA*

- writing knowledge about coded form entries and the slots they belong in on the form, for example

the current form is for CAS number some-CAS and for the year some-year and for buildings some-names

for the year that-year the max amount of pure chem that-CAS was some-number lbs

that-number lbs codes as some-code on the SARA 313 form

———————————————————————————————————-

EPA Form R non blank entry II.4.1 for CAS number that-CAS in that-year is that-code

- testing to make sure that all the form entries could be found as a simple table of answers by running the specification in Syllog.

- writing special syllogisms to be used when we generically drag and drop answers from the table into slots in the form

Form R some-year some-FID some-chem-name entry II.4.1 is some-maximum-code

———————————————————————————————————

Form R9page "II.4" : that-FID that-chem-name that-year

- writing the blank Form R for display on the screen using a simple extension of hypertext markup language [BCG92]

- using Syllog screens to couple the knowledge in the specification to the form, for example by dragging some-maximum-code from the sentence

  Form R some-year some-FID some-chem-name entry II.4.1 is some-maximum-code
  to the correct slot on in the form.

These steps were interleaved with one another, and with the testing of each item in Syllog as soon as it was written. While the whole process took several person-months, we estimate that doing similar tasks in future will take much less time and effort. This is because (a) the Form R specification now exists, and needs only to be adapted to changing circumstances (b) the underlying Syllog system is stable and has a useful set of online help screens, and (c) our skill levels in understanding EPA regulations and how to write them as runnable specifications are higher, and we can easily transfer these skills.

3.5 Automatically Generated Database Queries for Filling in EPA Form R

The Syllog system can be used without an underlying database management system when the data tables are small. When the data tables are large, or when a pre-existing relational database is to be used, Syllog automatically generates and executes queries in the SQL language [DaD93]. For example, if the table

number of lbs of a chem between this-min and this-max codes as this-code

is in a database instead of in the specification, then we write the syllogism

ext : cims.daphne.codesas MIN MAX CODE
———————————————————————————
number of lbs of a chem between this-min and this-max codes as this-code

in the specification. The syllogism says that there is an external database called "cims", containing a table belonging to "daphne", and that the table has the name "codesas" and the columns MIN, MAX, and CODE. It also says that the meaning of a row of this data table is that a number of lbs of a chemical between some min and some max codes as a certain code on Form R.

If the data are stored in a database, and are coupled into the specification by syllogisms like the one above, then the current version of the Syllog system makes about 250 SQL queries, using 50 tables, to fill in Form R for one chemical. This includes a number of data definition level queries such as

```
select NAME from sysibm.systables
where NAME = 'CODESAS' and CREATOR = 'DAPHNE'
```

that check to make sure that the necessary tables are present in the database. Among the queries that are actually used to fill in the form, some are as simple as

```
select distinct "CHEM", "CAS", "NAME"
from DAPHNE.GENERICNAME GENERICNAME
```

while others, like this one

```
select distinct 1992, X."NAME", '7664-93-9' from
   DAPHNE.CURRENTFORM W,
   DAPHNE.I313ISX X,
   DAPHNE.PURCHASED Y,
   DAPHNE.TOCONVERT Z,
   DAPHNE.GENERICNAME AZ,
   DAPHNE.S313REPOTHRESH BA,
   DAPHNE.ONSITECHEM BB
    where
     W."CAS" = '7664-93-9' and
     W."YEAR" = 1992 and
     X."CAS" = '7664-93-9' and
     X."NAME" = AZ."NAME" and
     X."NAME" = BA."NAME" and
     Y."UNITS" = Z."UNITS" and
     Y."CHEM" = Z."CHEM" and
     Y."CHEM" = AZ."CHEM" and
     Y."YEAR" = 1992 and
     AZ."CAS" = '7664-93-9' and
     AZ."CHEM" = BB."CHEM"
       group by Y."CHEM", Y."UNITS", Z."FACTOR", BA."THRESHOLD", X."NAME"
         having (sum( Y."AMOUNT") * Z."FACTOR") > BA."THRESHOLD
```

are more complex.

We emphasize that all of this SQL database activity is automatic, and is encapsulated inside the Syllog system. The specification author, and the user of a specification, never need to see any such activity. Rather, they interact with the system at a business level, via English and business forms, rather than a technical SQL level.

4 Implementation of the Technology

The Syllog technology has been implemented through about 10 major versions, over a number of years. Some design questions that were raised and solved in successive versions are described in [Wal93], [ABW88], [FRTW88], [WMSW90]. Early versions of Syllog were on an IBM 3090 mainframe, to take advantage of local databases. The current version runs on IBM RS6000 workstation under AIX, IBM's version of the Unix operating system. It automatically generates SQL queries and updates and executes them either locally or remotely. The database management systems supported are Oracle, and IBM's DB2/6000, which can in turn run SQL operations on DB2 on MVS and on SQL/DS on the VM operating system.

The branching to different DBMSs at the backend of Syllog is mirrored by a branching to different display screens at the front end. The Syllog user interface uses Xwindows and the Athena widget set, and we plan to replace Athena widgets by Motif (See e.g. [Bark91]). Since Xwindows/Athena/Motif is virtually platform independent, it is straightforward to display the Syllog user interface not only on Unix screens, but also on Windows, OS/2, and Mac screens. Conceptually, Syllog links business level English, forms, rules, and questions to system-level SQL databases. Architecturally, Syllog links most modern display screens to DBMSs from different software vendors.

The sophisticated behavior of the widgets that is needed for the Syllog user interface is programmed in the language TCL [Ous94] for the Wafe [NeN92] user interface package. The business forms at the Syllog user interface are written in a form description language that is mapped, via a simple generator, into Hypertext Markup Language [BCG92]. The Syllog system consists of TCL code, for the user interface, and of about three times as much Prolog code, for deduction (using a backchain iteration method [Wal91] that is different from Prolog's built in method), for explanations, for menu filling, and for a sophisticated Syllog-to-SQL compiler. From another viewpoint, one can say that the system is essentially implemented in C, using some extensions (TCL, Prolog) that are themselves in C. The TCL part of Syllog is interpreted at runtime, whereas the Prolog part is compiled when a Syllog runtime package is built. A complete Syllog runtime system occupies approximately 10Mb of disk space, plus the space that is needed for task specifications, and for the DBMS and its data.

5 Conclusions and Directions for Further Work

We have described a software technology that is "people oriented", in the sense that it allows a task specification to be written as English syllogisms and as tables of facts, and then allows us to run the specification directly to do the task. We have shown how this technology eliminates the expensive and troublesome gap that can arise between a task specification and a program that is supposed to do the task, by eliminating the program. As in the case of classical object orientation, our technology supports encapsulation, reuse, the picturing of familiar items on the screen, the use of large databases, and inheritance of properties. At present, we encapsulate and reuse programming knowledge, so that a task can be specified rather than programmed. An interesting direction for further work is to specify metaknowledge about how to reuse object level specifications, e.g. for reasoning about inheritance with overrides or about time. Some initial work in this direction is in [Lel91].

Our people oriented technology is used in environmental reporting, a demanding task for which engineering judgement and expertise are needed. Our English specification of the knowledge needed to fill in EPA Form R is about one quarter of the length of the corresponding EPA regulations, yet it is precise, readable, and executable. When the specification is used to fill in the form, we can also use it to generate a hypertexted explanation for each entry. An explanation serves as a trail, through the EPA regulations, environmental engineering expertise and the knowledge about the meaning of the organization's data, that justifies the entry in the form for audit purposes. If the data are stored in a relational database, the system generates approximately 250 SQL queries in order to fill in Form R. Some of these SQL queries are complex, and it would be a difficult task to use them to justify a form entry. Rather, their generation and use is encapsulated in the Syllog system, and explanation is at the business level.

In writing a conventional program, one is concerned not only with doing the required task correctly, but also with doing it efficiently. In addition one must deal with questions of data structures and storage management. It can be argued that some of the difficulty of large, conventional programming projects stems from the fact that changing a program to make it more efficient can cause it to do a different task from the intended one, and that the difference can be hard to detect. The difficulty is compounded if – as is usually the case – the task description leaves details to be filled in, and changes during programming.

The present Syllog system encapsulates many optimizations that seek to run a specification efficiently, whether or not the specification uses a database management system to hold tables of facts. So, the system is efficient on many real examples. One specifies a task, rather than saying how the task is to be done. Running the specification efficiently is the responsibility of the system implementers, rather than a problem for the programmers of each task. As is the case with database management systems, there is further work to be done on Syllog on the question of generic efficiency over families of tasks. Meanwhile, if a conventional program is needed for efficiency, it can be procedurally attached to a Syllog specification.

The user interface of Syllog, together with the supporting software, allows us to write a task specification almost in English. When we run a specification, we can ask questions in English, via menus, or via business forms. We get answers to questions as tables, as business forms, or as business charts. In the case of tables and business forms, automatically hypertexted explanations of results are available as needed. We believe that this people oriented software technology, and its future extensions, can make a major contribution to the accountability and surety of information systems, and that it can also contribute to the growing field of methods for interchanging executable knowledge.

6 References

[ABW88] Apt, K., H. Blair and A. Walker. Towards a Theory of Declarative Knowledge, In: Foundations of Deductive Databases and Logic Programming, J. Minker [Ed.], Morgan Kaufman 1988.

[Bark91] Barkakati, N. "Unix Desktop Guide to X/Motif", Hayden, 1991.

[BCG92] Berners-Lee, T.J, R. Cailliau and J.-F. Groff, The World-Wide Web, Computer Networks and ISDN Systems 25 (1992) 454-459. Noth-Holland.

[DaD93] Date, C. J., and Hugh Darwen. A Guide to the SQL Standard, Addison-Wesley, 1993.

[EPA94] "Toxic Chemical Release Inventory Reporting Form R and Instructions, Section 313 of the Emergency Planning and Community Right to Know Act, EPA EPA 745-K-94-001", U.S. Environmental Protection Agency, Washington D.C. .

[FRTW88] Foo, N., A. Rao, A. Taylor and A. Walker. Deduced Relevant Types and Constructive Negation. Proc. Fifth International Conference Symposium on Logic Programming, Seattle, Washington, 1988, 126-139.

[KRO93] Kroha, P. Objects and Databases, McGraw-Hill, 1993.

[Lel91] Lell, C. Using a Meta-knowlege Method for Developing an Educational Knowledge Based Application. Proc. 2nd Int. Conf. on Database and Expert System Applications – DEXA '91, Berlin, Springer Verlag, 1991.

[NeN92] Neumann, G. and S.Nusser. Wafe - An X Toolkit Based Frontend for Application Programs in Various Programming Languages, USENIX Winter 1993 Technical Conference, San Diego, California, January 25–29, 1993

[Ous94] Ousterhout, J.K. "Tcl and the Tk Toolkit", Addison-Wesley, 1994.

[Wal91] Walker, A. Position Statement on The Direction of Object-Oriented Technology in the Marketplace. In: "Object-Oriented Databases: Analysis, Design and Construction", Meersman, Kent and Khosla [Eds.], North-Holland, 1991.

[Wal93] Walker, A. Backchain Iteration: Towards a Practical Inference Method that is Simple Enough to be Proved Terminating, Sound and Complete. Journal of Automated Reasoning, 11:1-22. 1993.

[WMSW90] Walker, A., M. McCord, J. Sowa and W. Wilson. "Knowledge Systems and Prolog: A Logical Approach to Expert Systems and Natural Language Processing", second edition, Addison-Wesley, 1990.

7 Acknowledgements

In addition to the people whose work we have cited, many others have contributed to the Syllog system, and to its use in environmental reporting. We would like in particular to thank the users of the Syllog system for their patience when early versions of the system were less than user friendly, and for their many suggestions for improvements.