White Paper

# How do you decide between Git and a Component Content Management System?


## Rik Page – Bluestream

## Frank Miller – Ryffine

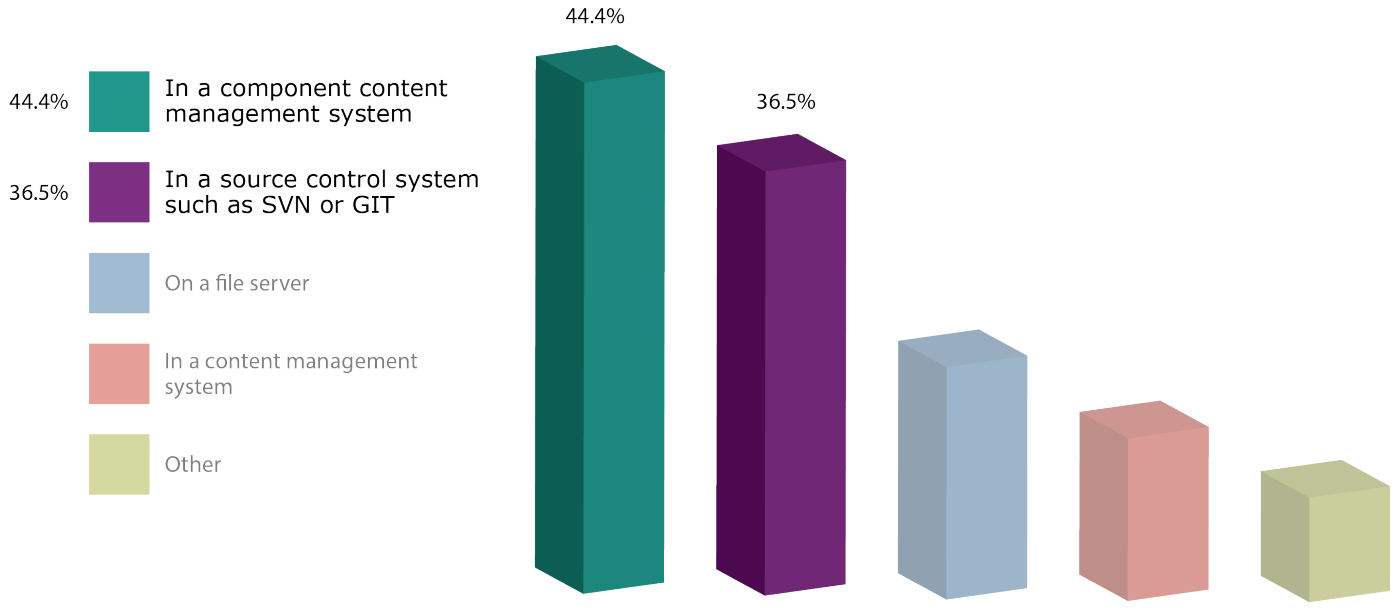

## Introduction

This white paper explores some of the things you should consider if you are using DITA, or thinking of moving to a DITA workflow for tech pubs.

## Background

A 2020 survey by CIDM into trends within the DITA community highlighted the dominant position of both Git and CCMS for content storage.

*Content storage in documentation organizations*




*Data courtesy of Comtech / DCL 2020 Trends Survey*

## What is needed by a modern Tech Pubs Team working in DITA

Whatever your industry, if you are working in DITA there are common requirements:

- A way to create content – Your authoring platform, Adobe FrameMaker, oXygen Author, XMetaL, SimplyXML etc.
- The need to Store, version, and share many component-based, structured (DITA) files. This can be addressed by an open source system platform such as Git or a commercial CCMS (Component Content Management System) such as Bluestream XDocs DITA CCMS.
- The ability to branch, merge and release content.
- The ability to manage the localization process.
- Send files for translation
- Receive translated content
- The need to publish your content. This will start with the DITA Open Toolkit but may also include other components such as Antenna House, Miramo, oXygen PDF Chemistry etc.

Looked at simply, or if your needs are simple, then a CCMS may not be necessary and Git might be the answer. If your needs are complex then a CCMS might prove to be the better approach.

## How Git can help

A lot of companies have chosen Git instead of a commercial CCMS. This is because Git has some very good properties:

- Git is good at storing, versioning, and sharing files.
- It allows authors to benefit from collaborating with teams already accustomed to Git.
- Git has a collaborative model.

Git also has a lower cost to entry; however, “open-source” does not mean “free”.

There is still the need to invest in: hard resources, infrastructure, and knowledge.

## The perfect Git customer

If we look at the capabilities of Git we can envisage the perfect customer:

- **Git is a culture fit** – why buy if I can build?, this is particularly true with software companies.
- You are already using Git in-house.
- oXygen is your XML IDE (integrated development environment) and you know how to squeeze every last bit from its capabilities.

- Aligning with development teams is a priority.
- You want to jump aboard automation that already exists.
- You understand the steep change management curve and embrace it.

A solution based on Git still requires additional components as shown in this diagram:


## Where Git falls short

Git was not designed to support DITA, it is just a happy coincidence that it can, however there are some issues to consider:

- Git takes time to set-up properly. You cannot simply download it and start work - or if you do it is likely you will be re-working something quite quickly.
- You have to support everything yourself. If anything goes wrong or breaks it will be down to you to fix.
- You will be reliant on internal expertise for assistance. There is always a risk that the internal expert who set-up the Git platform for tech pubs might leave the company, taking their knowledge with them.
- Solutions based on Git are not properly 'joined-up'. You may end up with components from multiple resources – there are no guarantees that these components will continue to work together as they are developed along different "Road-maps".

## Or does Git fall short?

### Myth #1 – Git is not designed for DITA

- No repositories are designed specifically for DITA. Database types run on multiple platforms including: XML, SQL (relational), no SQL, etc.
- Git isn't designed to store components, but with the addition of oXygen XML Author you can manage components.
- The bigger issue around DITA is managing reuse whatever the repository.

### Myth #2 - Takes time to set up properly

- Git can be deployed in the time it takes to fire up a Git repo.
- There are intricacies in setting up the entire infrastructure as shown in the diagram above, but many of those exist in implementing a commercial CCMS.
- DITA information design, repo design, branch strategy, workflows, Git tools all need to be thoughtfully implemented.

### Git Myth #3 – Reliant on internal expertise – which might up and leave

- An argument can be made that Git skill sets are more readily available than those with a combination of <fill in CCMS product here> and DITA. The challenge is capturing their interest in content rather than code.

### Git Myth #4 – Solutions based on Git are not properly 'joined-up'

- All-in-one software solutions tend to have numerous features that are neither wanted nor used. And they get in the way.
- With Git, you Git only what you need.

### Git Myth #5 – Different components can diverge

- You own it, you update it, for better or worse. There is a risk here, but if you understand the risk you can mitigate against it.

### Git Myth #6 – Unable to support complex localization needs

- This is true but many organizations already offload the highly complex localization work to their LSP (Localization Service Provider).
- Some organizations are having success with the oXygen add-on to build translation package.
- Smart repo design and branch strategy can give you greater control of your translated content

## When a CCMS makes sense

A CCMS has been designed from the ground up to support DITA – it isn't a compromise. There are several reasons for choosing a CCMS over an open source platform such as Git including:

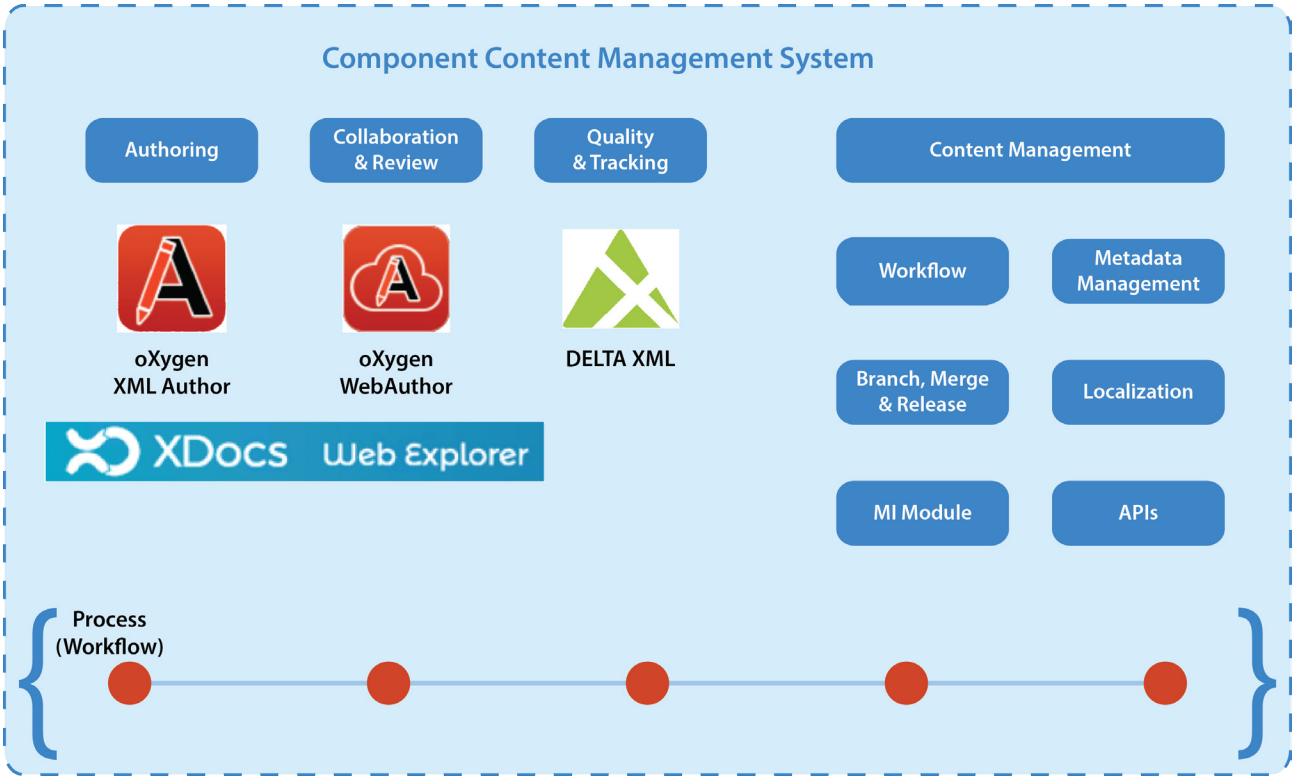
- You don't have in-house expertise on alternative platforms such as Git. If you are starting from zero a CCMS will be far quicker to get up and running and to maintain.
- Lots of projects/content. A CCMS will be easier to use if you are running multiple simultaneous projects. Branch and merge will be much easier with links between Topics fully supported.
- Multiple delivery requirements. A CCMS will be better if you have complex delivery requirements. Pushing content in multiple languages to online portals directly from the database immediately it is approved will be far simpler.
- Localization needs. Managing multiple languages with all the complex links between Topics and various conditions is supported within a CCMS.
- Complex relationships between Topics. Link management within DITA is essential, a CCMS such as XDocs validates all links on every import/export and provides tools to fix them.
- Integration with 3rd party tools. A CCMS supplied with comprehensive APIs opens the possibilities to connect to 3rd party systems and enhance the content lifecycle.

## The perfect CCMS customer

If we look at the capabilities of a CCMS we can also envisage the perfect fit:

- **You don't have in-house expertise on alternative platforms such as Git** – without the inhouse expertise you will be adding massively to your workload if you try to implement Git.
- **Lots of projects/content** – full control over reuse, branching, merging, releasing/publishing projects are a staple of a good CCMS.
- **Lots of delivery requirements** – a CCMS will be tighly integrated with the DITA OT (Open Toolkit) making multiple publications/single source delivery much easier.
- **Localization needs** – comprehensive management of large publications in multiple languages is a function of a good CCMS. Attempting to do this without automation is very time consuming and risks introducing costly errors.
- **Complex relationships between topics** – link management is a vital component of a DITA ecosystem. A CCMS such as XDocs will automatically validate all links and references every time content is checked-in/out.
- **Integration with 3rd party tools** – good APIs enable a CCMS to link with other data sources within the organization.

A solution based on a CCMS has the same functionality as one based on Git but they are within the CCMS umbrella.




## Where a CCMS falls short

- Cost
- Flexibility & extensibility
- Proprietary nature
- Implementation overhead

## Or does a CCMS fall short?

### CCMS Myth #1 – Cost

- A CCMS comes with functionality that you would otherwise have to develop – at a cost both in labor and in time.
- You will pay for support and maintenance but in return receive support from the provider which will minimize downtime in the event of problems.
- This is a long-term investment , year one costs will be almost certainly be higher with a CCMS, but full Return on Investment within 3 years should be a very realistic target.

## CCMS Myth #2 – Flexibility, extensibility

- A CCMS with good APIs offers flexible opportunities to link to other systems.
- The CCMS vendor may have already developed integrations which wil lsave time and development costs.

## CCMS Myth #3 – Proprietary nature

- Yes, but this in turn leads to reliability. A CCMS is a commercial product and has been built to meet commercial standards.
- Different vendors have different approaches to integration. You should discuss your needs with the vendors or an independent consultant prior to selecting a system.
- Some CCMS such as XDocs can be easily configured by the customer without the need of vendor involvement.

## CCMS Myth #4 – Implementation overhead

- Many CCMS are available as SaaS or Hosted solutions, this method of deployment will minimize the impact on your own IT department.
- Man-power overhead – this is arguably less with a CCMS where support is coming directly from the vendor.
- Existing functionality versus the time taken to create functionality with Git. The functionality within the CCMS has been developed to commercial satndards and wil llikely be superior.

# What you need to do

Irrespective of Git or CCMS, a successful deployment depends upon good preparation which can be broken down into the following.

## Hard resources

A business needs to spend money to make money. Smart investments need to be made around:

- **Time** – a deployment of a CCMS or Git platform needs to be allocated enough time to be developed, fine tuned and deployed. This means you should look for a time window where there will be less demands on the team from day-to-day workload. If you are going down the CCMS route take time to understand your needs and evaluate the various CCMS on the market to detaermine which best meets your needs/budget.
- **Additional tools** – although Git may be open source, other commercial tools may still be required to complete the solution, be prepared to invest in them.
- **Training** – new tools require training, don't try to scrimp on training costs. The sooner users are trained the faster you will start to see a return on the investment.

- **Outside expertise** – it is important to accept that you might not have sufficient in-house expertise. Consultancy firms and CCMS vendors have deployed hundreds of solutions, you should take advantage of this knowledge - it will pay dividends in the long run.

### Technical infrastructure

No tool is a silver bullet, a successful deployment needs to be/have:

- **Savvy and respectful practitioners** – users must understand how a system should work and know their own role within the content lifecycle.
- **Fit to requirements** – a solution needs to be configured to match the current needs of the organization and be able to address identified future needs.
- **Adequate maintenance and enhancement resources** – there must be available resources to support the solution.

### Specialized knowledge

When moving to a new way of working it is obvious that many teams will lack seasoned resources knowledgeable in the areas of:

- **Information Architecture** – the IA determines which topic types to use; helps select semantically appropriate tags, elements that will enable fastest content creation and best presentation. IA will also highlight the need for specialization and reuse techniques etc.
- **Taxonomy** – building effective taxonomies is a specialist skill and you should consider bringing in an external resource.
- **One-offs** – like XSLT transformation development and conversion options to transform unstructured content into valid DITA XML.
- **Branch, Merge and Release Management** – some ways of working that are unique to DITA will of course be unfamiliar to companies making the transition to structured authoring.

## Organizational maturity sets the conditions for innovation.

A mature organization has:

- **Standard processes** – put these in place if they are not already there.
- **Institutionalized best practices and documentation** – all procedures should be well documented and understood.
- **Specialized expertise in the different responsibilities required by the business –** if you are moving to DITA it is unlikely that you have specialized expertise already in place.

## Next steps

If you are going to start out with Git you need to take the following steps to get the best result:

- **Git training** – investment in training is seldom wasted, the better someone can use a tool the more effective they will be. Training gets people doing things properly and up to speed faster. It also ensures everyone is working in the same way.
- **Understand and develop your User Stories and Use Cases** – these determine how you will use the system, how different types of users will interact with the system.
- **Develop standard workflows, including Git** – think about and agree how you want the content lifecycle to look, then build workflows that support this.
- **Specialized roles for branch management** – agree on your approach to Branch, Merge and Release
- Specialized troubleshooting for conflict resolution

## Information design

Of utmost importance is standardization:

- Projects
- Maps
- Info types
- Reuse at all levels, with all mechanisms
- Enforcement/governance
- Roles & responsibilities
- No longer can everyone do everything. Move people into specialized Content Operation roles:
- Information Architecture
- Reuse Specialist
- Delivery (CI/CD or other)
- Release Management

## Leverage oXygen

Get the most out of your commercial tools, even if it's "just" your XML Editor (oXygen for example). These tools may fully or partially support your requirements in terms of:

- Projects
- Custom frameworks

- Rules checking
- CSS for authors
- Underutilized views

## Your content platform is a product – your product

Treat your platform as a product, because it is a product and it is “alive”.

- **Maximize learning velocity** – acquire skills as quickly as possible. The sooner you/the team are fully conversant with the solution the sooner everyone will benefit.
- **Development backlog** – try to avoid new functionality being delayed.
- **Maintenance plan** – plan for updates to the solution in advance and work towards them.
- **Enhance road-map** – build your own “Road-map” of how you want the solution to develop in the short, medium and long-term.

## If you’re staying in Git, do these things:

- Redefine model, processes, and what you want from your tools.
- Invest in change management and professional development opportunities (training, conferences).
- Treat your platform like a product that you own.

## If you’re moving on to a CCMS, do these things

Do all of the same things, plus:

- **Develop your use cases and user stories** – so that you fully understand how you want the system to perform.
- **Build a Proof of Concept system** – and test your use cases and user stories. You can do this with more than one system to determine which best addresses your use cases and user stories.
- Undertake training on all new components.
- **Pilot in the system** – start with a single project and give yourself time to complete it. You are using a new system, don’t put time pressures on yourself the first time you use it in production.
- Plan and conduct an intentional, managed roll-out.

## About the Authors

**Rik Page** is Director of Sales and Marketing EMEA at **Bluestream** - developers of the XDocs DITA Component Content Management System.

He has 19 years structured content and document management experience together with 14 years DITA.

Rik is based London, England.

email: **rikp@bluestream.com**

**Frank Miller** is President & Principal Consultant at **Ryffine** a leading independent consultancy helping companies plan, migrate to and improve their DITA strategies.

He has 14 years DITA implementations experience and a journalism background.

Frank is based in Denver, CO, USA

email: **frank.miller@ryffine.com**