# DITA Writing Methodology

Because DITA is interpreted and strictly-typed, writing with it can prove difficult and error-prone. To help alleviate these burdens, the following writing method attempts to supplant your typical writing habits to perhaps a more productive way when writing with DITA.

**Tools**

- Text editor with XML syntax highlighting and a directory view option
- DITA 1.2 spec + Open Toolkit (OT) 2.1
- Web browser
- PDF viewer
- Git/Github + Desktop app/Terminal
- Terminal

## Test-Driven Topic-Modeling

I developed the method below with this particular assignment and class-context in mind. It is also contingent on using the above writing tool configuration; namely, a simple code editor and the OT. That said, if you end up writing DITA in some other context, you may need to adapt in particular ways, based on the new writing technology.

The method walks you through 3 main goals: 1) *Drafting your topic model's task and scope*, 2) *Creating the initial architecture of your topic model*, and 3) *Writing, testing, and revising your topic model*. Please keep in mind that these tasks are not necessarily discrete. Writing in general, and true to writing with DITA, means iterative and recursive processes of writing, evaluating, and revising. The 3 phases will most likely blend in idiosyncratic ways. What's important is that you recognize that you must distribute your writing work over time to ensure a richer topic model with some basic, yet intelligent, reuse decisions.

*1.) Drafting your topic model's tasks and scope*

i. Write a list of user goals (tasks) and group them in different conceptual relationships that you think have reuse potential. For example, are the tasks sequential, grouped as conceptually related (e.g., grids, centering design strategies in CSS), or a choicetable path (e.g., use div or span).

   Tip: Write short little memos about these groupings or to create them! In other words, write this stuff out to help you think through these decisions and content-construction moments.

ii. Conduct a task analysis, which will help you scope your topic model for this assignment. Additionally, it will help you generate content to take to the next set of steps.

   Note: Note how you arrived at your decisions. These notes will prove invaluable to you when you write your memo that accompanies your topic model.

**Output: You should have draft material to take to your code editor.**

*2.) Creating the initial architecture of your topic model*

i. Create a Github repo with your desired project name. Keep it short and simple, but meaningful.

ii. Clone this repo to your local computer in your desired workspace.

iii. Set up your directory and file structure on your local computer. See the example structure on Github: https://github.com/Writing-with-Digital-Technologies-f15/DITA/tree/master/examples/css-grids

(a) Create a shared-topics folder and a folder for each user goal.
(b) Create a topics folder in each user goal directory to store your topic files.
(c) If you are incorporating media (images, etc.), create asset folders for each task/user goal.
(d) Also, create a shared-assets folder in the parent directory.

iv. Add a shell .ditamap in the parent directory.

Note: Remember to follow the naming-scheme standard.

v. Add shell task .dita topic files in each sub-directory/topic folder

Note: Remember to follow the naming-scheme standard.

vi. Add README.md files in all other sub-directories, so git will track these directories.

vii. Again, look at the example topic model structure to get a sense for how to set up your DITA topic model. If it looks ready to commit and push to Github, do so.

Note: Be sure to write a concise and meaningful commit message.

**Output: You should now have a basic architecture for your topic model.**

*3.) Writing, testing, and revising your topic model*

i. Start writing your topic model by writing your desired task topic. Before you begin writing it, write the topicref in your .ditamap file for testing purposes.

Note: Remember that Bellamy *et al*. discuss how task topics are supported by concept and reference topics, so if you start with one task, you can test it independently first. From there, address how the task needs particular concept and reference topic support.

ii. For each topic file that you write, be sure to define an "id" attribute for at least the parent tag.

iii. Include relevant tags specific to each type of topic-type.

iv. Once you complete a particular component of the topic, (e.g., a part of a step procedure), save it and test it with the OT.

v. Address any errors.

Note: Be sure to not to work on multiple files and components across files. Otherwise, you will compound the number of potential errors to address.

vi. Once the errors and content are at a place of resolve, commit the changes, and push it to Github.

vii. Repeat as necessary until you finish a drafted topic-type.

viii. Once you complete a draft topic-type, take a moment to jot any important design or content decisions, as useful moments to consider discussing in your memo.

ix. Finally, assess the content in the topics in relationship with other topics and the scope and goal of your topic model. Use the following questions to help you do so:

(a) Based on Bellamy *et al*. rubrics for each topic-type, do the topics include appropriate content?

(b) Are there any potential reuse cases?

(c) Do any relationships between topics suggest revising the .ditamap in any way?

Note: Do not limit yourself to these questions. Please be open to situational concerns. Jot them down and note any reason for one decision over a set of other potential pathways.

x. Repeat as needed.