

# DME: Documentation Management Environment for Software Product Lines – Tool Demo Proposal

Ha Duy Trung and Stan Jarzabek  
Department of Computer Science, School of Computing  
National University of Singapore  
+65 65162863  
stan@comp.nus.edu.sg

Tool demo at: <http://www.comp.nus.edu.sg/~stan/DME-video.html>

## ABSTRACT

Similar documents arise in software and business situations. Examples are user guides for different versions of a software product, contracts between vendors and clients, and legal documents. The usual practice is to capture similarities in templates that must be copied and manually customized to a new context – often a slow, tedious, and error-prone process. Document Management Environment (DME) automates routine tasks involved in creating and updating documents. DME provides functions to create templates and to generate custom documents from them. Any arbitrary document part can be designated as template's variation point, and the same inter-dependent customizations occurring at different variation points can be streamlined. We develop DME to automate generation of documentation of multiple software products (e.g., in reuse via Software Product Line approach), but also plan to explore its potentials in management of general business documents.

## Categories and Subject Descriptors

D.2.9 [Software Engineering]: Management - Productivity;  
D.2.13 [Software Engineering]: Reusable software – Domain engineering

## Keywords

Documentation, Software Product Lines, Reuse, Document Generation, Templates

## 1. INTRODUCTION

Document Management Environment (DME) facilitates and automates reuse of documents written in WORD. DME is useful in Software Product Line (SPL) engineering [1], where we manage a family of similar software products from a common set of reusable SPL core assets such as SPL architecture shared by products, source code components, documentation, test cases, etc.

Creation and evolution of documentation for SPL members involves much repetitive work. We can benefit from reuse of software documentation just as much as we benefit from reuse of SPL architecture and code components.

All the SPL members are similar, but each one also differs from others in client-specific features. The impact of features shows as many changes that must be applied through product code and documentation. SPL core assets help developers build

a custom product. They play the role of templates that are reused after suitable adaptations to derive custom products.

For example, User Guides for different SPL members are similar, but also different. The differences in User Guides reflect product-specific variant features implemented into some custom products, but absent from others.

The lifecycle of DME-supported documentation processing (Figure 1) fits into the usual SPL lifecycle, with two major phases, namely *Domain Engineering* and *Product Development*. First, DME helps Domain Engineers create parameterized, adaptable templates for each family of similar documents such as product User Guides. During Product Development, DME automates generation and evolution of custom User Guides from their respective templates.

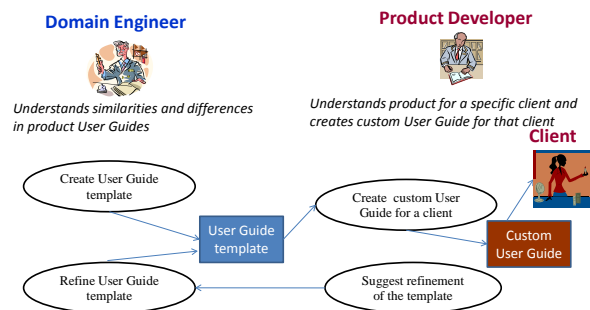


Figure 1. Managing User Guides with DME

## 2. EXAMPLE

Project Collaboration Environment (PCE) is a web portal supporting software development teams in project planning and execution. PCE facilitates sharing of project information within and across teams. In particular, PCE allows users to create and maintain records of domain entities such as projects, staff, tasks and relationships among them (e.g., tasks assigned to staff).

Suppose we developed a PCE for the mass market. There will be many PCE versions in use. All such PCEs will be similar but will also differ one from another depending on the team size, development style, and other project- and team-specific details.

A User Guide for PCE describes how to Create, Edit, Delete Staff record, Project, Task, records etc. For that, User Guide contains Staff-Section, Project-Section, Task-Section and in each section descriptions of relevant operations (Figure 2). The actual lists of domain entities (Staff, Project, or Task), their

respective operations and the details of operation description may differ across PCEs.

#### PCE for Agile Development: User Guide

Project Collaboration Environment is an integrated environment that supports project teams in software development. PCE stores staff, project data, facilitates project progress monitoring, communication in the team, etc. The following sections provide detail description of operations for domain entities supported by PCE.

##### Staff Section

This section describes operations to manage Staff information in a Project Collaboration Environment. A Staff profile contains the following information:

- Name: Full name of Staff
- ...
- Create a new Staff**  
Create operation allows users to add new staff data to PCE. Once added, this new information can be manipulated by using Edit, Delete or Display operations.
- Edit Staff information**  
Edit operation allows users to edit staff data. Once edited, this new information can be manipulated again by using Edit, Delete or Display operations.
- Delete Staff record**
- ...
- Display Staff information**
- ...
- Sort Staff**
- ...
- Print an individual Staff**
- ...

##### Project Section

- ...
- Create a new Project**
- ...
- Edit Project information**
- ...
- Link a Project with another Project**
- ...
- Delete a Project link**
- ...
- Delete Project record**
- ...
- Display Project information**
- ...
- Sort Projects**
- ...

##### Task Section

Figure 2. User Guide for PCE

### 3. TYPES OF DOCUMENT VARIATIONS

We analyzed families of similar documents such as PCE User Guides to understand how they differ one from another. Generalizing observations, we identified the following types of document variations:

*Comment:* Below, a “fragment” means any arbitrarily selected segment of contiguous text in a document such as word, sentence, paragraph, section, or any part of them.

#### VT 1. Parametric variation

Parameters are placeholders that appear throughout documents. Parameters have the same values within a given document, but may have different values across documents.

Examples: date, module name, syntactic variations, for example spelling (English or US), whether or not we put “;” before “and”, etc.

#### VT 2. Formatting variation

A fragment that can be formatted using different font type or color in different documents.

#### VT 3. Extra fragment

A fragment that appears in small number of documents. Extra fragment may recur in many places in each of such documents. Furthermore, each occurrence of such fragment may differ in arbitrary ways from other occurrences (in the same or in different document versions).

#### VT 4. “Almost common” fragment

A fragment that appears in most (but not all) of the documents. “Almost common” fragment may recur in many places in each of such documents. Furthermore, each occurrence of such fragment may differ in arbitrary ways

from other occurrences (in the same or in different document versions).

#### VT 5. Selection

A variation point in documents where we wish to select for inclusion one or more among pre-defined fragments.

#### VT 6. Repeated section

A section that recurs a number of times in a given document. Such section may occur different number of times in different documents. Furthermore, each occurrence of such section may differ in arbitrary ways from other occurrences (in the same or in different document versions).

Examples n in PCE User Guide: Subsections describing the same operation (Create, Delete) for various domain entities (Staff, Project). Also sections for Staff, Project, etc. (Figure 2)

#### VT 7. Linked document

Large documents can be decomposed to parts that are stored in separate files. A link can be placed in a document to show how documents should be composed together. Document composition rules may be affected by feature selection.

### 4. HOW DME WORKS

Playing role of a Domain Engineer, we create templates. We start with a sample document, say a User Guide of a typical PCE. We designate various document fragments – words, sentences, paragraphs, sections – as *variation points* that can differ from the sample document. We can also control each variation point’s characteristics, such as its format and repeatability. DME converts this annotated document into a template that Product Developer uses to generate User Guides for other PCEs.

Template variation points are of the seven variation types discussed in the previous section. DME provides functions to create template (for Domain Engineer) and to create custom document by customizing a template (for Product Developer). These functions are made available to users via WORD menu buttons that extend WORD with DME functionality (shown on the left-hand-side of the WORD toolbar in Figure 3).

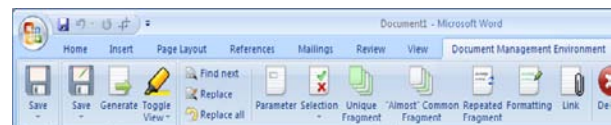


Figure 3. DME menu as an extension of WORD toolbar

#### 4.1. Creating document templates

We use PCE User Guide as an example. The task of Domain Engineer is to identify common and variant parts in User Guides. Common parts become “frozen” in a template, while variant parts are turned into variation points using DME buttons. A good start is to run WORD’s *Compare* function on existing User Guides. Differences highlighted by WORD are candidates for variation points in a template. Figure 4 shows common and variant parts in Staff and Project sections.

Suppose we observe that sections for Staff, Project, and Task are similar to each other. We could choose to create a Section-Template first. The advantage of creating this template is that it can be reused within User Guide (to create Staff, Project or Task sections), as well as across User Guides for different PCEs.

To convert Staff Section into a Section-Template, Domain Engineer positions cursor on fragments highlighted by WORD as different and clicks on suitable DME button to turn variant text into a variation point of a required type.

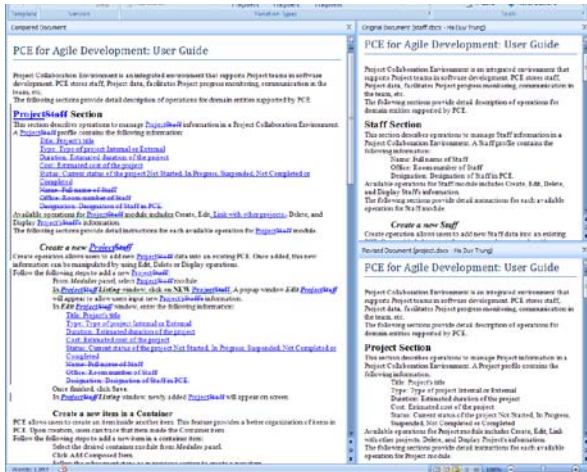


Figure 4. Staff and Project Modules compared



Figure 5. Section-Template created with DME

Domain Engineer can extend Section-Template with more variation points to address variant text found in yet other sections for other domain entities.

Figure 5 shows a Section-Template with variation points highlighted by DME in different colors).

Applying a similar analysis to other parts of User Guide, Domain Engineer completes User-Guide-Template.

## 4.2. Generating custom documents

To create a User Guide for specific PCE, Product Developer works with User-Guide-Template. For that she positions a cursor on template variation points and defines custom values. DME propagates values to all the other relevant variation points automatically.

To simplify template customization, all the variation points in a template are given default values. Furthermore, Product Developer can start with an existing User Guide that is most similar to the one she wants to create, and ask DME to impose a template view on it. This will show all the variation points, with default values from the selected existing User Guide. Now, Product Developer must provide only customizations that are unique in the User Guide to be created.

## 5. CONCLUSIONS

DME based on seven variation types (Section 3) is practical. We have not encountered differences among documents that could not be easily turned into a template. DME not only improves productivity and reduces errors, but it also keeps track of each document's unique details together with a reference library of standard templates. While we primarily target documentation of SPL, management of general business documents could also benefit from DME.

Still, we consider current DME too low level. Its usability can be enhanced by allowing a Domain Engineer to map features to template variation points. Then, during Product Development, having selected required features much of the customization data could be generated and fed to DME, rather than being manually entered. Domain-specific interfaces and wizards can further raise the abstraction level, and automate creation of custom documents.

DME is in the late stage of prototyping. We applied it in lab studies, but have not applied it yet in real world projects.

Internally, DME uses variability technique of XVCL [2][4] to build templates and then to customize templates in document generation. As XVCL is used to manage variability in code-related SPL core assets (SPL architecture and code components), we envision that future DME together with XVCL Workbench will provide integrated support for managing variability in code and its documentation.

## REFERENCES

- [1] Clements, P. and Northrop, L. *Software Product Lines: Practices and Patterns*, Addison-Wesley, 2002
- [2] Jarzabek, S. *Effective Software Maintenance and Evolution: Reused-based Approach*, CRC Press Taylor and Francis, 2007
- [3] Rabiser, R., et al "A Flexible Approach for Generating Product-Specific Documents in Product Lines," *Proc. Int. Software Product Line Conf, SPLC'10*, Jeju, S. Korea, Sept.2010, pp. 47-61
- [4] XVCL, XML-based Variant Configuration Language, a reuse method and tool <http://xvcl.comp.nus.edu.sg>