START THE "QUANTITY-DISTANCE ENGINE"

By
Frank R. Johnson, Jr. and Phillip C. Wager
Naval Facilities Engineering Service Center
Port Hueneme, CA

ABSTRACT

Explosives safety standards contain directive and commentary language. Both are founded upon experience derived from science and mishap history. Approval authorities expect full compliance with each directive requirement. Automation of such standards enhances the correct and complete application of the standard which in turn enhances the probability of safety, reduces approval review requirements, and generally improves productivity. However, development of automated explosives safety applications consumes both approval authority and software engineering resources resulting in substantial apparent and hidden costs. Experience has demonstrated that life-cycle maintenance of such applications, especially tasks associated with keeping current with approval authority revisions to the standards requires more resources than the initial development. Unresolved maintenance issues and the associated resource programming can and have made otherwise successful standards applications quickly obsolete. The tangible benefits associated with these applications indicate this type of conclusion is vastly inconsistent.

This paper discusses a methodology used to create an automated explosives safety standards application. The paper assumes the approval authority maintains the standard and drafts revisions to it using a word processing application such as Microsoft Word. The ideal methodology is envisioned to produce the automated version of the standard directly from the word processor document. Additionally the methodology ensures completeness, consistency and conciseness are properties of the automated process. An interim methodology is presented using Chapter 9 of DOD Ammunition And Explosives Safety Standards (DoD 6055.9-STD).

A computer program unit encapsulating all Chapter 9 site planning requirements produced using the interim methodology is discussed. This program unit may be used as a "quantity-distance engine[1]" for explosives safety site planning applications.

INTRODUCTION

Engineering standards and standard practices are usually founded upon the desire to produce a quality product that can be safely used by the public. Even in the defense environment, the military executes its operational and readiness mission using quality products that are safe to

[1] Engine is an alternate term for processor. Its usage implies the processor is a component of a host or container application.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **AUG 1998**	2. REPORT TYPE	3. DATES COVERED **00-00-1998 to 00-00-1998**
4. TITLE AND SUBTITLE **Start the 'Quantity-Distance Engine'**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Naval Facilities Engineering Service Center,Code ESC62,1100 23rd Avenue,Port Hueneme,CA,93043-4370**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
See also ADM001002. Proceedings of the Twenty-Eighth DoD Explosives Safety Seminar Held in Orlando, FL on 18-20 August 1998.

14. ABSTRACT
see report

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**13**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

employ. The ability to project military power must not be compromised by preventable accidents. Furthermore, the military establishment must be able to coexist within a nonmilitary environment without unnecessary and resource depleting litigation. A robust safety program with its accompanying standards and standard practices directly supports and enhances our defense posture.

Military explosives safety standards encompass the development, manufacture, storage, use, and disposal of a wide variety of explosive material. The associated safety program is managed by the Department of Defense Explosives Safety Board. Each of the military services and defense related agencies have various organizations that specialize in administering their explosives safety program. The material discussed in this paper may have application to any or all of these organizations, however any one of these organizations will be referenced simply as the approval authority.

Approval authorities issue various documents to implement their respective portion of the Department of Defense Explosives Safety program. These documents include various standards, handbooks, regulations, directives, instructions, technical manuals, and standard operating procedures. Some of these are binding, while others are advisory. The approval authority expects full compliance with the requirements contained in the binding documents. When advisory requirements are used, compliance with each of their provisions is also expected. The volume and complexity of the requirements including their application frustrate these expectations, on the part of both the regulator and the regulated. An ideal system based upon modern computer technology can ensure full application of the requirements and compliance.

An ideal system addressing each requirement in the context of a given application situation will ensure safety risks are minimized while improving productivity of the part of all involved parties. Each of the application characteristics are evaluated against the complete set of pertinent requirements, omitting none and ensuring approved interpretation of each requirement in the application context. The development, maintenance and validation of the ideal system must be accomplished through full participation of the approval authority to ensure the system properly implements each requirement. The resulting system certified by the approval authority would be used by field activities. Then approval requests require less scrutiny by approval authorities when such approved automated systems are employed. Produced products require little if any modification due to noncompliance with safety requirements. Thus the ideal system reduces the administrative burden associated with implementing and monitoring compliance.

The realities associated with creating a near ideal system can blur the vision. Experience has demonstrated that the development and maintenance of automated engineering systems, especially when criteria is incorporated, often produce less than desirable systems with a uneconomical short life-cycle. One particular lesson learned is the fact that life-cycle maintenance of engineering applications that incorporate criteria requires more resources than the initial development. This resource requirement includes revisions to the criteria, including associated revisions to the automated application and the associated validation. Thus future criteria revisions must include appropriate software modification and validation as an integral part of the of the revision process. To do otherwise will result in application obsolescence within a very short period of time, even within twelve months in some cases. This experience can be focused to create a systems environment that can produce and maintain automated explosives safety applications that are functional and within limited approval authority and software engineering resource constraints.

MORPHOLOGY OF A STANDARD

Explosives safety standards are derived from science and mishap history. Engineering and scientific studies are conducted and substantiated by extensive testing programs. Various methodologies are used to interchange data with interested communities throughout the country and our allied nations. Proposed standards are prepared, reviewed, and approved or rejected. The result is a set of documents that contain directive and commentary language. These documents are published and maintained, recently in digital format, by the appropriate approval authority.

The style and format of specification documents, themselves adhering to a government publications standard, usually follow a prescribed outline form that facilitate content reference and critical reading. Furthermore, the outline form simplifies small changes and major revisions, both are common with standards documentation, and tailoring to meet specialized applications.. Additional characteristics of importance include:

Coherence	Logical, orderly, consistent arrangement
Precision	Clearly expressed or delineated, strictly distinguished, conforming to rules
Directive	Rules, instructions
Descriptive	Convey an idea, characterize, represent pictorially, depict
Emphasis	Focused attention
Commentary	Explanations, interpretations

Tables, figures, footnotes, end notes, references, and appendices complete the characteristics of specification documents. Specifications that conform to these characteristics are not difficult to digest, implement, and systematize.

IDEAL SYSTEMATIZATION

The ideal systemization methodology is envisioned to produce the automated version of a standard directly from its digital form, which might be a particular word processor format, publishing format, or standard file format. This process includes the following functions:

Analysis	Study the standard to identify and relate each requirement including its required action and the parameters that trigger and control the action.
Design	Convert each and all requirements into a systematized procedure.
Instruction coding	Express the design in a particular set of executable statements associated with a particular programming language that can be translated into machine instructions.
Instruction compiling	Translate the programming language into instructions that can be executed by a computer. This is accomplished by a software engineering tool known as a compiler.
Linking	Convert compiled instructions to a form that can be loaded and executed by combining separate modules into one executable module, usually an executable file identified as the application. This is accomplished by a software engineering tool known as a linkage editor.
Loading	Initiate and control the process that executes the linked program in the processors of the computer. This process is a result of a command issued by the user.
Testing	Examine each standard requirement in the application context. An approved comprehensive validated suite of scenarios compliant with the standard are employed.
Modifying	Change the design and instruction coding to improve test performance and usability, and to incorporate changes and revision to the standard.
Installation	Move the application, including all of its parts and relations, from distribution media to the host computer culminating with a fully functional application.
Maintenance	Respond to functional enhancement requests, to computer system upgrade migration, and to changes and revision associated with the standard.

Software engineers have created development environments that automate the functions shown above in italic text. Software engineering tools used in the analysis, design, and instruction coding functions are also included in these development environments. The modification and maintenance functions are actually revisits to the other functions, which bring to bear the associated tools sets in the accomplishment of these functions. However, even with these tool sets the analysis, design, and instruction coding functions are primarily a set of intellectual human tasks. Thus the quest is to expand this tool set to improve efficiencies associated with the remaining manual tasks.

Batch processing standardized validated test sets is one way to bring the testing function closer to the ideal systemization methodology. Specialized test drivers and data sets devised to meet specific objectives have been successfully used to achieve more comprehensive testing and to improve productivity. The development of such test sets requires the participation of the approval authority to facilitate their compliance certification.

The hypothesis of this paper suggests one can locate advanced tools to extend the automation of the analysis, design, and instruction coding functions. The vision is to use language processing technology and advanced software engineering tools to move the digital text of the standard through the analysis, design, and instruction coding functions, then linking the result with existing software engineering tools associated with the italicized functions. The rational is to squarely put the automated standards application initial development and extended maintenance within the scant resource environment of the approval authority. It may even be possible to integrate this process with the standard development and maintenance process to produce the standard text and associated automated application simultaneously. With this full integration the automated application might be used during the process to investigate the proposed revisions before they are approved. This hypothesis has been tested to a certain extent within the context of developing an automated version of DoD 6055.9-STD.[2]

ANALYSIS OF DoD 6055.9-STD

The standard was critically read and studied to identify and relate each requirement including the required action and the parameters that trigger and control the desired action. Review of the derived annotations and ancillary notes indicated a level of complexity associated with the document that was not obvious at the commencement. Technology was sought to facilitate the analysis.

Critical reading initially dissected the document separating the commentary from the directive language. Attention was directed to the various function words such as prepositions, pronouns, or auxiliary verbs. Cross references were mapped into the directive context. The requirements were logically connected with the objective to achieve consistency. This effort resulted in various procedures, which include relational and case expressions.

Natural language processing technology usually associated with linguistics was attempted. The following concepts[3] are essential to implement this technology:

[2] Department of Defense, DoD 6055.9-STD, “DOD Ammunition and Explosives Safety Standards,” Under Secretary of Defense for Acquisition and Technology, August 1997.

[3] The American Heritage Dictionary of the English Language, Third Edition, Houghton Mifflin Company, 1992. Electronic version licensed from InfoSoft International, Inc.

Sentence	A grammatical unit that is syntactically independent and has a subject that is expressed or, as in imperative sentences, understood and a predicate that contains at least one finite verb.
Paragraph	A distinct division of written or printed matter that consists of one or more sentences, and typically deals with a single thought or topic.
Syntax	The pattern of formation of sentences or phrases in a language.
Semantics	The study of meaning in language forms.
Grammar	The system of rules implicit in a language, viewed as a mechanism for generating all sentences possible in that language.
Structure	The way which parts are arranged or put together to form a whole.
Parse	To break a sentence down into its component parts of speech with an explanation of the form, function, and syntactical relationship of each part.

In its present state this technology was not capable of interpreting the requirements of the DoD 6055.9-STD. Sentences were parsed and semantically analyzed, but paragraph and inter-paragraph semantics were not correctly rendered. Furthermore, the current document structure lacks coherence, a state that is clearly beyond the technology. Some revisions associated with inter-magazine distances that occurred between the October 1992 and the August 1997 editions of DoD 6055.9-STD were also beyond the semantic capability of the technology, because they involved a philosophical modification (PES vs ES barricades) of the standard. Further developments are required before this technology can be utilized to automate the development and maintenance of an automated DoD 6055.9-STD application.

Decision table[4], sometimes decisions logic table (DLT), technology was employed. This device is a concise, precis, and complete way to display a combination of conditions to be met and actions to be taken: they organize instructions into rules with structure. There are three areas in a basic decision table: condition to be met, action to be taken, and rules which are combinations of conditions and actions. A draft decision table for DoD 6055.9-STD paragraph 9.B.1 is presented in Table 1. This table has nine conditions, ten actions, and fourteen associated rules defined by Yes, No or "-" (denoting either Yes or No) for each condition. Software engineering tools[5] are available to complete and check consistency of decision tables. Some of these tools will also create relational if-then-else rules in English or a computer language.

[4] London, Keith R., Decision Tables, Auerbach Publishers Inc, Princeton, NJ, 1972.

[5] Logic Gem, by Logic Technologies, Yucca Valley, CA. or TableWise by Odyssey Research Associates, Ithaca, NY.

Table 1. An incomplete decision table for DoD 6055.9-STD paragrah 9.B.1

DLT	Identifier:	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14
C1	Hazard division 1.1 Mass-detonating explosives	Y	N	Y	Y	Y	Y	N	Y	N	Y	N	N	N	N
C2	Hazard division 1.2 Non-mass detonating explosives	N	Y	Y	Y	N	N	Y	Y	N	N	Y	N	N	N
C3	Hazard division 1.3	N	N	N	N	Y	Y	Y	Y	N	N	N	Y	N	N
C4	Hazard division 1.4	-	-	-	-	-	-	-	-	-	-	-	-	-	Y
C5	Hazard division 1.5 for transportation	N	N	N	N	N	N	N	N	Y	N	N	N	Y	N
C6	Hazard division 1.6	N	N	N	N	N	N	N	N	N	Y	Y	Y	Y	N
C7	HE equivalent weight for division 1.2 known	N	N	N	Y	N	N	N	N	N	N	N	N	N	N
C8	HE equivalent weight for division 1.3 known	N	N	N	N	N	Y	N	N	N	N	N	N	N	N
C9	DDESB buffer configurations provided	N	N	N	N	N	N	N	N	N	N	N	N	N	N
A1	Distance = f(NEW) 1.1 dist	X						X		X	X			X	X
A2	Use DLT 1.2 dist		X									X			
A3	TQ Distance = f(TQ NEW)			X		X			X						
A3	Distance = MAX(1.1 TQ Distance, 1.2 TQ Distance)			X											
A4	Distance = 1.1 distance					X									
A5	Distance = MAX(1.2 distance, 1.3 distance)							X							
A6	Distance =Max(1.1, 1.2,or 1.3 TQ distances)								X						
A7	Distance = f(1.6 NEW + 1.3 NEW)												X		
A8	Distance = f(1.1 NEW + 1.2 HE EQW) 1.1 dist				X										
A9	Distance = f(1.1 NEW + 1.3 HE EQW) 1.2 dist						X								
A10	Distance = f(NEW largest stack + buffer material wt)														

Automated flowchart[6] technology was employed. Figure 1 is an example flowchart derived from DoD 6055.9-STD chapter 9. The majority of the standard has been flowcharted using this technology[7]. Both the relational if-then-else expressions and the case expressions can be clearly represented. However, there are no automated completeness and consistency checks as there are with the decision table technology. Consistency and completeness qualities can be achieved by graphic observation, however. A significant example of revealing possible inconsistencies using a flowchart was found in the analysis of the fragment hazard provisions contained in paragraph 2.E and paragraph 9.C.1 of the standard. Some automated flowchart applications support automated software code generation. This automated software generation technology was not utilized because two flowcharts were required for its implementation. One flowchart based upon English language terminology found in the standard was used to analyze the standard. This format would not produce executable statements compatible with any programming language. Another flowchart using the desired programming language syntax could have been developed. It was found that this second chart was not necessary because of the relative ease to manually produce executable statements from the English language flowchart.

[6] Farina, Mario V., Flowcharting, Prentice-Hall Inc., Englewood Cliffs, NJ, 1970.
[7] FlowCharter7 by MicroGrafx Inc., Richardson, TX

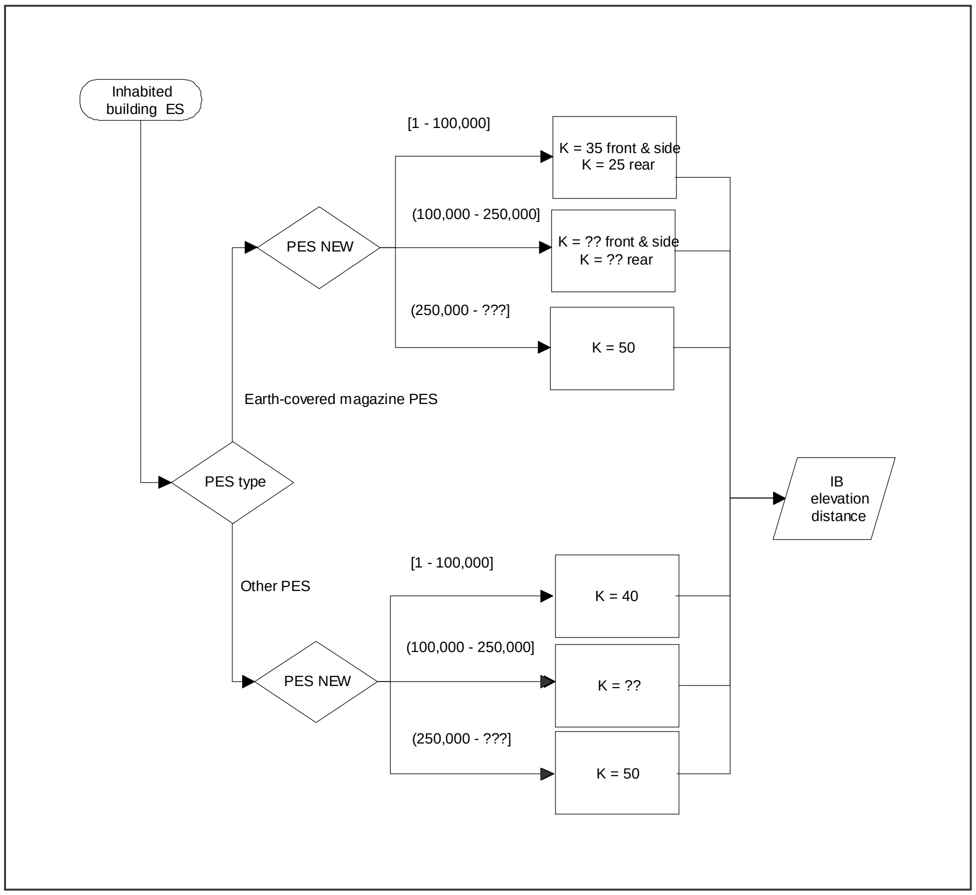

Figure 1. Inhabited building distance, paragraph 9.C.1.a.

DESIGN OF THE Q-D ENGINE

The analysis of the standard indicated a set of procedures, methods in object orientated programming[8] (OOP) terminology, could easily implement its requirements. Figure 2 shows how the some of these procedures were implemented in the Defense Explosives Safety Management Suite (DESMS) Explosives Safety Siting (ESS) Module sponsored by the Defense Environmental Security Information Management (DESCIM) Program. The current design includes methods for blast over-pressure, fragments, thermal, and chemical hazards within the confines of the engine. Future design revisions may also include additional methods such as measurement of PES-ES separation distance vectors and quantity-distance (Q-D) "arc" drawing templates, which are currently implemented outside the engine.

[8] Graham, Ian, Object Oriented Methods, Addison-Wesley Publishing Company, Harlow, England, 1996.

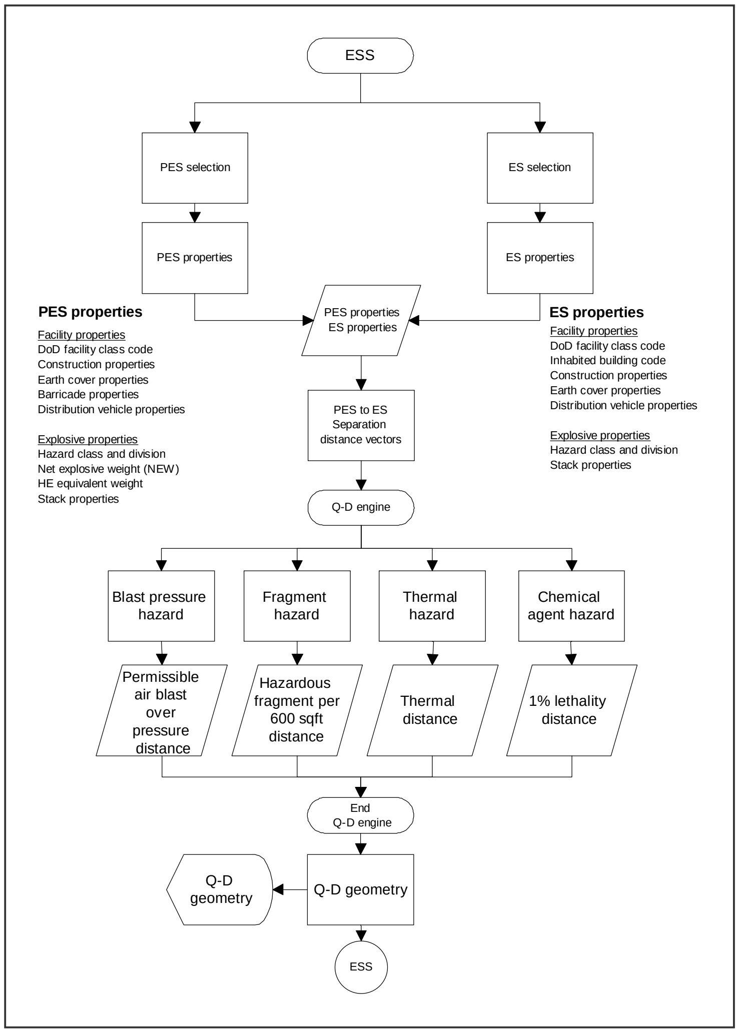

Figure 2. Q-D Engine design concept

A fundamental design premise is the use of the Q-D Engine in numerous applications. Discussions with the Q-D subject matter experts indicated a variety of requirements exist for automated Q-D criteria. These requirements include: calculators to determine the separation distance given the quantity of explosives and the quantity of explosives given the separation distance, Q-D computations compatible with commercially available mapping and geographic information systems (GIS), service specific Q-D applications, and a DoD standard application useable by each of the services. In response, a separate reusable piece of isolated functionality with a standard interface, even a component to a host or container application, was devised. Thus a prototype Q-D Engine component was created and successfully tested as a dynamic link library[9] (DLL) with C++, VisualBasic, PowerBuilder, and Delphi host siting applications.

[9] Dynamic link libraries (DLL) is a file that contains functions or subroutines implementing procedures (methods). The DLL is activated by a call to one of the methods contained in the library. Through the linking process the method is linked to the calling application while it is running, thus the term dynamic link descriptor. A DLL can be replaced by a simple copy function without modification to the host program (given certain conditions are met).

Maintenance resource requirements also impacted the design of the Q-D Engine. As an explosives safety subject matter expert, our organization noted the frequency of changes and revisions associated with the various explosives safety standards documents, especially those associated with separation distance requirements. Use of the automated decision table and flowchart technology coupled with an appropriate programming language demonstrated moderate resource requirements to modify the prototype Q-D engine to reflect the hazard division 1.1 changes occurring between the October 1992 and August 1997 versions of DoD 6055.9.

The design addresses the standard methodology to determine the required separation distances between a potential explosion site (PES) and a related exposed site (ES). The Department of Defense, Army, Navy, Air Force, and others develop and maintain standards to define the separation distance with respect to the quantity of explosives associated with the PES. With the participation of each of the respective approval authorities authorized and certified Q-D Engine components are envisioned. The DESMS will automate each of these four standards during FY99.

INSTRUCTION CODING

At least five concepts associated with instruction coding proved essential to achieving the objectives of the Q-D Engine component. These concepts are: the interface between the component and its host or container, the data environment within the component, Q-D procedures or methods within the component, if-then-else constructs, and case constructs. These internal notions will be briefly discussed because they can significantly impact development and maintenance resources.

Interface. The interface between the host and the component must be concise and persistent. This means the name, type, sequence, and number of arguments associated with each function or subroutine in the DLL must not change[10]. It also means there are restrictions associated with input and output arguments. The interface must only support parameters that are directly associated with the particular paragraphs of the standard that are being implemented within the associated procedure. Supplemental control parameters are highly discouraged. Single purpose functions and subroutines facilitate these requirements[11]. DLL replacement without modification to the host application can not be achieved with a variant interface.

Data environment. The data environment within each of the functions and subroutines must be hidden or inaccessible from the host. This is object oriented (OOP) characteristic known

[10] Plug compatibility in a residence electrical system is analogous to the DLL interface. A 220 volt plug is not plug compatible with a 110 volt plug. The interface arguments are analogous to the prongs on the plug and the receiving holes in the receptacle.

[11] A subroutine and a function are two types of subprograms that define particular computation operations within a more general main or host program. Thus an involved process may be subdivided into independent sub-processes. In this context the number of return arguments that are supported distinguishes a function from a subroutine. The function only allows a single value, while a subroutine may be multi-valued. The function notion has been derived from the function concept used in mathematics.

as encapsulation. Data visibility[12], type[13] and representation[14] are additional notions that are included within this concept.

Procedures. The implementation of the standard should be accomplished using single purpose Q-D procedures or methods. So doing will produce a one to one relationship between the written and automated standard. This relationship facilitates keeping current with changes and revisions to the standard by requiring replacement of only revised procedures. Many of these methods are hidden from the host application and are only used when certain values are passed through the interface, in other words they are invoked by a message from the host.

If-then-else constructs. The if-then-else constructs are one of two constructs that simplify instruction coding associated with the standard. If a relational expression, which includes the set of relational operator (equal to, greater than, less than, and, or, not) is true then a block of executable statements are executed, otherwise (else) another block is executed. The if-then-else construct might appear in a programming language as:

```
If (relational expression) then
        Block of executable statements
    Else If (relational expression) then
        Block of executable statements
    Else
        Block of executable statements
End if
```

Analysis of the standard demonstrates the necessity of this construct, a fundamental execution control construct associated with programming languages.

Case constructs. The case construct is the second type that simplifies automation of the standard. This construct uses a case expression and a case selector value to perform. The case expression computes to a single value which in turn defines which set or case of executable statements are executed. The case construct might appear in a programming language as:

```
Select case (case expression)
    Case (selector value)
        Block of executable statements
    Case default
        Block of executable statements
End select
```

[12] Data visibility determines which functions and subroutines have access to particular data entities.
[13] Data type distinguishes integer, real, complex, logical, and character data entities.
[14] Data representation determines how many machine dependent bits are used and how the bits are organized to represent a particular data value.

Due to the nature of the standard this control construct may be more frequently employed than the first. Although this construct is included in most programming languages, implementations differ resulting in either simple or straight forward or relatively difficult implementation.

VALIDATION AND CERTIFICATION

One of the benefits of automated explosives safety was the simplification of the approval request process standards was previously asserted. Critically reading the standard requires extensive familiarization, even memorization, of the contents to apply the requirements without error. Additionally, meticulous attention to detail is required. Approval authorities expend resources to ensure approval requests are reasonable and meet the specific requirements. The approval process is known to consume calendar time. To achieve the desired impact on the approval request process the automated technology must be acceptable to the approval authority.

Validation. The Q-D Engine should be validated with the results approved by the approval authority. This can be easily accomplished by defining a standard set of PES-ES relationships, each with specific characteristics designed to test specific requirements of the Q-D standard. The set should comprehensively address the standard. The approval authority should approve both the set of PES-ES relationships and the associated "standard" separation distances. A validation host application can be used to solve for the separation distance for each case using the Q-D Engine. Then the Q-D Engine results will be validated when they agree with the approved results.

Certification. The approval authority should certify and control the distribution of each version of the Q-D Engine. During the development or modification of the Q-D Engine involvement of the approval authority is essential, especially to interpret provisions in the standard and to resolve ambiguities. The approval authority should check the results of the validation solutions. When satisfied, the approval authority can certify the specific Q-D Engine version with an appropriate identify mark. The certification of the Q-D Engine might accompany the publication of the changed or revised standard.

Q-D ENGINE DLL

A prototype Q-D Engine DLL has been developed and successfully tested with C++, VisualBasic, PowerBuilder, and Delphi host applications. The Q-D Engine has started and is running. For example a Q-D Calculator application has been developed using this DLL and VisualBasic. The development was accomplished using the concepts addressed in this paper. The prototype addresses hazard division 1.1 according to DoD 6055.9-STD. It computes the required separation distances for over-pressure and primary and secondary fragments given explosive, PES, and ES properties. The Q-D Engine has not been either validated or certified by the Department of Defense Explosives Safety Board (DDESB).

The prototype has also been tested with other more robust applications. The Defense Explosives Safety Management Suite (DESMS) Explosives Safety Siting Application (ESS) has been tested with the prototype. Modifications are being made to make ESS fully compatible with the Q-D Engine DLL. ArcView, a commercially available geographic information sytem (GIS), has been tested with the prototype.

CONCLUSION

Experience with DoD 6055.9-STD has clearly indicated that the standard can be economically automated and maintained. Until natural language processing technology advances, critical reading and analysis of the standard by a explosive safety technologist is required to develop the automation procedure. Flowchart and decision table technology has reduced cost and time requirements. This technology has also identified anomalies in the text of the standard which need to be resolved by the approval authority.

The formation of a DLL compatible with Microsoft Windows 95 or NT using a robust programming language has proved the Q-D Engine concept. The DLL can be used with specialized applications such as a Q-D Calculator, with commercial applications such as ArcView[15], and with general purpose siting applications such as ESS. Validation and certification of the Q-D Engine was suggested and emphasized. Each Service and the DDESB can accomplish this using a validation suite of PES-ES relationships that have been documented and approved using the existing approval documentation and processes.

Long term utilization of the Q-D Engine was discussed. The integration of the maintenance and enhancement of the standard with the automation of the standard was emphasized. Accomplishing this with current staff personnel is feasible with some additional training associated with flowcharting, decision tables, and rudimentary programming.

RECOMMENDATION

It is recommended that the DDESB take necessary steps to validate and certify the Q-D Engine. Furthermore, the respective Army, Navy, and Air Force approval authorities should participate in the development of separate Q-D Engines for each of their standards. They should also validate and certify the final development products.

[15] ArcView is a geographic information systems product of Environmental Systems Research Institute, Inc. Redlands, CA.