

Adobe FrameMaker and ArborText Adept/Epic product comparison

This document summarizes questions and answers posted on the FrameMaker user lists as well as replies sent to me privately. The identity of some respondents has been removed at the request of the authors. The purpose of this document is only to inform those who may be curious about the two products with the objective of freely sharing information.

The identities and original contact information for public respondents appears as it was received in 1999 and 2000. Obviously, changes are likely to have occurred since then.

Now that I've had well over a year's worth of experience with Arbortext products, I have some of my own opinions on the subject. Unlike many of the responses shown below, I have no experience with FrameMaker+SGML.

Our publishing group decided on the Arbortext products for various reasons:

1. We can work using the native SGML all the time without an import/export process.
2. Our SGML database exists, as much as possible, without any footprints from proprietary tools.
3. Our development team had previous exposure to earlier versions of Adept.
4. FDK customizations were perceived as too expensive and proprietary.

A FrameMaker+SGML expert could probably refute all four points. Every software product leaves footprints. Both products can be largely customized with PERL and both require either FDK development (for Frame) or ACL development (for Adept/Epic).

Ultimately, other issues such as available expertise, budget, calendar time, project requirements, etc. will have a larger impact on the product decision than any of the views expressed herein.

Jason Aiken

Posted March 25, 1999

Greetings Framers,

We've heard a lot about the Frame vs. Word debate, but I'd be interested in hearing a debate about what seems to be the next level of argument --- Frame+SGML vs. ArborText.

Let's say that your company has made the jump to from clipper ships to space travel by abandoning Word and using Frame. What happens when you want to go to warp speed using SGML? Does the editor really matter?

From what I've seen and heard about SGML, proprietary software shouldn't matter. However, folks entrenched in SGML tend to see ArborText tools as the default. Can anyone, especially anyone from Adobe, dispute this notion? Is the SGML commitment going to be part of Adobe's goals in five years? Ten years?

I'm as hardcore a Framer as anyone else out there, but I'm afraid that the public misperceptions about Adobe and Frame may be applicable beyond the Frame vs. Word argument. Powerhouse publishing on a global enterprise scale could spell huge dollars for Adobe, as many corporations seek to modernize their publishing systems. InDesign shows a huge Adobe commitment to one aspect of the publishing market, but what about the SGML segment?

Adobe's website offers some case studies on Frame+SGML from Hitachi and others, but I'm interested in hearing a debate here. What are the strengths of Frame+SGML over Arbor Text? Are the two tools really moving closer together? If SGML isn't supposed to be proprietary, then why does it matter? Why might Frame+SGML be considered a weaker product?

I believe Frame+SGML must be an excellent product simply because of my experience with Frame sans SGML. I'm sure it has its idiosyncrasies, like any product, but please share any applicable thoughts.

Raising shields,
Jason

Response posted March 25, 1999

=====

I'm not familiar enough with the Arbor Text product to identify its advantages and disadvantages, but I can describe what I believe are the advantages and disadvantages of FM+SGML.

A. Main Advantages of FM+SGML as an authoring tool:

1. By virtue of context-sensitive format rules in the Element Definition Document (EDD) you get:
 - a. WYSIWYG authoring
 - b. Fully formatted, DTP-quality printed output without having to develop a FOSI, DSSL, or XSL stylesheets
2. Element catalog indicates validity of all possible elements that can be inserted at the current insertion point.
3. Element Merge/Split/Unwrap/Wrap/Change capabilities are excellent.
4. Interactive structure view facilitates text editing, editing/viewing of attribute values, insertion point positioning, navigation, etc.
5. Show Element Context utility displays the hierarchical context of any selected element, along with the format rules for that element. Any format tag appearing in the element's format rules can be selected so that you can examine the complete formatting details for the tag in the applicable designer dialog.
6. Document Validation Tool is excellent. Finds all instances of invalid or incomplete structures, as well as attributes having invalid values or missing values that are required, highlights the offending element in the structure view, and describes the nature of the problem.
7. You can leave required elements out (adding them later), and successfully save is as an FM+SGML doc.
8. Parses SGML document instances against its DTD, and produces an error log describing each anomaly.
9. Built-in structure generator for converting unstructured FM docs to structured docs, however, developing a structure rules table for this utility is difficult, and the results are usually problematic.

B. Disadvantages of FM+SGML as an authoring tool:

1. Authors have no control over, or knowledge of, the names that are assigned to entities (e.g., graphics, equations, text entities). In the case of native graphics, equations and multi-faceted graphics, even their exported filenames are unknown to the author. This is because the entity and file naming conventions are established by the import/export read/write rules. In the case of native graphics and equations, authors have no control over the graphic format in which they are written out on export to SGML, because, once again, this is determined by the import/export read/write rules.

2. Authors cannot create FM+SGML text fragments that are exportable as SGML text entities. If such a fragment does not begin with an element that is valid at the highest level, it is not exportable. If the fragment begins with an element that is valid at the highest level, FM+SGML exports it as an SGML document instance with an internal DTD subset, making it unusable as an SGML text entity. Consequently, although an FM+SGML structured doc can contain such fragments, imported by reference as text insets, that can (by means of read/write rules) be exported as entity references, there is no way in FM+SGML to create and export the referenced SGML text entities.

C. Advantages of FM+SGML in importing/exporting SGML/XML

None

D. Disadvantages of FM+SGML in importing/exporting SGML/XML

1. See also items B1 and B2 above.
2. Development of a complete SGML import/export application for a complex DTD/EDD is often a lengthy and costly process, usually requiring considerable API development with the FDK.
3. FM+SGML's representation of graphics and equations by means of a set of attributes may not conform to the representation of these entities in an existing DTD. In that event, it may not be possible to successfully import or export those entities.
4. For tables, FM+SGML only supports the CALS table model.
5. No tools are provided in FM+SGML for developing a screen FOSI, DSSL, or XSL stylesheet to permit on-line viewing of SGML/XML document instances produced by FM+SGML.
6. FM+SGML cannot create, import, or export SUBDOCS, nor can it preserve on import, or successfully export, marked sections.
7. FM+SGML has only limited capability to handle SGML processing instructions (PIs) on import or export.
8. An FM+SGML book file (required for exporting a set of FM+SGML files as a single SGML document) can have only one level of documents, whereas SGML documents can have more than one level of nesting. On export to SGML, FM+SGML exports each file in the book as an external SGML text entity.
9. When importing an SGML document, FM+SGML cannot subdivide it into SGML documents in a book, unless the SGML document contains special (FM+SGML-unique) processing instructions.
10. FM+SGML cannot successfully export or import cross-references to documents that are external to a book, or, in the case of an individual FM+SGML file, any external cross-references.
11. FM+SGML is currently incapable of properly importing or exporting well-formed XML containing Resource Description Frameworks (RDF), and resources (e.g., graphics, other documents) identified by means of Universal Resource Identifiers (URIs). The export capability provided in version 5.5.6 is nothing but a variation on V5.5's HTML export capability, requiring the mapping of paragraph tags (not elements). If the structure is complex, this mapping operation can be very difficult, if not impossible.

12. FM+SGML does not implement the XLink standard for XML links.

13. FM+SGML does not currently implement new language options (e.g., musical, mathematical, and chemical notation) that are possible in XML.

| Nullius in Verba |

Dan Emory, Dan Emory & Associates
FrameMaker/FrameMaker+SGML Document Design & Database Publishing
Voice/Fax: 949-722-8971 E-Mail: danemory@primenet.com
10044 Adams Ave. #208, Huntington Beach, CA 92646

Response posted on March 26, 1999

Jason,

My company has standardized on both FM+SGML and Arbortext Adept using a common DTD. Our documentation and training division has about 1100 writers in the US and Europe.

The decision to support both platforms came about for a variety of reasons. Among those reasons, but not limited to them are:

- a large base of existing Frame users
- the need to continue to support Frame projects that will never (for cost reasons) migrate to SGML
- an aversion to "take 3 steps backward" and give up a WYSIWYG interface.
- a perceived need to create "pure" SGML (Adept)

We have been going about this for at least 3 years and are still not ready for prime time. Much of this is because we are creating our own DTD. If you adopt an existing DTD, the route is much easier.

As for comparisons:

- Adept produces "pure" SGML. That's what Arbortext will say, but the truth is that the only way to produce pure SGML is to use an ASCII editor and hand code it. Adept does, however, come closer to this than FM does.
- Frame uses an output filter (like saving as PDF) to produce SGML and requires an intermediate program called an EDD to define the read/write rules for producing the SGML output. The output is only as good as the person that created the EDD.
- Adept only produces SGML, you have to create an entire production system using something like XYvision (not sure of spelling) to get any useful output. I believe Arbortext offers its own production system, but we found it to be inadequate for our purposes. The only output directly from Adept to paper is via a screen FOSSI which I believe is included with the editor. It is fine for technical reviews and such, but useful for little else.
- With FM+SGML you get to keep all the regular FM outputs such as PDF, HTML, XML etc. Since a template is involved, you also get to print to paper on day one.

One of the big deal claims of Arbortext was that, unlike FM, you did not have to worry about your people spending time on formatting since the interface was not WYSIWYG. Of course that did not stop the ballyhoo when they announced a "new and improved" interface that was much more WYSIWIG!

As a long time FM user, I am biased toward FM and make no bones about it. Having spent my time using systems such as troff and mm macros and using editors like the venerable "vi", I have no desire to go back to them. On the other hand, we have folks that went straight from troff to Adept and think it is wonderful.

Hope this helps some. I'm sure others may have different views or experiences.

Greg Henderson
Lucent Technologies Inc.
Voice: 614 860-6176
FAX: 614 868-3315

Response posted on March 26, 1999

Hi Jason,

This debate's been going on for quite a while! For an historical perspective, a search of the archives of comp.text.sgml should probably generate quite a few hits. The real debate is that Arbortext does not convert the source SGML/XML files into any other format, so you are continually working in true SGML/XML mode. Adobe, on the other hand, has a couple of tools that actually convert a source SGML file into Frame binary, and then you have the option of saving the file in Frame format or exporting back out to SGML. The import/export process requires some additional setup pieces to be created/maintained.

I don't know what industry you work in, but there are lots of Frame+SGML users out there that are happy; there are others that are not. A lot of it has to do with a well-designed SGML/XML architecture that users understand and use properly.

I'd be happy to discuss this further with you if you'd like, send contact info.

+++++

Mary P McRae

Principal XML/DMS Technologist

Document Management Solutions, Inc.

<!ELEMENT DMSi (tools, integration, performance, productivity)>

<http://www.dmsi-world.com>

mary.mcrae@dmsi-world.com <mailto:mary.mcrae@dmsi-world.com>

voice: 603 924 6578 fax: 603 924 6417

+++++

Anonymous response March 26, 1999

<snip name="Jason">

... What are the strengths of Frame+SGML over Arbor Text? ...Why might Frame+SGML be considered a weaker product?

</snip>

ArborText is considered to a stronger, more "pure" SGML authoring system, whereas FM+SGML is considered to be the better formatter. Frame has its problems exporting SGML.

<snip name="Greg">

- Adept only produces SGML, you have to create an entire production system using something like XYvision (not sure of spelling) to get any useful output.

</snip>

Xyvision is a ****much**** better formatter than FrameMaker. Of course, for the money... it should be! ;-)

Why pick one over the other. Depends on the project situation. If you have tons of data to format and distribute, then it makes sense to automate. You want to get as close to a black box situation as possible. What does that mean? You need an authoring system, a conversion tool, and a pagination engine. Tools, time, and high-level resources make this a serious time and money commitment. If you have that commitment, then you'll want the best tools. That's why Arbortext, Omnimark, and Xyvision all make sense in this situation.

Okay, now for some Frame-positive comments.

Why use Frame+SGML? Many reasons. Tech writers are comfortable with it. Development isn't easy, and there will most likely be a need for custom API's, but it's certainly not as complex as an Omnimark/Xyvision solution. You have the ability to control pagination while authoring. The costs associated are less. And, simple delivery needs... HTML, XML, PDF... are all handled painlessly (well, sorta). The EDD takes away the need for a conversion-tool (Omnimark, PERL).

Also, while developing an EDD, you have the ability to rapidly prototype format solutions. This is especially useful to be able to spot "unspec'ed" structure... or structure that will cause problems. For this reason alone, I've actually used it to "shake out" structure-->format problems, and then go back and implement in my Omnimark/Xyvision project!

****The ability to control format while authoring is a wonderful thing... but it can be VERY dangerous, too. It's very difficult to stop an author from invalidating a doc... while you can check validity, it's extremely difficult to tell someone to not deliver the postscript, because they used a <para> instead of a <stem>, even though they format the same way... especially when the prepress department is waiting and should have plated the job an hour ago.**

And, of course, in both scenarios, there are going to be problems due to weaknesses in the tool. It's inevitable. Which one is best? I think the jury is still out in many cases. I know many people who've implemented the first approach who feel the cost was just too great... while people who use FM+SGML feel that it didn't meet their expectations. Every project has its "grass is greener" contingent.

I'm interested in feedback you get from different sectors of the market. My first inclination is that you'll get some positive feedback for FM+SGML in highly technical sectors, like Aerospace and Semiconductor... vs. information markets like Medical and Legal.

Hope this wasn't too long-winded and gives you a little bit more insight.

Good luck,
Anonymous

Summary posted on March 26, 1999

Thanks to everyone who responded on my Frame+SGML vs. Arbor Text query. Dan Emory, Mary McRae, Greg Henderson, and another person (anonymous) all answered my post. I've learned a great deal, and hope others have as well.

This debate is not over. It seems that both products may have their own collection of headaches. It isn't as clear cut as a Frame vs. Word comparison. In the SGML world, one product's strengths seem to be the other's weaknesses.

The concept of "pure" SGML seems to be the biggest detriment to market perception of FM+SGML. However, many people still prefer the WYSIWYG interface. Adept Editor has made progress in the WYSIWYG arena, while FM+SGML's new features and true power are largely unknown to the SGML novice.

Frame folks at Adobe have a double challenge. Besides expanding the market for Frame as a whole with an amazing version 6 release, they must also dispel any notions about the FM+SGML product as an "impure" SGML editor. These are difficult challenges indeed.

The bean counter making the decision on Frame vs. Word is often criticized for not knowing enough about the products in question. Thus, we have the Frame vs. Word PDFs all over the place. The argument for FM+SGML over ArborText is a bit more muddled. It is safe to conclude that the decision makers are likely not versed in any SGML product. Therefore, they may also be likely to choose an SGML product perceived as "pure."

Anyone care to sit next to me and cheer for an end to the misperceptions about FM and FM+SGML before the buzzer sounds?

Jason
perceived Frame fanatic
pure

Response posted on March 27, 1999

Jason,

As is the case with the undying Wrod vs Frame debate, the answer is probably the same: horses for courses.

I have recently been through the process of assessing SGML documentation management systems and their associated editing environments (FrameMaker+SGML vs ArborText's Adept Editor). Some of our technical authors have been using FM+SGML for about a year in a stand-alone editing environment, but, because most of my communication is internal to people who only have MS Word, I have not become a skilled FM user despite arguing strenuously to get rid of the billygates product for group authored documents.

Therefore, in comparing FM and Adept Editor in a Windows NT environment, I was looking at both from the standpoints:

- a. of a fairly naive author who needed to come up to speed in a new environment, and
- b. of a systems specialist trying to understand what was required to improve, maintain and extend our authoring environment.

My (largely subjective) conclusions on various dimensions of comparison from this exercise are:

Speed. In theory Adept should be able to render an SGML instance for display faster than FM, because Adept uses SGML as its native format, while FM has to convert the SGML elements into equivalent elements in FM's proprietary EDD (Electronic Document Definition) structure. In practice I did not see any perceptible differences on short documents, and none of our technical authors have ever complained about FM's speed to render long and complex technical manuals to and from SGML.

GUI. Both systems are quite good, giving the user several options for displaying formatted versus structural views of the document being edited. On balance, I think the FM paradigm is probably closer to the familiar billygates standard than Adept's is - which may be considered good or bad depending on your biases. FM's on-screen WYSIWYG image corresponds closely to the output page format as both are generated from the same template and styles. With ArborText, the system used to format the screen display is not related to the page formatting engine which in many cases is provided by a third party.

Interfacing tools: We are particularly concerned to interface our SGML authoring environment to an SGML object management system, so I have looked at this area fairly closely. Both applications have already developed interfaces with most of the object management systems we have reviewed, but we will inevitably require extensions to the fairly basic interfaces provided. ArborText has its own complex procedure control and scripting language, which I understand to be quite powerful for developing custom applications and modifications. Based on a web search, a number of 3rd party solutions have been developed in this language. By contrast FrameMaker provides a FrameMaker Development Kit (FDK) which totally documents its wide range of API calls, allowing add-on applications to be developed in your preferred programming and scripting languages. Personally I have not been involved with either, but my impression is that there are many more consultants with FrameMaker FDK experience than with ArborText skills. This is certainly the case in Australia. FrameMaker is supported by an Adobe office, at least two FrameMaker specialist consulting houses with major skills, and a few independents who also appear to have significant FDK skills. Adept Editor is supported by a single computer services company - but Adept support is not a significant focus of its overall business.

Output and display formatting: In the environment I work in, we will need to manage a wide variety of document types, and will have to construct DTDs, display formats and print formats for many of them ourselves. If you have FrameMaker skills, understand SGML and have some understanding of programming principles, it is not a difficult job to build the SGML application required to convert the SGML DTD to an EDD (FrameMaker's proprietary equivalent of the EDD) and then map template styles to the EDD elements. FM automates much of the work. By comparison, before you can even create a formatted Adept Editor display for a new DTD, you will have to procure a Document Architect license, which costs (at least in Australia) about 5-6 times a single seat license for Arbor Text. And once you have it, it is a very complex system to understand. After a month working with Document Architect I had made less progress setting up a user interface for a single DTD than I did in three days with FrameMaker+SGML - without any additional software requirements.

To summarise:

If you are working on long documents, if your primary delivery is SGML, and your client can provide you with working display and proofing formats for the DTDs you are required to use; or if you are a very large organisation that can afford to retain the required skills to build the necessary user and output interfaces in house, Adept Editor would probably be the better tool.

If you are a small organisation and need to build your own DTDs, interfacing and output requirements, FrameMaker would appear to offer a much more cost-effective toolkit. That is certainly the decision we have made, where the authoring environment will have to be supported by only one or two staff who have a number of other responsibilities in addition to maintaining the authoring/editing tools.

Regards,
Bill Hall
Documentation Systems Specialist
Integrated Logistic Support
Naval Projects and Support
Tenix Defence Systems Pty Ltd
Williamstown, Vic. 3016 Australia
Phone: +61 3 9244 4170
Email: hallb@tenix.com
Fax: +61 3 9244 4242

Response received on March 28, 1999

Jason,

Good to hear from you.

I have subscribed to the Framers SGML list, though there seems to be no activity there (unless I've been unsubscribed). I occasionally get a message from there. It turns out it is really meant as a Frame+SGML list, as opposed to the more general FrameMaker list. Most people on the list seem to be staunch Frame supporters, and I doubt whether you would get objective responses from there (my opinion only).

There is a newsgroup called comp.text.sgml which often discusses SGML tools. I usually check it once a week, though have not had time lately. I recommend posting there.

We checked out available SGML tools towards the end of last year. We evaluated many of them. We encountered problems with Frame+SGML that it simply could not be set up to work with our DTDs. Because Frame filters the SGML files into its own proprietary format, we lost data in the conversion that forced FM+SGML out of the running. I checked version 5.5. I have not seen or heard of any improvements since.

We are using ArborText ADEPT series for authoring and printing, and Inso's DynaText and DynaWeb for online and web publishing. Feel free to contact me again if you need more info or even help setting up your SGML applications.

Out of interest, I have set up a conversion from Word to SGML using SEMA's rtf2rdc converter and Mark-It parser. I should have a conversion from SGML to rtf ready shortly, using Sema's rdc2rtf converter. I can provide more information on request.

I am beta testing Corel WordPerfect 9's SGML features, which are built into the kernel (as opposed to plugins used with Word, for example). I, and other beta testers, are extremely impressed with the SGML capabilities in WP9, and we are considering purchasing WP in the future for our SGML authoring instead of ArborText. We are going to suggest to one of our smaller subsidiaries that their documentation department purchase WP9 instead of ArborText, for both financial and feature reasons. WP costs about 15% the price of ArborText, its authoring features come close to those of ArborText, and its publishing features are better (out of the box). It also has the on-the-fly spell checker, which ArborText are not going to implement (they told me that explicitly).

I assume you are used to buggy software if you work with FM. ADEPT is no better. WP is less buggy than its competitors, and I'm basing that statement on beta testing.

Please feel free to contact me directly for any information/advice etc. you may need. We are planning on being 100% converted to SGML by May 15. I already have half out technical writers working in ADEPT, and have basic publishing capabilities in place.

Regards,
Gershon.

Gershon Leib Joseph
DOCUMENTATION TECHNOLOGIES MANAGER
Comverse Network Systems
Tel: +972-3-645-2001
Fax: +972-3-645-4088
email: gershon_joseph@icomverse.com

-----Original Message-----

From: Jason Aiken [mailto:jason.aiken@medtronic.com]

Sent: Friday, 26 March 1999 20:26

To: gershon_joseph@icomverse.com

Subject: New SGML users list -Reply

#####

Joseph,

Please excuse the private message. Last June you sent a message to the Framers list about an SGML list starting up. Did that list ever take off?

We're debating SGML tools and may make a conversion in the next couple of years. I've been interested in comparisons between Frame+SGML and Arbor Text.

I posted a message yesterday to Framers, but was wondering if it might be more applicable to post to a general SGML list. Due to political reasons, however, I don't feel comfortable posting my name and organization to any SGML list.

I'm going to at least try to subscribe to Brad's SGML list. Maybe I can find somebody there who can bring up this topic without creating an undesirable ripple effect.

Take care,
Jason

Response sent on March 29, 1999

I think FM+SGML's strength is not only related to the WYSIWYG interface: in fact, if you still need to produce a professional highly formatted output, FM+SGML is the only tool which can satisfy your need.

I have a lot of customers, here in Italy, which began writing SGML using Arbor Text, and later asked for help to produce a "paper" version of their manuals.

So they reached FM+SGML (through us) and asked themselves why don't use it also to write SGML. I think that depends only on marketing strategies. Adobe is too concentrated to push PDF to understand what has in its hands. If Adobe would spend 10% of its effort to push Acrobat to develop FM+SGML marketing, I think FM+SGML should become the product of choices in most cases.

Bye
Patrizia

Patrizia Roberti	email: proberti@logix.it
logiX s.r.l.	ph. +39-50-541755
Via delle Belle Torri, 18	fax. +39-50-543063
56127 PISA, Italy	<http://www.logix.it>

Another question: posted in November, 1999

In March of this year, I posted a query to compare Arbor Text and FM+SGML. Several users responded with very helpful information. In fact, the information acquired here has had a profound impact on our decision making process. Special thanks to both lists and those who responded. I also would like to extend thanks to those attendees of the FrameUsers Conference who took the time to share their opinions on this subject.

Several criticisms of FM+SGML have forced my department to consider other authoring tools. Please respond on the following:

1. SGML round-tripping: Can one go from FM+SGML to SGML to FM+SGML again without corrupting any data?
2. Is it true that FM+SGML cannot create, import, export or preserve SUBDOCS? What about marked sections?
3. Is it true that FM+SGML cannot natively be used for a component based SGML repository?
4. Case studies: is anyone in the world using FM+SGML with a controlled, component-based SGML repository to publish content in multiple output formats and multiple languages?
5. Performance: on a 400 page manual with scores of graphics and somewhat complex components, is there a serious performance issue parsing to SGML from FM+SGML (does it take seconds, minutes, hours...)?

Conversely, are there major performance concerns when compiling SGML chunks into a pretty 400 page book using Xyvision or Adept Editor?

6. Implementation: is it cheaper, faster and easier to implement an FM+SGML solution with complex DTD/EDD (and possibly FDK) development than (for example) an Arbor Text, Xyvision, DTD/FOSI combination?

Summary November 12, 1999

Here's the folks who have responded so far. Thanks to all those who responded with such insightful opinions and detailed analysis.

paraphrased from "Steve Schwedland" <steve@noonetime.com>

Steve mentioned that Astoria could likely handle the situation with some custom API work and brought up a tool called Lingua for localization. He also said that FM+SGML is a great publishing engine and would work in combination with other tools.

#####

"Kevin Brown" <kbrown01@worldnet.att.net>

> 3. Is it true that FM+SGML cannot be used for a component based SGML
> repository?
>

> Dan Emory, a renown FM+SGML guru, criticizes FM+SGML for its failure to
> create FM+SGML text fragments as exportable SGML entities---"if such a
> fragment does not begin with an element that is valid at the highest
> level, it is not exportable." Is it impossible to create, import,
> export, and preserve SGML text entities (i.e. chunks) in FM+SGML?

Absolutely untrue. Chrystal Software has many installations of our SGML component database management solution integrated to FM+SGML using our software bridge. It provides component-level management and component-level editing from within FM+SGML.

- > 4. Case studies: is anyone in the world using FM+SGML with a
- > controlled, component-based SGML repository to publish content in
- > multiple output formats and multiple languages?

I am sure we can provide you with contact information, but off-list. Send me an email and I will get you in-touch with representatives.

#####

"Vic Fragnito" <Vic.fragnito@alliedsignal.com>

To get round trip FULLY COMPLIANT SGML is difficult with both FM and Adept. Adept is more native in its handling of SGML and has a greater propensity to produce effective round trip SGML. FM requires addition of anchor elements for the display of graphics whereas Adept does not. This does pose a unique concern and requires after the fact programming or an in line FDK application. Adept handles entity selection on a pull down whereas FM requires creation of a palette. I am a FM advocate but it is a challenge.

As far as paper is concerned, both the FOSI and EDD are equally complex. They require specific skills and capabilities and it is not easy. If your DTD does not exactly match the order of presentation, both products suffer and require custom programming. The same is true of Xyvision. FM is more of a WYSIWYG tool whereas Adept and Xyvision are in the background and require expenditure of funds for the developer environment.

Check into a product by Advent Technologies, call 3B2. [www.3B2.com](http://WWW.3B2.com) <<http://WWW.3B2.com>>

This tool allows WYSIWYG batch rule based output and allows manipulation of text and graphics without affecting the SGML instance. It is the best, my opinion, tool to get an SGML document out to presentation and assure round trip SGML. It corrects the deficiencies in both FM and Adept.

Component SGML management is a problem to all SGML systems. Most reuse models do not address or handle ID/IREFS for reuse. Chrystal Software (Astoria Object based DB) is the best for its reuse model but it is still deficient in the reuse model by component.

#####

"Dan Emory" <danemory@primerenet.com>

- >1. SGML round-tripping: Can one go from FM+SGML to SGML to FM+SGML
- >again without corrupting any data?

=====

Yes, if you design the entire system correctly. By "system" I mean the EDD/DTD, the import/export application, and the SGML database repository. If you are designing your own DTD/EDD, it can be done. If you're stuck with an existing EDD that's difficult (e.g., MIL-M-38784), you're likely to have problems that can only be resolved with costly FDK development

=====

- >It is imperative that we get 100% consistency during each
- >edit/conversion cycle in order to work properly with translation memory
- >software. I understand FM+SGML creates SGML compliant data, but if you
- >ran a "diff" on the FM+SGML file before and after parsing to SGML and

>back, is it always the same? Is data such as variables or conditional
>text ever lost?

=====

Have you considered using TRADOS S-Tagger for translations? I've successfully round-tripped FM+SGML structured docs in MIF format through S-Tagger, eliminating the need to export to SGML for translation. However, for many languages, there are problems with misconversion of certain translated characters. This is where Unicode would help. No one seems to know whether the next release of FM+SGML will fully implement Unicode. Without Unicode, the most basic requirement to make translation work is to use the same font for the printed output that was used for doing the translation. Since most translation houses use Word 2000, that means TrueType fonts. Also, there are 5 locked code points (i.e., ANSI numbers that are unavailable) in FM+SGML, and these code points are needed in some central european and cyrillic languages. There's no workaround. Xyvision has no locked code points, and should (if not already then shortly) be Unicode-compliant. For that reason, Xyvision, not FM+SGML may be your best print engine.

If I were you, I would rule out any DTP/word processor/print engine that will not be fully Unicode compliant. Unless you can get a firm commitment from Adobe that the next FM+SGML release will be Unicode-compliant, I would advise you to remove FM+SGML from consideration.

=====

>2. Is it true that FM+SGML cannot create, import, export or preserve
>SUBDOCS? What about marked sections?

=====

Yes, those limitations still exist.

=====

>3. Is it true that FM+SGML cannot be used for a component based SGML
>repository?

=====

False! Chrystal Astoria has a bridge to FM+SGML, and I have seen it used successfully with FM+SGML as a component-based SGML database repository.

=====

>Dan Emory, a renown FM+SGML guru, criticizes FM+SGML for its failure to
>create FM+SGML text fragments as exportable SGML entities---"if such a
>fragment does not begin with an element that is valid at the highest
>level, it is not exportable." Is it impossible to create, import,
>export, and preserve SGML text entities (i.e. chunks) in FM+SGML?

=====

I should modify that statement somewhat. In order to export individual text insets as entities, each text inset source must be in a separately named flow within an FM+SGML file, and must be wrapped in an element named SGMLFragment. This element must be added to your EDD, but not to your DTD. In the EDD, the SGMLFragment element must be defined as valid at the highest level, with a general rule of <ANY>. When you export each individual text inset, FM+SGML writes it out as a text entity (without an internal DTD subset) with the filename you specify. Then, in the DTD, you must have an entity declaration for each such text entity of the form:

<!ENTITY entname SDATA "[SOI]" >

Where the bracketed SOI defines an indirect SOI (Storage Object Identifier) whose "lookup location" is specified in the read/write rules and/or in an entity catalog.

Now, you can you insert entity references to these text entities in your SGML documents.

However, if you are creating/editing your docs in FM+SGML and then exporting them to SGML for repository storage, you want to import individual FM+SGML text insets by reference into various places in your FM+SGML documents. Consequently, when you update the text inset itself, all documents that use the text inset will be updated. But, in order to replace those text insets within your documents with entity references when those documents are exported, you must create a read/write rule for each such text entity, having the following form:

```
entity "SOI"{  
is fm text inset "filename"  
in body (or reference)flow "flowname";  
}
```

Where:

"SOI" is the same name (in quotes) that appears in the bracketed SOI in the corresponding SGML entity declaration described previously.

"filename" is the FM+SGML document file that contains the text inset

"flowname" is the name of the text flow within the document file containing the text inset.

You should not include a directory path for the filename in these read/write rules. Instead, specify the directory locations of files containing text insets as entity search paths in the SGML application definition.

Now, the problem with this approach is that, on export of a document containing imported-by-reference text insets, FM+SGML writes an entity reference for each instance of one. Then, when you re-import the SGML document instance into FM+SGML, the entity references are replaced with the original FM+SGML text insets, not the SGML text entities that you exported separately as described above. So, if the text inset was updated in FM+SGML after the last time the text inset was exported to SGML as a text entity, or if the SGML text entity was edited in an SGML text editor, then the text inset in FM+SGML and the corresponding text entity (stored in the SGML repository) will not be the same.

There are other problems with text insets, including:

a. If a text inset contains variables, graphics, or other components that are replaced by entity references on export to SGML as a text entity, the DTD must contain declarations for all such entities, because a text entity has no internal DTD subset where these entities can be declared.

b. Cross-references to or from text insets can create some nasty problems.

=====

>4. Case studies: is anyone in the world using FM+SGML with a
>controlled, component-based SGML repository to publish content in
>multiple output formats and multiple languages?

=====

I would think so. You should check with Chrystal.

=====

>5. Performance: on a 400 page manual with scores of graphics and
>somewhat complex components, is there a serious performance issue
>parsing to SGML from FM+SGML (does it take seconds, minutes, hours...)?

>Conversely, are there major performance concerns when compiling SGML
>chunks into a pretty 400 page book using Xyvision or Adept Editor?

=====
There's no substitute for real-world testing, using documents that are typical in size and content.

=====
>6. Implementation: is it cheaper, faster and easier to implement an
>FM+SGML solution with complex DTD/EDD (and possibly FDK) development
>than (for example) an Arbor Text, Xyvision, DTD/FOSI combination?

=====
Obviously, with an SGML text editor plus Xyvision, you don't have to worry about developing an EDD or an SGML import/export application, thus these costs are unique to FM+SGML, and they can be considerable. These added costs may (at least partially) be offset by the (arguably superior) way to define formatting in the EDD and the FrameMaker template. Thus, if SGML docs are always imported into FM+SGML for formatting, you eliminate the Xyvision/FOSI/DSSL development costs.

=====
>BACKGROUND: We are developing a documentation factory. We must automate
>every aspect of publishing from authoring to translation to final output
>in such a way that allows scaling and guarantees 100% compliance with a
>controlled, component-based SGML repository in at least a dozen
>languages. If we can't make FM+SGML work, without exorbitant costs,
>delays or hassle, for items 1-3 listed above 100% of the time, we will
>have to abandon FrameMaker.

=====
You've got a tough decision to make. You first must define when and why a cost becomes "exorbitant". High one-time development costs may not be exorbitant if they yield a high Return on Investment (ROI) because of large reductions in labor costs, and/or improvements in the management of information.

=====
| Nullius in Verba |
=====

Dan Emory, Dan Emory & Associates
FrameMaker/FrameMaker+SGML Document Design & Database Publishing
Voice/Fax: 949-722-8971 E-Mail: danemory@primenet.com
10044 Adams Ave. #208, Huntington Beach, CA 92646

#####

###"Patti Nolan" <tsisink@earthlink.net> ##### Part one #####

>6. Implementation: is it cheaper, faster and easier to implement an
>FM+SGML solution with complex DTD/EDD (and possibly FDK) development
>than (for example) an Arbor Text, Xyvision, DTD/FOSI combination?

When you noted Xyvision I felt I needed to respond. I've been typesetting for 27 years, I've worked on several main frames; Penta, Xyvision, Miles. A year and a half ago my company decided to combine its editorial services, The Smart Editor (SES) and FrameMaker+SGML 5.5.6. I found the major differences to be:

- 1) Xyvision translation tables inable you to avoid most if not all programming needs that you will with Frame require.
- 2) With Xyvision you can rearrange text, search on text content, with Frame it will require programming intervention.
- 3) Speed and memory: Xyvision there is no problem, with Frame there is a memory leak, so you do have to be careful with how large your files are pior to composition.

I personally prefer Xyvision (old dogs ...), however, we have found that there is nothing that we can't do with FrameMaker+SGML as long as there is programming intervention. As for cost, FrameMaker+SGML is by far a better buy. But if you have a great deal of graphics (EPS), and your final output is a print product then, you will probably want to have Frame on a Mac platform. We have it on NT 4. Xyvision does support SGML with FrameMaker which would give you the best of both worlds. You should probably look into that combination.

I hope this has helped.

Regards,
Patti
tsisink@earthlink.net

PS I really like working with the EDD

#####

###"Patti Nolan" <tsisink@earthlink.net> ##### Part two #####

There are two other items that may be an issue for your company.

1) Tabular: If you have a lot of large tables that need to break automatically to the next page Frame does not do that automatically it will require programming intervention, Xyvision does. If your page layout (template) is 1 column and your tables could be 2-5 columns (not within the table, but should break evenly into columns) with Xyvision it's a simple macro <pc;2> could be defined to break the table into 2 equal colmns on the page. With Frame I don't believe that it is that easy, but look to Adobe about tabular before committing to Frame+SGML.

2) Fonts: If you are planning on putting Frame+SGML on an NT platform you will encounter problems (and need programming intervention) with fonts if you have the following situation: Lets say your using Helvetica Bold and Times both your document uses a lot of accented characters that do not esist on either font, so you've created two new fonts for this problem. Lets say your new fonts have an acute over a lowercase a (á). You now have 4 fonts Helvetica Bold, Helvetica Bold Accent, Times, Times accent.

<Head>Exámple: <Para1> Both of these require áñ accent over the lowercase a in Example and the a in an.</Para1></Head>

In Frame what you will have to do is for your Accent font create a variable in your template, such as aacute & aacute (for reg Helv. Bold & Times) HBaacute for lowercase, HBAacute for uppercase, and the same for Times, Taacute and TAacute. My assumption is that your new fonts do not have the same hex positions for an a with acute. Then programming will have to write a dll that will test to see if the variable <aacute> or <Aacute> is in <Head>, if in head change to <HBaacute> or <HBAacute>

If you're on a Mac with Frame+SGML fonts should not be a problem. I know that my example might be a bit confusing, so feel free to call me.

All of the projects that I do on Frame are completely automatic from start to finish, which does require programming. Any of the projects that I do on Xyvision can be completely automatic from start to finish with a simple tag line at the beginning of the ASCII file.

On all of the items I've listed on both emails I want you to understand that I really do like working with Frame+SGML and if what your needs are to maintain a database than I feel you are looking at the right software for the price.

#####

Anonymous response sent on November 12, 1999

However, we feel it is important that users considering Frame+SGML must have an accurate assesment of the various issues involved. Before giving you answers to your specific questions let me make a couple of general points.

SGML authoring tools (including Frame, Arbor Text, and Xyvision) are like owning a compiler you do not have anything useful until you spend time (read as dollars) developing your application. Our firms experience has been that SGML development is not cheap. So if you expect to buy any tool and implement your SGML for nothing then give it up before you start. Although our firm does SGML development in all the tools listed in your EMail, we prefer Frame+SGML. Our experience has shown that SGML applications can be developed more cheaply in FrameMaker+SGML then the other tools listed in your EMail. Based on the sketchy information in your EMail I would guess that you are looking at a mid siz figure development budget no matter what tool you use. If anyone trys to tell you that it can be done for less I would be very carefull about following the plans they may present you with. Unfortunately tool vendors (including Adobe) are prone to minimizing the development costs associated with their tools.

There is a lack for competant developers out there for SGML applications (Frame or Otherwise), but there are a lot a amatuers out there. You are going to get EMail from some of them that will directly conflict with what you are going to here in this EMail. A first indication of this is that the amatuer tends to tell you a lot of things can't be done in Frame that can in fact be done. This lack of compentant development knowledge is so bad that Adobe in their manuals will tell you that there are things that cannot be done that can in fact be done in Frame.

See below for answers to your specific questions.

> 1. SGML round-tripping: Can one go from FM+SGML to SGML to FM+SGML
> again without corrupting any data?

This is not necessarily a fault of Frame. It has to do with how various "diff" programs work including those that are built into many translation memory tools. These tools fall into two categories:

The first generation tools are line oriented "diff" engines. That is for text to be considered a match it must fall on the same line in two versions of the files being compared. Some slightly improved versions of these tools recognize unchanged lines anywhere in the file so long as the line breaks fall in the same place. In otherwords they recognize inserted and deleted lines. Unfortunately this type of "diff" engine is built into older translation memory tools like Transit and TM2. Frame does not round trip well with this class of "diff" engine. Although our developers were able to get Frame to round trip with this class of tool through careful read/write rule development that insured the line breaks were in the right place algromethically.

Second generation "diff" engines are more SGML aware and they compare files based on element content no matter where the line breaks fall within the element content. Examples of tools with second generation "diff" engines are Astoria and Texcel. Frame out of the box interfaces well with these type of products.

> 2. Is it true that FM+SGML cannot create, import, export or preserve
> SUBDOCS? What about marked sections?

This is correct although I personally think that marked sections is a bad way to do conditional text. There is an alternative for conditional text when designing your DTD and that is to base conditional text on attribute values rather than marked sections. If you use this technique Frame supports it really well. This is the preferred conditional technique of most translation tools as well. Attribute based conditional text is supported by tools like Adept and Xyvision.

> 3. Is it true that FM+SGML cannot be used for a component based SGML repository?

This information is a complete lie. Frame interfaces well with component repositories like Astoria. There are a number of fortune 100 companies that have implemented Frame with repositories. All you have to do is change the read/write rules and the error Dan complains about goes away.

> 4. Case studies: is anyone in the world using FM+SGML with a controlled, component-based SGML repository to publish content in multiple output formats and multiple languages?

There are a number of companies that have done or are in the process of doing this. These include:

IBM

Compuware

United Airlines

Philips Semiconductor

> 5. Performance: on a 400 page manual with scores of graphics and somewhat complex components, is there a serious performance issue parsing to SGML from FM+SGML (does it take seconds, minutes, hours...)?

When you get involved in this issue make sure you are comparing apples to apples and not apples to oranges. The favorite trick of the Adept folks is that once you open a file in Adept it laces the file with cryptic comments that are used to speed up the parsing of the file on subsequent opens. The adept rep then uses such a preprocessed file in a comparison Frame's ability to open the same SGML. In my mind if you are using the preprocessed file then you should compare to the time that it takes to open a Frame binary file.

If you take the preprocessor ability away from Adept then the file open times are about the same for the two tools with Frame actually being faster when certain complex structures (like tables) are involved.

The problem is with either tool the run times for a 400 page manual are too long for interactive users. The thing that Adept/Xyvision reps point to is their batch capability. Unfortunately the typical Frame user/consultant is unaware that a very powerful batch capability comes with Frame (manuals are buried where you will never find them).

> 6. Implementation: is it cheaper, faster and easier to implement an FM+SGML solution with complex DTD/EDD (and possibly FDK) development than (for example) an Arbor Text, Xyvision, DTD/FOSI combination?

As noted above costs vary, but Frame is usually slightly cheaper. An important issue that you can't easily quantify is author productivity and happiness. The user interfaces in Adept and Xyvision have to be one of the worst of all time.

I can tell you the story of one fortune 100 company that implemented Adept for about 2000 writers. Dissatisfaction was so high (people were willing to quit rather than use Adept) that the company is now replacing Adept with Frame in the same place.

Response sent on November 24, 1999

Hi

Got a copy of this list of issues, some I can answer so I thought I would XyEnterprise has a content manager using SGML which manages objects of data and assembles these into documents and tracks revision and updates on an object and document/delivery basis. Our system supports a number of SGML editors including ArborText and FrameMaker + SGML. Hence the basis for my response. (responses to enumerated items in brackets-JA)

1. SGML round-tripping: Can one go from FM+SGML to SGML to FM+SGML again without corrupting any data?

[We have heard that sometimes FM will create different SGML markup than what went in, but so far we have found that this is very consistent and since we do a diff on the way back in, we will detect markup errors.]

It is imperative that we get 100% consistency during each edit/conversion cycle in order to work properly with translation memory software. I understand FM+SGML creates SGML compliant data, but if you ran a "diff" on the FM+SGML file before and after parsing to SGML and back, is it always the same? Is data such as variables or conditional text ever lost?

[Nothing is one hundred percent consistent, but the data should not disappear. But since SGML is not sensitive to line ending/carriage controls, but FM and AT may send back records that have broken differently so if you diff based on CR?LF then you could get differences that donot exist since the editors don't care, nor should they based on the Spec.]

2. Is it true that FM+SGML cannot create, import, export or preserve SUBDOCS? [YES] What about marked sections?

[Yes, it does not support MS - We have pretty much decided to refrain from using MS, too limiting and go to element/revision control process to delimit versions/effectivities, etc.]

3. Is it true that FM+SGML cannot be used for a component based SGML repository?

[No, component repository exists outside FM. It is true that to use FM effectively in my opinion, the FM user needs to see documents in logical sections (like chapters or page blocks, etc. so that pagination works correctly and his context of pages is preserved, but the content manager has the responsibility to put the objects together to form the proper viewing product for the user - at least that is the approach we have taken. We use the same concept for ArborText to allow a user to see a document assembled at any level of the hierarchy he chooses]

Dan Emory, a renown FM+SGML guru, criticizes FM+SGML for its failure to create FM+SGML text fragments as exportable SGML entities---"if such a fragment does not begin with an element that is valid at the highest level, it is not exportable." Is it impossible to create, import, export, and preserve SGML text entities (i.e. chunks) in FM+SGML?

[Yes if the content manager does it. Note, Arbortext is not (or was not a revision or two back) good at doing this either. In or content manager we determine the level in the hierarchy that the user has selected and wrap it with the proper tags to provide valid parsing. Even SGML editors that claim that they can take chunk and dynamically parse often have difficulties in not supporting inclusion or exclusion that occurred in the hierarchy at a higher level than the chunk being imported.]

4. Case studies: is anyone in the world using FM+SGML with a controlled, component-based SGML repository to publish content in multiple output formats and multiple languages?

[Yes we have a number of customers currently, two being Swiss Air Airlines, and RAAF MPLM Squadron in Australia. Information can be provided on these organizations.

5. Performance: on a 400 page manual with scores of graphics and somewhat complex components, is there a serious performance issue parsing to SGML from FM+SGML (does it take seconds, minutes, hours...)?

Conversely, are there major performance concerns when compiling SGML chunks into a pretty 400 page book using Xyvision or Adept Editor?

[Going from FM+SGML to SGML is reasonably quick. SGML to FM+SGML and a display is very slow since it has to convert to a EDD form and parse, and format (typeset the data) for display. Arbortext is faster. It also depends if you are importing graphics for editing. In AT most users do not edit graphics, just view them. If you edit the graphics in FM then you have a reasonable time to import and export them.]

6. Implementation: is it cheaper, faster and easier to implement an FM+SGML solution with complex DTD/EDD (and possibly FDK) development than (for example) an Arbor Text, Xyvision, DTD/FOSI combination?

[If you are using a Xyvision (XyEnterprise) FM+SGML then the requisite FDK work to support the application to the database is done. The EDD and style setup takes more time than an equivalent operation in ArborText (I believe, but I am not skilled in EDD/FM styles, we usually have subcontractors do this for us). AS for FDK/ArborText develop language work, this may be required in all cases depending on what special features or tools your users require or you want to add to the make editing or review easier. In all likely hood work would be required in either editing approach you chose, but this is application dependant entirely assuming the content manager already has the products integrated together.]

Hope this helps and let me know if I can be of further assistance.

Raymond H. Stachowiak
EMail: rhstacho@compuserve.com
Xyvision Enterprise Solutions, Inc.
Tel: +1-847-506-1630
US Mobile: +1 847-922-3724
Australia Mobile: 61-(0)41-731-7275

Summary: September 7, 2000

As you have probably guessed, we went with ArborText (well, are in the process of going...it takes many many months). The bottom line for us was that we wanted to manage a database of text modules in SGML all the time and FM+SGML was seen as a "filtered" SGML tool compared to the native editing capabilities of ArborText. Moreover, we need to work on improving our translation memory work flows and FM+SGML was somehow perceived as incapable of 100% consistency when round-tripping from .fm to .sgm formats. I dunno.

In truth, I think any SGML system requires so much customization that you are forced to go with FM+SGML if you have an FDK programmer in house and ArborText if you've got an ACL programmer in house. (ArborText does also offer a full suite of training in support of their product, whereas Frame is still the neglected bastard child.) ArborText may be more expensive, but FDK programmers cost a pretty penny, too. If you're in CA, you should have no trouble finding one. Here in the Midwest, it's a bit different.

Adobe has done a lot in recent years to reverse the public misconceptions about FrameMaker, but the company still has a lot more to do to realize the full marketing potential of the product IMHO.