AUTOMATED PSA PLATFORM CONVERSION

Carroll Trull[a*], Go Tanaka[b], Yuji Komori[c]
[a] Engineering Planning and Management (EPM): Raleigh, NC, USA, cxt@epm-inc.com
[b] Toshiba Energy Systems & Solutions Corporation: Yokohama, Japan, go1.tanaka@toshiba.co.jp
[c] Toshiba Energy Systems & Solutions Corporation: Yokohama, Japan, yuji1.komori@toshiba.co.jp

Abstract: A case study of converting a utility Probabilistic Safety/Risk Analysis (PSA) model from one platform to another will be discussed. The model consists of fault trees, event trees, analysis cases, data, etc. A significant focus of this paper is on efforts to automate the processes using Visual Basic for Applications (VBA), which is not overly sophisticated, but is an intuitive and versatile programming language. The source PSA code is essentially a "large event tree, small fault tree" model, and the target code is EPRI's CAFTA. The process involved conversion of Level 1 and Level 2 fault trees and event trees and associated data, while developing algorithms to facilitate macros to decompose the source PSA model's structure and translate that into a structure that could be recognized and imported to CAFTA. Manual efforts were needed to finalize the top logic structure and bridge trees for Level 1 and Level 2. However, the use of macros using VBA greatly simplified the troubleshooting efforts typical of model conversion. The intent of this paper is to illustrate the utility of VBA macros in an example case study, which happens to be model conversion. This project was a joint effort between Japan and the U.S.

Keywords: PRA, PSA, fault tree, model conversion automation.

1. PROJECT GOALS AND BACKGROUND

The main objective of this project was to convert a PSA model to CAFTA in as efficient a way as possible, as a joint venture between a U.S. vendor and Japanese vendor. In meeting the efficiency criterion, several aspects of the conversion were considered for simple macro development. In particular, it was immediately recognized that the source PSA code was compatible with Microsoft Excel, which facilitates the use of Visual Basic (VBA), which is embedded with Microsoft Excel. CAFTA also has import/export options that are compatible with Microsoft software, which means that many tasks could be reduced to basic operations, such as copy/paste, with proper validation.

The intent of this paper is to illustrate the model conversion as an example case for using macros to greatly simplify tasks associated with PSA development and maintenance. The author has experience with PSA model conversion for various other platforms (including RiskSpectrum to CAFTA), providing necessary expertise to critique the benefits of including automation in this case, as well as development of macros for other purposes, such as a tool for rendering thermal-hydraulic output, or simple data organization tools.

The set of tools developed for this project were designed to:

1. Extract the fault tree structure from the source PSA model files and recreate it in a format that is readable by CAFTA,
2. Extract and organize data from the underlying source PSA model database files for ease of import to CAFTA,
3. Create links between the fault trees to match support system interfaces in the model, and
4. Provide a simple interface for operation.

2. MODEL CONVERSION

2.1. Event Trees

The event trees from the source PSA model were converted to the CAFTA .ETA format. The event tree definitions and branches were relatively few and defined simply, such that this process was largely manual, with the exception of some model integration tasks and verification, discussed later, in Sections 2.5 and 2.6. Otherwise, event tree logic was developed in CAFTA manually based on the input data because it was straightforward and brief.

2.2. Fault Trees

Each system fault tree required for the CDF and LERF logic was converted from the source PSA model format to the CAFTA .CAF format. The logic structure for the fault trees was based on text block input that defined the structure and required inputs with specific arguments. This structure and related arguments were based on a specific set of formatting instructions, such as a string of numbers and characters, which could be exploited by developing algorithms that map the logic structure to a new format, which could be recognized by CAFTA. The recognized fault tree structure could be imported to CAFTA via the Input Logic module (CTRL-I in CAFTA) or database structure, which could be edited in Microsoft Access.

As an example of the algorithms needed to create these macros, an event tree might be defined by a text block with a series of arguments represented by numbers and characters, separated by a delimiter (e.g., a comma) or specific vertical or horizontal spacing, based on a particular structure. An algorithm is needed to decompose this structure into elements or sub-elements of the structure that CAFTA recognizes. For example, a macro might find the nth argument of the third line of a block, and identify it as the failure mode of a particular type of basic event. The algorithm is built on that decomposition and could become quite complicated, such that the use of macros can significantly reduce errors in tracking and applying this mapping.

To illustrate this with a simplified example, the block of values shown in Figure 1 represents a fault tree in which:

- Column 1 represents an event type (1 – gate, 2 – basic event, 3 – transfer gate).
- Column 2 represents a gate type (X – AND, Y – OR).
- Column 3 represents the number of inputs.
- Column 4 represents subsequent rows of the block that represent the inputs.
- Column 5 represents some event text.
- Column 6 represents an external file that represents a data source for some additional input.
- Column 7 represents a basic event failure mode.
- Etc.

1	X	3	234	Gate 1 Text	File1.xyz	-	…
1	Y	4	5678	Gate 2 Text	File1.xyz	-	…
2	-	-	-	Component 1	File2.xyz	R	…
2	-	-	-	Component 2	File2.xyz	R	…
2	-	-	-	Component 3	File2.xyz	S	…
2	-	-	-	Component 4	File2.xyz	S	…
2	-	-	-	Component 4	File2.xyz	T	…
3	-	-	-	Transfer Text	File2.xyz	-	…
…							

Figure 1. Simplified Example of Input Block

For the actual project, the blocks were not necessarily structured in well-defined rows and columns, nor was the cell data necessarily as well-defined or consistent. However, this serves to illustrate the concept. Translating the above definitions into a fault tree results in Figure 2.

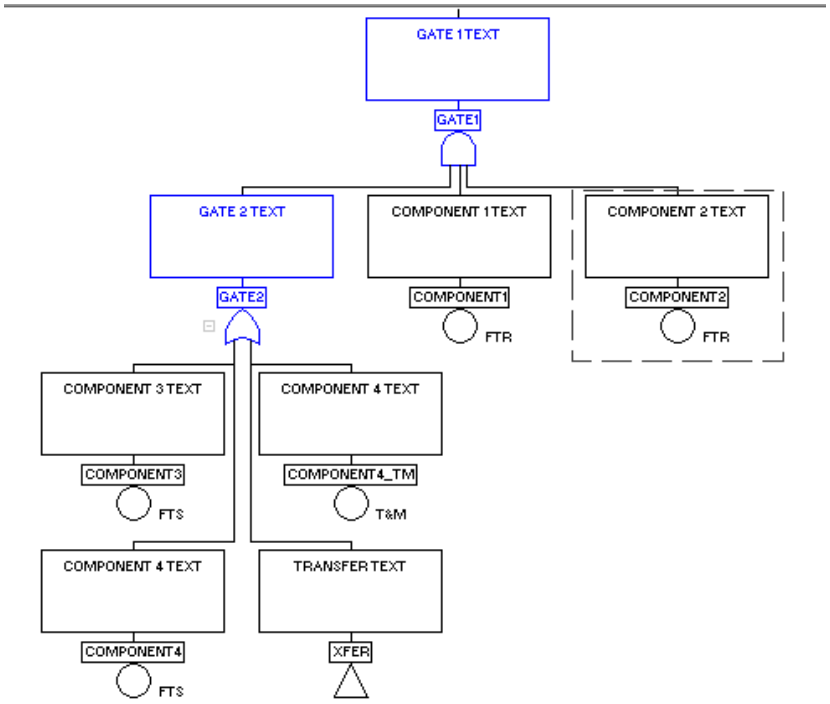

Figure 2. Simplified Example Resulting Fault Tree

The set of macros developed to convert the fault tree logic was developed in a macro-enabled spreadsheet (.xlsm). These macros performed the following types of functions, generally.

- Define the folder location for the fault tree files and call additional macros to operate on each file in the folder.
- Open each fault tree file and organize the data into a format that is recognizable by CAFTA. To do this, the filename and location are passed from the file retrieval macro, and the file contents are copied to tabs within the spreadsheet. It then deconstructs the fault tree format and recreates it on additional tabs in the new format. At this point, every object that is included in the structure would be treated as a "gate," for example, and the gate names would use the filename for the fault tree and the same index used in the source PSA structure.
- Identify gate types, such as OR, AND, or COM.
- Generate tabs in the spreadsheet to organize gate and basic event data.
- Create transfer gate links in the fault tree logic based on information contained within the source PSA model files, to replace converted gate names for the respective transfers. If a linked fault tree is not found in the file (because it has not been loaded), the macro would record it as missing.
- Create internal transfer links. That is, within each fault tree file, there could be several blocks of data describing the fault tree structure. These are considered "internally linked," and are identified and updated in the CAFTA structure via macro.

- Identify "gates" that should be basic events and replace the gate names with the corresponding basic event names on the corresponding tabs.
- Rename basic events to follow new type codes that are better suited for the CAFTA type code structure, and ensure the CAFTA character limit is not exceeded by removing extraneous characters when necessary.
- Pull all of the finalized logic from each of the fault tree structure tabs in the spreadsheet into one tab for ease of import into CAFTA.
- Update gate and basic event names for consistency with associated data.
- Test features of the larger macros and check for errors.
- Make corrections for sets of events that were outside the algorithms of the conversion macros.

There was also a macro dedicated to deleting all tabs and memory of anything related to the model or conversion, for the purpose of starting over from scratch. This was necessary in order to troubleshoot the various macros efficiently.

2.3. Data

The source PSA data and type code files were converted to the CAFTA .RR format. As with the fault tree structure, these data were formatted in particular ways that made them amenable to manipulation by macro. Thus, an additional set of macros was developed to extract and organize these data based on algorithms that converted the data format and type to something recognizable by the CAFTA (Microsoft Access) database.

Macros associated specifically with data conversion were maintained in a separate spreadsheet from the fault tree macros. Its goal was to populate the CAFTA database tables with all relevant data. This spreadsheet was largely built manually from the source PSA model database files, but with help from a set of relatively small macros. The database files were imported to the spreadsheet and then the data from these files was reorganized and updated to match a format that can be imported to CAFTA's database via Microsoft Access. There are several small macros that were used to assist in the development, including the following types of functions.

- Populate specific columns of the spreadsheet with data from the source PSA model database files. Some columns were populated via equations, lookups for associated data types (e.g., failure mode units, type code specific exposure times, distribution types, etc.), or manual actions.
- Rename basic events to follow the new CAFTA type code structures.
- Create the type code format to be imported to CAFTA's TC table.
- Generate the type code format on the relevant spreadsheet tabs, to create a relationship with the associated basic events.
- Remove extraneous characters from basic event names longer than the 31-character limit of CAFTA after being renamed to follow the new type codes, such that the database would match what was done in the fault trees.
- Pull the Description data for basic events together and combine it for each basic event.
- Troubleshoot issues, such as identifying mismatches between the Type and Factor fields.

2.4. Common-Cause Failure

With respect to common-cause failures, the source PSA did not use special treatment in the fault tree structure or database that would impact the process described so far. Common-cause basic events would appear with multiple parents just as some other basic events or gates, and data would be populated manually as any other basic event data, so that the processes described in Sections 2.2 and 2.3 were sufficient to capture common-cause impacts.

2.5. Model Integration

The model integration and quantification involved generation of a linked, single-top fault tree from the event tree logic and results of the conversion. This was created and verified both manually and with the help of additional macros. The output of the previous sections was a set of spreadsheets/tabs with data formatted for CAFTA, which could be copy/pasted or otherwise imported as appropriate. For example, in order to efficiently import basic event data to the CAFTA database, the following instruction set would be followed:

1. Open the CAFTA database with Microsoft Access.
2. Create a copy of the BE table (structure only). In the copy, delete the UNITS field and all fields between DESC and TYPE.
3. Copy and paste columns A through E of the relevant tab (minus the headers) into the copy of the BE table.
4. Create an Update Query to match the basic event names between the BE table and its copy, and populate the C, FACTOR, DESC and TYPE columns of the BE table with the data from the copy.

Or to import type code data:

1. Copy and paste column A (minus the header) of the relevant tab into the TC table of the database.
2. Create a copy of the TC table (structure only). In the copy, delete the SOURCE and EQUATION fields.
3. Copy and paste columns B through G of the relevant tab (minus the headers) into the copy of the TC table.
4. Create an Update Query to match the type code names (TYPE) between the TC table and its copy, and populate the RATE, UNITS, DESC, DIST, PARM2(EF) and NOTES columns of the TC table with the data from the copy.

Generation of the top logic was a largely manual, collaborative effort to reproduce the accident sequences defined in the source PSA model, while significantly simplifying the top logic. The majority of system linking was captured in the previous steps with the judicious use of macros. Simplifications were warranted because CAFTA is a “large fault tree, small event tree” platform, unlike the source PSA code, meaning that many event tree branches could be subsumed under functional tops in CAFTA when properly linked.

Model integration and quantification were not without challenges. The original model was relatively simple, relative to U.S. and emerging Japanese standards, which presented a challenge in matching expectations for accident sequence logic. Similarly, the model results were somewhat sparse with which to compare results from CAFTA.

2.6. Interface and Validation

Several relatively large macros were developed to maintain the mapping algorithms, while many smaller macros were developed for specific tasks, such as reorganizing data, and then called by the larger macro set in proper order. For the majority of the automation, these were compiled and linked to a set of buttons in the user interface (Microsoft Excel). The macro-enabled spreadsheets contain instructions to display these buttons upon opening the

files. Some macros were not included in these sets and had to be run as needed via the code interface in Microsoft Excel, which is accessed with Alt-F11, when macros are enabled. These included test versions of macros, troubleshooting and error-checking macros, etc.

In principle, every macro important to the model conversion could have been called in turn by a single parent macro, linked to a single button. The primary reasons for separating the macros to some extent, and using a set of half a dozen separate buttons, were (1) to develop and maintain specific macro functions for major tasks independently and (2) to avoid computational limitations. Some of the larger macros would operate for several hours, depending on the number of external files being processed, and this could cause the process to run into memory limitations.

For verification, several individual fault trees were manually verified to have been converted correctly. The logic produced from the automated conversion was compared against these manual conversions and the fault tree, when imported to CAFTA, was compared with the associated source PSA model fault trees. Beyond these verifications, the code itself was able to identify certain errors (either through message boxes or printed on a default tab), and with the final conversion, none were identified. Finally, CAFTA has tools for verifying data population and other potential problems with the logic, which were used, and resulted in identifying no errors.

As mentioned in Section 2.5, there were challenges with integrating and finalizing the CAFTA model, which, in general, is not uncharacteristic of a model conversion. However, with the completion of the project, the CAFTA model was quantifiable and was able to reproduce the available results exactly.

3. CONCLUSIONS

As discussed in Section 2.6, the fidelity of the model conversion was verified. Typically, model conversion verification is characterized in part by the percentage to which the new results match the original, and anything better than 90-95% is considered adequate, with justification for known differences. In this case, with the available model and results, it could be said that this conversion was a perfect match. Model conversion typically requires roughly on the order of a month for bulk file conversion and then often an additional 1-3 months of logic structure reorganization, troubleshooting, and refinement. In this case, most of the effort was in developing and troubleshooting necessary macros, which took roughly 2-3 weeks, and another week completing the top logic and simplification. This process eliminated almost all subsequent necessity for troubleshooting and refinement. And the conversion of this particular model can now be modified as necessary and entirely reproduced within days, which is not typical of a model conversion without the use of such automation.

Automation of model conversion is just an example of the use of macros to simplify PSA tasks. Different model platforms facilitate the use of VBA to automate conversion or manipulation to different degrees (for example, CAFTA can import some data formats directly, precluding the need for some of the more significant examples of this project). But the benefits of applying relatively unsophisticated programming skills for repetitive or data-intensive tasks should be evident. For this project, the benefits included:

- Time/effort reduction.
- Error reduction.
- Automatic verification and error checking.
- Accuracy of conversion.

These types of benefits are commonplace when using macros to assist with repetitive or data-intensive tasks, of which PSA has many. So, in conclusion, it behooves the PSA analyst to put in the effort to learn some basic programming skills. This is particularly important in light of increasing industry interest in dynamic PSA, artificial intelligence and machine learning (e.g., for efficient data collection efforts), and application

programming interface (API) structures for PSA software tools (e.g., the ability to manipulate PSA models and results directly through external programming languages).

3.1. Final Thoughts

This paper highlights benefits of developing macros, in general, as a timesaving and error-reduction exercise, but with a focus on PSA model conversion. The automation discussed herein was developed for a specific purpose and model structure, and without the specific intent of long-term use and reproducibility. With some additional effort, a more permanent, robust solution could be developed to make model conversion repeatable and consistent, with fewer user steps required. This could be useful, for example, in freely converting between the two major platforms used in Japan (RiskSpectrum and CAFTA) to meet requirements of the Japanese regulators (NRA) to show adequacy of PSA results by comparison with the results calculated by other PSA codes or to use specific features of each platform for different purposes.