# Abstract -
# Developing an Automated Documentation Environment

PATRICIA MESSIER ADAMS
SCIENCE APPLICATIONS INTERNATIONAL CORPORATION
ORLANDO, FLORIDA, USA

A vast amount of data is generated in the production of modern software systems such as training systems, requiring well-monitored yet heretofore time-consuming maintenance and configuration management. The paper's premise is that development and management of software documentation for training systems can now benefit from the utilization of recently innovated automated documentation tools and CASE tools. Tools now at the disposal of the industry have the ability to automate document generation and management for integration within a distributed environment. The time and thereby cost savings associated with such automation is incalculable.

An example is demonstrated using the development of Software Development Folders (SDFs) for the Joint Simulation System (JSIMS) Enterprise. The SDFs, considered prototypical for Build 0, were created and refined using an Enterprise-selected document automation tool, in conjunction with a modeling CASE tool. The document automation tool facilitated the insertion of script commands to document various types of data associated with the object-oriented Categories and Classes. The tools' capability to so "link" to diverse data in a distributed environment is further augmented by utilizing hyperlink capability and Internet Web directories. Such ability to automatically gather distributed information provides a training audience with "at-a-glance" views of specific documentation. Pertinent data can be accessed in real time from just one source.

Using an automated system is advantageous simply because the data can be updated on a requisite basis without the obvious risk of missing some crucial piece of the puzzle. Automation allows training participants to more fully concentrate on their lesson activities without the necessity of opening many sources to ensure they are in possession of the latest data. Of greatest consequence, however, is not the automation itself, but the potential widespread usage of available technology to enable such a networked system.

## Biographical Sketch -

Ms. Adams is highly specialized in software documentation, with particular expertise in those produced by MIL-STDs 498, 2167A and 1644B. Extensive experience in the publications field has afforded her numerous topics with which to work, with emphasis on software development manuals and guides for F-18 simulated aircraft maintenance trainers. In addition, she developed curriculum materials to facilitate training in C-7 and C-11 aircraft carrier catapults. She is presently a member of the Joint Simulation System (JSIMS) Mission Space Objects (MSO) Integrated Product Team (IPT) based in Orlando, Florida, USA. The Software Development Folders generated and maintained by Ms. Adams for MS0 Categories and Object Classes were created with scripts she wrote using tools enabling the automated generation and update of the folders. She is a member of the JSIMS Enterprise Process Group (JEPG) and the SAIC Software Engineering Process Group (SEPG).

# Developing an Automated Documentation Environment

**Patricia Messier Adams**
**Science applications international corporation**
**Orlando, Florida, USA**

## OVERVIEW

Documentation for training software systems has heretofore been burdened with vast amounts of data requiring well-monitored yet time-consuming maintenance and configuration management. Development and management of software documentation for training systems can now benefit from the utilization of recently devised automated documentation tools and CASE tools. In recent months, the facilitation of such management through automation has been examined with regard to such newly innovated document automation tools as well as the utilization of Internet Web sites for automated data updates within a distributed environment. As an example, Software Development Folders (SDFs) for the Joint Simulation System (JSIMS) Build 0 were developed using an automated documentation tool created for a word processing application in conjunction with a modeling CASE tool. The modeling CASE tool creates a "Model" containing Categories and Classes and their respective documentation and appropriate diagrams. The automation tool extracts such pertinent data from the Model and creates a document comprised of "links" to the actual work products amid a textual environment. Any automated features not readily provided for by the automation tool and not residing in the Model are generated using hyperlink capability and Internet Web directories.

## The Case for Automation

The greatest advantage to using an automated system is, simply, that the data can be updated and managed on a requisite basis without the obvious risk of missing some piece of the puzzle. Automation allows training participants to more fully concentrate on their lesson activities without the necessity of opening many sources to ensure that they are in possession of the latest data. The training audience can access a full set of continually updated information on a given Class or Category, for example, a Sensor Category. In an automated environment, the Model is modified by the developer wherever it resides. In turn, the documentation manager has merely to regenerate the appropriate SDF and the updated data will be automatically included. Such ability to automatically gather distributed information provides a training audience with "at-a-glance" views of specific documentation. Pertinent data can be accessed in real time from just one source.

## Contents

The SDFs, as a case in point, exist as electronic media, albeit in a word processing file. In this case, the majority of the data is available on-line only, since it can be accessed solely through "links" to the actual work products, such as the model itself, hyperlinks to Web pages (or files), source code files, test cases, the System Problem Report (SPR) system, etc. The data contained therein can exist as a composite of each of the Categories and their decomposed levels at given points in time to enable the preservation of historical data. The SDF manager and, later, the training manager have only to make a copy before regenerating.

## Maintenance

SDFs are captured at end-of-build points and can thereafter repose in a Web-based document management system that allows them to be saved as documents that can fulfill contractual requirements. Formatting anomalies may arise as a result of the conversion and need to be corrected. Additionally, the management system allows restricted access to the documents, so that only persons such as QA, the SDF manager, the IMS manager, and a designated customer representative can view them during their

inception. Such restricted access is necessary given the proprietary and/or classified nature of the source code and other products. Thereafter the training audience would have access to the SDFs to flesh out their understanding of the training process.

SDFs can also be maintained, throughout the Build cycle, in the same Web-based document management system, although the SDF manager, in this case, must be also given "write" privileges to enable incremental document regeneration. The unique nature of SDFs necessitates explicit project-specific provisos which outline how validation will proceed and at what intervals the SDFs are to be configured.

## CONSTRUCTING THE DOCUMENTS

Documentation automation tools often provide various pre-smithed templates which may give the documentation manager an idea of how to proceed. In this case, none of the templates were utilized as originally devised, requiring significantly more tailored scripts. Although such tailoring of templates requires a great amount of toolsmithing, in the end the automation process makes the effort worthwhile. A template assures that documentation can be standardized across training systems, projects, teams, et al. The Software Development Plan (SDP) could be the definitive authority on document organization; any documentation required by a project follows the plan explicitly.

Automation tools insert, at the manager's prompting, a series of commands which "instruct" the word processing document to traverse to the selected Model and extract specified blocks of data. The commands are not "links" per se, since they do not automatically update whenever the Model is updated. Updates occur only when the document is generated or regenerated.

### Adding to the Template

Other kinds of data might be necessary to include in the documentation. For example, SDFs contain additional requisite data not necessarily residing in the Model, e.g. source code, test cases, SPRs, review minutes, etc. In that case, the word processing application enables the insertion of hyperlinks that can connect to this data. Updates to such sites would necessitate privileges being to those who need to make the changes. Ideally, the sites will be Web-based to facilitate communication between the office PC (where the automated documents reside) and the software development environment (where source code and test cases would reside). Otherwise an inordinate amount of time copying files will be required, defeating the purpose of an automated environment.

Links can also be added which point to documentation required for specific exercises associated with a given Class or Category. The greatest advantage is that, during training, the student can access all data required via one single file, a file essentially created and managed by the push of a button.

### Generating the Documents

It is suggested that a minimally populated Model be utilized to test the scripts, one that contains, at the very least, the features that need to be extracted. Upon generation, a file separate from the template is created, called <filename>Gen.doc. The template remains as originally created and can be used limitlessly. Thus, when a Class template is created, it can be utilized for each Class residing in the Model. Likewise, when a Category template is created, it can be utilized for each Category in the Model.

When the test is concluded with no errors, then the template can be generated utilizing the definitive Model.

Similarly, other types of document templates can be utilized cross-product teams with their respective Models. It is simply a matter of entering into the command what Model needs to be used. Again, the templates need to be tested with mini-models prior to actual generation with the team's own model.

## MAINTAINING THE DOCUMENTS

The intervals at which regeneration needs to be enacted are project-specific decisions. In JSIMS' case, Build 0 was a prototype, and SDFs were generated as an exercise in developing the automated environment. Ideally, the SDFs should be captured as preliminaries when Classes and Categories have been defined, and finalized after integration, when the code should be considered a final product. Any changes made thereafter are documented through SPRs in the DDTS system, and automatically updated in the SDFs through the hot link to that Web-based system. At this point the documentation is controlled by the configuration management system. The configuration manager would assume the primary responsibility of managing the documentation and ensuring that it is made readily available to the pertinent training audience. The time savings involved with automating the previously labor-intensive update process increases productivity, making the updates quick and less likely to contain erroneous data. There are cost and time savings implications of real-time updates.

### Identifying Milestones

Project-discretionary milestones can be identified to capture "snapshots" of the documentation. These milestones could be design reviews, peer reviews, and/or management reviews. Similarly, the project may define coordination or synchronization points at which certain software products are expected to be completed and captured in the documentation. If a project is designed as a series of builds with phases contained therein, then identification is required of what phases would be yielding what products and whether those products require documentation updates. In each case, the designated documentation manager should be included in the review, coordination point, or phase identification process to expedite the updates.

### Configuration Management

Typically, documentation is updated when products are configured (controlled) by configuration management and thereby archived in the CM library. When data will be regenerated and captured is determined by the project design and is a management decision; some do not configure products until certain phases and/or coordination points are completed, yet documentation updates are crucial. A dynamic project that generates products and product updates at accelerated intervals would require the documentation updates to occur at regular intervals as well without waiting for review, coordination points, or phase completion. The "Preliminary/Final" recommendation fits nicely into this scenario: SDFs established, initially generated with a sketchily populated Model, configured as a "Preliminary." Thereafter, updates occur as the development process unfolds, until after integration, when the "Finals" will contain a fully populated document which are dynamically current for the training system.

### Security and Storage

The archival of documentation differs according to the sensitivity of the data contained therein. SDFs, for example, contain source code which, while not necessarily classified, still are of a proprietary nature that would require security provisions. Storing the documents on a Web-based drive, preferably an FTP (File Transfer Protocol) site, is ideal: visible links need not exist and privileges given only to those who need to view/manage them.

## SUMMATION

Investigating which automated documentation tools exist and which would operate at optimum capacity is a matter of examining what the initial software development environment will contain, i.e., what platform, what software applications are presently at disposal, and what is stipulated by the contract. On occasion, one tool is preferred over another but is constrained by the operating system. The selection of a tool that

can function on multiple platforms is always preferable. That way, if the software environment changes, the tool is flexible enough to change with it.

For developing an automated documentation environment, the primary focus should be on the automated nature of the data management and to maintain that integrity throughout the development process, constantly ensuring that all avenues are utilized and explored. Of greatest consequence, however, is not the automation itself, but the potential widespread usage of available technology to enable efficient management of a networked training system.