Explosives Safety Siting Quantity-Distance Engine and Flowcharts Criteria Merge and Software Rewrite

Authors:

Cameron Stewart; NAVFAC EXWC; Port Hueneme, California, USA.

Michael Oesterle PhD, PE; NAVFAC EXWC; Port Hueneme, California, USA.

Steven Willis; NAVFAC EXWC; Port Hueneme, California, USA.

Lea Ann Cotton; DDESB; Alexandria, Virginia, USA.

Key Words:

Explosives, Criteria, ESS, Quantity-Distance Engine, Flowcharts, Programming

Abstract:

The Quantity-Distance (QD) Engine for the Explosives Safety Siting (ESS) software is being rewritten to simplify the programming, reduce maintenance costs, and switch the programming language to promote future maintainability and adaptability.

Introduction

Explosives safety site plans are required whenever new construction or explosives operations change at any installation within the United States Department of Defense (DoD). The complexity, time-investment, and high level of risk involved with this process led to the development and deployment of ESS software to automate the majority of it. While much of the ESS software uses its Geographic Information System (GIS) platform, a key component has been programmed separately to calculate the Explosives Safety Quantity Distance (ESQD) arcs based on the criteria found in the DoD 6055.09-M (DoD criteria), NAVSEA OP5 (United States Navy criteria), and AFMAN 91-201 (United States Air Force criteria). This separate component is called the QD—for "Quantity-Distance"—Engine and it was programmed in VB.NET. The QD Engine was actually broken down into three separate engines—the DoD Engine, the Navy Engine, and the Air Force Engine, as three separate software files, one for each criteria manual. In addition, several other files were created for additional testing, validating, filtering, and configuration support. Three sets of flowcharts, one for each engine, were created in conjunction with these QD Engines as programming guides to demonstrate in each case how the criteria would be used to calculate ESQD arcs for any scenario.

Creating the QD Engine and flowcharts necessitated the creation of a whole set of variables to describe each type of explosive, facility or location, and defining attributes necessary to automate the calculation of ESQD arcs according to the criteria found in the manuals. Two key descriptions not found in any of the manuals are the type of Potential Explosion Site, or PES, and the type of Exposed Site, or ES. Any location which could be used for the handling or storage of ammunition and explosives could be a PES, and any location exposed to a detonation at the PES is identified as an ES. Not all types of locations can be a PES, but all locations may serve as an ES. Other variables describe the attributes of the PES and ES, their relationship to each other, and the nature of the ammunition or explosive located at the PES. Some of these variables have explicit reference to explosives safety criteria, others

merely facilitate the automated ESQD arc calculation process. Examples of PES and ES types are found in Figure 1 below; examples of PES attributes are found in Figure 2 below.

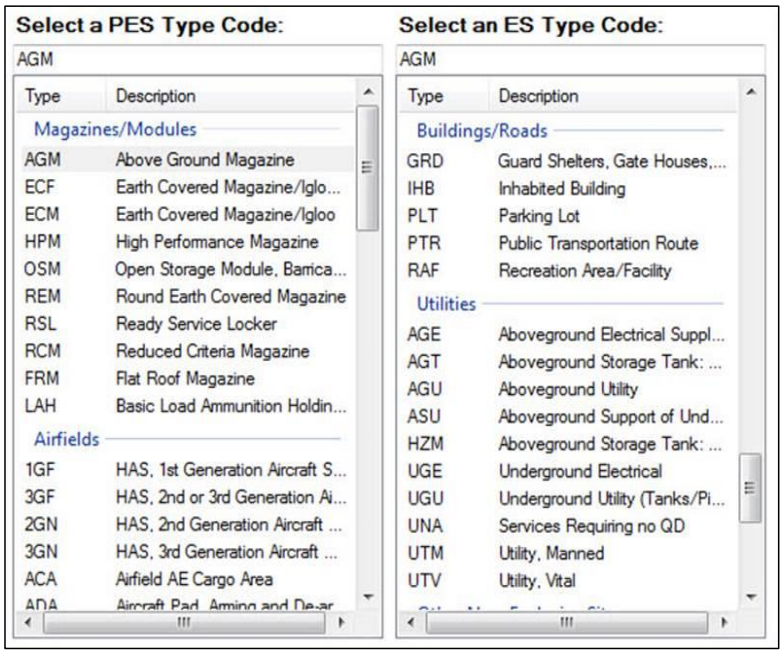

Figure 1 PES and ES Examples

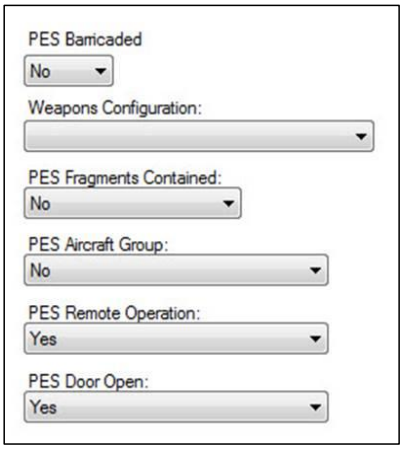

Figure 2 Sample PES Attributes

The DoD Engine and Navy Engine and their associated flowcharts were developed and maintained by the Naval Facilities Engineering Service Center (since renamed the Naval Facilities Engineering and Expeditionary Warfare Center or EXWC), while the Air Force Engine and its associated flowcharts were developed and maintained by a DoD contractor. This has resulted in unnecessary redundancies in the flowcharts and QD Engine and added complexity whenever updates to the criteria necessitated updates to the flowcharts and QD Engine. In order to promote the long-term sustainability of ESS, it was decided to merge all of the flowcharts and engines. At the same time, the development of a web-based application of ESS led EXWC to recommend changing the format of the QD Engine from VB.NET to C#. Both the merging of the engines and transition between programming languages has resulted in significant changes to both the flowcharts and the QD Engine.

Commented [OMGCNEC1]: I removed ISA because I don't want to imply that ISA was doing something inappropriate, which they weren't, but someone may perceive it that way;

Commented [OMGCNEC2R1]:

Flowchart Merge

All three versions of the flowcharts consisted of different sections corresponding to different sections of criteria. The general flow consisted of going from a common starting point to the section where an exposure—Inhabited Building Distance, Public Transport Route Distance, Intraline Distance, or Intermagazine Distance—related to the level of risk accepted for the pairing of a specific PES with a specific ES would be assigned. Based on that assigned exposure, a required minimum distance between the paired PES and ES would be calculated based on the type and amount of explosive associated with the PES. The different sections where the exposures would be assigned corresponded to the type of ES and PES paired together. These types included locations associated with airfields, piers and wharves, utilities, and explosives storage and operating locations. There was also a section reserved for intentional detonation activities. Lastly, there were sections for the distances associated with each class or hazard division of ammunition or explosive—1.1, 1.2.1, 1.2.2, 1.2.3, 1.3, 1.4, 1.5, and 1.6.

As EXWC developed the DoD and Navy Engines and flowcharts, and it was evident at the time that there were only minor differences between the criteria associated with both, the differences between the engines and flowcharts were minor. Having a different author behind the Air Force Engine and flowcharts led to a great many differences between that engine and flowchart set and those developed by EXWC. Therefore, early in the merge process a meeting was held with representatives from the DDESB and each service who were experts in the explosive safety criteria to review the flowcharts. This revealed that there were many more similarities between the different criteria

manuals than had been thought previously, and that the merged flowcharts could be greatly simplified. As a result, more changes were made to the flowcharts, which were reviewed by the same group of experts. An example of this process can be seen in Figure 3 and Figure 4 below. For the case where "ES = EOL", the branching to the right is much more complex in the former, which was presented at the initial review, than it is in the latter, which was created as a result of these reviews. This simplification was duplicated throughout the flowcharts.

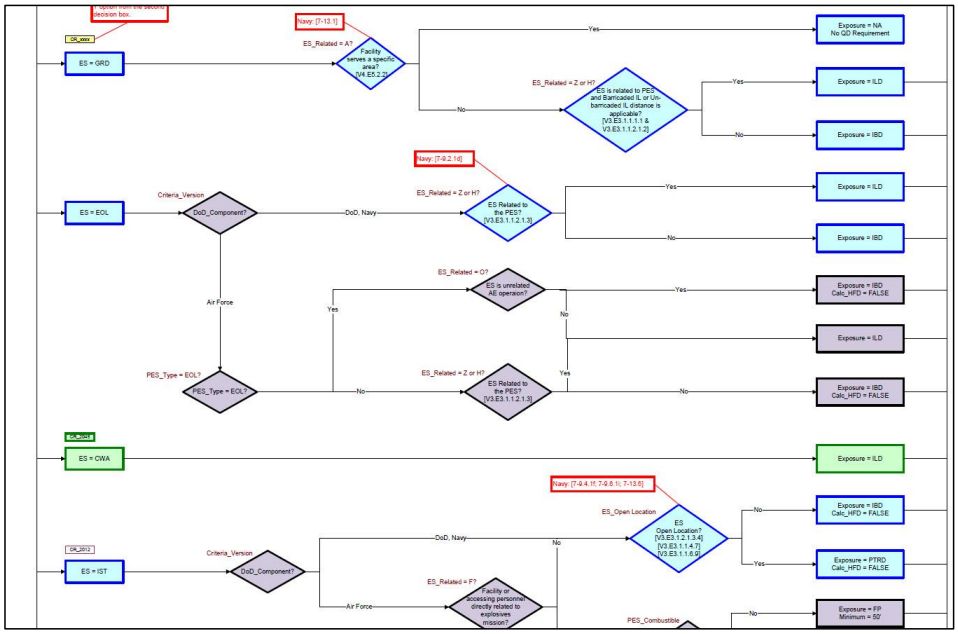

Figure 3 Initial Merged Flowchart Draft—Exposures

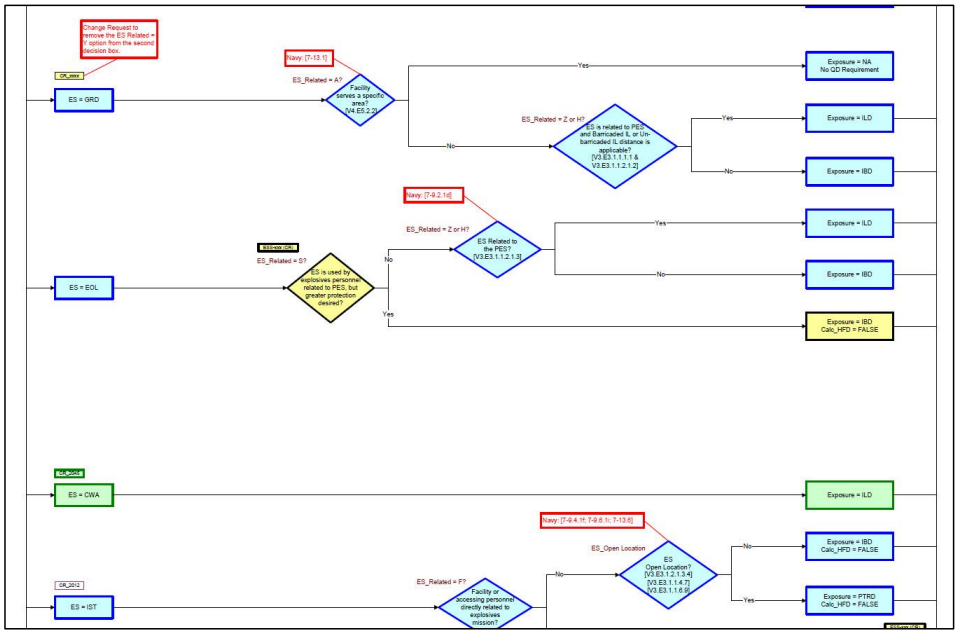

Figure 4 Revised Merged Flowchart--Exposures

In addition to reconciling differences and simplifying whenever possible, another aspect of the flowchart revision was the inclusion of new criteria which had been approved but not yet published in any of the manuals nor included in ESS. Most of these new additions pertained to the criteria governing intentional detonations and utilities. The flowcharting logic for each of these additions had to be drafted and then subjected to the same process of review and revision as with the merging of the criteria. An example of the new criteria drafted into the flowcharts is found in Figure 5 below.

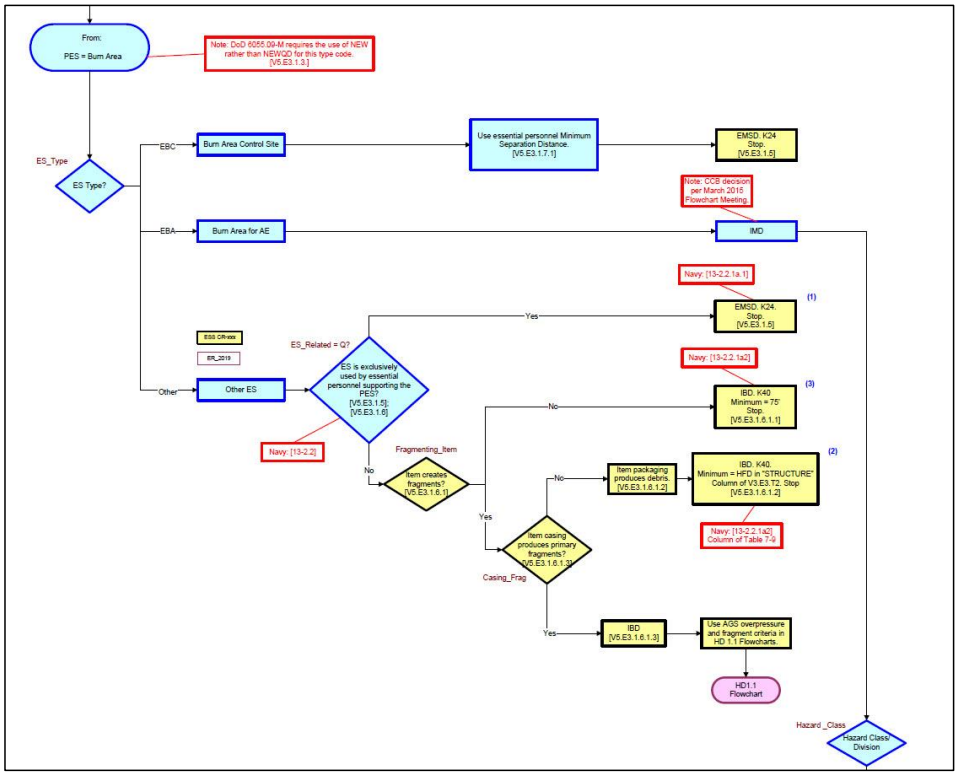

Figure 5 New Criteria for Intentional Detonations

By merging the flowcharts from three separate sets totaling 276 pages down to one common set totaling 152, less time and effort will need to be spent in maintaining them in the future. Readers will not have to look at three different sets to check between three different sets of criteria, but will find all the criteria in one page, and, in many cases, will see that the criteria between all three manuals is the same. Also, by merging the flowcharts first, there is now a blueprint for merging the QD Engine that has made that process quicker and more efficient.

QD Engine Rewrite

Before going over the actual program rewrite process, some explanation of how the QD Engine is validated is useful. Concurrent with the original development of the QD Engine was the creation by hand of a database of thousands of problems with expected answers from the criteria manuals. In this database there is at least one problem for each unique flowchart path in the DoD and Navy flowcharts, and many of the paths in the Air Force flowcharts have validation problems as well. While the validation problems is not comprehensive, as that would be infeasible given the time, money, and manpower available to EXWC, enough problems have been created to give EXWC a high level of confidence few if any bugs exist in the QD Engine when it is tested using this database. In order to test the QD Engine with this database, a validation program was written as one of the supporting programs for the QD Engine, and it can be run either directly from the compiler used by EXWC, Microsoft Visual Studio, or

from a stand-alone application from ESS called the QD Calculator (the QD Calculator allows a user to calculate ESQD arcs as a minimum required distance between a PES and an ES separate from the GIS interface in ESS). This validation program will then run all of the analyses in the database associated with a particular engine (DoD, Navy, or Air Force) and compare the distances calculated by the QD Engine and compare them to the expected distances entered into the database (see Figure 6 and Figure 7 below). Although there are thousands of validation problems in the database, it takes the program only a couple of seconds to run through all of them. Thus, whenever a change has been made to any of the engines, the validation program can be run and quickly reveal most if not all of the errors the change may have created unintentionally. Thus, this database and its associated program has become one of the most important tools to ensure the QD Engine is reliable and stable. It does have one big drawback, which is that necessary changes to the database must be performed by hand. When major changes are made to the explosives safety siting criteria—which is rare—a lot of work must be done to locate and update all of the affected problems in the database and, when necessary, create enough new problems to cover the newer criteria.

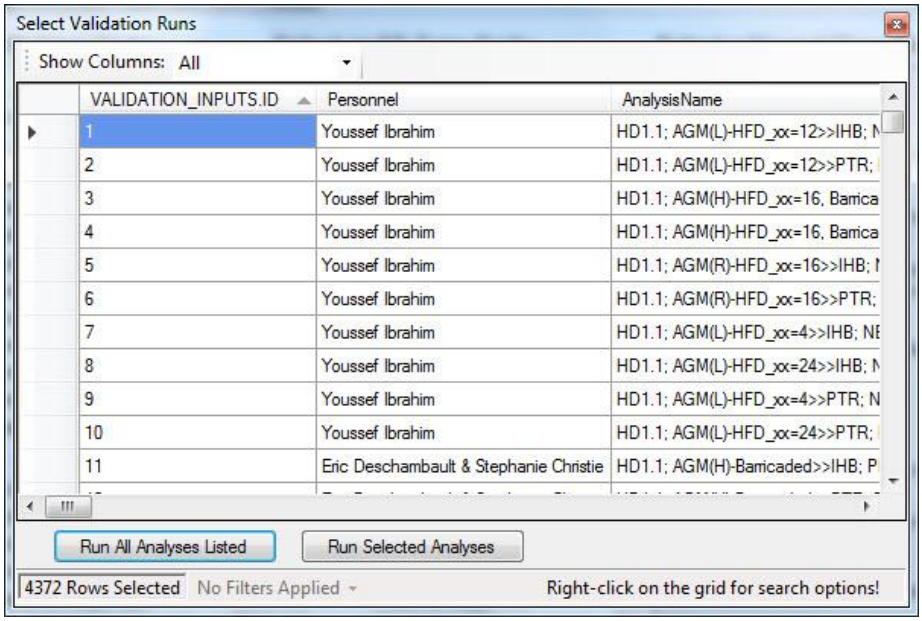

Figure 6 QD Calculator Validation Initiation

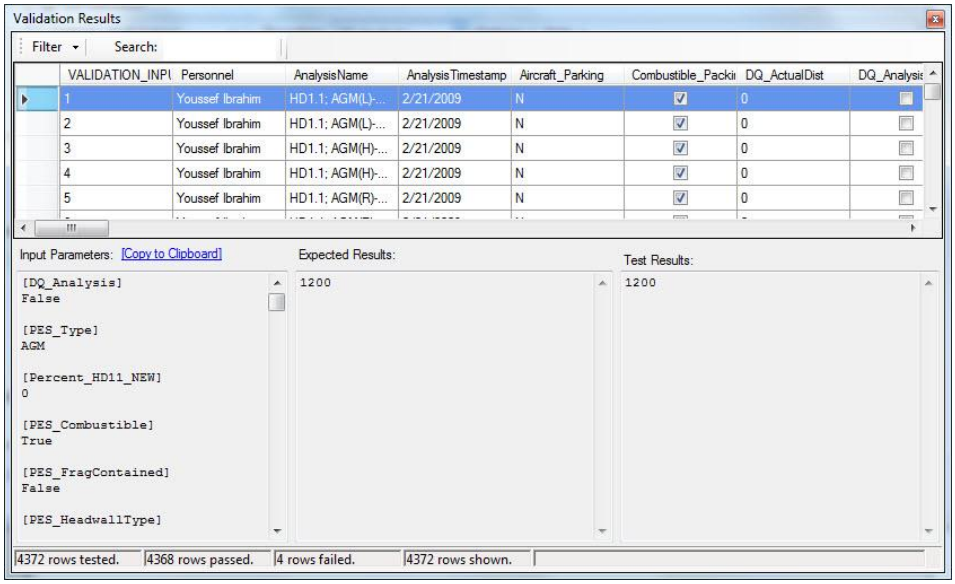

Figure 7 QD Calculator Validation Results

The process to rewrite the QD Engine was broken down into stages in order to make the transition from the multiple engines in VB.NET to one engine in C# as smooth as possible. Stage 1 involved using an automatic tool to convert the VB.NET code into C# and then correcting any remaining bugs preventing the existing, unmerged engines from running without errors. No changes were made to the way the engines calculated criteria, and nothing was done further at this stage towards merging the engines. This allowed EXWC to use a library of thousands of problems created to validate the engines in VB.NET to also validate the C# conversion and ensure that the new programming language would work and that the merging and revisions necessary would be based off a stable program. Several errors were discovered and corrected this way. At the conclusion of this stage, the entire VB.NET code for the QD Engine was in C# and the results exactly matched those obtained from the VB.NET version.

Stage 2 commenced with using the DoD Engine as the starting point, and going through each portion of the code that related to the sections in the merged flowcharts and making changes as necessary so that the logic in the code matched the logic in the flowcharts. While much effort was made to reuse existing code in order to expedite the revision process, complete rewrites of portions of the code were done when more efficient programming techniques than were used previously were possible. Such revisions only made sense if the techniques were not too advanced, as EXWC programming policy has been that internally developed and maintained programs could be understood by those with basic training in programming but did not hold degrees in programming. Since EXWC does not employ many trained programmers but has many engineers with programming experience, this allows greater flexibility in those at EXWC who may contribute to internally developed and maintained programs while also ensuring that programming does not get so specialized that turnover in project personnel jeopardizes EXWC's ability to support its software.

At the time of this writing stage 2 is over half-way done. Limited testing has been done as stage 2 has progressed to ensure that there is confidence in at least partial functionality of the code that has been rewritten. Complete testing will not be possible until at a later stage as the validation library mentioned previously cannot be used until changes

are made to the programming supporting the QD Engine. Thus, stage 3 will involve reconfiguring the programming related to the use of the validation library so that it can be used to test the merged QD Engine, as well as revise and add to the existing library problems to account for changes in criteria and ensure that each branch in the merged flowcharts has a corresponding validation problem. When this is complete, EXWC will be able to use the merged QD Engine with ESS and with a high degree of confidence that any remaining bugs in the QD Engine will be minimal.

The last stage of the QD Engine rewrite will be to make any additional changes to the structure of the QD Engine and its supporting files to enable seamless integration with ESS in both its desktop and web-application platforms. This will require coordination between the general ESS development team and the QDE development team. Once this is complete, most future updates to the QD Engine should be relatively simple with minimal time and effort required by either ESS or QDE development teams to complete and integrate into ESS. In addition, unlike in the past where criteria changes often had to be made two or three times across the different engines, now these will only need to be made once—reducing the chances for error as well as the amount of work required.

Commented [OMGCNEC3]: I think this is more appropriate because it does not call out a specific company.

Commented [OMGCNEC4R3]:

Commented [OMGCNEC5R3]:

Conclusion

Merging the flowcharts and rewriting the QD Engine has been a significant task which will pay dividends in the future. The cost of maintenance and risk of error have both been reduced. A by-product of this process has been the minimization of differences between each individual component's implementation of criteria into the QD Engine, thereby increasing uniformity across all components. By changing the programming language of the QD Engine from VB.NET to C#, the program has become more efficient and is positioned for continued relevance as a commonly-used, well supported language and one that is more adaptable to different applications. With ESS being developed into a web-based application, with the potential for mobile application development being considered in the future, that flexibility in the QD Engine code will support these changes. As a key component of ESS, the QD Engine and supporting flowcharts are well positioned for sustainable maintainability and adaptable to future development of ESS as it evolves to better serve the warfighter.