NOTE: *My prose is in the DTD.xml file itself. According to this answer this should be allowed https://campuswire.com/c/G25D0F0A9/feed/155*

How did you decide to represent the data in the way that you did? Why did you choose the elements and attributes that you did?

I decided to represent the data based on reusability, readability, and data independence. The textual document I choose is structure and allowed for logical break down of the text. I extracted what appears to be repeated and generic enough and created a shared element for it, for example: I intended for the element 'section' to be generic, and reused it in other element definitions throughout the document. The element 'section' has a 'header' attribute that corresponds to an actual header section on the represented document. In addition to that, I thoughtfully worked to keep the DTD flow readable and easy to understand with clear descriptive inline documentation and well named elements and attributes.

What were the hardest decisions you had to make in this design process?

It was whether to represent the further details of the element as attributes or children. I based my decision on what makes the document more readable and reusable

How does your DTD design support data independence?

The design support data independence on different levels. For example: The element 'section' allows for textual blobs to be added, changed, or removed from the represented textual document without the need to change the DTD. In addition to that, there are child elements that allow the represented document to grow as needed like: bbi, tsi, and ti (refer to my DTD for a refresher on what each one means).

How may your DTD design support the overarching goals of data curation (revisit objectives and activities of Week 1)?

- Collection: provides a path for collecting and acquiring data from repositories that utilize the same readme template
- Organization: uses DTD standards in naming elements and attributes in addition to documentation
- Preservation: Ensure that data will be understandable and useable in the future through well-named elements and attributes. In addition to descriptive documentation
- Workflow: supports a systematic way to run data through workflow (e.g.: find all repositories that belong to a specific software engineering team)
- Sharing: Supports sharing data between different teams and entities because of the simplicity of the language
- Modification: the DTD is open for modification and changes

What are the design's pros and cons?

Pros: Organized - documented - scalable - reusable