A Plant Documentation Information System Design

Dr Bing Wang
Computer Science Department
University of Hull
Hull, HU6 7RX, UK

Ma Feicheng
Center for Studies of Information Resources,
Wuhan University
Hubei Wuhan, 430072
The People's Republic of China

Abstract

Traditional systems do not have the descriptive ability to represent the complex, multi-faceted nature of complex documentation structures such as engineering systems descriptions. This has led to significant information management difficulties in maintaining documentation which reflects, in a consistent and up-to-date way, the current state of the documentation systems. The formal method has been widely recognized as a precise way to define the structure of a complex documentation system. In this paper, a formal approach to model petrochemical plan information is presented. The motivations of using formal methods in the design of a heterogeneous documentation management system are described. The paper focuses on the issue of using formal specification to clarify our understanding of the problem and as a basis for the design of a multi-purpose documentation supporting tool, which can link different databases and provide users different modelling methods to support the operations of complex document structures.

1. Introduction

Emerging technologies such as distributed and object databases, client-server architectures, electronic data interchange (EDI) and information standards such as STEP and SGML, hypermedia and open document architectures offer potential solutions. Although methodologies such as SSADM and Yourdon exist for designing and managing large-scale information systems, these do not provide a mechanism for integrating these different technologies into an overall architecture to support the heterogeneous information found in a petrochemical plant environment. Also, due to mainly the absence of suitable information and documentation standards, it has been difficult to develop open configurations with adequate accommodation for maintenance and extensibility. Petrochemical plants have inherent economic and personal risks associated with their operation. New York Times of 19th June 1991 listed 14 petrochemical disasters from 1987 to 1991 in the USA alone. Figures for these incidents were 7 dead, 138 injured and a cost of $246 million. A root cause of these incidents leading to personal and economic loss is the lack of a trusted means for operation staff to access up-to-date, consistent information about the plant's overall operational status and procedures for operating it. Due to the inherent inflexibility of

DOI : 10.5121/ijcsit.2010.2510

traditional documentation systems, documentation may be inaccurate with different versions containing inconsistent plant details. As a result operational staff may not be well informed about the current plant status, and different individuals may have conflicting views of the plant. This can lead to a lack of awareness of potentially hazardous developments or actions, and the consequent development of a hazardous situation that could have been avoided.

In the following sections, I will first study the current situation of petrochemical plants described in section 2. Section 3 explains objectives of this industrial project, its main deliverables and guiding principles. Section 4 describes the formal specification of the integrated documentation supporting tool and associated design methodologies on the basis of the defined designing principles. Chemical documents are unique both in their formats and structures. There are few researches in this area, in particular the documents about the system design. Thus, in section 5, we will present some related research which are relevant to our research in order to provide readers a broad view of documentation supporting systems. Finally, in section 6, we conclude the research by summarizing the benefits of the project.

2. Chemical Information Systems ---Background Review

The current installed technology architecture used within the petrochemical industry to manage and control chemical processes has limitations relating to the storage, maintenance and management of plant information and documentation, where a particular issue is the non-use of information standards; the technology of the distributed control system (DCS); and human factor issues relating to staff access, presentation and application of plant information.

2.1 Management and Storage

Plant documentation and information should be viewed as a resource for supporting the information and decision-making needs of all those involved in the plant. If this is to be the case, then the information must be managed over the whole lifetime of the plant - from design through to operation, with the plant design information and documentation formatted and structured with the requirements of specific operational staff and tasks in mind. It should also be of sufficient quality to inspire confidence and ensure that it is used. Unfortunately documentation produced during design activities is not always written to address the specific needs of operational staff, and it tends not to be structured or presented in a form suitable for ready operational access. For traditional paper-based documentation, inherent access and indexing limitations mean that relating a large documentation set to a specific plant situation may not be easy. For example, following procedures defined in paper form can be difficult because of the need to use complex indexing, documentation may not be directly on-hand, and the relationship to displayed plant details on the operator console may not be immediately apparent.

A further point is that the management and maintenance of plant documentation is often inadequate, with poor accommodation of on-going changes to the plant and its mode operation. This leads to incompatibility between different document versions, and between the documentation and the actual plant. This is in part due to plant documentation and operating procedures not in general being in an easily updateable and accessible form such as that provided by electronic document storage. Ideally plant information should be stored without duplication

in a form which is accessible to all operational staff, with no copying by operational staff of this information permitted.

For operators, these information management and storage limitations have two important consequences. Firstly, the operator usually has no mechanism for ensuring that work procedures are kept up to date and consistent. Secondly, there is poor feedback of operational experience into the documentation updating process. When an operator spots an inconsistency in a procedure, its existence needs to be recorded for incorporation in the subsequent version of the procedure documentation. The absence of a single mechanism for recording such observations to a central point so that they are quickly dealt with can lead to unacceptable delays between the inconsistency being noted and the wider awareness of its existence.

For example, start-up procedures are with increased operating experience. Operator experience should be an on-going aspect of such revisions. Unfortunately there is often no record of ongoing notes and activities on which to base revisions to the procedures. One aspect of this is the lack of a mechanism for operators to log and share comments on current operational experience between themselves and between supervisory staff. Shift notes alone are inadequate - a general system log which is generally accessible and which allows the accumulated knowledge to be incorporated into the plant documentation is required.

A further problem is that documented procedures and operating guide-lines are given in terms of tag numbers. However, present documentation provides inadequate cross-referencing between the process display and the written procedures. Also, inadequate use is made of diagrammatic forms. For example, the documented source should be able to create or call a relevant schematic diagram. This tends to lead to the specification of functional relationships and an overly simplistic parameter representation of the plant provided by traditional operator consoles.

2.2 Problems from Non-Use of Standards

Due to the lack of appropriate information standards in the past, sharing of information between computer-based stores of plant information has been impractical. The result is that it is impossible to maintain consistency between different versions of stored information entities. There are no present standards for conversion from electronic forms of data or operating procedures. Non-adherence to standards would have deleterious implications for the cross-domain information integration and access needs of this project. To support the multi-faceted information presentation, proper attention must be paid to the available and emerging standards in order to ensure consistent search capabilities across different databases containing different types of information. At present search procedures are not compatible across information-domains. Also the ability to extend an implemented decision support system to accommodate changing plant requirements would be more difficult. A readily extendable architecture requires the use of recognized standards both to store the information in a non-proprietary form which can be easily updated over the lifetime of a plant, and to connect together the various parts of the overall system which, for practical reasons, needs to be of a distributed nature.

2.3 Limitations of the Distributed Control System

The distributed control system (DCS) is the control system that is designed to provide the interface between the physical plant and the operators who control it. It includes the instrumentation circuitry for displaying the plant variables as well as the remote-control facilities for allowing the operator to change plant variables. The main purpose of the DCS system is to provide a platform for process control and status monitoring; there is an attempt to include some internal capability to monitor specific logical combinations of alarm states.
However, decision support assistance within the DCS platform is essentially restricted to the text and logical combinations configured into the system database at the time of the project engineering. This has to be maintained using proprietary DCS systems tools.

As plants become more complex and operating needs more stringent, the DCS and plant information systems must become more integrated. However, the present DCS architectures are limited in their ability to incorporate operating manuals and procedures. Where decision support details are included in the DCS, the inflexibility of the architecture means that updating is time consuming. Also, the closed nature of the technology architecture along with its complexity means that ready access to this information is difficult and so this lack of transparency has the effect of reducing operator confidence.

3. Research Objectives and Deliverables

The research aims to define the overall requirements for a flexible, computer-based decision support framework for operational staff to access plant documentation in a timely and useful format. A central part of this framework is a mechanism for safe maintenance of this information by entering new details or by modifying existing records over the plant lifetime. In order to both help develop these techniques and to show how an installed on-line system would benefit plant operators, an integrated documentation system is implemented. There are three main aspects to the work:

3.1 Objectives

1. First, in order to develop, maintain and extend in a phased way such support systems, an information model and associated tools to structure the information and maintain its quality are necessary. A supporting architecture based on standard database and integration concepts along with recognized electronic documentation and information standards will be used to construct the prototype information model; this should illustrate the consistent integration of engineering information necessary to describe the plant and its operation. Such plant information may include operating instructions, procedures and guide-lines, chemical hazard details, geographical installation information and current work-permit allocations in the form of tables, text, images and diagrams. Information and quality management issues will be considered with reference to recognized standards.

2. Secondly, description of user-oriented presentation mechanisms are required which provide ready access to the stored information for decision support tasks.

3. Thirdly, the development of a prototype user-oriented decision support system to demonstrate from a practical perspective the increased value of electronic management and retrieval of plant information stored in standard formats. This will be influenced by

present work involving electronic operating manuals, and will aim to increase the awareness of and perception of the potential risks involved. This project is concerned with the higher risk associated with non-routine normal operations, such as start-up and shut-down, which requires a change from steady-state plant operations. Safety risks will be specifically identified, and the benefits of the prototype system will be demonstrated with reference to these.

The aim of this project is to have provided, on completion, material to help guide a future project which would both aim to broaden the work done here to cover a wider range of tasks and applications and also produce a more significant pilot system.

3.2 Main Deliverables

The research focuses on the design of a complex documentation environment. The main deliverables of this project are as follows:

- A statement of the information needs of operators, based on a detailed analysis of a task of a specific facility at a current plant that requires an awareness of aspects contributing to the safe execution of the task.
- A description of an open information model and architecture for plant documentation, illustrated against the information needed for the chosen task and against existing electronic operating instructions. Relevant plant information includes operating instructions, procedures and guide-lines, chemical hazard details, geographical installation information, design analysis reports, and current work-permit allocations in the form of tables, text, images and diagrams.
- An integrated documentation prototype implementation and written material emphasizing the principles required of a full-scale system that would support the operational needs of the plant facility. This is concordant with present work involving electronic operating manuals, and aims to support the move towards an increased awareness of and perception of the potential risks involved.
- A qualitative assessment of the potential risk minimization and savings resulting from future on-line installation of systems of this type in terms of reduced operator error and downtime. Risks related to information are explicitly identified.
- An understanding of the principles of management and maintenance of the information architecture and the demonstrator system on an on-going basis. This includes consideration of the use of appropriate technologies, methods and tools for information structuring, access and maintenance.

3.3 Guiding Principles for Each Phase

The research aims to integrate a range of information sources; there is a risk of becoming overburdened with the details of too many new problems. In order to minimize this risk, our intention is to investigate only a very well defined and understood operator task. The following provides the list of those important points which guides the designing of integrated documentation maintenance system in the various phases.

- Analysis Phase

1. A clear definition of the management, work place and user requirements that the information architecture and management framework must support.
2. A clear understanding of the operability weaknesses of the present system, and a description of how the proposed support system would overcome these weaknesses. This includes identification of the available plant information, limitations and costs associated with operational access to this information. This may involve identification of formal and informal communication channels and their associated limitations. The development of information architecture is necessary to facilitate this.
3. Identification of recognized standards for document management (for example, XML)[4,5], distributed databases (for example, extended SQL), relational databases, and engineering data (STEP). This is important for ensuring long-term compatibility with future developments.

- Conceptual Design and Prototype Implementation Phases

1. Operator interest and commitment to the project's ongoing success is important to the ultimate on-line use of any system. For this reason, operators will be fully consulted and provide input to the conceptual design phase.
2. New software development for the prototype will be kept to a minimum. Any software development will be compatible with the existing DCS system. The aim is to make use of and add value to existing systems (such as existing commercial off-the-shelf software) as far as possible.
3. A central issue in any full-scale information system implementation will be that of populating the databases with suitably formatted information. During the design phase, attention will be directed at selecting a plant facility where much of the necessary plant information is in an electronic form which can be easily translated into a standard format.

The above guiding principles are only the brief design requirements for the integrated documentation system design. In the following section, we will further study the strategies of the system design and why we need a formal method to help us to clarify the understanding of problem domains.

4. A Comprehensive Documentation Supporting Environment

On the basis of the above defined design principles, we further clarify that the documentation supporting system for the project should have the following functions:

1. It is a method independent supporting tool. This means that the tool should support different application environments by using different modeling methods. This is a special requirement for the project. There are several standard and non-standard data modeling methods used. Different sites of a plant use different methods to describe documentation environments. It is therefore difficult for users of one site to understand a documentation environment defined by another site, in particular, if they are not familiar with the method used to define the documentation structures.
2. It is a database independent interface. This means that the tool can understand different database systems. Data stored in different databases can be easily linked and displayed in a single interface.
3. It is a user-friendly interface. This means that the tool should provide enough helpful information when users defining a documentation environment.

By achieving these design aims, in particular, in order to minimize the risk of linking the existing software with the documentation supporting tool, we firstly proposed a conceptual structure of such documentation supporting environment. The Information Exchange Mechanisms are the rules defined either to display information stored in a database or to manipulate screen objects defined by a user. In order to clearly define these rules, we formalize these rules by using Z schema definitions.

The formal specification language Z [1] is developed by Oxford University's Programming Research Group in the late seventies. The Z specification language is currently popular both in industry and academia. Spivey has given the following explanation: `` *Formal specifications use mathematical notation to describe in a precise way the properties which an information system must have, without unduly constraining the way in which these properties are achieved*".
Formal specification is widely recognized as a precise way to define the structure of a complex software system. The roles of formal specification can be summarized as follows:

- It is used to clarify your understanding of the problem.
- It is used to communicate your intention to your customer.
- It is used to provide a prototype to demonstrate your ideas.
- It is used to use as a basis for design.
- It is used to provide requirements against which the software implementation can be proved correct.
- It used to explore mathematically the consequences of the specification.

The Z specification language is based on first-order logic and set theory. Thus, specification defined in Z can be unambiguously, precisely and concisely proven. More importantly, formal methods have play an important role in software engineering. During the design of the documentation supporting tool, the Z formal specification was used to formalize the structures of information exchange mechanisms precisely. Especially, we can,

- give a full and accurate description of the rationale of those essential operations of the tool, which is very difficult to realize using plain sentences.
- identify errors and inconsistencies in the specification process.
- define the complex documentation constructors recursively by using Z.

4.1 Information Exchange Mechanisms (IEM)

The documentation supporting tool can be simply defined as an interface between
users and databases. However, such interface can provide users enough information to help them to define a documentation environment structure. In order to clarify data transformation between the interface and databases, we propose a set of information exchange mechanisms which explain the essential data flow mechanisms among users, interface and databases.
In fact, there are two data flows which are the data transferred from databases and displayed on the interface, and the data drawn by the user and stored by databases.

To separate data flows among users, interface and databases; we can easily define different data types and concentrate on the aspect of how to define a unified structure to capture semantics of both data flows. The advantages of separating data flows in the documentation supporting tool design are:

- System components can be separately constructed and the supporting tool can be designed in an object oriented way.
- A formal method can be used to explore mathematically the consequence of the final system.

We emphasize the user-friendly interface. This means that firstly, users do not need to know the details of the documentation supporting tool, and secondly, users do not have to understand how database systems store objects defined by them. As we have described in the above section, users of petrochemical plants only need to know how to define their application structures by using their familiar data modeling methods. The information exchange mechanisms defined in the project allow users to achieve this by separating information exchanged among the users, the interface and databases according to the nature of data flows. From the documentation supporting system design point of view, there are two separate sub-systems of the documentation supporting system. The first sub-system is to transfer data flows between screen objects and interface itself, and the second sub-system maps data flows between interface and underneath database systems.

The IEM of this project makes it different from the traditional database interface (TDI) approach where the interface is designed only to interchange information between end users and specified databases. There are two disadvantages in the TDI approach. The first is that when the database systems are changed its interface must be manually altered to take an account of the corresponding changes. The second is that the interface restricts users' abilities to model and maintain complex processing environments since users are forced to rely mainly on the facilities provided by the host database system to manage the processing environments.

The IEM approach is also different from that taken in what have become known as user interface management systems (UIMSs). UIMSs emphasize more the issue of how to make the communications between users and computers more efficient, easier and friendlier. UIMSs are primarily interactive tools for supporting interface function design. In contrast, this approach stresses not only efficient communication between end users and computer systems, but also independence between both users and supporting system itself; and also between supporting system and the underlying information base [2,3,6].

4.2 Formal Rules to Control Data Modeling

The main task of the documentation supporting tool is how to maintain different semantics defined by different data model methods. Several rules, which are open-ended central control unit to maintain the semantics of different data model methods, are defined.
These rules act as modeling guidelines to control the semantics of any data model method. The advantage of doing so is that we can easily not only deal with the semantics of data models but also add a new data model method without changing the existing system.

From the documentation supporting system point of view, a data model can be abstracted in three main components: methods for defining data, methods for manipulating data and semantic

constraints defined for objects. Methods for manipulating data and controlling data semantic constraints are the most difficult and complex issues in developing a data model. The defined rules are to help users to define data manipulation methods and semantic constraints among data. This means that users can define data semantics according to their needs. To best understand formal rules defined in our approach, it is necessary for us to first review the three semantic constraints of a data model.

- *Inherent Constraints* --- They are inherent properties of a data model. They are originally defined.
- *Explicit Constraints* --- They are the properties which can be explicitly defined by a user.
- *Implicit Constraints* --- They are the properties which can be inferred from either the inherent or the explicit constraints.

In this approach, a rule has two functions. Firstly, a rule explicitly defines data semantics and the way of how to execute a data operation. Secondly, a rule is a specification. It specifies both the semantic constraints and restrictions of operations. Three types of rules are defined.

They are *Inherent Constraint Rules, Explicitly Constraint Rules,* and *Implicit Constraints Rules*

4.2.1 Inherent Constraint Rules

These are pre-defined rules and cannot be modified by users as the definition of inherent constraint defined by a data model. These rules specify essential properties of a data model.
The syntax of an inherent constraint rule is simple. It consists of keywords and descriptions which are separated by colon(:).
For example:
Model: *ssadm*; this means the model a user uses is a data model called ssadm.

Inherent constraints can be further classified as the object and relationship inherent constraints. Because different data models use different definitions for objects and relationships, in practice, the documentation supporting tool of our approach uses different files to store and control the inherent constraints of a data model. An object inherent constraint is thus defined as follows:
ObjectProperty: *domain-name, domain-value*;
The following Z schema further clarifies its nature.

Object_Inherent_Constraint

$$
\begin{aligned}
&ObjectProperty: \mathbb{P} Reserved\_Keyword \\
&domain\text{-}name: \mathbb{P} Attribute \\
&domain\text{-}value: \mathbb{P} Value
\end{aligned}
$$

This schema expresses such semantic meaning that an object inherent constraint is defined by using a set of reserved keyword. The reserved keyword used in this project is one which specifies a particular meaning of an aspect of a specific data model. The *domain-name* specifies the nature of the object which can be a simple or complex object. If it is a complex object, it can have sub-objects or component objects which in turn are either simple or complex objects. Thus,

the abstract description of this complex implication is simply expressed as an element which belongs to given set [Attribute]. A given set used in Z gives us a chance to ignore some details of an object. The *domain-value* indicates the value from which the object type can draw its values.

By using the above Z schema definition, we can easily and clearly express our intention to describe the inherent properties of a screen object. That is, an inherent property of an object is defined by a specified keyword and set of values. It is the same for us to further define a relationship inherent constraint by the following schema:

Relationship_Inherent_Constraint

$RelationshipProperty_n$: $\mathbb{P}$ *Reserved_Keywords*

domain−name: $\mathbb{P}$*Relation_Attribute*

domain−value: $\mathbb{P}$*Cardinality*

The above schema represents the same semantics as the object inherent constraint definition. However, $RelationshipProperty_n$ which specifies the *nth* relationship type is the one pre-defined and recognized by the supporting tool. A relationship inherent constraint is simply define as $RelationshipProperty_n$: domain-name, domain-value; which is stored in a file in a real system application. The *domain-value* is defined by a given set Cardinality which specifies the cardinalities among relationship types.

4.2.2 Explicit Constraint Rules

The explicit constraint rule is a first-order statement which explicitly defines a required property of a data model. By using an explicit constraint rule, we can force an existing data model to change its properties or to enhance the functionalities according to our needs. A constructor is a building block for organizing a data model. There are two constructors. The first constructor is the object constructor and the second one is the relationship constructor. The object constructor is defined as follows:

ObjectConstructorTypem ::=null_object | inheritanceobject << Node_Constructor>>| compositeobject<< Composite_Node_Constructor>>

This Z type definition uses free type structure of Z to recursively define that an object type is either a composite object type denoted by *compositeobject*, or a sub-object inherited from a super-object denoted by *inheritanceobject*, or a null object, *null_object*. Because we use hypertext concept, *node*, to describe a screen object. In the prototype, users can navigate among objects and relationships they define. The concepts used in the above definition, *Node_Constructor* and *Composite_Node_Constructor*, are used to represent simple objects and composite objects, respectively.

From the data modeling point of view, an object defined by a user represents a certain semantic meaning of a real world enterprise. However, on a computer screen, this object is a screen object which can contain different type of information such as a text, audio or an image. Thus, in order to define a node constructor, we have the following three schemata which describe a screen object in our approach.

Multimedia Type

The information within a screen object can be different types such as text, audio, video and image. This is defined as the following type in Z: Multimedia ::= text|audio|video|image

Components

The object structure of the documentation supporting tool is constructed in an object oriented way. The visible part of an object is the multimedia information stored in a database and it has a system created identity. The node constructor is to organize the multimedia information and to semantically represent an application structure. In Z, we can use a given set [ComponentID] to represent the system generated unique identities. Thus, the information stored in a screen object has a system generated identity and the combination of different multimedia types, and defined as follows:

Components

$$Media\_part : \mathbb{P} Multimedia$$

Node

The node has the same semantic meaning as the conventional node defined in hypermedia and it is described as follows:

Node

$$source, destination : Components$$

$$\exists m : \mathbb{P} Multimedi \mid source.media \twoheadrightarrow m \wedge destination.media \twoheadrightarrow m$$

The above Z schemata express only the basic rationale of the conventional hypertext node structure.

Node Constructor

The most important feature of our approach is that screen objects generated by users are constructed in a semantic way. That is, a screen object is not a system generated object. It is a semantic object which represents a real world entity of an application. This is the fundamental and important difference between our approach and other object oriented hypermedia data structures. In most object oriented hypermedia approaches nodes are treated as objects not constructors. Thus, the hypermedia data structure is passively to represent an application structure. That is the major problem of the current approaches which force users to fix their application structures by a pre-defined data structure.

Composite Node Constructor

We believe that complex and composite node structures are the basic constructors to define the structure of an application. In our approach, any meaningless node are not allowed to be related.

This is specified by the explicit constraint rule schema. Before that, we give the definition of the composite node constructor which makes the existing node as the component part of another node.

Composite_Node_Constructor

Node_Constructor

Composite_parts: ℙ*Multimedia*

$\exists n{:}Node \mid n = node \wedge composite\_parts = n.source.media\_part \wedge n.destination.media\_part$

This schema says that a composite node is a set of the existing nodes. This is represented by the combination of the multimedia information of the component part of a node. A composite node is not a new created node since it is only a container to hold other nodes. That is why there is no new corresponding identity generated for a composite node. The methodology of the approach is to avoid the complexity of using the object oriented approach to model every possible object. This is another difference between this approach and the existing object oriented hypermedia approaches which treat every node as a unique object. Our approach uses a simple mechanism to treat component information only as a unique and fundamental object. Two constructors defined above are used to structurally define the application structure. Our aim is to model an application structure in a more semantic way and capture more semantics without losing both the flexible usability of hypermedia to navigate in the hyper-space and the ability of using database to maintain hypermedia information.

Implicit Constraint Rules

An implicit constraint rule is functional statement which can produce new rules on the basis of the existing ones. It will help users to infer other semantic constraints from the existing constraints. Thus, it is used to directly support the implicit constraints of a data model illustrated at the beginning of this section. Normally, a user does not need to define implicit constraints unless it is necessary. In some circumstances, users can use implicit constraint rules to automatically generate some implicit constraint definitions which can help them to understand these semantics of their data models or be used for further application definitions. The syntax of the generated rules is the same as that of inherent and explicit constraint rules. An important issue of using implicit constraint rule is that users can merge new rules with the existing old constraints in order to construct a new implicit constraint rule.

In this section, we briefly discuss how to use Z formal specification language to formalize semantic rules used in our approach. The purpose of using formal method is to illustrate the internal structure of a complex rule definition, in particular, the different semantic connections among the defined object types and how the defined types can be related in order to represent an application structure. However, the formal definitions presented in this paper only describe the static properties of our approach. We also define thirty-four Z function definitions on the basis of these schemata. For the purpose of this paper is only to illustrate our intention of using formal methods in an industrial project, we are not going to discuss them into details.

5. Related Research

Formal methods are currently being increasingly introduced and frequently used in hypertext research. When compared with other research areas. From 1988 onwards, several research results have been published. The earliest person to use mathematical methods to formally define the structure of a hypertext system was Garg[7]. In his PhD thesis, Garg used set theory and first order logic to abstract his hypertext model, where the fundamental hypertext concepts, nodes and links, are characterized and defined by mathematical set definitions. The important contribution of Garg's work is the definition of the abstraction mechanisms --- aggregation and generalization --- which were constructed on the basis of the mathematical definitions of nodes and links. Garg's work shows that we are able to define the hypertext structure using set theory and first-order logic.Furthermore, Garg's approach convinces us that the highly precise abstraction essentials of mathematical approaches are of great significance in research concerned with combining databases with hypertext. Based on Garg's research, Afrati enhanced his model. More advanced features of hypertext systems, such as structured nodes, attributed links and scripts, can be modelled in Afrati's approach [8]. Afrati, like Garg, based his approach purely on set theory and first-order logic. No formal specification language was used. Since the 90's, formal specification languages such as VDM and Z have been introduced into hypertext research. Lange [9] used VDM to define the hypertext structure. His work experiments with the use of an OODBMS to build a hypertext system, and he formally defines his hypertext data model in an object oriented way. Halasz and Schwartz [10] used Z to formalize the Dexter hypertext reference model. The Dexter model divides a hypertext system into three layers, these being the runtime, storage and within-component layers. Dexter's specification focuses mainly on the storage layer which is used to model the basic hypertext node/link structure. In the specifications, an important and basic concept is that of a unique identifier (UID). As Halasz and Schwartz have described, "UIDs are primitive in the model, but they are assumed to be uniquely assigned to components across the entire universe of discourse (not just within the context of a single hypertext) "[10]. By using UIDs, the Dexter model defines hypertext operations clearly and unambiguously, and guarantees addressing for any hypertext component. It is also beneficial to us when defining operations, constraints and relationship types in our approach. From Lange's approach and the Dexter model, we see that the formal specification of hypertext structures can provide the foundation for understanding the essentials of hypertext systems. However, besides these formal approaches to define hypertext data models, it is hardly to see any other formal specifications describing the mechanisms for combining databases with hypertext to support complex documentation environments, like the approach described in this paper. In recent years, XML and its related technologies have shown their power to maintain complex documentation systems [11, 12]. These approaches focus on purely on how to use XML to present documents but lack the formal description of the system structures.

Another aspect of the related research for this project is the digital library research. The research on using database to support library information management can be tracked back more than twenty eight years ago [13]. The early researches on using databases to support library systems mainly concentrated on the aspect of how to use record-based structures to represent bibliographic information. This is because the bibliographic data can be naturally defined by a record, i.e., attributes of a record can be used directly to represent components of a bibliographic data such as title, reference number, date and so on. The most important result of early research revealed that the power of a database is not only to structurally define a bibliographic data but to capture semantics among documents. But, since 90s, very few researches on this area can be found. It is, however, worth to review some typical approaches which are still used in large. The

Stanford Digital Library project is a typical example [14]. This system uses meta-information to support digital libraries. This mechanism of the approach is still widely use in most digital libraries but the media is changed to XML based text format. From the design point of view, the system designers used a software engineering approach to define the requirements of the metadata. Thus, a meta-information, like XML schema, is set up on the basis of requirements. The meta-data requirements can be classified as query model for searching relevant information. The most interesting part of this approach is InfoBus, the infrastructure of the kernel structure of the digital library. The InfoBus approach is quite close to our approach. InfoBus is a distributed and heterogeneous infrastructure for digital library. It uses proxy implemented in CORBRA whereas our prototype is built on top of SQL Server 2005 and use XML as the supporting media. Other features of InfoBus such as remote method calls to access proxies and placing document objects in a machine and accessing them through the network, are quite similar to our approach. The only difference is that our approach is built on .Net Framework approach [15, 16, 17]. Due to the nature of this paper is to mainly illustrate the mechanism of our approach; it is not our intention to provide more details about the prototype. More detailed description for the corresponding prototype design is in the process and will be submitted to this journal in near future. In summary, we benefit the above described relevant researches and more importantly, our approach implemented both the formal description of complex chemical plan documents and its implementation mechanisms which are missing in the current approaches.

6. Conclusion

A complete set of design principles for the complex documentation application environments in general and for petrochemical plants in particular were developed in this project. It also proposed a possible solution of linking different tools harmoniously in order to meet different users' requirements. The key issue of such integration is the formal data model described in the above section. In conclusion, it is worthwhile to summarize the benefit of the project.

The project defines the requirements for, and demonstrates the implementation principles of a decision support system for operations management in the petrochemical industry. The areas directly addressed are refining, chemical and off-shore process control and process training.
This type of system is particularly applicable for operations involving grade changing, process plant reconfiguration and re-routing, and complex start-up/shut-down procedures.

The direct operational benefits of the project are:

- Reduced risk to individuals and the environment.
- Improved compliance with safety legislation, with less risk of enforced plant closure.
- Less economic loss due to downtime or inefficient production.
- The decision support concepts developed in this project will be applicable on all medium to large process plants in general, and on all small/large plants with specific safety environmental considerations.

The project creates a `blueprint' for the information and electronic documentation architectures which should result in reduced risk of plant incidents due to better informed operational staff and robust plant information management, and less risk of enforced plant closure due to improved conformance with safety legislative requirements. This work also provides the opportunity to draw on a related present international collaborative research between universities on the design

and implementation of conceptual and knowledge-based models of large information systems. The techniques used to develop the prototype, the decision support architecture, and the documentation and information standards used for this architecture, are all relevant to both safety-critical and non-critical operations management involving other large-scale technological systems.
The work is supported by the Natural Science Foundation of China (中国国家自然科学基金重点・目支). Special thanks to the Natural Science Foundation of China (NSFC, No.70833005) for carrying out the research presented in this paper.

7. References

1. JM Spivey, The Z notation: a reference manual, *Prentice-Hall International Series In Computer Science* (1989)
2. John J. Leggett and Frank M. Shipman, "Directions for Hypertext Research: Exploring the Design Space for Interactive Scholarly Communication"**,** Proceedings of the nineteenth ACM conference on Hypertext and hypermedia, Santa Cruz, CA, USA 2004, pp 2-11.
3. X.Shi, M.Bonner, etc, "The very small world of the well-connected", Proceedings of the nineteenth ACM conference on Hypertext and hypermedia, Pittsburgh, PA, USA, 2008,pp 61-70
4. Utz Westermann and Wolfgang Klas, "An analysis of XML database solutions for the management of MPEG-7 media descriptions", ACM Computing Surveys, 2003, pp 331-373.
5. Surajit Chaudhuri and Kyuseok Shim," Storage and Retrieval of XML Data Using Relational Databases", Proceedings of the 27th International Conference on Very Large Data Bases, 2001, pp 730
6. Jaakko Kangasharju and Sasu Tarkoma, "Benefits of alternate XML serialization formats in scientific computing", Proceedings of the 2007 workshop on Service-oriented computing performance: aspects, issues, and approaches", 2007, pp23-30.
7. Garg, P K, "Information management in software engineering: a hypertext based approach", DPhil thesis Computer Science Deppartment, University of Southern California, USA, 1989.
8.Afrati, F and Koutras, C D "A hypertext model supporting query mechanism", Proceedings of the First European Conference of Hypertext, France, Nov. 1990, pp 53-63.
9. Lange, D B "A formal approach to hypertext using post-prototype formal specification ", Lecture Notes in Computer Science (Vol 428) Springer-Verlag , 1990, pp 99-121.
10.Halasz, F and Schwartz, M "The Dexter hypertext reference model", Proceedings of Hypertext Standardization Workshop, Jan, 1990 pp.95-131.
11. Jonathan G.K and Foss, Alexandra I. Cristea, "The next generation authoring adaptive hypermedia: using and evaluating the MOT3.0 and PEAL tools ", Proceedings of the 21st ACM conference on Hypertext and hypermedia, 2010, Toronto, Ontario, Canada June 13 - 16, 2010, pp 83-92.
12. Iñaki Paz, Oscar Díaz, "Providing resilient XPaths for external adaptation engines", Proceedings of the 21st ACM conference on Hypertext and hypermedia, 2010, Toronto, Ontario, Canada June 13 - 16, 2010, pp67-76.
13. Macleod, I A, "A database management system for document retrieval applications", Information System s (6), 1982, pp 131-137.
14. Nielsen, J, "Hypertext and Hypermedia",(Chapter: Applicationsof Hypertext), Academic Press, Inc., 1990, pp 43-82.
15. Sells, C and Weinhardt, M, "Window Forms 2.0 Programming", ISBN 032126796, Addison-Wesley, 2006.
16. Amiano, M, D'Cruz, C and others, "XML Problem, Design and Solution", ISBN 0471791199, Wiley Publishing, Inc, 2006
17.Klein, Scott, "Professional SQL Server 2005 XML",ISBN 0764597922, Wiley Publishing, Inc, 2006.