SGML (Standard General Markup Language) Workshop (1996 for Systor AG)

SGML (Standard General Markup Language) Workshop (1996 for Systor AG)

Table of Contents

Chapter 1. Roadmap

Due to the success of HTML many people have heard about SGML but have only vague ideas about its capabilities and use. After a short introduction to the basic ideas behind SGML we will first discuss how SGML relates to HTML. This will carry us over to a gentle introduction to the concept of markup, document type descriptions (DTDs) and instances and validation of documents.

The advantages of modelling the "sea of information" by using a markup language will be shown next: why is content centric markup better than presentation oriented markup ? What can I do with documents in SGML form that I can’t do without ? Why do SGML people call documents "source" ?

Next come examples of successful uses of SGML production systems in various industries. This should wipe out any fears of using some kind of "exotic" technology. Many companies have drawn enormous financial benefits by restructuring their information base into a reusable, SGML based form.

SGML Tools are the next topic. We will cover the core of a SGML production system, the validating parser and discuss the concept of entity management. Is there an automatic way to tell if my HTML source contains vendor specific extensions ? Other tools like editors/browsers, transformation engines and SGML databases can be covered only very sketchy.

During the workshop we will have a look at SGML related Products and Standards like Hytime (a standard for powerful hypertext engines) and DSSSL (a standard for transformation and formatting engines that provide the final layout of SGML documents)

Our last topic will be how to provide SGML documents and processing frontends with advanced "pluggable" semantics by using java applets and how frameworks and SGML fit together.

Chapter 2. THE MAIN IDEA BEHIND SGML: INFORMATION REUSE

1.1 THE DIFFERENCE BETWEEN "DATA" AND "INFORMATION"

consider file foo.xyz: *This is DATA, not reusable*

123456789 Walter Kriha A47G92 UONF WKR Heitersheim

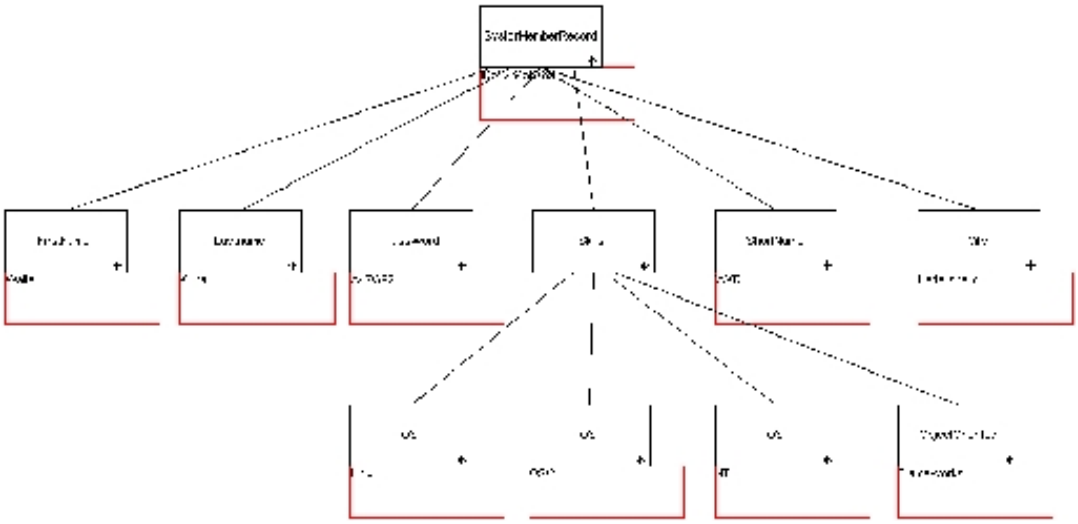
and now consider foo.sgm: *This is INFORMATION, reusable*

```
<!DOCTYPE SystorMemberRecord >
[
<! definition for SystorMemberRecord -->
<!Element SystorMemberRecord O O (FirstName,Lastname,Password,Skills
Shortame,City) >
<!ATTLIST SystorMemberRecod
ID #required>
<!Element (FirstName,LastName,Password,ShortName,City) O O (#PCDATA)>
<!Element Skills O O (OS*,ObjectOriented?)>
]
<SystorMemberRecord ID=123456789 >
<Firstname>Walter</FirstName>
<Lastname>Kriha</LastName>
<Password>A47G92</Password>
<Skills>
<OS>Unix</OS>
<OS>OS2</OS>
<OS>NT</OS>
<ObjectOriented>Framework Development</ObjectOriented>
</Skills>
<ShortName>WKR</SortName>
<City>Heitersheim</City>
```

</SystorMemberRecord>

What happens if you need more information in a Systor member record element ? In the first case you'll have to modify source code because the structure of the data is somewhere hidden in algorithms. Those algorithms probably rely on the fixed position of certain data parts. The structure is defined somewhere in a design document and probably outdated. No other tools can handle those data.

The second file contains all the information necessary to validate the content and to discover how it is organized, structured:



This is what SGML tools usually operate on and what they can offer other programs via an api. The document content tree together with the information contained in a certain documents complies with the Document Type Definition for this document.

Automatic transformation tools could map the second information to any format necessary. Screen masks could be generated as well as database schemes. Even old versions of processing tools can read it. They can go through it and find elements that they know about and warn the user about new ones. (No more : sorry , can't read your visio,paradigm plus, word, powerpoint data - it was created was a newer version. How dare you not to use the latest and greatest update of proprietary data format technology ?)

If every visio/powerpoint/word etc. update causes your information to become inaccessible what format should an insurance company use that has to store insurance documents for 100 years ?

Object technology reuses algorithms, frameworks reuse design and processing. SGML helps to reuse information. We can build software that uses SGML but at the same time our productions process is a framework that could benefit from using SGML too.

1.2 SGML RULES

- Say what you mean ! (Without meta information your information turns into "data")
- Make things explicit ! (e.g. Information exchange between applications, use markup to structure your information)

- Do not copy information, reuse it !
- Do not hide information in proprietary formats ! (source code is just one of those)
- Do not mix your information with processing instructions !

Chapter 3. 2. HOW DOES SGML RELATE TO HTML ?

HTML is a Document Type Description (DTD), specified using the SGML markup language. A HTML file, e.g."MyHomePage.html" is a document instance of class/type HTML.

2.1 WHAT DOES TYPE/CLASS AND INSTANCE/OBJECT MEAN ?

A comparison with a typed object oriented language:

C++ SGML

BNF specification of Core reference syntax

c++ grammar

special language implementation SGML declaration

(e.g. how long can identifiers be ?) (markup characters identifier length,

code sets etc)

customer.hpp: customer.dtd

class customer <!Element customer (name, address) >

{ <!Element name (#PCDATA) >

name x; <!Element address (#PCDATA)>

address y;

}

main.cpp: customers.sgm

customer <!DOCTYPE customer ...>

newcustomer <customer>

(Kriha, Heitersheim) <name>Kriha</name>

<address>Heitersheim</address>

</customer>

A language type or class means that there is an *automatic* way to make sure that instances/objects that claim to be of a certain type really ARE of this type. The c++ compiler will make sure, that nobody can place "accountnumber" into a customer object and the SGML Parser will do the same for the customer element in customers.sgm.

What does this mean for HTML ?

Having a HTML Document Type Description means, that the markup used in HTML document instances is declared and defined. The names and attributes that show up in HTML documents are defined and restricted to those mentioned in the Document Type Description. And not only the literals

are defined: their structural composition is defined too:

can a <H1> Element in HTML have any other Elements embedded ? Which ones ? What attributes are legal for <H1> Elements ?

There is ONLY ONE PLACE get an answer to those questions: the HTML Document Type Description ! NO written document defines HTML ! Or have you ever seen a C or C++ Compiler refer to user documentation when it compares a class definition with its use in a source file ?

Unfortunately many HTML Applications do not validate (compare) HTML instances against the DTD. This is the reason why there are so many broken HTML Homepages etc. around. The Browsers are very forgiving and will simply skip over wrong elements or attributes or eventually generate errors.

2.2 HTML BROWSERS COMPARED TO FULL SGML APPLICATIONS

HTML Browsers are SGML applications that know ONLY ONE Document Type Description: HTML.dtd. This is simply a waste of programming resources.

An SGML Browser will process any document instance of any DTD.

2.3 HTML EDITORS COMPARED TO SGML EDITORS

Many HTML Editors do not read or understand the HTML Document Type Description: They will let you insert wrong elements or attributes or violate the structural composition. SGML Editors will read the DTD (any) and they will offer only legal elements or attributes at a certain position in document instance: e.g. in our customer example, you would not be allowed to enter an element accountnumber within a customer element.

And even if they read the HTML DTD and validate it against your input: its a waste of programming ressources because most of the code to validate against ANY DTD must be implemented anyway.

2.4 THE INFAMOUS „PROPRIETARY" EXTENSIONS TO HTML

The discussion about „private" extensions to HTML by Netscape or Microsoft is pretty much incomprehensible for any SGML person. In technical terms, we talk about new Document Type Descriptions e.g. NetscapeHTML.dtd or MicrosoftHTML.dtd. This is what SGML was made for ! To structure YOUR information in any way YOU like. Those are YOUR informations, aren’t they ? SGML even provides ways to register your new document types as a publicly known type with so called „formal public identifiers".

When a SGML editor finds a document that claims to be of type NetscapeHTML

<!DOCTYPE NetscapeHTML .....

it will load the NetscapeHTML.dtd and offer you exactly those elements and attributes that are legal

according to NetscapeHTML !

2.5 USING THE SGML PARSER TO DETECT PROPRIETARY EXTENSIONS AND BUGS IN HTML DOCUMENTS

Anybody who extends a document type description is creating a new type of document that does no longer comply to the old DTD.

Problem:

Suppose you are a consultant doing home page design for customers. How do you make sure, that your HTML code is correct ? It may run on the netscape browser. Will it run on other browsers ? And if not, is it your fault ? Who has to pay for fixes ? If you could PROVE that your HTML code is structurally correct, the customer would have to pay for adjustments for buggy browsers.

Hint:

Run your HTML documents through a validating SGML Parser (e.g. SP by James Clark). The parser will use the HTML document type description and compare your document with it. If it does not complain, your documents are valid HTML instances.

Problem:

Do you want to know if you have used Microsoft or Netscape special elements or attributes ?

Hint:

Just run the parser with the regular HTML DTD. The parser will tell you about non-conforming parts.

Problem:

Do you want to make sure that you have used only netscape extensions ?

Hint:

Run the parser with the NetscapeHTML.dtd and your document. Everything in your document that violates the netscape document type description will show up.

The idea behind is: Use an automatic way to validate instances against type definitions. And the background comes from mission critical documents like airline repair manuals: if an importand element about crucial service points in a boing 747 does not make it into the final release of the maintenance manual (probably due to cut and paste errors) this plane might be doomed.

By comparing the service manual document against a document type description, the parser detects a missing element. This is what QA and ISO9000 people like to hear.

2.6 USING THE SGML ENTITY MANAGER TO BREAK UP HTML INSTANCES IN REUSABLE CHUNKS

SGML knows the *element structure* (everything that is defined with <!ELEMENT xxxx .. > belongs to this „logical" structure and forms a tree of named elements) and the *entity structure*, the

physical way your information is divided in different chunks.

Lets use an analogy from C++: typically programms written in this language are physically split in .cpp files and .hpp header files. The header files contain definitions and get included into the .cpp by the preprocessor directives. The C++ compiler finally sees one big file that contains everything. Not beeing able to structure your Information in physically small chunks means no reuse but copy and paste adjustments with all their problems.

SGML allows you to create information chunks (entities) and include them into your document via an Entity Declaration

<!Entity WK „Walter Kriha" >

After that you can use the entity WK anywhere in your document (if the document type declarations allows text at this position). When the SGML parser finds something like

<text> &WK <text>

in your document, it will replace „&WK" with „Walter Kriha". You can put frequently used text fragments together in a file and include this file at the beginning of your document.

If the entity refers to some external information (file, database content, a page on the web etc.) the SGML parser asks the entity manager to resolve the reference and return the content.

While the former case with „&WK" is more like the usual macro mechanism in C++, the latter case is special: The rquested information could be CREATED by some mechanism exactly at this moment (e.g. by a database lookup or a web access)

Problem:

You would like to reuse more of your HTML documents without falling back to prehistoric cut and paste technology.

Hint:

You can use the entity mechanism with your HTML files and break them up into reusable chunks. Then, at page creation time, call the „spam" utility from James Clark from a cgi script. Spam will use the SGML parser and entitymanager to resolve the entity references and produce the final HTML document. Works just like the C/C++ or any other macro preprocessor. This would allow you to reuse your HTML information in different pages.

2.7 HTML MARKUP VERSUS SGML MARKUP

From an SGML point of view the HTML markup has been one of the most criticized points. It is a mixture between content centric markup and presentation oriented markup. And therefore does neither right. HTML markup does not allow advanced layout specifications like Adobes PDF does and it is not content centric enough to allow more intelligent queries.

2.7.1 content and structural markup:

HTML other DTDs

H1,H2, HEAD,BODY e.g. research,medical,tumors,treatment,

2.7.2 presentation markup:

HTML other DTDs

BR,BOLD, etc.

NONE !! „Style sheets" are used to specify layout outside the document

Stylesheet: (using pseudo commands)

Select Element „tumors" in all elements research (transformation)and print them bold(formatting).

The real language used for that is DSSSL (pron. „dissl" , Document Style Semantics and Specification Language). A subset of it will be used to format HTML documents and is part of the XML effort. The steps „transformation" (selecting parts from a SGML document) and formatting (processing those parts) are typical for SGML applications. (see below)

What is the advantage of keeping all layout/presentation issues outside of your document ? The content markup can concentrate on the information structures only. Consider a search engine that can use more content centered markup like the medical DTD example above. It can use the information in tags for searches too. A request like:

find all information in „research" elements and give me the content of element „tumors"

sounds a little bit more specific than:

go through all H1,H2 ..... elements

The acid test for element markup: would you put an element with name „H1" into a database ?

Would you like to model your companies information base (e.g. SBCs codedb using metainformation tags like „H1" ?

Another advantage of keeping processing information out of your document:

Different „formatters" (layout producing engines) can be attached, e.g. tex or a printing system for the visually impared. Postprocessing programs can use the „pure" content of the document for whatever they want to do. There is nothing in the document that makes it hard for postprocessing systems to use it.

The conclusion is, that HTML documents are a very good medium for the web but they should be generated from more content centered markup. According to Sun Microsystems, their new generation of Web servers will do just that.

2.8 A NEVER ENDING STORY: LOST LINKS IN HTML DOCUMENTS

A link is not only a convenient means to guide users through an information base. It is also a means to avoid copy and paste approaches - the dreaded duplication problem - and to guarantee that only the latest version of a referenced document is used. As we have seen above, it could even be generated on the fly if (lazy evaluation technic). If only the referenced document could be found !

SGML and related standards can help here too. The main reason why HTML documents are hard to maintain is the incorporation of links into documents and a lack of addressing mechanisms. This is what Hytime (Hypermedia time based structuring language) is all about:

Suppose we have:

- a piece from an audio stream
- some frames from a video
- some selected elements from a SGML document
- a moment in time, e.g. a workflow item
- a segment from a 3D object
- a chunk from a cd

addressing: (What's in a name ?)

How do I address these things in the real world ?

This requires an addressing mechanism that is capable using structures within real world objects to navigate and find requested pieces of information It needs a scheme to specifiy start and end points, dimensions etc.

linking: (Solve problems with another level of indirection)

How do I link these different pieces of information together ?

It needs a scheme to specify different kinds of links: bidirectional links, links that are kept outside of the documents for easy manipulation, aggregate links etc.

XML (extensible markup language) is a proposed standard within the www community that will combine SGML, DSSSL and Hytime into an easy to use hypermedia system capable of advanced rendering and linking.

Chapter 4. 3. PROCESSING OF STRUCTURED DOCUMENTS

Processing of SGML document instances is the applications job. Since the markup of any SGML document can be defined in any way the user wants, the application will need hints how to process specific elements. We have to add the semantics of processing.

There are basically 2 different way to extend the application for new document types:

use other documents that specify what the application is supposed to to. This is kind of a mapping process: elements from the new document type get mapped to already known processing steps via a specification language (e.g. Stylesheets in DSSSL).

Recent advantages in software technology made another approach possible: "plug ins" or "applets", provided together with the document, could extend the behavior of the processing application. The application becomes kind of a virtual machine that gets extended for each document type.

Both approaches have advantages.

The first one, using a language to specify processing requires a powerful machine to interpret the processing instructions but makes those processing instructions explicit. Other tools could manipulate those processing instructions on a meta level.

The second one seems to be more convenient. Just download the behavior and extend the processing capabilities. The disadvantage is, that the processing instructions are hidden inside the applet or plug in. The be really flexible, those applets should be generated instead of programmed. But where do the instructions to generate processing applets come from ? At this moment your back at the specification language, just on a meta level.

Chapter 5. 4. COMPANIES USING SGML MARKUP AND TOOLS

Because SGML tools were fairly expensive and the process of restructuring the information base was expensive, mostly companies with a huge information base or special safety concerns used SGML, e.g. airline industries, military and government organisations, automotive industries etc. Right now SGML based production systems are entering medium sized companies.

4.1 GENERAL AREAS OF USE

- Publishing (so called database publishing)
- Information exchange between companies (like Edifact but more powerful)
- Systems management (Help systems e.g. Unix desktop)
- Information retrieval
- logistics (parts databases)
- Hyper- and multimedia Applications

4.2 EXAMPLES

4.2.1 Semiconductor companies

In 1996 those companies agreed on a standard for their complete information base on integrated circuits. What are they doing with it ?

- publish application notes
- create marketing information
- use it as input to their production lines
- use it as input to their design and development tools
- exchange this information with other companies that would like to be a second source

What are the gains ?

- NO duplicated information
- Every element is defined and can be checked automatically

- The same information base can be processed by different tools for different purposes: reuse of information
- Ability to transform the document source into any format requested
- High quality queries are possible, using the markup (meta) information in the source
- Version control and releases are easily handled
- Independence of national code sets
- long term archiving is no problem: SGML is an ISO standard

4.2.2 Airline industries

Several years ago they agreed to exchange information on airplanes in SGML form only. Why did they decide to go SGML ?

Their major concerns are:

the massive amount of documents that come with an airplane (Lufthanse gets about 1.3 million pages of documentation to EACH airplane from Boing. EVERY YEAR ! With 3 -4 updates EVERY YEAR.

Manual processing of those updates is simply impossible. Not only because of labor costs but because many of those informations are mission critical. The service handbooks have to be updated too and no item is allowed to get lost. Only SGML structured documents provide the markup that is necessary to do automatic validation and transformation.

4.2.3 Automotive Industries and Telecommunications companies

Every car or part of it that is going to be exported to the US has to have complete documentation in SGML form. The big car makers (e.g. BMW) use SGML heavily for their internal information processing.

In 1996 the telecommunications companies have agreed to use SGML for data exchange between telcos.

Chapter 6. 5. COULD WE USE SGML WITHIN SYSTOR AND FOR SYSTOR CUSTOMERS ?

5.1 INFORMATION MODELLING: SBCS CODEDB

SBC surely has a huge information base that is mission critical. Processing this information means:

- putting it into User Interface Ressource file to get screen masks,
- putting it into source code for processing,
- putting it - or parts of it - into configuration files to control application behavior.
- Converting it into different formats and languages for access to other databases (transformation/ translation).
- Generate queries from it
- document it, generate reports

A lot of this information is duplicated in one way or the other and becomes part of ressource files or source code. This is a maintenance nightmare.

If this information base is structured using SGML, every application could use a SGML processing sub- framework to read from this base. Screen masks could be generated automatically during runtime. The basic issue is: the element "customer" from the information base has been turned into a strictly defined TYPE that is machine readable. This definition is *COMPANY WIDE*! No programmer will need extra documentation that declares what type it is. Its all in the document type description to this type - and it is *machine readable* !

If the SGML framework has a CORBA Interface , any processing application (c++, c, smalltalk, java etc.) could get access to it. The codebase itselft could be converted on the fly to html and everybody could access it with a web browser. The Panorama SGML Browser has been converted to a netscape plug in and would even save us the transformation to HTML.

Other departments then software engineering would also get benefits from this new information base. Besides beeing able to use standard tools to view and edit the information they could use post-processing tools to generate whatever information they need internally.

5.2 SGML AND OBJECT ORIENTED FRAMEWORKS

There is a natural fit between those two approaches. Both concentrate on reuse and automatic processing issues. Both need to be highly configurable to fit various needs. Both use complex data structures like trees and directed acyclic graphs to build generic processing machines. Frameworks need a means to get access to meta information on data to enable generic processing. Documents structured with SGML provide just that.

A frequent problem with generic processing is the translation between different symbolic representations of information. Most frameworks have "wrapper" or "translator" type classes that convert from one format to the other. Where do those wrappers/translators get their mapping information ? Is it hard coded in the source code or is it kept in SGML documents ? If a translator needs special

processing algorithms, a plug in can provide it easily. Where does it come from ? From a processing or workflow document that gets mapped to the document format that should be translated into a different form.

Ressource files, database schemes, repositories etc. are just documents and they can be defined using SGML.

Chapter 7. 6. SGML TOOLS AND APPLICATIONS

6.1 AVAILABILITY AND SGML "AWARENESS"

One important aspect when choosing tools is their SGML "awareness". A database can handle SGML documents by storing them as "blobs". This may be ok for a publishing system that has to produce books from finalized versions of documents. It is definitely not enough if you want to use queries or reuse your document type descriptions for new types. The system simply does not know about markup. Entity Managers or Element managers can work on the markup directly and answer complicated queries.

Practically every aspect of an SGML production system can be handled with user friendly tools. There are

- Browsers,
- Editors,
- Databases,
- Transformation engines, to sgml or between DTDs,
- Formatters,
- Document Type editors and analysers.

There are even "visual recognition engines" that try to create markup by scanning and interpreting visual characteristics of non - SGML documents. Tools help to convert huge databases into SGML form.

see: www.falch.no/infotek for the authoritative guide to all SGML tools.

especially interesting for software engineers: system development kits for parsing, entity management, styhle sheet processing and browser/editor generation, many of them are public domain and can be easily integrated into a framework.

6.2 ARCHITECTURE OF A SGML PARSER AND ENTITY MANAGER KIT

The parser toolkit by James Clark is a very flexible framework that shields most applications from SGML interna through a generic api. Clients can receive parser events and build whatever data structures they need to represent the document markup and content.

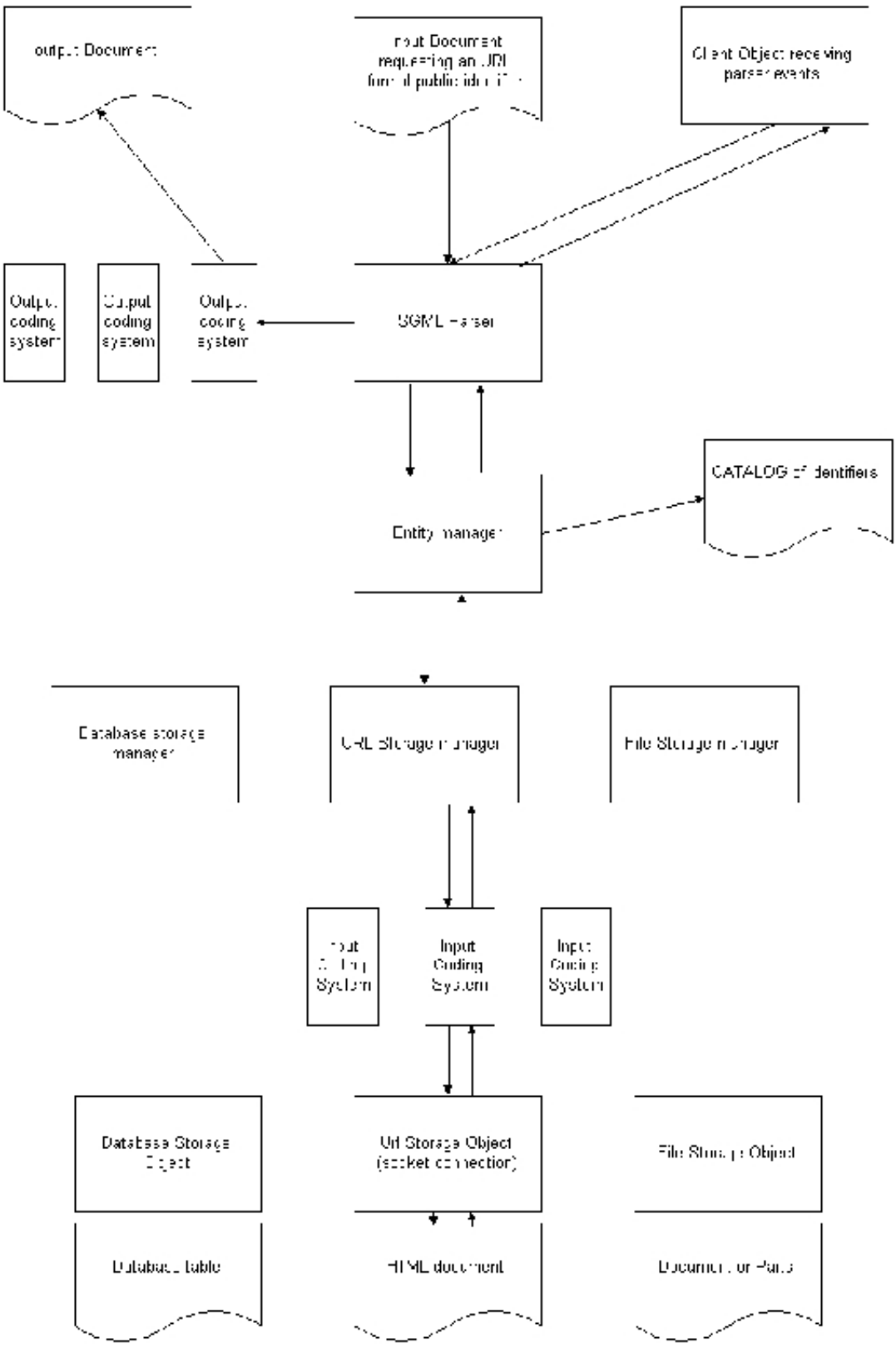
Input and output conding systems can read or create arbitrary formats and coding systems (Unicode etc.) This is important since SGML is an ISO Standard for international document exchange too.

The Entity Manager, when he receives a request for a certain entity, consults the CATALOG file(s) to find a proper mapping to a system identifier. This Identifier goes via the chain of responsibility to all available storage managers. The one responsible for this type of entity creates a storage objects and wrapps it into intput storage systems that do the translations from SGML names and entities to

system known types.

This is a pretty complex process and the good knews is, that it is of no concern to you.

Of course, this parser and entity manager framework is extensible to support new storage types.


output Document
input Document requesting an URI from a public identifier
Client Object receiving parser events
Output coding system
Output coding system
Output coding system
SGML Parser
Entity manager
CATALOG of identifiers
Database storage manager
URL Storage manager
File Storage manager
Input Coding System
Input Coding System
Input Coding System
Database Storage Object
Url Storage Object (socket connection)
File Storage Object
Database table
HTML document
Document or Page

Chapter 8. 7. SO WHAT IS THE PROBLEM WITH USING SGML ?

Again there is a strikting similarity between Object Oriented Development like frameworks and SGML production systems:

The hardest part is NOT the technical side. NOT the syntax or grammar or algorithms. After almost 3 years of framework development I can say that the hardest part is to make people think about what they are doing. To categorize, find commonalities and abstractions and to define them. It is this meta position that es necessary but makes people often feel uncomfortable:

I can’t prototype anymore ! Why do I have to make classes and types ? Why do I need a document type description ? I just want to write documentation. I am a pragmatic - just do it !

You will hear those arguments over and over. This position is characterized by a concentration on a small aspect of a companies business: that persons own work. Lifecycle issues, maintenance and service, reuse and final costs are excluded. It needs good financial controlling to keep hackers from creating junkyards of data instead of reusable information.

Chapter 9. 8. USEFULL POINTERS AND LITERATURE TO SGML, DSSSL AND HYTIME

8.1 THE BEST POINT TO START

www.sil.org/sgml: robin covers sgml web page. the best starting point for everything about SGML and related subjects.

8.2 TOOLS AND VENDORS

www.falch.no/infotek Steve Peppers Whirlwind Guide to Tools and Vendors, excellent and complete Reference of everything thats available, where you can find it and what projects are running.

8.3 INFORMATION MODELLING AND REUSE

Eve Mahler and Jeanne El Andaloussi, Developing Document Type Descriptions.

A very good book about information modelling

8.4 IMPLEMENTING A COMPLETE SGML PRODUCTION SYSTEM

B.Travis, SGML Implementation Guide

explains what is necessary to create a SGML production system for a company and how to gain the benefits from it.

8.5 SYNTAX, GRAMMAR AND CONCEPTS OF SGML AND HTML

Wolfgang Rieger, Einführung in SGML

A well structured introduction

8.6 THE SGML BIBLE

Charles F.Goldfarb, The SGML Handbook

"THE" book about SGML, contains the ISO standard to

8.7 SYSTEM DEVELOPMENT KITS, PROGRAMMERS STUFF

for developers:

www.jclark.com: homepage for the famous sp parser and entity manager and the jade DSSSL engine. All public domain, well designed and easy to integrate in your own software. SP supports Hytime too.

www.synex.se:

makers of Viewport, a system dev. package to add sgml and hytime browsing technology to your application FAST. Several browsers have been written using Viewport.

8.8 WORK ON STANDARDS (SGML, DSSSL, HYTIME)

www.sgmlopen.com: homepage of the SGML standards body. Documentation on Standards, new releases, DSSSL etc.

8.9 HYTIME

www.technoteacher.com Hytime specialists, public domain hytime browser and literature on Hytime

David Durand, Steve DeRose, Making Hypermedia work. A guide to the concepts behing Hytime. Explains addressing and linking.

8.10 CURRENT DISCUSSIONS ON MARKUP, HYPTERTEXT AND THE WEB

and last but not least:

comp.text.sgml, the SGML newsgroup. Practically all major SGML developers take part in this newsgroup. A must.