Event Summary

Information Authoring: Applications, Features and Futures…

Bruce G. Mills
www.lonepinestudio.com
SLO-STC February 27, 2006

Description: Wondering what the heck is going on with the information design software you've been using forever, or trying to decide which programs to learn to buff out your résumé? Find out where you should invest your efforts to ensure your professional future at **"Information Authoring Applications, Features, and Futures."** We will gaze into the crystal ball at some of our longstanding design, layout, Help, and text processing favorites to find out where they appear to be heading.

In the last few years, we've seen many products evolve and devolve, as major software companies merged and new industry standards emerged. For example, open source standards and 'Webification' are rapidly changing the character of software and content development. What does all this mean, and how can you benefit from — or avoid being hurt by — the many changes? Should you invest in application skills or technology-based knowledge? Get the scoop on "creeping incompatibility," obsolescence, Webification and other future trends that will make some applications winners and some losers at this timely event!

Speaker: Bruce Mills, B.F.A., M.B.A. is the principal of Lone Pine Studio in Paso Robles providing graphic design, Web design and development , illustration, documentation, technical publishing, marketing, and business consulting to clients in SLO County and beyond. He has evaluated, used, developed, manufactured and marketed computer graphic systems and software since 1979 holding a series of creative and executive positions with training, communications, marketing and manufacturing organizations. Bruce had been an independent contractor and consultant since 1998.

For complete speaker notes and online presentation go to
http://www.lonepinestudio.com/pages_media/media_STC-022706.php

Where did my favorite app go?

As technical communication professionals and information authors, we are asking ourselves that question more and more often these days. Through acquisitions, mergers, failures, technology migration and creeping incompatibility, the applications that we have become a skilled expert using are no more. We are left facing the challenge of figuring out where to invest our time, effort and dollars to acquire and become proficient with a new tool. Bruce shared his experiences and offered his time tested guidelines for evaluating alternatives as well as some provocative insights into future directions in software development.

What is Job One?

Bruce points out that before we start looking at new applications we really need to stop and define our primary job activity.

Doing the job that generates invoices or ensures that you won't be fired or laid off is always "Job One." "Job Two" is dealing with all of the other things that make the performance of Job One better.

If a tool doesn't effectively facilitate or enhance the performance of Job One or if Job Two becomes all consuming, you probably shouldn't waste your time on it. No one wants to be an expert at Job Two if Job One goes away.

We must ask ourselves:

- What do I need to accomplish?

- What application areas are most critical?

- What are my biggest concerns?

- What criteria should be considered in evaluating new applications?

Criteria for Evaluation

Next, how should we approach the challenge of evaluating new products so that we get the best return on out investment?

Bruce mapped out five dimensions for evaluation:

1. Skill vs. Knowledge – Experience Building
2. Learning vs. Doing – Usability
3. Productivity vs. Creativity – Staying Focused
4. Big Company vs. Little Co. vs. No Company – Consider the Source
5. Old vs. New – Transferability

Skill vs. Knowledge – Experience Building

Does learning a new product expand our overall knowledge base or is it just an expedient tool that we will have to abandon sooner than later? .

Some applications that we might consider are highly specialized, targeting unique technical tasks. Some are more general purpose, addressing common tasks. On the surface, all are intended to increase productivity, in some cases by streamlining repetitive tasks, in others by offering previously unavailable capabilities. If the tasks being targeted are product or vendor specific, they are vulnerable to obsolescence. If the skills required to use the product mask underlying technology or process, you become more vulnerable to the whim's of the developer and market forces. In other words, you risk becoming an expert with a non-existent product.

Conversely, products that enable you to become more knowledgeable about underlying technology and processes, enable you to protect your investment in learning new skills. For example, becoming an expert with a particular graphical HTML editing tool that masks the underlying properties and behaviors of HTML contributes to specific skill development and short term productivity but not to longterm knowledge of HTML, Internet standards or architectures. Acquiring knowledge as well as skill ensures transferability of skills in the future.

Learning vs. Doing – Usability

What is the startup ROI on learning a new product?

Given the cost of acquiring and maintaining software, we must carefully evaluate the trade-offs between

getting up and running quickly at a lower productivity level vs. investing in learning/training first then implementing a more productive overall process later. Circumstances may dictate which approach is taken, however, documentation, tutorials, support and user feedback should be thoroughly evaluated with this in mind. For instance, OpenOffice and Mozilla suites offer significant advantages over Microsoft suites as well as excellent documentation, and extensive training opportunities but may require a more significant up front investment in learning how to use them effectively, or unlearning how to use a familiar product. Yet in the longrun they may be more productive and contribute more to our knowedge base than the old product.

Productivity vs. Creativity – Staying Focused

How important is high productivity relative to extensive and perhaps esoteric creative options?

Competitive market forces drive proprietary application developers or continually add WOW! factor to flagship applications. While some such enhancements may be useful others may be used rarely. Esoteric effects and functions could be attractive if our primary focus is on developing creative content. If our primary focus is on performing certain routine tasks as efficiently as possible, its just extra overhead and a potential loss of productivity. All information authors (content creators) do some of both. Most will say as in other fields, its 90% labor and 10% creativity. At what point do impressive features that are difficult to use or rarely called on become obstacles to Job One?

If you are working in an environment where the goal is uniqueness and productivity is of less priority, such specialized features may be an asset. Flash is perhaps an example of a product long on creative capability and short on productivity at least in the short term. Illustrator and Photoshop are good examples of software that balance creativity with productivity.

Big Company vs. Little Co. vs. No Company – Consider the Source

Who develops and maintains the software?

There are three general categories from which we can acquire software, assuming we don't want to create them ourselves:

1) Large vertically dominant software and technology developers
2) Smaller independent software developers
3) Open Source software organizations

Large vertically dominant vendors offer vertically integrated product lines that cover a broad range of related application areas. They offer sophisticated product suites that claim to offer high-level compatibility and extensive support. The software is expensive and success is not always assured. This option, however, may increase our dependence on a singe source and their ability to upgrade and support the product in the future.

Smaller independent developers typically offer leading edge or niche products that are below the radar of the vertical dominators. The risk with smaller independent developers, on the other hand, is whether or not they can sustain a product line, stay in business, and avoid being bought or forced out by the bigger fish. Yet some of the best values can be found in from independent developers but careful scrutiny of demo software is essential.

An alternative route is Open Source products and technologies (if they offer appropriate features and

capabilities for the task at hand), such as OpenOffice, the Mozilla suite and W3C language specifications. They are developed and maintained by students, professors, and engineers who are themselves users. In some cases, they are the product of government funded research payed for with taxpayer dollars. In other words, we already own them. They should be free. New users can use and contribute to the continued development under GNU-GPL (GNU General Public License) public licensing agreements. See http://www.gnu.org/ and http://www.fsf.org.

Old vs. New – Transferability

How is this going to change my work environment and what is it going to cost to get back to where I was before?

Replacing or upgrading means making a transition from old to new. Whenever face the need to upgrade or replace software, we are faced with questions such as:

- What are the risks, unexpected consequences?

- Will it be compatible with other tools that I still depend on?

- Will I have to upgrade other software and equipment?

- Will I have to buy addons and extentions to get back where I was before?

- What will it cost for future upgrades and maintenance?

And, when evaluating the product:

- Does the product offer a clear future upgrade path?

- Does it offer extendability?

- Is the released product reliable and complete?

- Are there known deficiencies?

- Is the documentation accurate and comprehensive?

- Are current skills and knowledge transferable to the new product?

- How long will it take to get back to where I was before?

- Am I giving up anything critical?

- Will new skills and knowledge be transferable to the next product?

- Does it support my current peripherals and drivers?

While current trends toward the convergence of computing environments may reduce some of the issues related to software compatibility, availability and performance, operating systems continue to evolve, I/O standards continue to change, and the all pervasive influence of the Internet continue to introduce uncertainty.

Industry & Technology Trends

Having armed us with some useful criteria for evaluation, Bruce offered a spapshot of some fundemental changes that are taking place in the evolution of content creation software.

Webification

The single most significant impact on the direction of information authoring software development is the success of the Internet. The all pervasive Internet has become the primary vehicle for business and personal communications as well as entertainment worldwide and is the primary target of the majority of authoring and content development applications.

Its existence is the result of several public domain and publicly licensed technologies and applications such as: W3C Standards, Open Source/GNU, XML, XHTML+Mobile, XSL/XSLT, SVG, PDF, SWF, Unicode (UTF-8, UTF-16), FTP/HTTP Protocols, client and server-side scripting (JavaScript, Perl, PHP), Java, and more. These technologies taken together comprise the Internet and constitute a distributed global application environment.

Because of its growing complexity and roll of specialized languages and code writing, there is a growing need for collaboration and integrated development capabilities.

That's “Webification.”

New Publishing Paradigm

What does Webification mean to information authors and content creators?

The scope and spontaneity of the Internet and the portability of content have now given rise to significant changes in the definition of documents and publishing. Simultaneous cross-media publishing based on a single-source content model, the inclusion of dynamic and data-driven content, and international distribution have turned the concept of a traditional page-based document upside down. Content can now exist independently of any specific page-bound product or document format.

Based largely on Open Source technology, Web-centric documents are in essence dynamic software conatiners for XML formatted content. And as a result, applications are increasingly becoming Web centric! XML is at the heart of not only the single-source publishing model but for virtually all state-of-the-art data portability models. Bruce strongly emphasised attention to XML-based features and support.

Unixification – Platform Convergence

The convergence of manufacturers on Unix-like operating systems and the platform independent nature of the Internet application environment mean that software design can become more universal across platform boundaries. It also means that user skills and knowledge may become more portable.

Open Computing

Bruce brought to out attention the often overlooked roll of Open Source.

The Open Source movement has provided many of the key technologies that are the foundation of Internet computing. Not only are these technologies readily available to end users as standards, languages and products, they are supported or extended in some fashion by all major application developers. XML is now supported by mainstream browser makers, Javascript (Mozilla/EMCAscript) is supported by Windows and most major publishing products including Adobe's CS2 suite and Flash. XML with CSS and XML transforms can even be run directly in emerging server environments. Perl, Ruby and MySQL can be run as standalone applications on NT and Unix-flavor platforms. Any application that is not fully compliant with these standards or that does not support their use will be less useful in the Internet computing environment.

With platform convergence and the productization of Open Source technologies such as Apache2, a

personal computer is no longer just a passive receiving node on the Internet. Every personal computer can be equipped with a server, publishing technology and databases to become a powerful Internet computing platform and personal interactive information domain.

Market Forces – Vertical Dominance

How do market forces shape the technology landscape?

The Internet offers vast opportunities for the benefit of all users but also offers vast opportunities for profit to proprietary software vendors. While the Internet may be built around public domain technologies, sophisticated proprietary software can bestow significant productivity benefits and provide vertical integration and support for enterprise-scale customers. As the application environment becomes more Webified, bigger companies seek to grow revenues by controlling ever larger vertical slices of the Internet, enterprise and consumer application markets.

Bigger companies gobble up market share, try to diminish competition often along with innovation, product quality, and competitive pricing. Yet much of the core technology upon which the Internet is built is Open Source and in the public domain. In fact, this is where much of the innovation takes place that drives Webification.

Smaller independent developers who are more versatile are more likely to develop leading edge applications in niches that the bigger developers have not yet identified or will not approach because they may jeopardize current product lines. The smaller guys may break ground both in application design and in markets served. If they are successful, they will either be bought out or out-marketed by the big guys.

This creates a continuous cycle of innovation, adaptation and between public domain organizations, independent developers and vertical dominators. At the balance point of this unstable triangular is Open Source which continues to be a beacon for future development trends.

Stratification, Collaboration & Integration

How are Webified applications evolving to deal with the increasing complexity of information/content content development tasks?

With increasing complexity comes increasing specialization. As the development environment for Web-centric content has become more like software development, technologists have sought to achieve rational separation of different tasks based on expertise. The result is a move toward stratification in the technology and in authoring and development software.

While some of us may try to wear only one hat, we often perform a variety of different rolls whether we know it or not. The door should be open to operating on as many levels as necessary without extensive upgrading or add-ons.

Increasing complexity may also mean working in teams, possibly in remote locations. Webified application environments must offer a means for collaboration and process control as well as the ability to seamlessly integrate the large scale projects. Hence the recent proliferation of software “suites” with collaboration and project management capabilities.

Features and Futures – Meta Software

Does it walk like a duck?

The future of information authoring and content creation is being driven by the success of the Internet, technology standardization (W3C compliance) and the Open Source movement. With the prospect of every computer and every mobile device being connected through a vast distributed application environment, content creation is no longer an isolated task performed by subject matter experts. It is a complex collaborative development environment with vague boundaries between authoring and development.

Information authoring applications are becoming "meta" software. Software for creating software as well as content. Going forward, applications of choice will support this trend.

If its not "Webified" its probably not worth investing your time, effort and dollars. You'll be right back at this same decision point again in no time.

Roadmap to the Future

Here are a dozen or so tips Bruce compiled for deciding where to invest your effort so that you get the greatest return on your investment and will ensure that you build transferable knowledge as well as proficiency:

1. Define Job #1. Content creator, producer, developer, system administrator…?
2. Learn as much as possible about emerging technologies and trends driving your primary application environment.
3. Analyze and document your work environment, your requirements, and your expectations.
4. Evaluate new tools in terms of the effort invested in skills vs. contribution to core transferable knowledge.
5. Consider transferability of skills and compatibility with your current environment. Will you have to upgrade other tools and equipment?
6. Evaluate products in terms of the Usability ROI — learning vs. doing. Checkout the documentation first!
7. Consider production vs. creative trade-offs.
8. Evaluate your needs in terms of vertical integration and support vs. independent or Open Source alternatives.
9. Look for 'Webified' architectures and features that support Open Source and emerging standards (XML).
10. Evaluate features and upgrade paths for an appropriate task stratification architecture. All-in-one or expensive add-ons?
11. Get first hand feedback from peers or peer journals — not from data sheets or sales literature.
12. Avoid surprises. Have a transition plan for implementation — parallel systems.
13. Don't use new products or technologies on a mission critical deadline.
14. Try to avoid 1.0 and X.0 releases.

Winners & Losers

With some additional suggestions of possible winners and losers in the move to Webification Bruce launched lively discussion after the break. Various product categories were evaluated according to the criteria outlined in the first half of the meeting.

Evaluation Matrix

Bruce offered a sample matrix (see STC-022706-matrix.pdf) as a possible tool for evaluating applications

relative to the criteria presented earlier, their degree of Webification and their suitablility for Job One. Albeit somewhat subjective, the matrix provides a framework for rating the suitablility of a particular product or product line for our needs. To use this or similar matrix answer the following questions:

1. What is Job One?
2. Does learning a new product imply add another notch to our tool belt or does it expand our overall knowledge base?
3. What is the startup ROI on learning the product?
4. How important is high productivity relative to extensive and perhaps esoteric creative options?
5. Who develops and maintains the software?
6. How is this going to change my work environment and what is it going to cost to get back to where I was before?
7. Is the product going to be viable in a Webcentric application environment?
    - Does it support HTML/XHTML/XML?
    - Does it support Adobe Acrobat?
    - Is it cross-platform compatible?
    - Is it feature compatibile with other applications that I will be using?
    - Does it support collaborative activities?
    - Does it offer appropriate levels of functionality for the rolls that I will likely play?
    - Is there an Open Source alternative?
    - Does it have a future: upgrade path, support, market viability?

Resource References:

http://www.adobe.com
All things Adobe/Macromedia

http://www.adobe.com/svg/
Adobe SVG Demos

http://www.microsoft.com
MS Office

http://www.openoffice.org
OpenOffice.org

http://www.mozilla.org
FireFox, Thunderbird, Sunbird, Mozilla Suite

http://www.gnu.org/
General Public Software Licensing

http://www.fsf.org
Free Software Foundation

http://www.java.sun.com/
Java Technology

http://www.rubyonrails.org
Ruby and Rails Development Tools

http://www.runrev.com/
Runtime Revolution/MetaCard Cross Platform Media Development

http://www.wbtexpress.com/
4System.com (Poland) HTML/XML eLearning and courseware development tools

http://sources.sourcetool.com
Ref for information on eLearning and online Help products

http://www.toolbook.com
Online help and eLearning development tool

http://www.madcapsoftware.com/
The new RoboHelp

http://www.pinnaclesys.com
Consumer to professional video editing

http://usa.autodesk.com/
Discreet Fire – Professional video editing

http://www.corel.com/
WordPerfect, CorelDraw, Painter, PaintShop Pro