

LASSELLERAMSAY
Business Content Development

Secrets to Cheaper, Faster, and Better DITA Implementations

Tom Voltz & Tom Idleman
Fall 2009

Summary: Specialization is DITA's "Secret Weapon"

When moving to DITA, most organizations start with a generic DITA implementation, using the default content elements and structures.

While a generic approach minimizes the upfront implementation effort, it significantly increases the downstream costs including training information developers, the complexity of authoring new information, quality and consistency issues, and lost opportunities to repurpose and reuse information.

If instead of using the default DITA structures, you use DITA specialization in the core implementation strategy, you are likely to experience faster, less error-prone authoring, less training requirements, and more opportunities to re-use and repurpose information.

The bottom line is that, in many cases, investing in specializing DITA for your organization leads to a cheaper, faster and better DITA implementation.



Secrets to Cheaper, Faster, and Better DITA Implementations

Typical DITA Implementation: Generic DITA

Organizations face many challenges along the way to a successful transition to DITA.

Content developers need to learn a new way to write, taking information chunking and minimalist authoring to a whole new level. Authors need to learn new applications. Writers and content architects need to review and reorganize a proliferation of files. And last, but certainly not least, content contributors need to learn new and often more restrictive approaches to organizing and working with information.

Documentation and publication infrastructure departments typically face this significant set of challenges by looking for opportunities to simplify the challenge. Hard pressed by the demands of the transition to DITA, a common reaction is to use DITA “as is” – using generic DITA without creating a restricted DTD more tailored to their needs – in an attempt to minimize extra work, unwanted restrictions on the authoring process, and possible technical complications.

Unfortunately, this attempt to simplify the task of adopting DITA usually complicates the task, instead.

The Problem with Generic DITA

Using generic DITA as a baseline for transition comes with its own issues. What appears to be a rather rational and measured decision on the surface ultimately leads to numerous extra problems when transitioning to DITA.

It's true that generic DITA provides everything authors need, but it also provides authors everything they don't need.

The base information types in DITA, such as Topic, Reference, Concept, and map, are purposely designed to minimize constraints so you can use multiple approaches to organize information.

But leaving them loosely defined for an actual implementation makes the content author's job more difficult than it needs to be. For example, in a <title> element for a generic topic, there are 35 different elements to choose from when composing the title. 25 of those 35 elements are a group of elements collectively called the programming domain, which includes items such as <codeblock>, and <apiname>. By removing these irrelevant items, the job of selecting the right elements is much simpler, especially if you are writing in a domain that does not have anything to do with software.

Another way of looking at this issue is to compare using generic DITA to produce documentation to using Adobe FrameMaker or Microsoft Word without templates. Without



Secrets to Cheaper, Faster, and Better DITA Implementations

templates, consider the amount of extra training authors would have to attend and retain in order to create a company's documentation suite. And even after extensive training, consider how much time authors would waste trying to remember what formatting to apply to each chunk of information day to day as they write. What do you think the chances would be that two different authors would format the same type of information the same way? Slim to none.

But using generic DITA is even more problematic than using FrameMaker or Word without templates. Because DITA requires authors to format information logically rather than typographically, there are far more pitfalls for authors to fall into. For example, in a documentation environment, command names, menu options, and what users type in an application might all appear in bold text according to a company's in-house style guide. In FrameMaker or Word, that's one rule authors need to remember and one rule authors might break. In DITA, it's three: apply the command name tag to command names, the menu option tag to menu options, and the user input tag to what users type in an application.

Furthermore, if a formatting element is misapplied in FrameMaker or Word, at worst you have a transgression against the in-house style. True, such transgressions do affect the quality of documentation, ultimately affecting customer satisfaction and the customers' ability to learn from documentation.

A misapplied formatting element in DITA costs so much more. Misapplied formatting elements can lead to transgressions against the in-house style, but because of the strict rules of DITA coding, they can also lead to broken documentation builds that must be tracked down and corrected. They also seriously degrade the ability to re-use information coded in DITA, which is one of the key reasons to switch to authoring in DITA in the first place.

Specialization: The True Power of DITA

What really makes DITA a powerful authoring framework?

Certainly, the fact that DITA enforces a topic-based architecture helps promote the modularity of documentation and the goal of re-use. The XML foundation also makes DITA an architecture that can be supported by multiple tools and applications.

But the "D" in DITA stands for "Darwin" for a reason. The concepts of inheritance and specialization are central to Charles Darwin's theory of evolution. And those very concepts are built into DITA so that it can be specialized effectively.

In fact, the Task, Reference, and Concept topics that are considered basic components of DITA are specialized versions of the generic DITA topic. Bookmaps are also a specialized version of the generic DITA map. So whether you want to or not, you are already using DITA



specializations. Doesn't it make more sense to use your own custom specializations rather than the 'one size fits all' versions?

Specialization and Semantic Richness

Using specialization to customize your content elements and content structures can create semantically rich content, which can significantly increase the potential value of your content.

"Semantic richness" is an important concept, but often hard to understand in the abstract. To make it more concrete, consider the difference between off-the-shelf cash registers and cash registers that have been customized for fast food restaurants.

Like generic DITA, the off-the-shelf cash register can accommodate ringing up anything that is for sale, from bowling balls to sailboats. But if you're only selling fast food items, that flexibility can slow down both effectiveness and efficiency.

Consider ringing up an order for a hamburger (\$3.59) and a milkshake (\$1.99) using a generic cash register:

press then , then , then , then
 then , then , then then

And the same transaction on a 'specialized' cash register:

Press the  button
Press the  button

Using a generic cash register requires knowledge about the food prices and about 10 keystrokes, whereas the customized cash register requires pressing only two or three buttons. But there's a much deeper and more important difference – the second transaction is much more descriptive about what is actually happening – in other words, it's a **semantically rich** transaction.

Because the system knows what was ordered, not just the price of what was ordered, it is possible to use this information to:



- Notify the kitchen exactly what needs to be cooked as the order is being entered in the register.
- Track inventory automatically by simply capturing the register button pushes.
- Log hourly trends of what customers are buying as the semantic content of the orders is passed on.



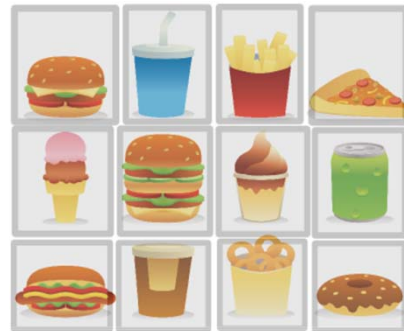
Of course the ‘semantically impoverished’ generic cash-register would not enable any of these uses of the information. A system that allows the cashier to provide semantically rich information creates opportunities to use that information in new ways.

This example also helps to demonstrate a few other benefits of creating a system that provides a more semantically rich information environment:

- Training is now quicker and easier, and more intuitive.
- There are fewer entry errors.
- It’s easier to change elements, such as prices, without requiring retraining.

Using specialized DITA is very similar to using specialized cash registers.

By creating DITA elements that are semantically specific instead of generic, for example `<marketing description>` for a paragraph containing a marketing description of a product as opposed to a plain `<p>` paragraph tag, you can build templates that actually guide authors’ input rather than relying on the authors remembering to provide the marketing description information in the appropriate place.



Furthermore, your opportunities for re-use increase and you can target those opportunities more specifically. With a `<marketing description>` tag, for example, you could automatically pull that information from a user’s guide into a white paper or vice-versa. You could also change the way that marketing description information is rendered specifically rather than having to change the way all paragraphs are rendered. You could even pull all of those marketing descriptions into a product-line description page on your company’s Web site.

So, if specialization provides all of these significant benefits, from lowering training requirements, preventing tagging mistakes, and increasing the focus and opportunities for re-

use, why don't more corporations specialize DITA? Unfortunately, there are a lot of myths about specializing DITA out there in the industry. Let's address the most prominent of those myths, now.

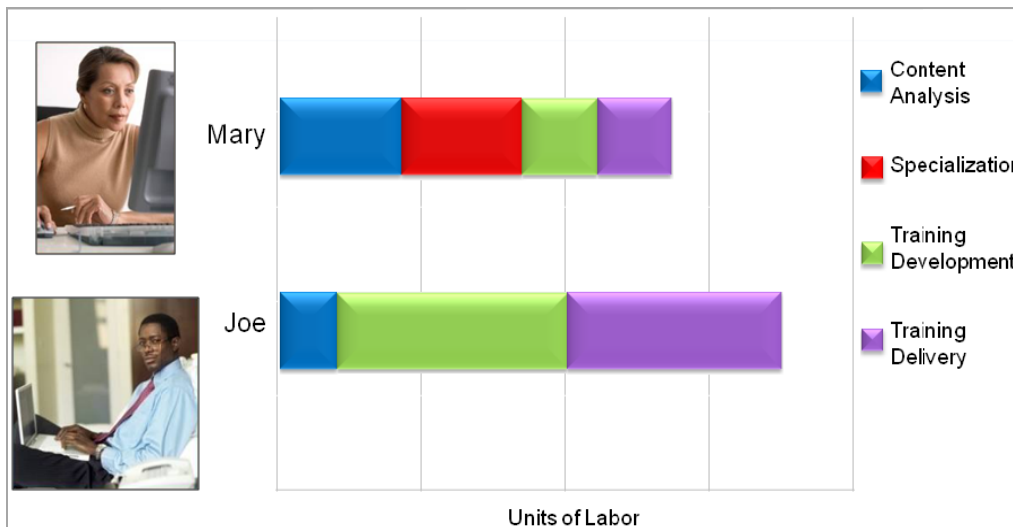
DITA Specialization Myths

“Specializing DITA is a complicated process that will slow down our adoption and use of DITA.”

Specializing DITA is certainly not a trivial process, but overall, it actually saves time in many ways if it's done correctly:

- It reduces the amount of time required to train current content providers as well as future content providers that you hire.
- It reduces the amount of code review your team needs to conduct to ensure that similar information is tagged the same way and that tagging is done to standard.
- It increases the amount of time your content providers spend writing content as opposed to trying to recall what DITA tag to use in each particular context.

In the end, you do have to put in more effort and time to analyze the information architecture, audience, and information development process. But you more than make up for that initial effort in so many different ways.



Secrets to Cheaper, Faster, and Better DITA Implementations

“You should only specialize DITA by *extending existing types*.”

Many think that specialization is only relevant in DITA to cover edge cases in information structures that generic DITA can't handle.

It *is* true that specialization allows you to accommodate information structures that can't normally be handled by generic DITA. But the real reason to specialize is to restrict generic DITA to cover only what you need. If you conduct a proper information analysis and build a process that is flexible enough to begin with, even if you end up initially restricting DITA too much, it is relatively easy to expand it again.

The last challenge a documentation department needs to face is a superset of tags larger than generic DITA. It's no wonder organizations that believe this myth don't want to get anywhere near specialization.

“We'll be stranded if we specialize DITA.”

This is often a matter of an organization misunderstanding the limitations of the documentation tools they are using rather than the true limitations of DITA.

Well-constructed specializations will never leave you stranded. As long as the authoring tool you use supports specialized DITA, you can mix and match any and all combinations of information authored in specialized DITA and generic DITA. You can even quickly roll back specialized DITA to generic DITA, if necessary. Indeed, DITA was built with this seamless ability of specialized and generic DITA to work together. As long as your specialized tags inherit the properties of generic tags the correct way, you can change your mind or work with other information sources that don't use specialized DITA any time you want.

Now, if you start with the wrong *tool*, a tool that doesn't support specialization properly, that can lead to significant problems. But DITA itself is constructed precisely so that you are never stranded if you specialize correctly.

“We'll have to customize our stylesheets if we specialize DITA.”

This is absolutely not the case if you set up your specialized elements with the proper inheritance from generic elements.

For example, to return to our <marketing description> example, if you set up <marketing description> to inherit the properties of the generic <p> element, all information tagged with <marketing description> will simply be rendered the same way as information tagged with the <p> element. No changes to your stylesheets are necessary.

Secrets to Cheaper, Faster, and Better DITA Implementations

Now, if you want information tagged with <marketing description> rendered differently from information tagged with a <p> tag, you would have to customize your stylesheet or rendering process to accommodate that difference whether you specialize DITA or not.

Furthermore, by restricting DITA through specialization as we recommend, you're actually *reducing* the amount of styles you need to track and specify in your stylesheets if you do decide to customize your stylesheets to change how your output is rendered.

“If we restrict DITA too much through specialization, we’ll be stuck using a subset of DITA that doesn’t fit our needs.”

Again, this is a myth based on method rather than the true limitations of DITA.

Certainly, if the content analysis you conduct up front is seriously off the mark, if you don't have a clear understanding of how your publication process will work while using DITA, and you don't fully comprehend the needs of your users through a proper task analysis, then the specialized DITA you create won't fit your needs from the very beginning.

Moreover, if the process you use to modify DITA isn't based on a sophisticated understanding of the DITA specializations that already exist and the properties of generic DITA elements, you'll have problems. And if that same process poorly tracks and adjusts to the needs of your authors should those needs not fit your specialized DITA model, you'll waste time and resources.

Final Thoughts

Don't begin your adoption of DITA by establishing a legacy that requires new writers to learn a superset of DITA elements that don't semantically describe the information that they're actually writing and that they don't even fully use. Don't take on a documentation cycle that includes excessive DITA code reviews to ensure that authors have coded their information correctly and haven't broken documentation builds. Don't start with a DITA methodology that seriously limits your ability to effectively re-use information in a focused way and your ability to adapt to future documentation requirements.

The best way to reap the benefits of DITA are to invest upfront in consulting services from a qualified provider to guide you through the process, and train you to maintain and improve your information architecture, including specializations over time.

Works Cited

If you found this useful, you may also enjoy our two part webinar, "Secretes of Cheaper, Better, Faster DITA Implementations" at <http://LR.com/Our-Content/>

About Lasselle-Ramsay

Lasselle-Ramsay is a leading content development services organization that develops and delivers critical business information and learning solutions for new products, business initiatives, and regulatory requirements. We integrate content, content development processes, and technology to provide solutions that are up-to-date, on-demand, and targeted to increase productivity and drive know-how. We focus on technical documentation, training development, and web-enabled content solutions

Since 1982, Lasselle-Ramsay has provided information and learning solutions to leading companies in a variety of industries, such as Fireman's Fund (insurance), Genentech (biotechnology), Tyco (finance), Cisco Systems and Hewlett-Packard (technology). Contact us at www.LR.com for effective solutions to your next content project.

Legal

COPYRIGHT © 2009 by Lasselle-Ramsay.

Lasselle-Ramsay and LR are trademarks of Lasselle-Ramsay Inc. All other trademarks and logos are the property of their respective owners. All rights reserved.

Reuse and reprinting of this material by permission only. To obtain permission contact us at:

Lasselle-Ramsay, Inc

101 Redwood Shores Parkway,

Redwood City, CA 94043

Tel: 650.968.1220

FAX: 650.968.0949

www.LR.com

