Implementing a viable architecture for standardized Intelligent Graphics

Co-Author: Maryse DA PONTE
Title: Research Engineer.
Company : Aerospatiale Division AIRBUS
Address
Service A/BIS/A, BP D0611
316, route de Bayonne
Toulouse
France
31060 TOULOUSE Cedex 03
Email:maryse.da-ponte@avions.aerospatiale.fr.

Biography

Maryse DA-PONTE joined Aerospatiale in 1989 and has worked in the electronic documentation since 1992. From 1994, she has been involved in the graphics area and has joined the ***ATA*** (Air Transport Association) /AIA (Aerospace Industry Association) Graphics Working Group which is working on the concept of ***Intelligent Graphics***.
Maryse is a graduate engineer in Computer Science from the CNAM (Conservatoire National des Arts et Métiers) (France).

Co-Author: Marco GOERTZ
Title: Software Engineer
Company: Inso Corporation
Address:
299 Promenade Street
Providence, RI 02908
USA
Email: mgoertz@inso.com

Biography

Marco GOERTZ joined Inso Corporation in 1998 for his technical expertise in "intelligent graphics". He created the "intelligent graphics" project MetaWeb during his graduation thesis in a joint venture with Ematek GmbH (Germany) in early 1996. In August 1996, he was hired by Ematek and has been responsible for the MetaWeb development since then. Over the last 3 years he has been working closely with Henderson Software Inc. (HSI) on multiple projects from Germany and in the USA.

He has made significant contributions for the WebCGM profile developed by CGM Open and the W3C.

Marco is a graduate engineer in Mechanical Engineering from the Technical

University of Braunschweig (Germany). He is specialized in Graphical User Interface (GUI) design and development and has created applications for companies such as Siemens and Volkswagen.

Co-Author: Lofton HENDERSON
Title: Dir. Adv. Graphics Devt.
Company : Inso Corporation
Address
1919 Fourteenth St,
Ste 610,
Boulder
USA
CO 80302
Email:lofton@cgm.com.

Biography

Prior to joining Inso Corporation in 1997, Lofton Henderson presided over Henderson Software, Inc., specializing in CGM technology, products, and services for 12 years. He has taught and lectured extensively on CGM and other computer graphics topics, and is co-author of the "The CGM Handbook" (Henderson and Mumford, Academic Press, 1993, 450 pp.), an in-depth look at the standard and its application in the real world. He has worked on the ANSI and ISO committees responsible for graphics standards for 15 years, during which time he has had roles ranging from CGM document editor to Convenor of the ISO SC24/WG6 Metafiles Working Group. He has made substantial contributions to the definition of the significant industry CGM profiles: ATA GRexchange and IGexchange, PIP, J2008, RIF, and the recently approved WebCGM W3C Recommendation. He currently is chairman of the CGM Open Consortium.

Foreword

The first parts of this paper recap the contents of "Intelligent Graphics: Towards a viable architecture using the most appropriate standards", Da-Ponte, Duluc, and Henderson, 1998. That article discussed the origin and requirements of Intelligent Graphics, and proposed the general terms of an optimal, open, and interoperable architecture. The later parts of the present paper develop the ideas of the original paper further, illustrating them with a mockup, and exploring further standardization of critical components of the system.

Introduction

The WEB technologies offer powerful and user-friendly functionality for consultation of electronic documentation. They enable in particular a quick access to the correct information. This functionality is available for text content. But documentation also includes numerous graphics which enable a rapid understanding of information. However, the electronic consultation of graphics does not offer the same functionality as text does for on-line consultation. Current standardized electronic graphics are limited. They do not enable for example, links from a graphic's objects to another information or retrieval on the content they include. These limitations are because they do not enable any computer processing. Only human eyes are able to interpret them.

Therefore, work has been initiated to specify more powerful electronic graphics. The ATA (Air Transport Association) has developed the concept of "Intelligent Graphics" and has worked on this concept for the aeronautical domain for several years. Recently, the W3C has defined a set of requirements to use interactive graphics on the WEB. This article will present work done in this domain, and more particularly in the aeronautical domain.

We will introduce, first, the specific needs of the aeronautical domain and the work done by ATA around the "Intelligent Graphics" concept.

Second, there is no single solution today to implement this concept. In addition, new standards such as HyTime and XML offer new possibilities, which could also be useful for graphics. So, Aerospatiale initiated a study to define a viable architecture using the most appropriated standards for standardized Intelligent Graphics. We will present the results of this study in a second part.

What about "third part". To validate the results of the study and explore the concept of "Intelligent Graphics" in more detail , a mock-up has been developed. We will present it.

Finally, the study and the implementation of the mockup has initiated some discussions on issues, general characteristics of the companion file, and the need for a standard API. These issues will be discussed in a fourth part and will enable to conclude on work to be done in the future.

ATA work

Let's see first the work done by the ATA around the "Intelligent Graphics " concept. A short introduction on the specific characteristics of aeronautical technical documentation will present the needs of this domain.

Aeronautical context

The aeronautical technical documentation, including AIRBUS documentation, has specific characteristics. It is varied (different domains, different types of information), and it is also characterized by large quantities of data, numerous customizations, short revision cycles and a long life. A high level of standardization is also needed.

The aeronautical technical documentation includes many graphics. An AIRBUS Illustrated Parts Catalog can include more than one hundred thousand graphics. Their uses are frequent and varied. Sometimes, experienced mechanics only take the graphic to repair a failure. Technical documentation users also use them as a means of navigation in the large-volume documentation.

In order to manage all these characteristics, technical documentation is migrating to Electronic Structured Documentation which offers powerful potentialities in terms of consultation, edition and management for text. To obtain the same potentialities for graphics, the ATA has developed the " Intelligent graphics" concept.

ATA "Intelligent Graphics" concept

The ***ATA*** concept of "***Intelligent Graphics***" defines standardized structured graphics, which could be used by applications in an interactive way. The "***Intelligent Graphics*** Requirements" specifications [IGREQ22] gives some base principles for the structuring requirements and defines the functionality that "***Intelligent Graphics***" must support.

ATA MODELING

ATA specifications [IGREQ22] give some basic Object-oriented requirements for structuring graphics. As the basic need is to access objects smaller than the whole picture, the picture is structured in objects, called "graphical objects". These objects are logical units, such as an engine or a locator. They contain a graphical representation of the object, the semantics associated with these objects and can include others objects.

To be accessible, each graphical object must be identified. A logical graphical object can have relationships to other objects (graphical or ***SGML***). The semantics of the logical graphical objects, also called properties by the ***ATA*** specifications [IGREQ22], is described via attributes.

"*INTELLIGENT GRAPHICS*" FUNCTIONALITY

ATA defines four types of functionality. Three of them will be presented here. In regard of the state of the art, the last functionality, the analysis seems more difficult to be implemented today in a standard way and will not therefore be studied here.

✈ Navigation needs

The first functionality specified in the "***Intelligent Graphics***" concept is navigation. It represents the most important functionality in terms of needs. It permits, for example, the navigation from an illustration of a component to another illustration of the same component containing more details. This functionality represents all the capabilities of browsing from a logical graphical object to another documentation unit. It will be done from a logical graphical object to another logical graphical object and/or textual element. For example, a fault code can refer to two figures.

✈ Query needs

The second functionality specified in the "***Intelligent Graphics***" concept is the query. One example is "based on a reference designator or an equipment list number, being able to access illustrations containing a particular part". This functionality represents all the capabilities of accessing a logical graphical object using query mechanisms.

Different types of query are necessary, full text search, queries on the structure, e.g., find all the illustrations of chapter "31-30-00" and queries on properties, e.g. finding the graphical object representing the part whose part number is "D60192". Some queries mix both types.

✈ Extraction needs

The third functionality specified in the "***Intelligent Graphics*** Requirements" specifications is the extraction of data, e.g., identification of the reference designator or equipment list number associated with a component in a graphical object. This functionality will enable the end user to access the non-graphic information from an illustration. The extracted information could be or not a part of the visible illustration.

Despite there being a subtle difference between the goals of extraction and query, in terms of exchange specifications, the extraction requirements are almost the same as the query requirements. The extraction is done by the applications after the query.

Let's see now how this functionality is supported by graphics.

Aerospatiale study

The ATA 2100 specifications recommend two ways to exchange structured graphics. , a pure ***CGM*** solution or a mixed solution using ***CGM*** plus ***SGML***. Thus, the "semantic" content may reside either in the ***CGM***, or in the ***SGML.*** The specification of the latter solution needs to be completed to specify the roles of each standard in a detailed way. In addition, the ATA specifications have not yet taken in account new standards such as XML, Hytime, XLL.

So Aerospatiale, in cooperation with HSI/Inso, made a study to evaluate the following standards: SGML, Hytime, XML and determine if these standards could be used to model structured graphics. For each standard, the advantages and drawbacks have been evaluated. This study has enabled us to choose the best standards for implementing a viable architecture for structured and interactive graphics.

Study of the standards

SGML

The ***SGML*** (Standard Generalized Markup Language) enables representation of the structure in a rigorous way enabling the analysis and treatment of this structure by computers, independently of the platforms and of the software.

SGML enables the definition of generic structures for particular types of document - the DTD (Document Type Definition). A markup language is built to express the different constraints existing on the structure and contents of a given class of documents. The DTD is interpretable by computers.

The ***SGML*** standard does not define a query language. However, SDQL (Structured

Documents Query Language), included in DSSSL (Document Style and Semantic Specifications Language) defines a query language well adapted for handling structured documents (***SGML*** in particular).

SGML presents some limitations for links. Links on objects are possible only in the scope of the same ***SGML*** document. They are enabled only between special elements which have been identified as target elements (using ID(s) attribute) and special elements that have been identified as targeting elements (using IDREF(s) attribute). For external links, ***SGML*** interprets the external document as an indivisible document, because there are no means to measure, locate and count within data. The ***XML*** standards family tries to remove these limitations.

XML AND XLL

The standard ***XML*** (eXtensible Markup Language), based on ***SGML***, introduces a new type of document, the well-formed document: a structured document, which can be processed without its DTD. It makes the processing simpler.

Another standard of ***XML*** family, ***XLL*** (eXtensible Linking Language) is the process of specifying high-powered hypermedia linking and pointers to very specific locations within ***XML*** data. This standard has been strongly influenced by HyTime and TEI extended pointers.

The ***XLL*** subset, XPointer provides a simple syntax for pointing to elements and other parts of ***XML*** documents regardless of whether they have IDs.

The ***XLL*** subset, XLink provides a way to collect groups of these pointers to make entire links: connections between arbitrary numbers of data objects. It permits you to identify your own linking elements with any names you want, and defines several sub-types of link. The simplest link, called "simple inline" link, is a one-way link from the link's own location to one other place and works like the HTML elements. XLink also provides a way to collect links outside of the documents that they reference, the "out of line" link. XLink links can also include metadata about individual links and pointers. For example, they can specify a formal role or function for each individual end. Finally, XLink provides some very rudimentary behavior control.

CGM

CGM (Computer Graphics Metafile) is the ISO standardized language to represent two-dimensional graphics with vector or raster data, independently of the platforms and of the software. ***CGM*** defines a functional specification, independent of the encoding of the metafiles, and three types of encoding. As it supports compressed raster and symbols libraries, ***CGM*** is well adapted for technical documentation.

The ***CGM*** Application Profile (A.P.) defines how a domain uses ***CGM***, e.g. the ***ATA*** Application Profile defined for the civil aeronautical domain. A parser checks the conformance of the metafile to ***CGM*** and to the Application profiles. This latter, which also enables definition of semantics specific to the domain, will become more important with "structured graphics". However, as the Application profile is not written in a computer-understandable way, ***CGM*** tools are not able to adapt automatically to all profiles.

The APplication Structure (APS) ***CGM*** element enables the grouping of graphical elements into logical objects meaningful to and accessible by applications. APS are based on the mark-up principle and delimited by mandatory tags. The APS has a type which permits classes of APS to be defined. It may be described by attributes, which provide the capability to associate non-graphical information with the logical graphical object. APS enables the metafile to be structured in a hierarchical structure. The WEB ***CGM*** profile, recently defined by ***CGM*** Open and W3C specifies interactive graphics with APS.

Two models of APS are available, the embedded APS which includes the graphical primitives of the objects and implicit boundaries, and the overlay APS which includes explicit boundaries of the object. The embedded APS offers possibilities that are more powerful: no data redundancy, exact boundaries and easy maintenance. The overlay model is useful for legacy, and raster data.

APS are specified and standardized by means of the A.P. However, today, no formalism exists in ***CGM*** for describing content model and APS relationships, such as DTDs. No occurrence symbols, connectors or reduced possibilities are available for typing objects. Specifying at least the metadata content models should be considered as a high priority requirement for the widespread use of structured graphics profiles.

No standard way is defined in ***CGM*** to specify the links between graphics or between graphics and other types of data. The only thing useable by a link application is the capability of defining an ID for a given APS. This is a step forward to the link capabilities. However, a uniform solution is necessary.

No query language is available today to manipulate ***CGM*** data and in particular Version 4 data.

The choice of standard

The study of the standards has led to the following conclusions. ***SGML***/***XML*** is a well-suited standard for describing structured documents. Regarding ***SGML***, it is obvious that it lacks graphic specifications to handle graphics. ***CGM*** is a well-suited standard for describing 2D graphics despite the fact that some standardized formalisms are missing to handle structure. However, each standard offers part of the required functionality. If we use both standards, where is the frontier between use of each?

STRUCTURE MODEL

Studying structured graphics has highlighted that the use of something equivalent to a DTD to define a structure model of a graphic is necessary. As today, ***CGM*** does not offer a formalism to define a complete structure model, it seems that ***SGML***/***XML*** DTD is a good means to describe this structure. As specified by the ***XML*** principles, this structure model will not be mandatory for consultation.

Due to the growth of ***XML***, we think this standard will make numerous tools available and will be the substitute for many uses of ***SGML***. Therefore, we recommend that every new feature defined with ***SGML*** must be "***XML***-compatible" (conversion to ***XML*** must be possible at consultation).

SPECIFY AND STRUCTURE GRAPHICAL DATA

CGM is already known as a powerful standard for representing graphics. ***CGM*** functional specifications are the most appropriate specifications to describe graphics. All the graphical data, such as the graphical primitives and the boundaries of the objects, will be specified with ***CGM***. The logical objects will be defined by using ***CGM*** APS (Application Structure feature). For new graphics, the embedded model, more powerful, is recommended. Thus, the boundaries of the objects will be automatically managed by the tools. The overlay model will only be used for the raster data, particular for photos and legacy ***CGM***.

PROPERTY SPECIFICATIONS

Considering the current state of the art, especially the fact that ***CGM*** has no modeling capability and the fact that we choose to use a DTD for model control we decided to use a ***SGML***/***XML*** file for handling properties of the ***CGM*** metafile. This file, called "companion file" will be attached to the ***CGM*** file. The structure of the graphic, described with APS, will be duplicated in this ***SGML*** companion. Properties will then be added to the resulting structure in this ***SGML*** file. This companion file will enable the graphics to be checked against the model defined by the DTD.

LINK SPECIFICATIONS

Today, the most powerful standard to define exchangeable links is ***XLL*** of the ***XML*** family. As ***XML*** has a more pragmatic approach than HyTime, it seems ***XML*** has a good future and numerous tools will probably be available.

QUERY LANGUAGE

Concerning the "Data Extraction" and "Query" needs, the first step for implementing intelligent graphics is to verify that the data on which queries and data extractions have to be achieved are correctly identified. Then the query and data extraction could be performed using conventional query languages. Standards for more powerful query languages are in development in several forums, but not yet advanced enough for review or choice. So, in the short term, existing query languages should be used and a survey of the new query languages should be done.

CHOICE CONCLUSION

The chosen solution is not perfect. It raises the classic issue of duplicating information. However, these choices are made in regard of the state of the art. We considered that ***SGML/XML*** tools are already available and that it will be easier to handle this issue than to make ***CGM*** standards evolve and ***CGM*** vendors implement them in a short time. Besides, we can imagine that a structure will be generated automatically from another.

Let's see now how these results have been implemented in a mock-up.

Presentation of the mock-up

From the results of the study, Aerospatiale has developed a research tool to investigate further the concept of "Intelligent Graphics". We will see first the mock-up in terms of functionality. In a second part, the basic principles of the mock-up will be presented.

Mock-up functionality

The mockup illustrates the consultation of interactive graphics and presents three types of functionality defined by the ***ATA "Intelligent Graphics"*** concept, navigation, query and extraction.

INTERACTIVITY PRINCIPLE

The mock-up displays interactive graphics. When the cursor passes through an object, the cursor changes shape and the object is highlighted. If you click on it, a window appears and displays the semantics associated with the object. If links are associated to the objects, they are also displayed in the windows.

NAVIGATION FUNCTIONALITY

The mock-up implements the navigation functionality, for graphical objects as well as for text elements. The innovation here is the possibility to navigate from an object of the graphic. Each object of a same graphic can have specific links.

Let's now take a concrete example : When a user reads a maintenance procedure, he can, for example, click on the reference of the figure and navigate immediately to the figure. He can also click on a particular part number within the maintenance procedure and navigate to this particular part within the graphic. In this case, the graphic is displayed with the part concerned highlighted (cf. Fig. 1 : Navigation to objects inside the graphic). Thus, the user immediately sees the correct part. This functionality is very useful for wiring schematics which include numerous small objects. On a graphic, the user also has the possibility to click on a particular part. If a link is specified on this particular part, the user can navigate to another graphic which gives for example a more detailed view of the part. When the user studies the graphic of a maintenance procedure, he also has the possibility to navigate from this graphic to the equivalent graphic of the parts catalog.

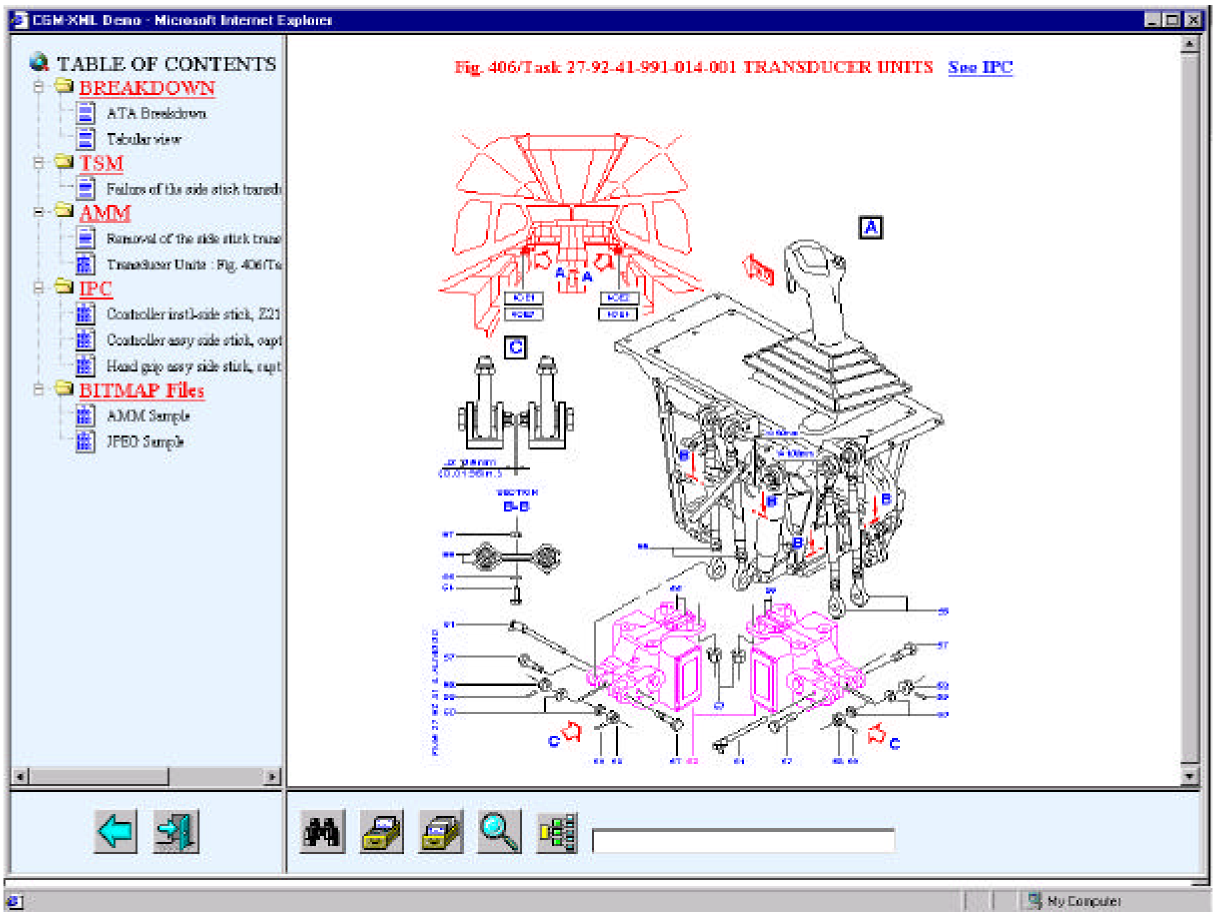

CGM-XML Demo - Microsoft Internet Explorer
TABLE OF CONTENTS
BREAKDOWN
ATA Breakdown
Tabular view
TSM
Failure of the side stick transdu
AMM
Removal of the side stick trans
Transducer Units : Fig. 406/Ta
IPC
Controller instl-side stick, Z21
Controller assy side stick, capt
Hand grip assy side stick, capt
BITMAP Files
AMM Sample
JPEG Sample
Fig. 406/Task 27-92-41-991-014-001 TRANSDUCER UNITS
See IPC
A
B
C
SECTION B-B
My Computer

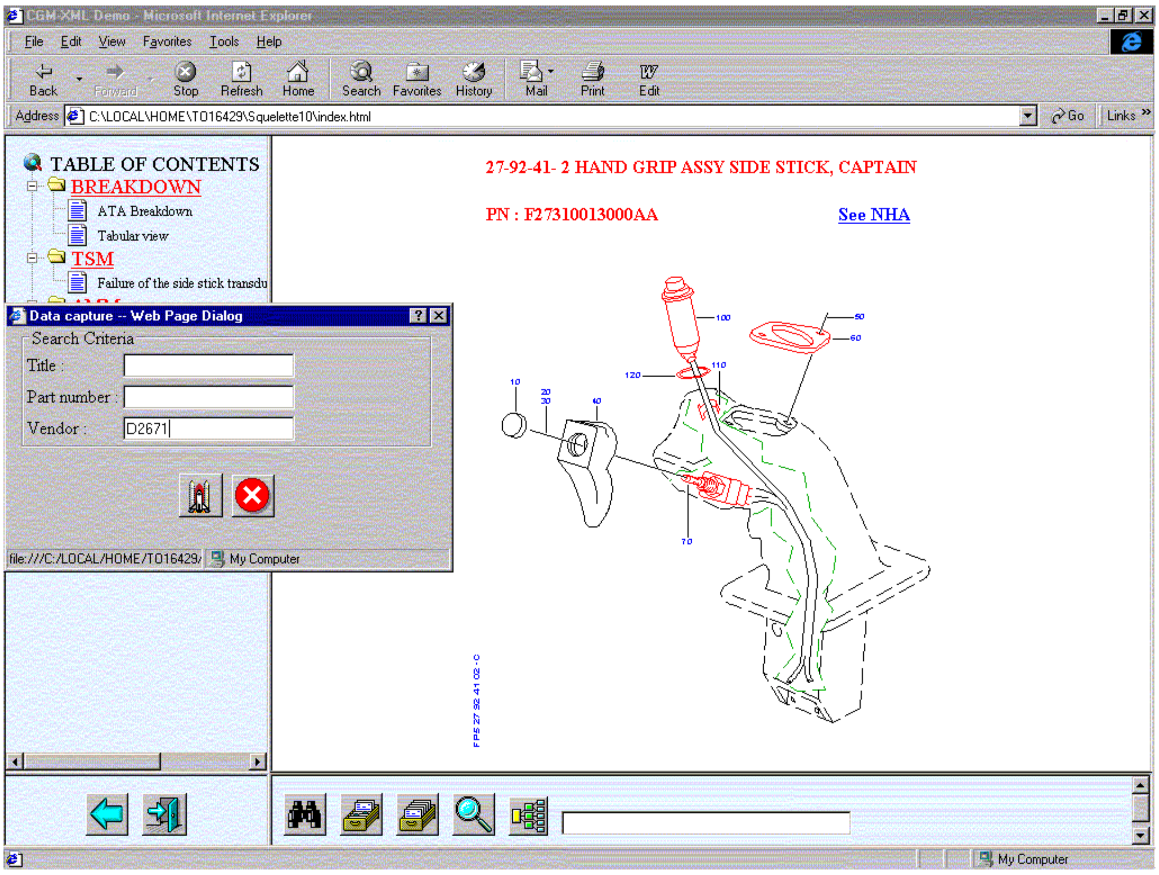
Fig. 1 : Navigation to objects inside the graphic

QUERY FUNCTIONALITY

The mock-up implements the query functionality on the graphics. The purpose of this second functionality is to find objects within the graphics based on data associated to the objects as search criteria, e.g. finding the graphical object representing the part whose part number is "D60192".. This functionality also enables queries on the structure, e.g., find all the illustrations of chapter "31-30-00". Often, queries mix both types.

Let's now take a concrete example. When a user views a graphic, he can for example make a query to locate all the parts sold by the same vendor. He can also make a query to highlight all the "screws" of the graphic. To do this query, the mock-up displays a window in which the user enter his search criteria. Thus, the user is not obliged to use query language, sometimes too complicated for documentation users. When the user validates the search criteria, all the objects which match the search criteria are highlighted in the graphic.

The mock-up only implements some criteria to do queries (cf. Fig. 2: Query on semantic). But these criteria could be more numerous, depending on the graphic modeling.


CGM-XML Demo - Microsoft Internet Explorer

Address C:\LOCAL\HOME\T016429\Squelette10\index.html

27-92-41- 2 HAND GRIP ASSY SIDE STICK, CAPTAIN

PN : F27310013000AA See NHA

Data capture -- Web Page Dialog

Search Criteria

Title :

Part number :

Vendor : D2671

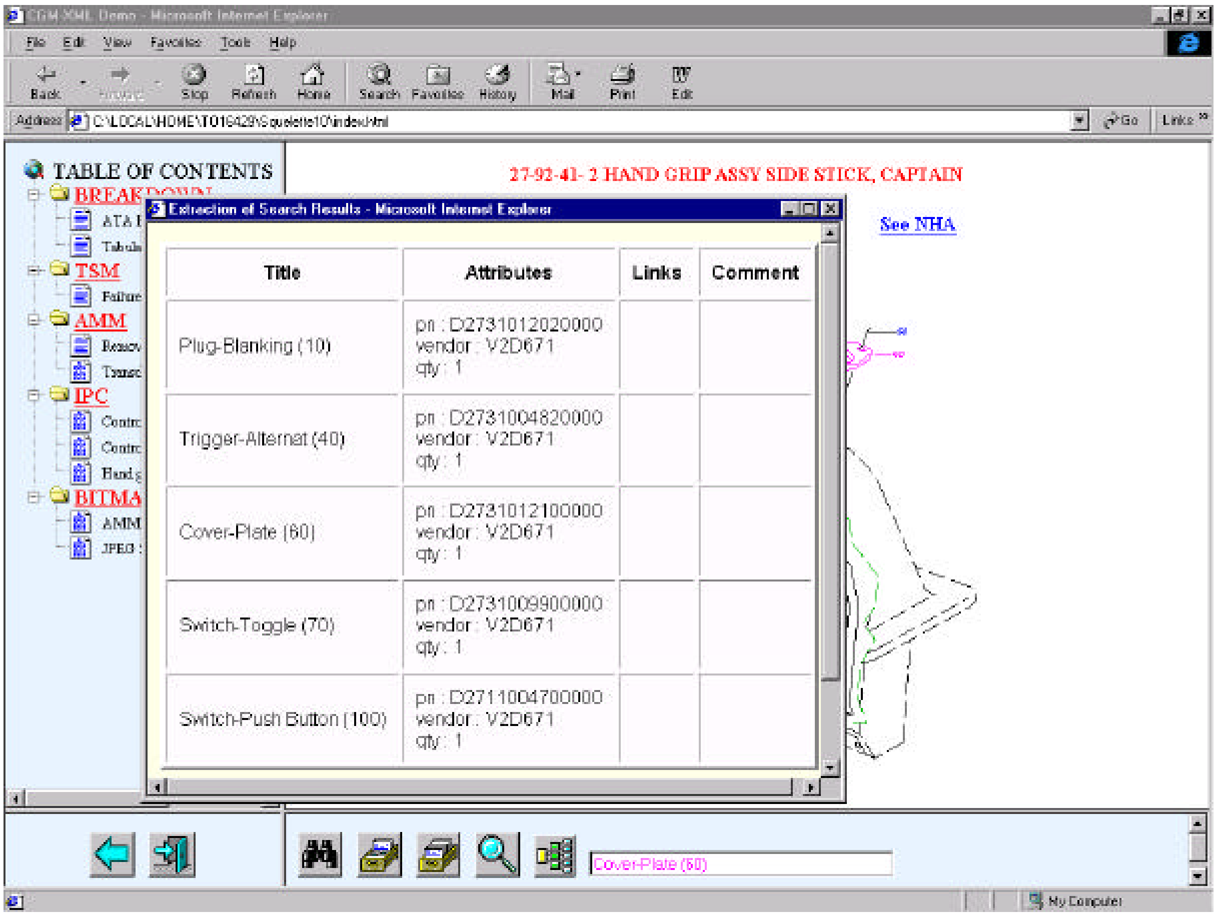
Fig. 2 : Query on semantic

EXTRACTION FUNCTIONALITY

The third functionality implemented by the mock-up is the extraction of data, e.g., extraction of the reference designator or equipment list number associated with a component in a graphical object.

Let's come back to a concrete example. In the last example, after a query, the user has seen all the parts sold by a particular vendor. With the extract functionality, the user may display all the information concerning these parts. Another possibility of the mockup enables the user to extract all the information associated with all the objects of the graphic. (cf. Fig 3: Extraction of data).

Today, the mock-up only enables query on one graphic at any one time. To implement the queries on several graphics, a new step is necessary; the management of graphics in a repository.


CGM-XML Demo - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit
Address C:\LOCAL\HOME\T016429\Squelette10\index.html
Go Links
TABLE OF CONTENTS
BREAKDOWN
TSM
AMM
IPC
BITMA
27-92-41- 2 HAND GRIP ASSY SIDE STICK, CAPTAIN
See NHA
Extraction of Search Results - Microsoft Internet Explorer
Cover-Plate (60)
My Computer

Title	Attributes	Links	Comment
Plug-Blanking (10)	pn : D2731012020000 vendor : V2D671 qty : 1		
Trigger-Alternat (40)	pn : D2731004820000 vendor : V2D671 qty : 1		
Cover-Plate (60)	pn : D2731012100000 vendor : V2D671 qty : 1		
Switch-Toggle (70)	pn : D2731009900000 vendor : V2D671 qty : 1		
Switch-Push Button (100)	pn : D2711004700000 vendor : V2D671 qty : 1		

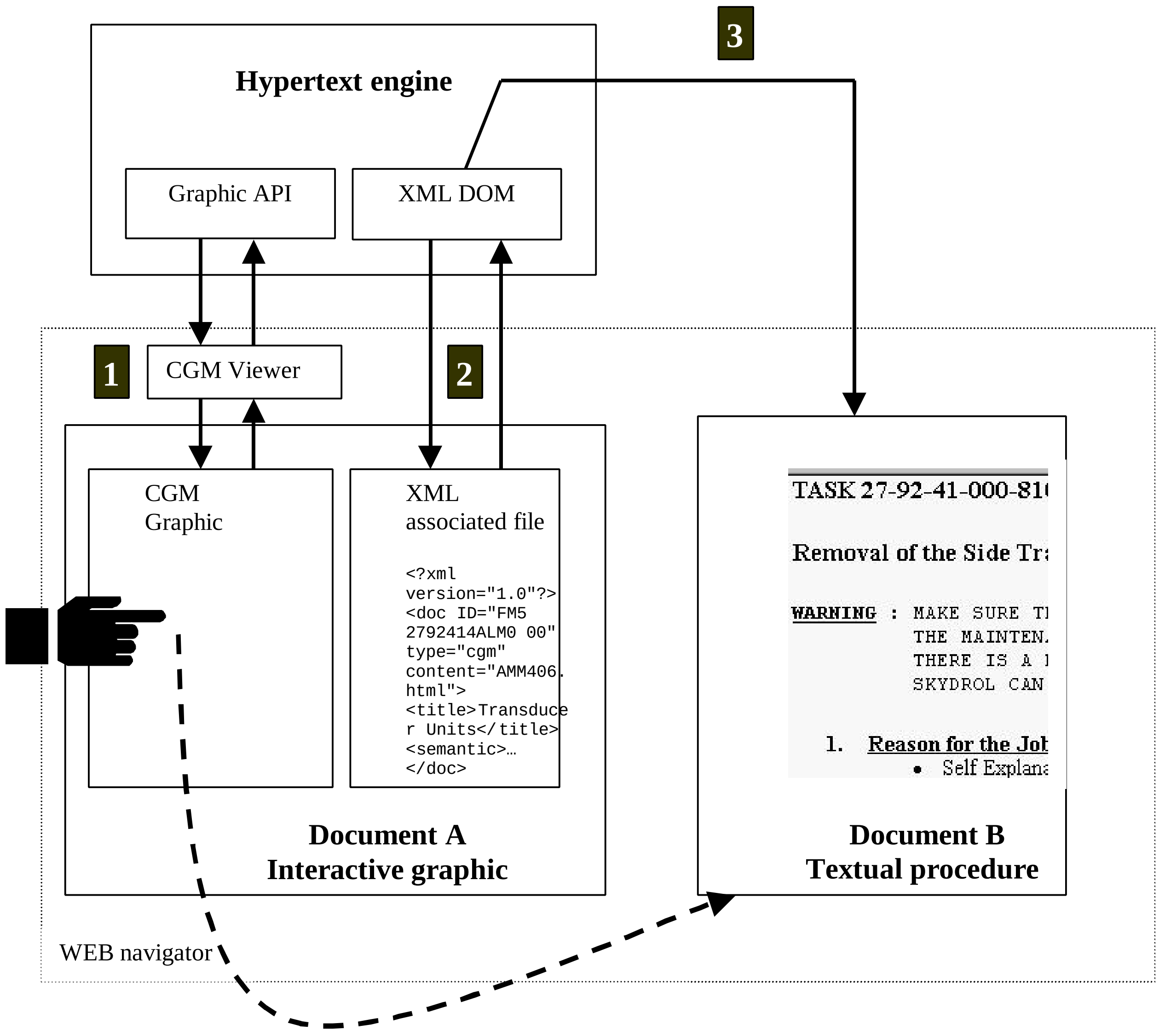
Fig 3: Extraction of data

Intelligent Graphics architecture

Having seen the functionality offered by the mock-up, we will present the principles of the mock-up which enable this functionality to be implemented.

The mock-up manipulates graphics specified in conformance with the results of the study, a CGM Version 4 associated to an XML companion file. This latter includes the ids, the structure and the semantic of each object.. The XML files are manipulated via a DOM (Document Object Model) API.

Let’s see the architecture and the general working of the mock-up with an example. In this latter, the user follows a hypertext link which links a graphical object (Document A) to a textual procedure (Document B).



```
<?xml
version="1.0"?>
<doc ID="FM5
2792414ALM0 00"
type="cgm"
content="AMM406.
html">
<title>Transduce
r Units</title>
<semantic>…
</doc>
```

Fig 4: The mock-up architecture

The working of the mock-up is as follows :

Step 1 : When the user clicks on the CGM graphical object, the event is sent to the hypertext engine with the object identifier.

Step 2 : The hypertext engine interrogates the XML file to extract the data and the links associated to this object.

Step 3: The hypertext engine commands the display of the textual procedure which is the target of the link.

In terms of languages and tools, the mock-up was developed in JavaScript which enables quick development. It uses an Internet navigator (Microsoft Internet explorer 5.0. version beta 2) which offers interesting XML possibilities and a viewer CGM, (ActiveCGM browser from InterCAP). Vbasic script has been used to program orders to the viewer.

The work on this mock-up and the previous study have opened several issues:

The first issue concerns the consultation ergonomics. Currently, the interactive graphics offer new possibilities (selection, navigation, query, extraction). These new functions must be accessible by the user through an interface as simple and natural as possible.

The second issue is linked to a job study on the content of the graphics : How we will model the graphics ? Do we need one or several types of modeling depending on the types of graphics ? Which navigation links do we need to implement? Which are the most pertinent search criteria?

It is also important to study the possibilities to upgrade the creation environment of these new graphics.

The need for a standardized and powerful graphics API has been highlighted during the development of the mock-up.

Two of these open issues, the companion file and the CGM Viewer API Considerations will be presented in a more detailed way.

Companion File and CGM Viewer API

Motivation

The mock-up work illustrates the feasibility of the proposed IG architecture, and has demonstrated some interesting techniques. However, it is based on ad hoc and proprietary definitions of at least: companion file format; and, the interface between the browser (hypertext engine) and the CGM viewer.

Because a principal goal of this work is an optimal architecture that is open and interoperable, then standard, open specifications of the companion file and browser-viewer interface (the "graphics API") are necessary.

We consider both of these to be significant future work projects, but we address some of the basic principles now, in particular in furtherance of a suitable standard "graphics API".

Companion File

CONCEPTS

A **Structured Graphic (SG)**, is a CGM graphic, which is marked up with V4 Application Structures, to define objects of application interest in the metafile, but which contain no metadata (intelligence) other than the APS 'id', APS 'type', and possibly an optional APS Attribute of type 'region'.

An **Intelligent Graphic (IG)**, is a Structured Graphic together with other associated metadata (e.g., hyperlinks, search keys, etc), which may either be internal to the CGM or external (in a "companion file").

A principal conclusion of the Optimal IG Architecture study is: the intelligence of an

Intelligent Graphic should be external to the CGM, segregated into a separate "Companion File" (coded in XML).

Goals of Externalizing Intelligence

The principal goals of externalizing the metadata from the CGM are to:

1. Improve the maintainability of the IG, by separating the relatively volatile metadata (intelligence) from the less volatile graphics (SG).
2. Increase the re-usability of the graphic, by a allowing multiple distinct sets of metadata (intelligence) to be associated with a single graphic.

These two goals, along with the interoperability goal, raise some key issues about a standardized companion file.

A Key Issue

We consider that the detailed discussion of a standard companion file is beyond the scope of this paper (for details see, for example, "Graphical Hotspot Definition - A Common ATA/AECMA Approach", Cruikshank & Zimmerman, 1999).

However some issues about companion file format affect the overall system architecture, and in particular the graphics viewer API specifications.

One principal issue illustrates this: Should the companion file replicate the structure and hierarchy that exists within the SG, or should it only associate the appropriate properties and metadata with the objects in the SG?

Replicating the whole hierarchy increases maintenance problems of the IG, and compromises the maintainability goal (above) of the companion file architecture. On the other hand, if the companion file does not replicate structure, then it is required that the browser (hypertext engine) be able to obtain this from the CGM instance (via the CGM viewer, presumably).

About this issue we conclude: In an optimal I.G. architecture, a standard companion file should not replicate hierarchy, but rather should only associate intelligence properties (metadata) to the objects in the SG, by reference to the IDs of those objects.

This leads us to the API requirement: The graphics API must have sufficient power to expose all structure and hierarchy of SG instances, whether the IG metadata is external (companion file) or internal.

CGM Viewer API

CGM Viewer API Considerations

In order to achieve the three main goals of an optimal architecture, navigation, query, and extraction, a powerful graphics API is required. Furthermore, in order to demonstrate interoperability a standard API is required.

A conventional API based on interface functions only may seem like a good idea but it is most likely proprietary, not very flexible, and also doesn't quite fit into the fast growing world of object-orientation.

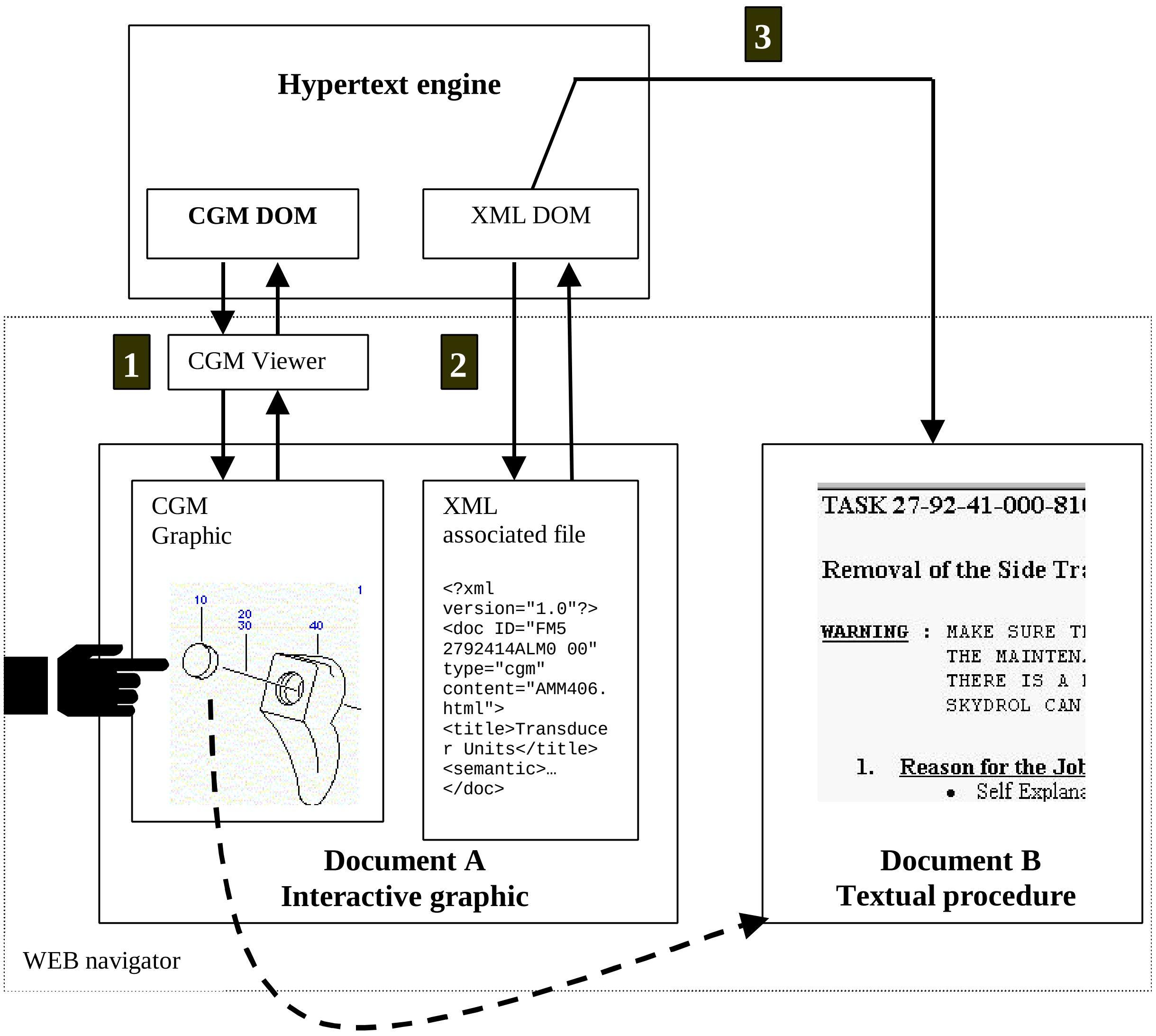
The solution we propose offers the following key advantages:

- Profile independence
- Internal and external intelligence
- Exposes the entire hierarchical structure of a CGM
- Off-the-shelf web browser support
- Highest flexibility for implementing additional functionality through automation

Considering the environment CGM viewers are likely to work in — Web browsers, office applications, etc. — and the high-level programming techniques used in these environments — automation through JavaScript, JScript, VBScript, ECMA Script, and VBA — a similar approach seems to be best suitable.

Taking Web browsers and HTML as an example, a big effort has been made to programmatically access each individual element of an HTML document. First results were limited object models implemented by the 3rd Web browser generation. These models have been extended to the, unfortunately still proprietary, Document Object Models currently supported by the 4th Web browser generation. An effort to standardize the Document Object Model for HTML and XML has resulted in the W3C recommendation "Document Object Model (DOM) Level 1 Specification" [W3CDOM].

With this proposal, we would like to initiate open discussions with the goal of creating an equivalent standardized specification for CGM, the "CGM Document Object Model".



```
<?xml
version="1.0"?>
<doc ID="FM5
2792414ALM0 00"
type="cgm"
content="AMM406.
html">
<title>Transduce
r Units</title>
<semantic>…
</doc>
```

TASK 27-92-41-000-81

Removal of the Side Tr

WARNING : MAKE SURE T
THE MAINTEN.
THERE IS A
SKYDROL CAN

1. Reason for the Jo
• Self Explan

Fig 5: An extended architecture

The illustration above presents an extended architecture based on the standardized Document Object Model for XML and our proposed Document Object Model for CGM. This architecture fulfills all needs for the mock-up, plus adding a new level of flexibility and generalization by introducing a standard two-way graphics interface.

CGM Document Object Model

The CGM Document Object Model allows direct, programmable access to individual components of CGM graphics. This access, combined with an event model, allows the CGM viewer to react to user input, and display different views of the graphical content. The CGM DOM puts sophisticated interactivity within easy reach without any profile dependency. With a CGM DOM compliant viewer, there would be no need for customizing viewers in order to support different profiles.

WHAT IS THE OBJECT MODEL?

The object model is the mechanism that makes CGM programmable. The object model builds on functionality similar to that being used by Web authors creating Dynamic HTML for Internet browsers.

The initial object model draft provides access to the intelligent structure model of a CGM graphic. This means that every CGM APS element in the graphic can have a script behind it that can be used to interact with user actions and change the view of the graphic content dynamically. This event model lets a document react when the user has done something on the graphic, such as move the mouse over a particular element, or click a mouse button. Each event can be linked to a script that tells the viewer to modify the view of the graphic on the fly, without having to go back to the server for a new file. The advantages to this model are that authors will be able to create dynamic Web sites with interactive graphical content, the basic requirement for sophisticated Interactive Electronic Technical Manuals (IETM).

Let’s take a look at a simple example:

```
...
<object id="viewer" src="test.cgm" type="image/ cgm"
 width=100 height=100>
</object>

<script language="JavaScript">
if (viewer.metafile.title != "")
  alert("The title of this metafile is " +  viewer.metafile.title);
}
</script>
...
```

This piece of HTML/JavaScript code demonstrates how to access the CGM DOM in a Web browser via scripting. The script reads the **title** property of **metafile** object contained in the **viewer** instance and displays the result, if not empty, in a message box. The whole structure of a CGM file could be extracted this way and by specifying scripting functions for the events exposed by the objects of the CGM DOM the next level of sophistication is reached. The CGM has become interactive. It can react to user input by executing the specified scripts.

The proposed solution for a flexible, dynamic and standardized graphics API can be summarized by the following formula:

Object Model + Event Model + Object Methods = API

SCOPE OF THIS DRAFT

The CGM Document Object Model represents a substantial project that we have just started working on. The present, preliminary concept can be found in further detail in the appendix of this document.

This draft does not attempt to offer a complete model for accessing all aspects of a CGM file, which has to be considered as the ultimate goal. Instead, it focuses on the "intelligent" graphic object structure expressed by V4 Application Structure (APS) elements. With this limited, but important, subset of a complete CGM DOM, it will be possible to implement the three main goals of an optimal architecture, navigation, query, and extraction, outside (and therefore independent of) the viewer.

Conclusion

This paper has presented a viable, conceptual architecture for ***Intelligent Graphics*** using current standards, and has illustrated concretely how this architecture can be implemented using currently available technologies. The demonstration of the architecture in a mock-up experiment illustrates some key issues, which must be further addressed in order that the conceptual architecture can be realized in an open, interoperable, and implementable standard architecture for Intelligent Graphics. We have proposed a solution to perhaps the most critical issue — the "standard viewer API" issue — with a CGM DOM proposal, and have illustrated the overall concepts and content of work-in-progress to fully define the CGM DOM.

Appendix A

A.1: CGM DOM Objects Reference

Note: Since this is a very early state of this project, and subject to rapid change, only the first object, the **metafile** object, and the first collection, the **picture** collection, will be described in detail. This will illustrate the concepts of the CGM DOM proposal.

The other object references have been developed to a similar level of detail, but are listed here only in brief tabular form, to indicate the scope and content of this work-in-progress.

Terminology Definitions

An **Object**, such as the CGM viewer itself, is a programming structure encapsulating both data and functionality that are defined and allocated as a single unit and for which the only public access is through the programming structure's interfaces.

A **Property** is a data member of an exposed object, such as the metafile description string of a CGM. Properties are set or returned by their names. Internally, properties are implemented as get and put accessor functions, which allows a property to be read-only, for example, by omitting the put accessor implementation.

A **Collection** is an array of elements contained by an object, such as an array of pictures contained in a CGM. Collections usually expose a length property, which returns the number of elements in the array. An element can generally be accessed via its zero-based index.

A **Method** is a member function of an exposed object that performs some action on the object, such as changing the zoom factor of a picture.

An **Event** is an action recognized by an object, such as clicking the mouse or pressing a key, and for which you can write code to respond. In Automation, an event is a method that is called, rather than implemented, by an Automation object. Reporting an event to a hosting application is often referred to as “firing” an event.

metafile

Properties	Collections	Methods	Events
title	pictures	...	onload
version	...		onunload
description			...
picture			
location			

...			

DESCRIPTION

Represents the Computer Graphics Metafile in a given CGM viewer. You can use the metafile object to retrieve information about the metafile, to examine the pictures within the metafile, and to process events.
The metafile object is available at all times. You can retrieve the object by applying the **metafile** property to a viewer object.

EXAMPLES

The following example defines a function that displays a JavaScript alert (message) box. It then assigns this function to the **onload** event of the metafile object contained in the CGM viewer object with the id “viewer”. The viewer fires the **onload** event after loading a metafile and causes the browser to execute the loaded function, which displays the the string " Metafile has been loaded." in a message box.

```
function loaded()
{
  alert("Metafile has been loaded.");
}

viewer.metafile.onLoad="loaded()";
```

PROPERTIES

title	The BeginMetafile string.
version	The MetafileVersion element.
description	The MetafileDescription string.
picture	The currently displayed picture object.
location	The URI of the metafile.

COLLECTIONS

pictures	A collection of all picture objects contained in the metafile.

METHODS

(*none so far*)

EVENTS

onload	Fires immediately after the viewer loads the metafile.
onunload	Fires immediately prior to the metafile being unloaded.

picture

Properties	Collections	Methods	Events
title	aps	load	onload
scalemode	...	...	onunload
vdcextent			onmouseover
...			onmouseout
			onmousemove
			onclick
			ondblclick
			...

aps

Properties	Collections	Methods	Events
id	attributes	...	onmouseover
type	aps		onmouseout
...	...		onmousemove
			onclick
			ondblclick
			...

attribute

Properties	Collections	Methods	Events
name	...	...	...
value			
...			

sdr

Properties	Collections	Methods	Events
...	sdritems	...	...
	...		

sdritem

Properties	Collections	Methods	Events
type	sdritemelements	...	...
...	...		

sdritemelement

Properties	Collections	Methods	Events
value	...	...	...
...			

A.2: CGM DOM Collections Reference

pictures

Syntax	Properties	Methods	Applies to
object.**pictures(**index**)**	length	...	metafile
	...		...

DESCRIPTION

Is a collection of all picture objects contained in a metafile.

SYNTAX

object.**pictures(**index**)**

Parameter	Description
object	The metafile object.
(*index*)	Optional. An integer or a string specifying the index value of the object to retrieve. Integer indexes are zero-based, meaning the first object in the collection has index 0. A string index is valid only if the string is an identifier of a picture in the metafile.

REMARKS

The returned objects are of the type **picture**. The pictures collection can be used to query information about all pictures of the metafile. Note however, that in order to improve performance and reduce memory footprint the collections of the picture objects will only be available for the currently loaded picture. In other words, you will only be able to retrieve picture descriptor information for the pictures not currently loaded by the viewer.

PROPERTY

length	Returns the number of elements in a collection.

METHODS

(*none so far*)

APPLIES TO

metafile

aps

Syntax	Properties	Methods	Applies to
object.**aps(**index**)**	length	…	picture
	…		…

attributes

Syntax	Properties	Methods	Applies to
object.**attributes(**index**)**	length	…	aps
	…		…

sdritems

Syntax	Properties	Methods	Applies to
object.**sdritems(**index**)**	length	…	sdr
	…		…

sdritemelements

Syntax	Properties	Methods	Applies to
object.**sdritemelements(**index**)**	length	…	sdritem

	…		…

Bibliography

[*CGM*O98] ***CGM*** Open GroupWEB site: http://www.CGMopen.org

[HENM93] Henderson, Lofton R, Mumford, Anne M, The ***CGM*** Handbook, Academic Press, 1993

[IGREQ22] ***ATA*** Graphics Working Group, ***ATA*** 2100 specifications, Chapter 3.3.3 "***Intelligent Graphics*** Requirements", ***ATA*** (Air transport Association) 2100 Specifications, Revision 1998

[IGEX24] ***ATA*** Graphics Working Group, ***ATA*** 2100 specifications, Chapter 3.3.4 "***Intelligent Graphics*** Exchange", ***ATA*** (Air transport Association) 2100 Specifications, Revision 1998

[ISO8632] ISO (International Standards Organization) Information technology - Computer graphics - Metafile for the storage and transfer of picture description information ISO/IEC 8632 -1992- 1994

[W3CDOM] W3C "Document Object Model (DOM) Level 1 Specification" web site: http://www.w3.org/TR/REC-DOM-Level-1