DITA Case Study: The Joseph Smith Papers Project

Eric **Severson**

Abstract

Brigham Young University, in cooperation with the Church of Jesus Christ of Latter Day Saints and assisted by Flatirons Solutions, is in the midst of an ambitious project to XML-encode all of the historical papers of LDS founder Joseph Smith. Starting with Joseph Smith's journals, the encoded XML is being used to produce both high-quality printed books, and to build a sophisticated online repository for scholarly research. Even though DITA was designed for technical documentation, it was selected for this project because of its ability to create a flexible set of information objects that can be re-purposed across many potential uses and contexts. For example, a daily entry in one of Joseph Smith's journals, together with scholarly annotations, can be treated as a set of specialized DITA topics. This whitepaper illustrates how the content was encoded in a TEI specialization of DITA, including an analysis of the applicability and limitations of DITA specialization techniques. It also shows how this XML fit into the complex workflow associated with scholarly encoding efforts.

Table of Contents

1. Introduction

Brigham Young University, in cooperation with the Church of Jesus Christ of Latter Day Saints and assisted by Flatirons Solutions, is in the midst of an ambitious project to XML-encode all of the historical papers of LDS founder Joseph Smith. Starting with Joseph Smith's journals, the encoded XML is being used to produce both high-quality printed books, and to build a sophisticated online repository for scholarly research.

Even though DITA was designed for technical documentation, it was selected for this project because of its ability to create a flexible set of information objects that can be re-purposed across many potential uses and contexts. For example, a daily entry in one of Joseph Smith's journals, together with scholarly annotations, can be treated as a set of specialized DITA topics. In the printed form of the journal these topics would simply be output in sequence. In the research database, however, a particular entry of interest might be linked to other Joseph Smith artifacts such as correspondence, legal documents, and financial records. Each of these, in turn, would also be encoded as specialized DITA topics.

Because it is crucial that this project stand up to rigorous scholarly standards, it was necessary to encode original erasures and corrections, interlineations, marginalia, spelling errors and normalizations of name and place abbreviations. Of course, the obvious choice for this kind of markup was TEI (Text Encoding Initiative). To accomplish this, we used a relevant subset of TEI as a DITA specialization. The result is that the Joseph Smith papers are encoded as a combination of specialized DITA topics, but the lower-level markup of the source text follows TEI encoding standards. This blended format is at most "one transform away" from both valid TEI and valid DITA markup.

This whitepaper illustrates how the content was actually encoded in DITA-based XML, including an analysis of the applicability and limitations of DITA specialization techniques. It also shows how this XML fit into the complex workflow associated with scholarly encoding efforts. This workflow includes preparing XML-based transcriptions, three steps of validation against original source material, proposing and refining scholarly annotations, multiple copy edit and verification steps, and formal internal and external peer reviews.

The material in this whitepaper is presented with the kind permission of Brigham Young University and the Church of Jesus Christ of Latter Day Saints. Flatirons Solutions worked on the JSP project with a brilliant team from Brigham Young University, including Shawn Jordan and Anthony ("AJ") Mott, who actually designed the DITA and TEI-based DTDs, Mark Ashurst-McGee, who provided expert advice on scholarly standards and text verification, Ron Esplin, the JSP Editor-in-Chief, David Willden, JSP executive sponsor and project manager, and a host of other editors and scholars.

2. Overview of the JSP Project

2.1. Goals and Objectives

The Joseph Smith Papers (JSP) project has two primary objectives:

- *To produce a set of high-quality printed books containing Joseph Smith's journals, correspondence, legal documents, financial records and other historical papers.*These books must be a faithful rendition of the original source material, including both an accurate transcription of content, and appropriate indications of original abbreviations, misspellings, erasures, and other source issues. They also need to be supplemented by scholarly annotations that help clarify context, provide historical background, and provide linkages to related material.

- *To be able to use the same source files to create online material for scholarly research.* This includes the ability to find collections of individual items meeting specific search criteria, and the ability to automatically link related items (e.g. all items for a given subject or week). These "items" would include individual journal entries, letters, legal briefs and financial records, as well as relevant individual scholarly annotations.

The following tactical objectives were also established:

- Establish an orderly and efficient editing / review process.
- Ensure all sub-processes are visible and collaborative.
- Preserve the absolute integrity of source text.
- Optimize the effective use of scholarly annotations.
- Meet established scholarly standards for text markup and verification.

2.2. Chosen Technology

The JSP project uses an XML-based electronic format, based on a combination of DITA and TEI standards. XML is edited using Arbortext Epic and stored in Documentum, while images of original pages are displayed in PDF. Documentum workflows control the entire editorial and review cycle, with final print production handled by a separate Quark-based process.

2.3. Challenges of Scholarly Markup

As the figure below illustrates, faithfully rendering original content is no small task. Now carefully preserved in Church archives, the pages of Joseph Smith's original journals are faded, smudged, and difficult to read. To meet established scholarly standards, three independent levels of text verification are required – once when the text is initially transcribed to electronic format, once by scholars as they analyze the text for potential annotations, and once by an expert text verifier who does a final validation and may use high-tech spectrographic techniques to resolve any remaining ambiguities. Each time text verification is completed, the electronic source content must be "locked down" so that its integrity is preserved.

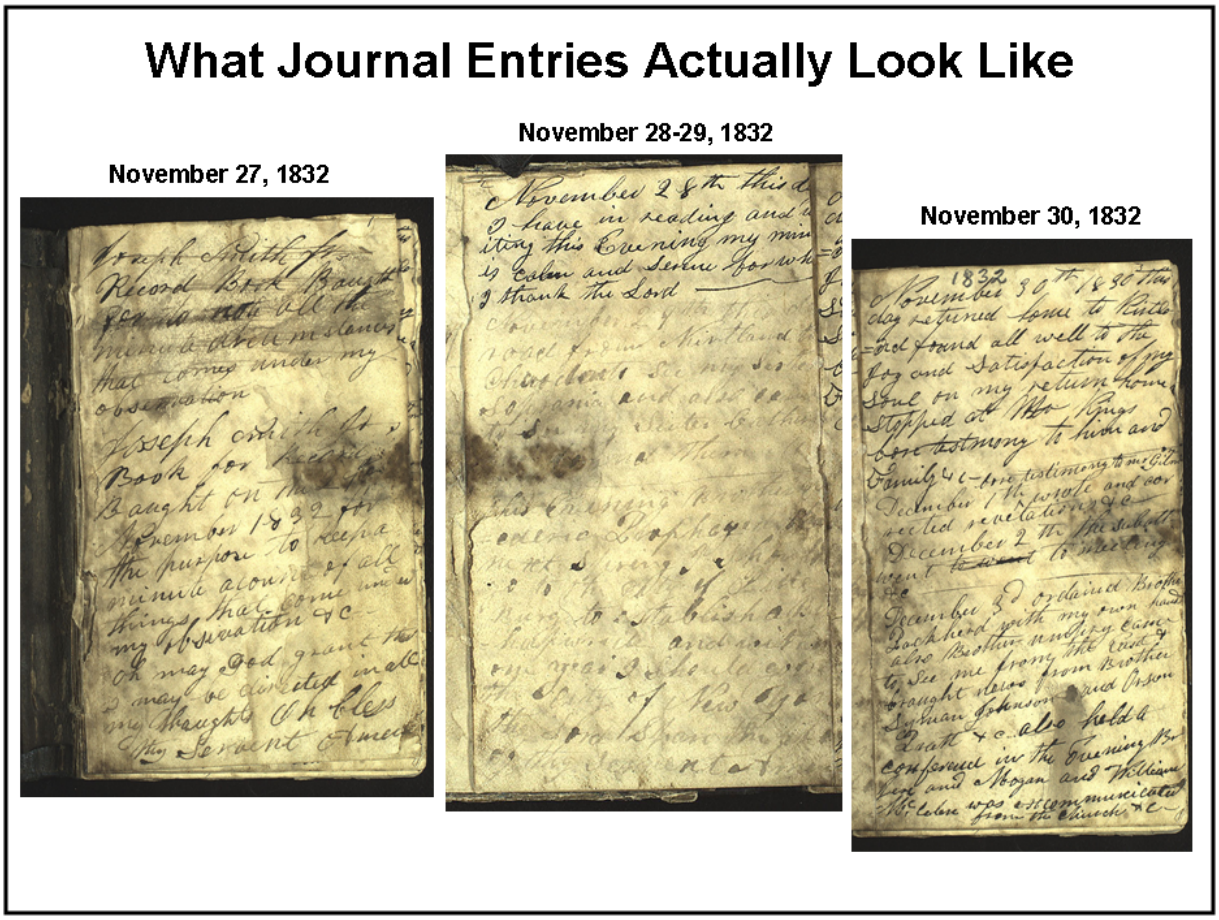

Figure 1. Faithfully Rendering Original Source Isn't Easy

Scholarly standards also require that original erasures, additions and deletions, abbreviations, spelling errors and notations are faithfully preserved. This allows other scholars to have the raw material necessary to form their own opinions about potential interpretations and resolution of ambiguities. Such markup may include:

- Original additions, deletions, erasures and corrections.
- Faded text and damaged pages, as well as unclear or unreadable text.
- Original spelling errors.
- Implied text (words apparently left out of the original).
- Normalized name and place abbreviations.
- Marginalia (notes in the margins of the original page).
- Interlineations (things added in between lines).
- Change in handwriting.
- Original line and page breaks.

Scholarly annotations are also used to clarify and enhance understanding of the original content, including:

- Providing historical context for conversations and meetings.
- Providing bibliographical information for the people involved.
- Providing geographical information for the places mentioned.
- Cross-referencing other related materials.
- Providing a variety of other clarifications.

Figure 2 illustrates the complexity of this process for the three journal entries shown in Figure 1.

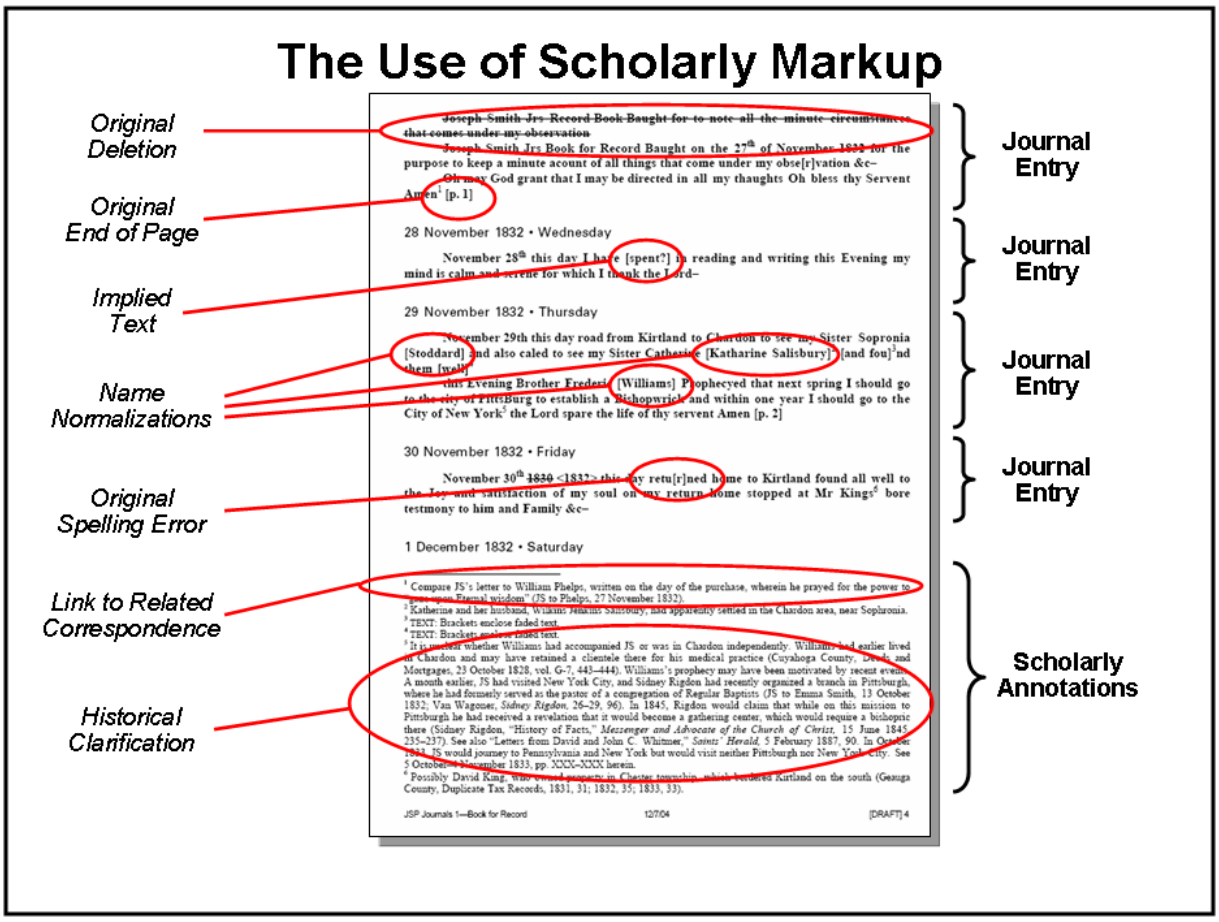

Figure 2. A Transcription of These Pages with Markup and Annotations

2.4. Rigorous Editorial and Review Workflow

To maintain the necessary degree of scholarly rigor, the JSP project uses a rigorous editorial and review workflow. As illustrated in Figure 3, this is broken into eight major pieces:

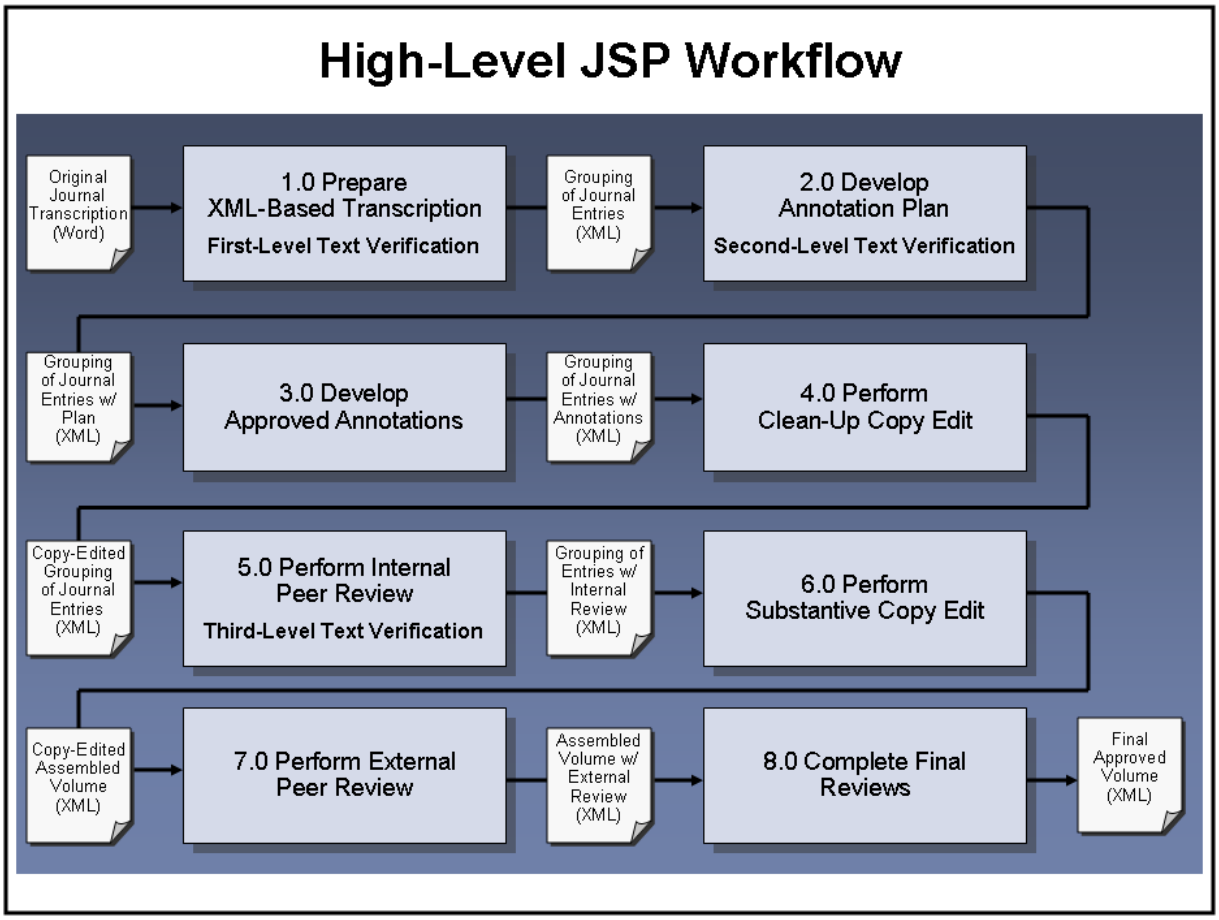

Figure 3. JSP Uses a Rigorous Editorial and Review Workflow

First, the "Prepare XML-Based Transcription" process (1.0) takes original Word-based transcriptions prepared by the Church Archives group, and transforms them into XML-based documents that can be used by Volume Editors as the basis for preparing annotations. As part of the transformation process, text is verified against images of original manuscripts, and names and places are normalized (e.g. "M.A.S.", "Mary" and "M. Smith" are all mapped to "Mary Ann Smith"). Also, journal entries are combined into groups (approximately 50 pages each) that form logical units of work for Volume Editors and reviewers as they go through the rest of the process.

The "Develop Annotation Plan" process (2.0) includes second-level text verification and completion of name / place references, along with the development of a set of proposed annotations. During this phase, multiple Volume Editors work on a designated grouping of journal entries, each able to view the entire group but with the collaborative ability to check out and edit individual journal entries.

Next, the "Develop Approved Annotations" process (3.0) includes the development of actual text for all proposed annotations approved in the prior phase. As in the previous phase, multiple Volume Editors work on a designated grouping of journal entries, each able to view the entire group but with the collaborative ability to check out and edit individual journal entries. Any research necessary to complete the entries is also performed at this point.

The "Perform Clean-Up Copy Edit" process (4.0) occurs after all annotations have been approved, and before the journal entry group is sent for Internal Peer Review. This process includes both source checking and copy editing for spelling and grammar. During this phase, Volume Editors are involved in responding to Copy Edit issues, but the Copy Editor retains control of the content.

The “Perform Internal Peer Review” process (5.0) allows senior editors and other key internal reviewers to provide comments after the Series Editor has approved all annotations. In addition, the third and final text verification is performed at this point. After review comments are submitted, Volume Editors formally correlate them and the result is approved by the Series Editor.

The “Perform Substantive Copy Edit” process (6.0) occurs after all journal entry groups have gone through Internal Peer Review. At this point the entire volume is assembled and reviewed for overall style, tone, and consistency, and addition to spelling and grammar issues. Once the Substantive Copy Edit is complete, the volume is sent on for External Peer Review.

The “Perform External Peer Review” process (7.0) then routes the volume for review and comment by outside scholars. After comments are submitted, Volume Editors formally correlate them and the result is approved by the Series Editor.

At this point, the “Complete Final Reviews” process (8.0) includes the final steps necessary to approve the finished volume and prepare it for the publishing process. These include a final copy edit, Editorial Board review, and GA approval.

3. JSP’s Use of XML: Combining DITA and TEI

Since JSP is a scholarly research effort, the Text Encoding Initiative (TEI)[1] is the natural choice for XML markup. And in fact, not only does JSP require TEI encoding, but it would probably have used TEI exclusively if printed books were the only objective.

However, TEI was not designed for the modular chunking and linking required by an online research database. For this purpose, the flexibility, reusability and topic orientation of DITA[2] seemed like the right choice. TEI, with its lack of modularity, would not suffice for this purpose.

Although DITA was developed for technical documentation – not scholarly markup – it is highly extensible, offering a combination of *structural or “topic” specialization* (new information types derived from standard DITA types) and *domain specialization* (extended vocabulary that can apply to any information type). For JSP, the key questions in using DITA were as follows:

- *Topic (Structural) Specialization.* Could DITA, which normally specializes to technical documentation topic types such as concept, reference, and task, be flexible enough to handle JSP information types like journal entries and annotations?

- *Domain Specialization.* Would DITA be flexible enough to handle the complexities of TEI as an extension to its “tech doc” oriented element set?

- *TEI Interoperability.* Even if DITA could be effectively specialized for TEI, would it still be possible to easily produce valid TEI as an output? As a JSP requirement, electronic content needed to be at most “one transform away” from valid TEI.

- *DITA Validity.* If TEI can be accommodated, would the result still be valid from a DITA perspective? If not, could we also be at most “one transform away” from valid DITA?

The remainder of this section describes how these issues were resolved in the JSP application, and summarizes the “lessons learned” in regard to the limits of DITA specialization.

[1]The TEI (Text Encoding Initiative) was developed by Lou Burnard, Michael Sperberg-McQueen and others in the late 1980s. It was officially released as an SGML application in 1994 and was converted to XML in 2002. TEI is now maintained by the TEI Consortium, and can be accessed at www.tei-c.org.

[2]DITA (Darwin Information Typing Architecture) was originally developed by Michael Priestly, Don Day, David Schell and others in IBM’s XML Workgroup and ID Workbench teams. Since early 2005, DITA has been a public standard maintained by OASIS, accessible at http://www.oasis-open.org.

3.1. JSP Topic Architecture

For the initial JSP application, the basic information unit (topic) is an individual journal entry. Later, as the scope is expanded to include other types of Joseph Smith papers, specialized topic types for letters, legal briefs and financial records will be added.

In the JSP architecture, each of the major topic types is used to contain header information about the journal entry or other artifact. These topics can be used as the basis for higher-level scholarly research and as the basis for creating summaries and indices. Under each of these major topic types, two other "sub-topic" types are attached. The first, called the "transcription," contains the actual content of the journal entry in TEI format. The second is used for scholarly annotations (one topic instance per annotation).

Meanwhile, from a DITA perspective each of these topic types is specialized from a common "JSP Topic," which in turn is derived from the standard DITA Topic. The standard DITA specializations for technical documentation – Concept, Reference and Task – are not used in JSP.

Figure 4 summarizes this architecture:

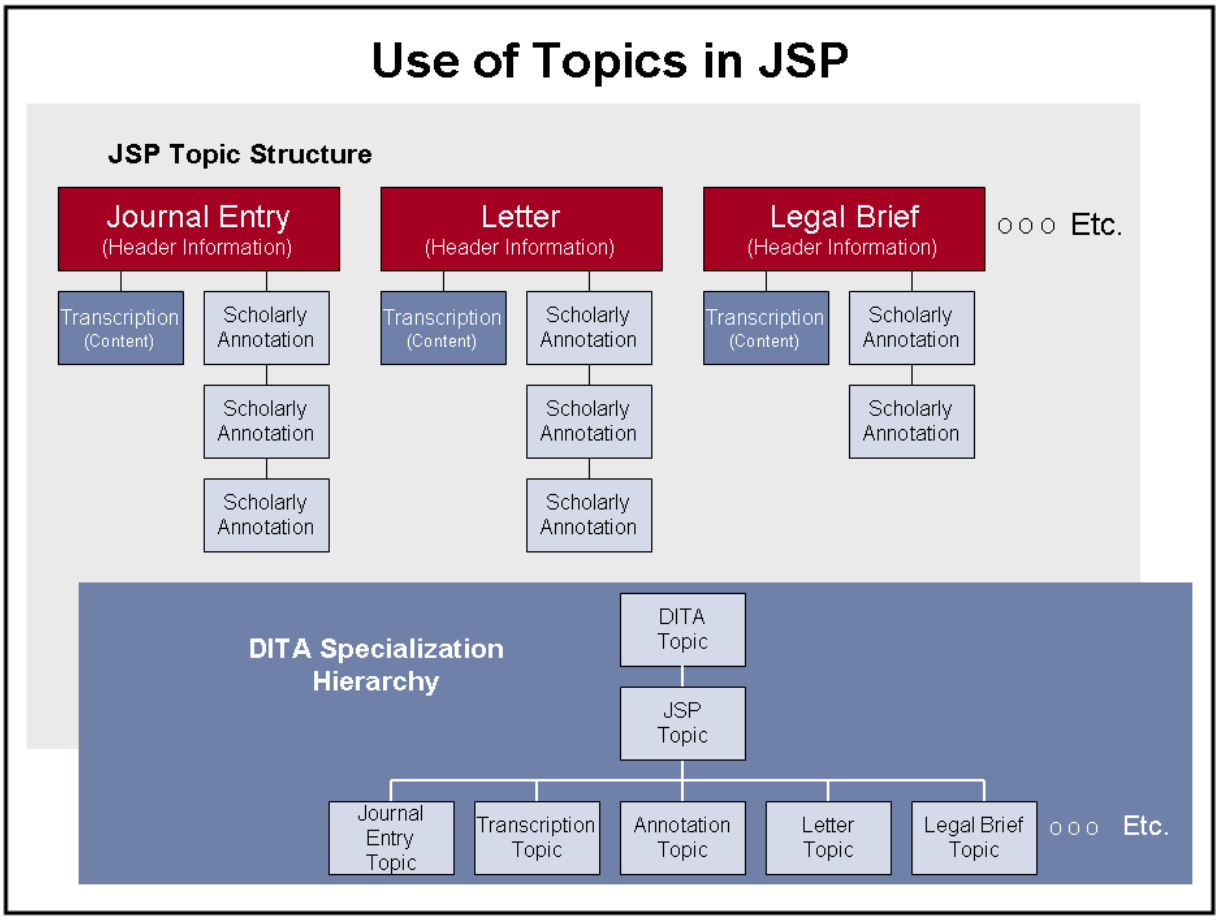

Figure 4. The JSP "Topic" Architecture

3.2. Specializing the Overall JSP Topic

Certain JSP-specific information is common to all major topics, as well as to transcriptions and scholarly annotations. These specializations are done in a higher-level “JSP Topic” so they can be automatically inherited by all the other topic types. Figure 5 shows a sample of these elements, consisting of summaries of persons, places and doctrines mentioned in the content, together with index and glossary terms:

```
<!-- Define the content element entities -->
<!ENTITY % persons        "persons" >
<!ENTITY % places         "places" >
<!ENTITY % doctrines  "doctrines" >
<!ENTITY % indexTerms     "indexTerms" >
<!ENTITY % glossaryTerms  "glossaryTerms" >
Etc.

<!-- Define the elements -->
<!ELEMENT %persons; (%tei.persName;)* >
<!ELEMENT %places; (%tei.placeName; | %tei.geogName;)* >
<!ELEMENT %doctrines; EMPTY >
<!ELEMENT %indexTerms; (%tei.term;)* >
<!ELEMENT %glossaryTerms; (%tei.term;)* >
Etc.

<!-- Define the specialization hierarchy -->
<!ATTLIST %persons; class  CDATA "- topic/p jsp.topic/persons" >
<!ATTLIST %places; class  CDATA "- topic/p jsp.topic/places" >
<!ATTLIST %doctrines; class  CDATA "- topic/p jsp.topic/doctrines" >
<!ATTLIST %indexTerms; class  CDATA "- topic/p jsp.topic/indexTerms" >
<!ATTLIST %glossaryTerms; class  CDATA "- topic/p jsp.topic/glossaryTerms" >
Etc.
```

Figure 5. Sample JSP Topic Module

Please note that the overall JSP Topic module defines common specialized elements that can be inherited by other JSP-specific topics, but does not provide the topic specializations themselves. These are done in the individual topic modules, as described in the following sections.

3.3. Specializing the Journal Entry Topic

Figure 6 illustrates the topic specializations used for a JSP Journal Entry. Similar specializations will be done for letters, legal briefs and financial records. Note that the standard DITA topic title and short description are retained, but the DITA topic body is replaced with the Journal Entry Header information, the components of which are inherited from the overall JSP Topic. As in the standard Topic structure, this is followed by related links and a set of sub-topics. In the case of JSP, however, these sub-topics are specifically restricted to a set of transcriptions (at each of the verification levels), followed by scholarly annotations. Figure 7 provides a sample of how these specializations are applied in XML.

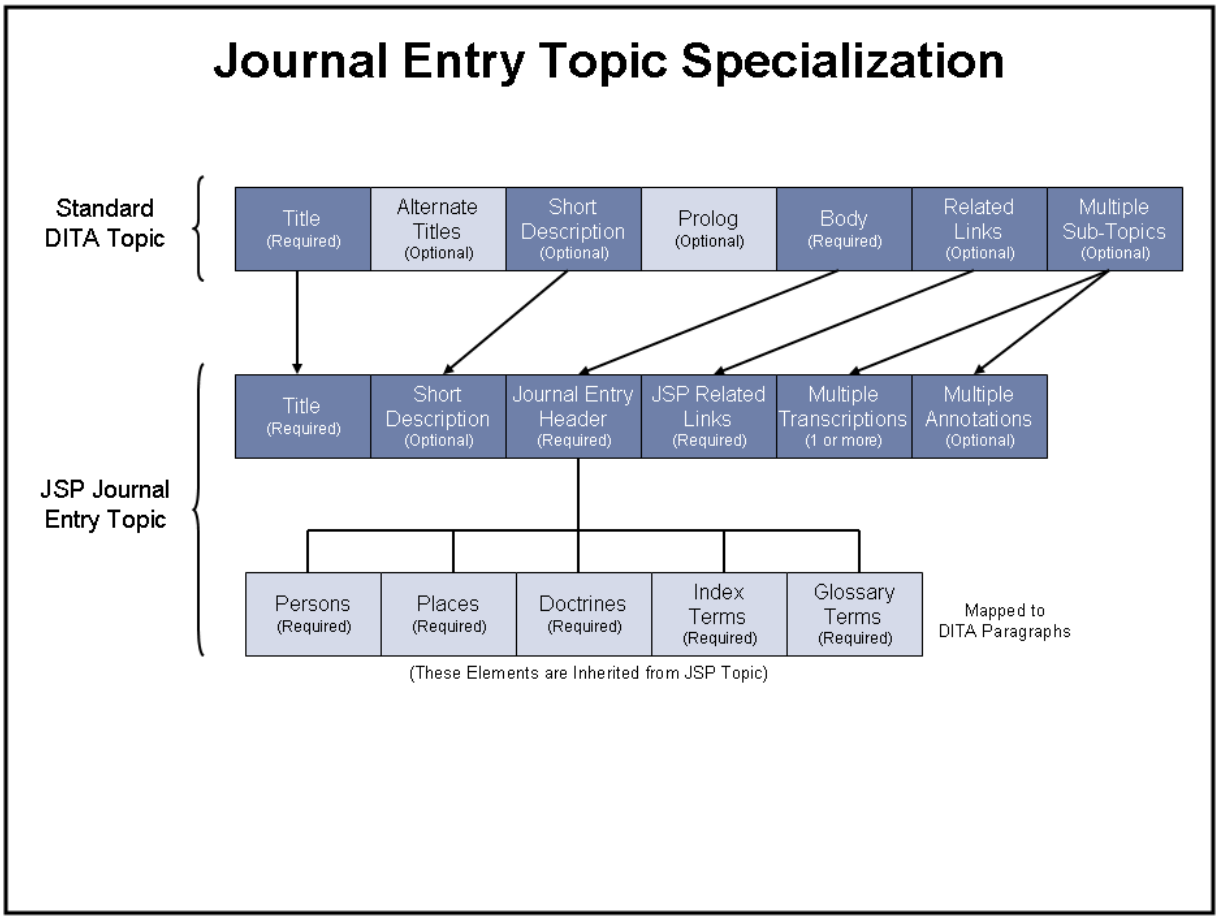

Figure 6. Specialization of the JSP Journal Entry Topic

```
<!-- Define the content element entities -->
<!ENTITY % journEntry "journEntry">
<!ENTITY % jeHeader "jeHeader">
Etc.

<!-- Define the elements -->
<!ELEMENT journEntry  (%title;, (%shortdesc;)?, %jeHeader;,
 %jsp.related-links;,  (%info-types-transcription;)+,
 (%info-types-annotation;)*)>
<!ELEMENT %jeHeader;  ((%persons;), (%places;), (%doctrines;),
 (%indexTerms;), (%glossaryTerms;))>
Etc.

<!-- Define the specialization hierarchy -->
<!ATTLIST %journEntry; class CDATA "- topic/topic jsp.topic/topic
 journEntry/journEntry">
<!ATTLIST %jeHeader; class CDATA "- topic/body jsp.topic/body
 journEntry/jeHeader">
Etc.
```

Figure 7. Sample JSP Journal Entry Topic Module

3.4. Specializing the Transcription Topic

The “Transcription” topic contains the original source content in TEI format, saved at each of the required verification levels. Transcriptions are also specialized from the standard DITA Topic, as illustrated in Figure 8. In this case, the DITA “body” element is mapped to the JSP “transcription” data. This in turn is split into “header” and “source text” components, both treated as specializations of DITA “section” elements. Source text resolves to individual paragraphs containing TEI-based content.

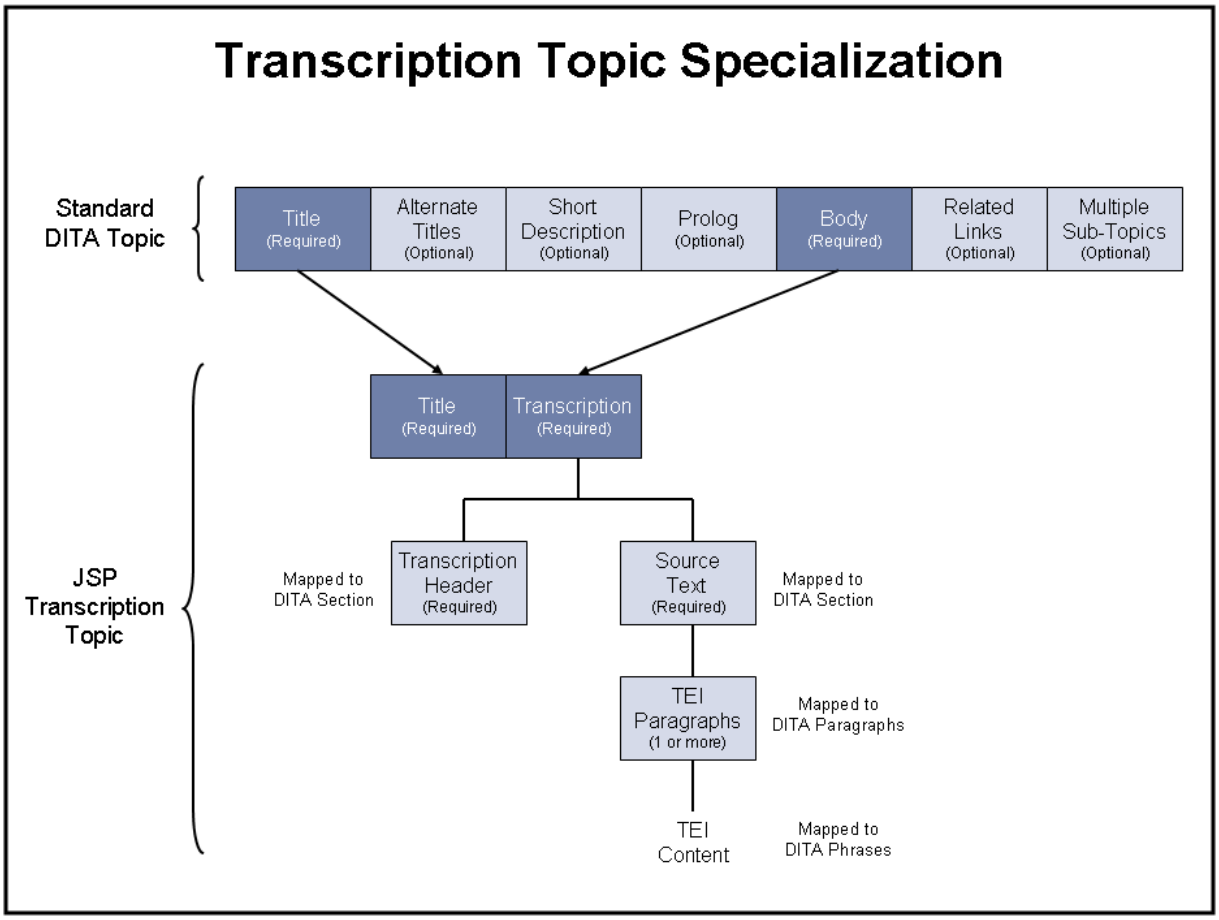

Figure 8. Specialization of the JSP Transcription Topic

3.5. Specializing the Annotation Topic

Planning, development and review of scholarly annotations is the most complex portion of the JSP workflow. Before investments are made in time-consuming research, editors review each proposed annotation and provide significant feedback in the form of embedded comments. For those annotations that are approved, research notes may be developed by external researchers, and original editorial comments may be incorporated into the final annotation. As annotations are completed and sent back for approval, it is also important to track which annotations are ready for review and which are not.

Figure 9 shows how the Annotation Topic was specialized for JSP. Here the standard DITA “body” becomes the annotation, which in turn is split into header information, the annotation content itself, and attached research notes. Annotations use a subset of TEI content, and also contain elements for editorial comments. XML attributes are used to track the status of each annotation and control the annotation editorial cycle.

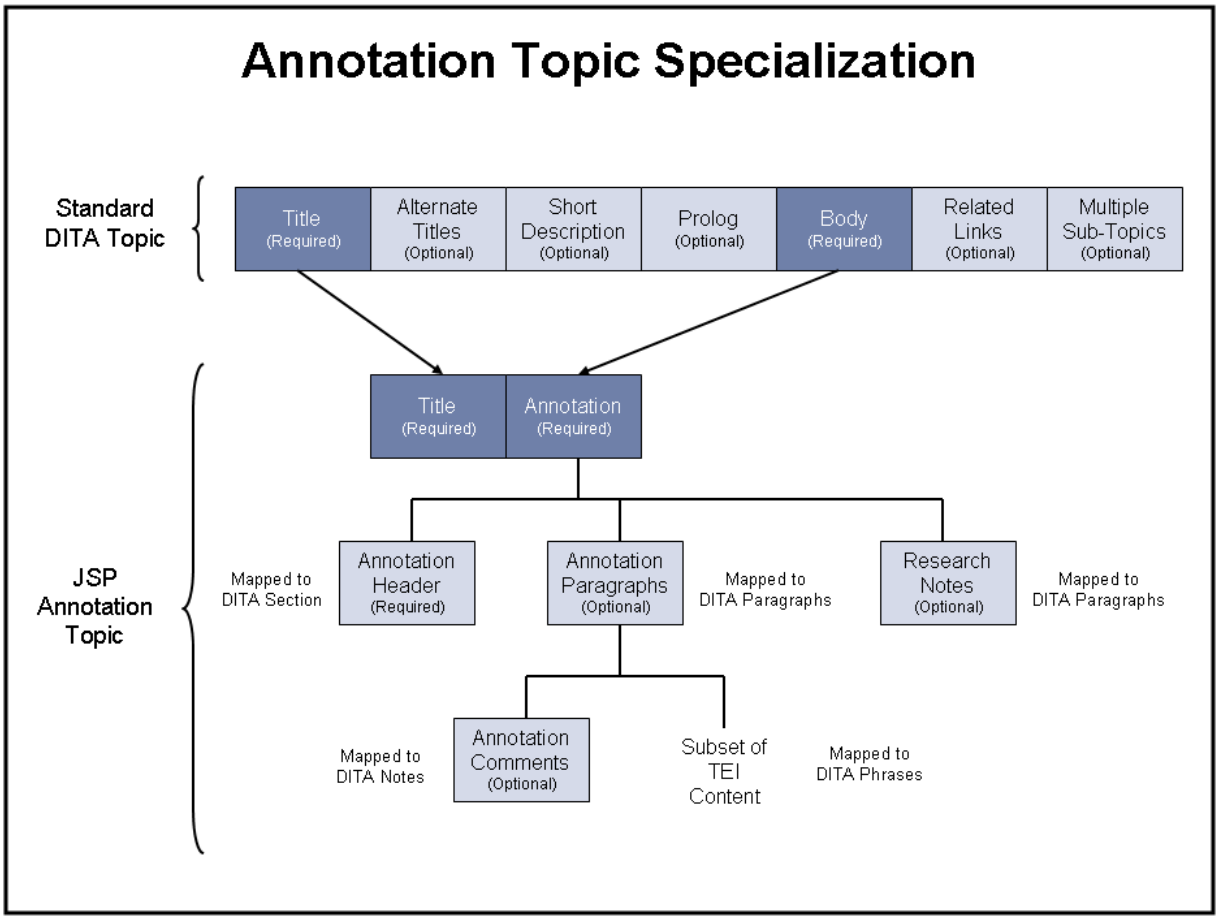

Figure 9. Specialization of the JSP Annotation Topic

3.6. TEI Domain Specialization

Ideally, the TEI vocabulary would be treated as a standard domain specialization that could be included within all JSP topic types. This section explores our attempts to make this work within DITA, and analyzes the constraints and limitations we discovered along the way.

Most TEI tags mark specific issues within text, such as names, places, abbreviations, or a phrase that is spelled incorrectly, unclear in the original, originally erased and so forth. These elements can be mapped in a relatively straightforward manner to the DITA phrase element (“ph”), which can consist of text (#PCDATA) and other nested phrase elements. Some elements – such as bibliographic information and monograph – have very complex internal structures. However, from a DITA perspective all of these structures can still be mapped to some combination of nested phrases.

Below is a portion of the Element Declaration File for a TEI domain specialization that uses this approach. To maintain TEI validity, parameter entities must be used to model the internal structure of the various TEI elements; because all internal components resolve to DITA phrase elements, the result is also valid from a DITA perspective:

```
<!-- Define the content element entities -->
<!ENTITY % tei.abbr "abbr">
<!ENTITY % tei.add "add">
<!ENTITY % tei.addName "addName">
<!ENTITY % tei.analytic "analytic">
Etc.

<!-- Define entities used for TEI content models -->
<!ENTITY % m.data  "%tei.abbr; | %tei.geogName; | %tei.name; |
 %tei.persName; | %tei.placeName;">
<!ENTITY % m.date "%tei.date; | %tei.dateRange; | %tei.dateStruct;">
<!ENTITY % m.edit "%tei.add; | %tei.corr; | %tei.damage; |
 %tei.del; | %tei.expan; | %tei.sic; | %tei.space; | %tei.supplied; |
 %tei.unclear;">
<!ENTITY % m.Incl "%tei.anchor; ">
<!ENTITY % m.loc  "%tei.ptr;">
<!ENTITY % m.phrase     "%m.data; | %m.date; | %m.edit; |
 %tei.handShift; | %m.hqphrase; | %m.loc;">
<!ENTITY % tei.phrase  "#PCDATA | %m.phrase; | %m.Incl;">
<!ENTITY % tei.phrase.seq "(%tei.phrase;)*">
<!ENTITY % specialPara  "(#PCDATA | %m.phrase;)*">
Etc.

<!-- Define the TEI elements -->
<!ELEMENT %tei.abbr; %specialPara;>
<!ELEMENT %tei.add; %specialPara;>
<!ELEMENT %tei.addName; %tei.phrase.seq;>
<!ELEMENT %tei.analytic; (%tei.teiAuthor; | %tei.editor; |
 %tei.respStmt; | %tei.teiTitle; | %m.Incl;)*>
Etc.

<! -- Define the specialization hierarchy -->
<!ATTLIST %tei.abbr; class  CDATA "+ topic/ph tei-d/abbr">
<!ATTLIST %tei.add; class  CDATA "+ topic/ph tei-d/add">
<!ATTLIST %tei.addName;  class  CDATA "+ topic/ph tei-d/addName">
<!ATTLIST %tei.analytic; class  CDATA "+ topic/ph tei-d/analytic">
Etc.
```

Figure 10. Element Declaration File for a TEI Domain Specialization

Here is how these phrase-based specializations would be summarized in the corresponding Entity Declaration File:

```
<!-- Define the TEI domain specialization entities -->
<!ENTITY % tei-d-ph "abbr | add | addname | analytic | …">
Etc.

<!-- Define the domain identification entity -->
<!ENTITY % tei-d-att "(topic tei-d)">
```

Figure 11. Entity Declaration File for a TEI Domain Specialization

Finally, the figure below shows how the resulting TEI domain specialization could be applied to the JSP shell DTD.

```
<!-- Declare the entities for the domains -->
<!ENTITY % tei-d-dec PUBLIC "-//JSP//ENTITIES TEI Domain//EN"
 "tei-domain.ent">
%tei-d-dec;

<!--Redefine the entities for the base content elements to
 add the specialized content elements from the domains
-->
<!ENTITY % ph "ph | %tei-d-ph;">

<!-- Define the domains attribute of the topic elements to
 declare the domains represented in the document -->
<!ATTLIST journEntry domains CDATA "&tei-d-att;">
<!ATTLIST annotation domains CDATA "&tei-d-att;">
<!ATTLIST jeTrans domains CDATA "&tei-d-att;">

<!--Embed DITA topic to get generic elements -->
<!ENTITY % topic-type PUBLIC "-//IBM//ELEMENTS DITA Topic//EN" "topic.mod">
%topic-type;

<!--Embed specialized JSP Topic to get common JSP elements -->
<!ENTITY % jsp-topic-type PUBLIC "-//JSP//ELEMENTS JSP Topic//EN"
 "jsp-topic.mod">
  %jsp-topic-type;

<!--Embed Journal Entry, Annotation and Transcription topic types-->
<!ENTITY % journEntry-typemod PUBLIC "-//JSP//ELEMENTS JournalEntry//EN"
 "journEntry.mod">
  %journEntry-typemod;
<!ENTITY % annotation-typemod PUBLIC "-//JSP//ELEMENTS Annotation//EN"
 "annotation.mod">
  %annotation-typemod;
<!ENTITY % jeTrans-typemod PUBLIC "-//JSP//ELEMENTS Transcription//EN"
 "jeTrans.mod">
  %jeTrans-typemod;

<!-- Include TEI vocabulary definitions -->
<!ENTITY % tei-d-def PUBLIC "-//JSP//ELEMENTS TEI Domain//EN"
 "tei-domain.mod">
%tei-d-def;
```

Figure 12. Shell DTD for a TEI Domain Specialization

At face value, all this looks good. In practice, however, TEI domain specialization didn't actually work. There were several reasons for this, all of which were driven by the fact that we needed to be able to produce valid TEI – not just use TEI elements within our DITA markup.

Let's take a simple example that shows why this is true. As part of the standard JSP header defined in JSP Topic (see Figure 5), we have two specialized elements called "persons" and "places". The first contains a series of TEI "persName" elements; the second is made up of a combination of TEI "placeName" and "geogName" elements. These content models are repeated below:

```
<!ELEMENT %persons;        (%tei.persName;)* >
<!ELEMENT %places;         (%tei.placeName; | %tei.geogName;)*  >
```

If we perform a TEI domain specialization, the “persName”, “placeName” and “geogName” elements would be defined *inside* the domain specialization. Based on DITA’s rules, this means that they *cannot* be referred to directly in the topic specializations. DITA requires that domain specializations are completely independent from topic specializations, so that topic and domain specializations can be freely mixed and matched.

As specializations of the DITA paragraph (“p”), the “person” and “places” elements can have more restricted content models, but they cannot assume any particular domain specialization. With this constraint, the best we can do is to restrict both of these to a simple series of phrase elements, as follows:

```
<!ELEMENT %persons;        (%ph;)* >
<!ELEMENT %places;         (%ph;)* >
```

From the Entity Declaration File (Figure 11) and Shell DTD (Figure 12), the domain specialization would be applied as follows:

```
<!ENTITY % tei-d-ph "abbr | add | addname | analytic | …">
<!ENTITY % ph "ph | %tei-d-ph;">
```

When this set of entities is resolved, each of the elements is specialized in the same way, containing *all* of the TEI elements, plus the original “ph” element which is legal in DITA but not in TEI. The end result is that we have TEI elements at the low level, but no way to control the proper subsets of TEI to map to elements at a higher level.

```
<!ELEMENT %persons;       (ph | abbr | add | addname | analytic | …)* >
<!ELEMENT %places;        (ph | abbr | add | addname | analytic | …)* >
```

It’s tempting to try to solve this problem by moving the distinctions up a level. For example, what if we invented new specializations at the JSP Topic level that could distinguish between person, place and geographical names? Then, without violating DITA rules, we could make our “persons” and “places” content models more specific:

```
<!ELEMENT %persons;        (%person;)* >
<!ELEMENT %places;         (%place; | %geography;)* >
```

The problem, however, is that we still can’t assume a particular TEI tag to be mapped to these new specialized elements. Otherwise we violate DITA modularity constraints. The full picture looks like this:

```
<!ELEMENT %persons;        (%person;)* >
<!ELEMENT %places;         (%place; | %geography;)* >
<!ELEMENT %person;         (%ph;)* >
<!ELEMENT %place;          (%ph;)* >
<!ELEMENT %geography;      (%ph;)* >
```

So we’re left with the same issue: we still are unable to make the proper distinctions between these elements. The fact of the matter is that no matter how many times you move the elements “up a level”, you end up with the same problem – until you have essentially performed the entire domain specialization up at the topic level. This is what actually happened in the case of JSP – we ended up dropping the idea of a TEI domain specialization and instead defined all the TEI elements and content models within the overall JSP Topic. TEI elements could then be freely referred to in the Journal Entry, Transcription and Annotation topic specializations – without violating any DITA constraints.

There was one other thing that occurred to us. What if we could create the domain specialization based on elements defined in our topic specializations? In the current example, that would create a specialization hierarchy something like this:

```
<! -- Define the specialization hierarchy -->
<!ATTLIST %tei.persName; class CDATA "+ topic/ph jsp.topic/person
 tei-d/persName">
<!ATTLIST %tei.placeName; class CDATA "+ topic/ph jsp.topic/place
 tei-d/placeName">
<!ATTLIST %tei.geogName; class CDATA "+ topic/ph jsp.topic/geography
 tei-d/geogName">
```

With this technique, the distinctions can be made in the topic specializations, and then directly mapped to the TEI elements in the domain specialization. Unfortunately, however, this is also *not* allowed in DITA – domain specializations can only be done at the base DITA Topic level. The reason is the same as for the other constraints: if this kind of dependency were allowed, then domain specializations could not be freely applied to any valid topic specialization.

3.7. TEI-Specific Attributes

DITA's rules require that specializations use an equivalent set or subset of the attributes defined for the base element. For TEI validity, this restriction could not be met. However, since the TEI-specific attributes could be dropped when generalizing back to DITA Topic, we could still meet the requirement of being "one transform way" from both valid TEI and valid DITA.

4. Conclusion

The Joseph Smith Papers project has successfully used DITA-based XML for scholarly text transcription and encoding, an area that has previously been the exclusive realm of TEI. This is particularly interesting because JSP has taken DITA far out of its "comfort zone", and therefore serves as a notable proof point of DITA's extensibility.

In general, DITA proved workable for the JSP application, and it was reasonable to think of the JSP content in terms of topics (e.g., journal entries) and sub-topics (e.g., transcriptions and annotations). Furthermore, although many of the standard DITA elements were not used, the DITA Topic structure was sufficiently general to support required JSP topic specializations.

Domain specialization, however, was a different story. Although formal TEI domain specialization was attempted, JSP reverted to defining the TEI elements in the overall JSP Topic – treating TEI as a *topic* rather than *domain* specialization. This was necessary for two reasons. First, we had the requirement to create truly valid TEI markup, not just to use TEI elements as an extension to DITA markup. This meant that we had to connect elements in the TEI domain specialization directly to higher-level constructs defined in topic specializations. Second, DITA requires that domain specializations are completely *independent* from topic specializations, so that topic and domain specializations can be freely mixed and matched. This meant that it was actually impossible to establish the connections we needed. The only choice was to treat TEI as part of the topic specializations.

In the end, while TEI domain specialization didn't work, the resulting solution was both effective and successful. And, true to its requirements, it was no more than "one transform away" from both valid TEI and valid DITA markup. The Joseph Smith Papers project now has an XML-based solution that can support both printed books and a topic-based research database, while meeting rigorous standards for scholarly markup and editorial / verification processes.

Biography

Eric **Severson**
Chief Technology Officer
Flatirons Solutions Corporation [http://www.flatironssolutions.com/]
2555 55th Street Suite 100D
Boulder
Colorado
80301
United States of America

An internationally recognized XML pioneer and content management industry expert, Mr. Severson has over twenty years of experience in the technology field, ranging from hands-on product development and consulting to senior management roles in engineering and marketing.

Mr. Severson worked directly on the design of the JSP solution, and is currently Chief Technology Officer and a co-founder of Flatirons Solutions Corporation, a consulting and systems integration firm specializing in content management and XML-based publishing. Prior to joining Flatirons, Mr. Severson was an Executive Consultant for IBM Global Services, where he focused on content management and XML, and was a principal developer of IBM's XML certification test. He also served as Vice President and Chief Strategist for Interleaf, Inc., an XML-based content management company, and was co-founder, CTO and principal developer for Avalanche (a startup SGML/XML software development company).

Mr. Severson is a past President of OASIS, and was a member of the advisory boards of the AIIM Document Management Alliance and the Xerox Executive Roundtable for Document Technology. Mr. Severson holds an undergraduate degree from the University of Wisconsin, and completed Ph.D. coursework in Computer Science at the University of Colorado. He also holds an IBM consultant certification, an IBM XML developer certification, a CPA certificate, and a Certificate in Data Processing (CDP).