

PVP2009-77871

**ARTIFICIAL INTELLIGENCE (AI) TOOLS FOR DATA ACQUISITION AND
PROBABILITY RISK ANALYSIS OF NUCLEAR PIPING FAILURE DATABASES (*)**

Pedro V. Marcal

MPave Corp.
3606 Calico Ranch Gate
Julian, CA 92036 USA
marcalpv@cox.net

Jeffrey T. Fong

Mathematical & Computational Sciences Division
National Inst. of Standards & Technology (NIST)
Gaithersburg, MD 20899-8910 USA
fong@nist.gov

Nobuki Yamagata

Nihon ESI K.k., The Virtual Try-Out Company
Shinjuku Green Tower Bldg. 6-14-1 Nishi-Shinjuku
Shinjuku-ku Tokyo 160-0023 JAPAN
nym@esi.co.jp

ABSTRACT

Over the last thirty years, much research has been done on the development and application of in-service inspection (ISI) and failure event databases for pressure vessels and piping, as reported in two recent symposia: (1) ASME 2007 PVP Symposium (in honor of the late Dr. Spencer Bush), San Antonio, Texas, on "Engineering Safety, Applied Mechanics, and Nondestructive Evaluation (NDE)." (2) ASME 2008 PVP Symposium, Chicago, Illinois, on "Failure Prevention via Robust Design and Continuous NDE Monitoring." The two symposia concluded that those databases, if properly documented and maintained on a worldwide basis, could hold the key to the continued safe and profitable operation of numerous aging nuclear power or petro-chemical processing plants. During the 2008 symposium, four uncertainty categories associated with causing uncertainty in fatigue life estimates

were identified, namely, (1) Uncertainty-1 in failure event databases, (2) Uncertainty-2 in NDE databases, (3) Uncertainty-3 in material property databases, and (4) Uncertainty-M in crack-growth and damage modeling. In this paper, which is one of a series of four to address all those four uncertainty categories, we introduce an automatic natural language abstracting and processing (ANLAP) tool to address Uncertainty-1. Three examples are presented and discussed.

Keywords: Aging structures; ANLAP; artificial intelligence; computational linguistics; conceptual dependency; crack propagation; database software; Dataplot; design of experiments; failure event database; fatigue; flaw detection; information extraction; in-service inspection; life extension; material property database; mathematical modeling; natural language processing; NDE database; nondestructive evaluation; nuclear power plants; nuclear safety; petro-chemical plants; Python; risk-informed analysis; risk-informed engineering economics; SQL; statistical data analysis; uncertainty analysis.

(*) Contribution of the U.S. National Institute of Standards and Technology. Not subject to copyright.

Disclaimer: The views expressed in this paper are strictly those of the authors and do not necessarily reflect those of their affiliated institutions. The mention of names of all commercial vendors and their products is intended to illustrate the capabilities of existing products, and should not be construed as endorsement by the authors or their affiliated institutions.

I. INTRODUCTION

Over the last thirty years following the 1979 nuclear accident at Three Mile Island [1]¹, much research has been done on the development and application of in-service inspection (ISI), failure event databases, and risk-informed fatigue modeling of defect management for pressure vessels and piping [2-37]. The good news is reflected in Figure 1, where Pietrangelo

[38, 39] showed in 2008 that the U.S. nuclear power plant average capacity factor increased from 66 % in 1990 to an astonishingly high 91.8 % in 2007, resulting in a huge 39 % increase in the annual energy production of 104 power reactors over a span of 17 years -- from 580 to 807 billion kilowatt-hours per year -- without building since 1979 a single new commercial reactor. On the other hand, some not-so-good news was hinted in 2004 by Teather [40], who reported that

" . . . The fleet of reactors in the U.S. is aging, however, and many are now applying for licenses to extend their lives.

"By the end of this year (2004), a third of the existing plants, built to last for 40 years, will have applied for licenses to continue operating for another 20."

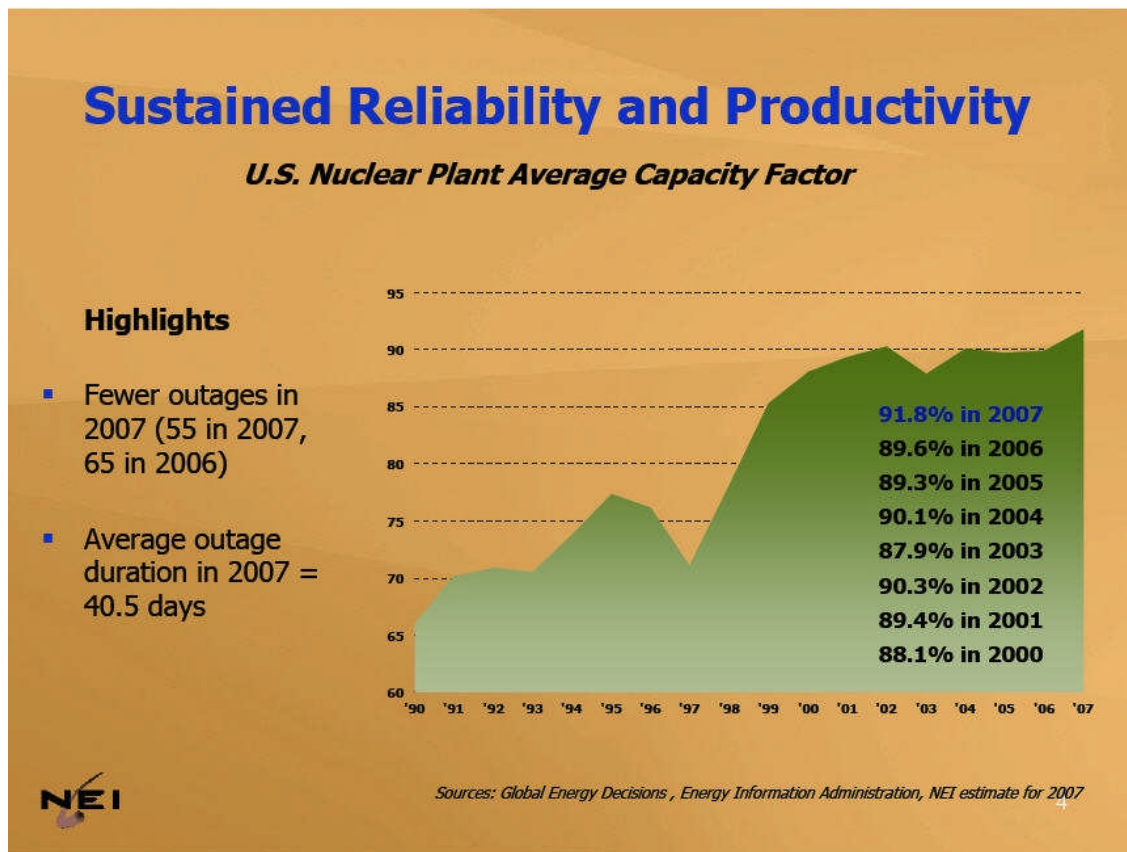


Figure 1. U.S. Nuclear Plant Average Capacity Factor (1990 - 2007), after Pietrangelo [39].

¹Figures in square brackets denote references listed at the end of this paper.

I. INTRODUCTION (CONT'D)

The problem of maintaining and safely operating an aging equipment or structure is not unique to a nuclear power plant, as reported recently in a New York Times article by Cooper [41]:

" . . . More than a quarter of the nation's bridges are structurally deficient or functionally obsolete. Leaky pipes lose an estimated seven billion gallons of clean drinking water every day. And aging sewage systems send billions of gallons of untreated wastewater cascading into the nation's waterways each year."

Getting back to the highly researched and government-regulated problem of permitting a 40-year-old nuclear power plant to operate for another 20 years, we show in Figure 2 a specific result of a simulation experiment on the failure

probabilities predicted by a computer code named "PC-PRAISE" for the surge-line elbow of a Combustion Engineering plant in terms of probabilities of crack initiation and through-wall cracks as a function of time (after a 2000 NUREG/CR6674 report by Khaleel, Simonen, Phan, Harris, and Dedhia [17]). In that report, the authors [17, p. 9.7] stated:

" . . . It is seen that cracks initiate rather early in the plant life. There is about a 50-percent probability of initiating a fatigue crack after only 10 years of operation.

" . . . Over this 10 years, about 50 percent of these initiated cracks are predicted to grow to become leaking (through-wall) cracks.

" . . . The frequency of through-wall cracks (lower curve) increases significantly over this 10 years and then remains relatively constant over the remainder of the 60-year plant life."

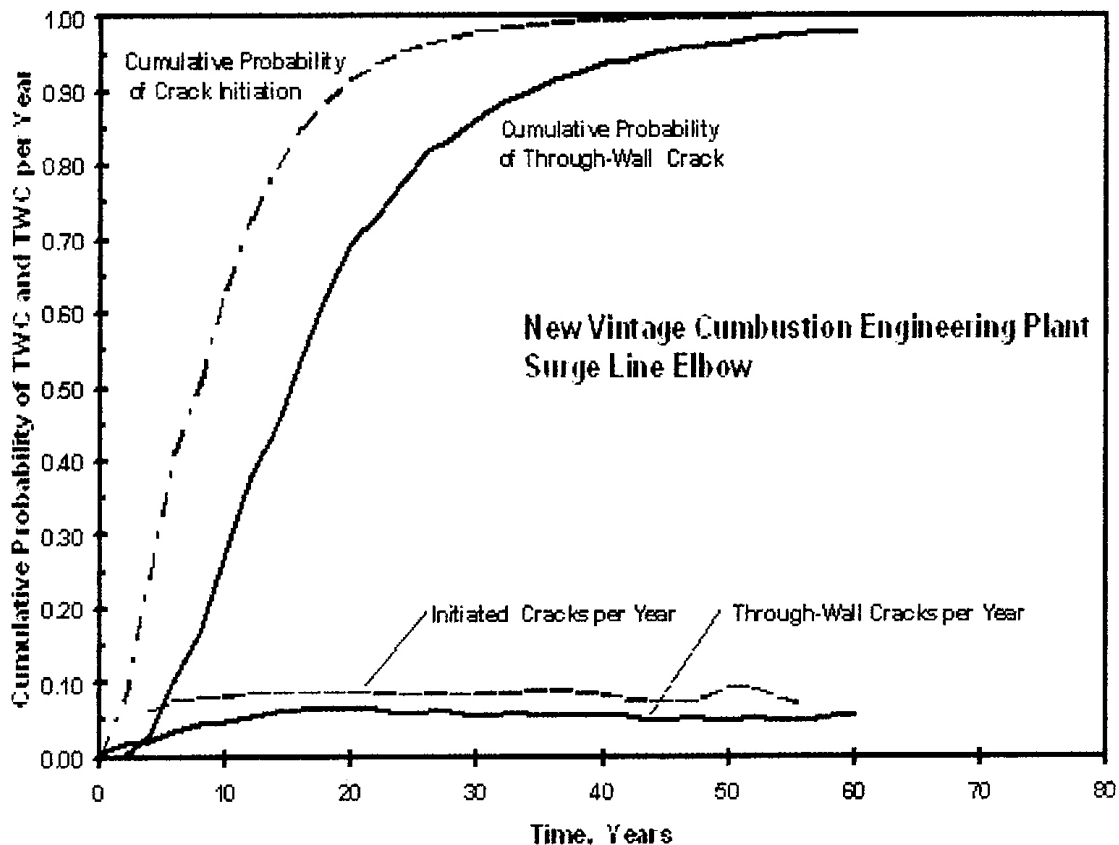


Figure 2. Calculated Probabilities of Crack Initiation and Through-Wall Crack for the Surge-Line Elbow of the Newer Vintage Combustion Engineering Plant (after Khaleel, Simonen, Phan, Harris, and Dedhia [17]).

I. INTRODUCTION (CONT'D)

In that same report [17, p. 10.1], the authors concluded that,

" . . . The calculations gave a wide range of failure probabilities for the selected components, with some components having end-of-life probabilities of through-wall cracks of nearly 100 percent and others with probabilities of less than 10^{-6} .

" . . . It is recognized that there are uncertainties in these calculated failure probabilities and core damage frequencies. Sources of the uncertainties come from assumptions made in the fracture mechanics and probabilistic risk analysis models themselves and from the inputs to the models.

" . . . In particular, the inputs for cyclic stresses were based on design-basis data, which could differ from the stresses occurring during the actual plant operation."

On the role of failure data in plant aging management, Chockie and Gregor [29] presented in 2008 an assessment and a more rational approach to the complex problem of failure event and in-service inspection data collection, analysis, interpretation and life extension decision making:

" . . . After almost forty years there is a vast amount of data on operational performance of nuclear plants and their systems, structures, and components (SSCs).

" . . . By understanding some of the key **limitations of the data sources**, more effective use can be made of the information gained from the analysis of the data.

" . . . This operational performance data and the resulting information, in combination with an **economic assessment of the benefits and costs** of various options, is essential for effective aging management and life extension decisions of the nuclear power plants."

(boldface furnished by authors of this paper.)

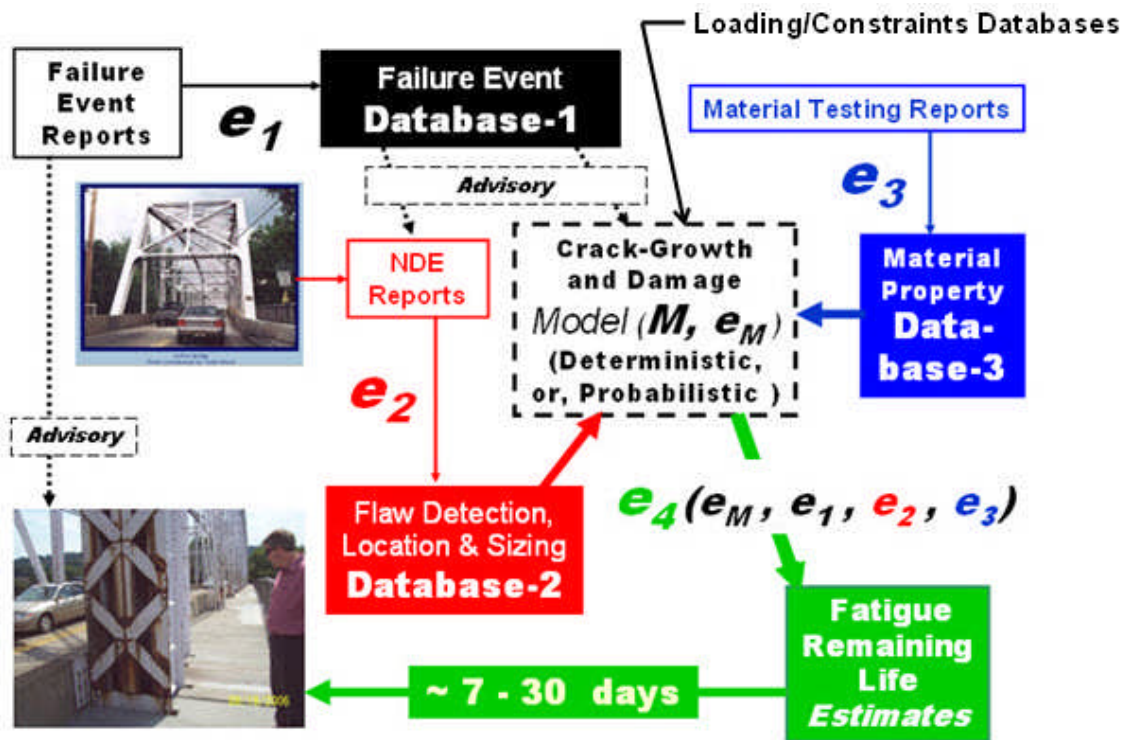


Figure 3. A conceptual representation (after Fong and Marcal [30] and Fong, Ranson, Vachon, and Marcal [33]) of the information flow plus the uncertainties and potential errors associated with and inherent in Loading/ Constraints databases, Failure Event Database-1 (*Uncertainty-1, or, e_1*), Flaw Detection, Location & Sizing Database-2 (*Uncertainty-2, or, e_2*), Material Property Database-3 (*Uncertainty-3, or, e_3*), Deterministic or Probabilistic Damage Models (*Uncertainty-M, or, e_M*) and Remaining Life Estimates (*Uncertainty-4, e_4*). Photo at the upper left corner is from the 70-year-old Jonathan Hulton Bridge, built in 1909, of Pittsburgh, PA, courtesy of reference [42]. Photo at the lower left corner was taken by the first author (Fong) during a site visit to the bridge in 2006.

I. INTRODUCTION (CONT'D)

As shown in Figure 3 (after Fong and Marcal [30] and Fong, Ranson, Vachon, and Marcal [33]), the complex flow of information, both qualitative and quantitative, from failure event reports, NDE reports, material testing reports, computer modeling simulations, and remaining life estimates for decision making is associated with all sorts of uncertainties and potential errors due to data collection, interpretation, analysis, and modeling. For our purposes here, we identify four distinct categories of uncertainties and errors as the main causes of uncertainty in remaining life estimation, namely,

- e_1 - associated with *failure event database-1*,
- e_2 - associated with *flaw detection/location/sizing-2*,
- e_3 - associated with *material property database-3*,
- e_M - associated with *crack-growth/damage modeling*,

such that the remaining life estimates will have an uncertainty denoted by

- e_4 - associated with *remaining life estimates*,

where conceptually speaking, the model result uncertainty, e_4 , may be an implicit or explicit function of the source uncertainties, e_1 , e_2 , e_3 , e_M , etc. (incl. loading/constraints).

Since in engineering, the only way one can verify a model result uncertainty, e_4 , is to check it against e_1 , the failure event statistics, we propose to use artificial intelligence (AI) tools linked to statistical analysis codes to address e_1 . We also address all the other source uncertainties, e_2 , e_3 , and e_M in a series of 4 papers as shown below with a summary of results on e_4 to be given in the 4th part of the series [45]:

- e_1 - Fong-Marcal-Yamagata [This paper].
- e_2 - Fong-Marcal-Hedden-Chao-Lam [43].
- e_3 - Fong-and-Marcal [44].
- e_M - Fong-deWit-Marcal-Filliben-Heckert-Gosselin [45].

In Sect. II, we present a brief review of the application of artificial intelligence (AI) tools to computational linguistics. In Sect. III, we present our AI tool named ANLAP 1.0 [46] using a natural language toolkit, NLTK, first developed by Loper and Bird [47]. In Sect. IV, we present Example-1, a simple application of ANLAP to illustrate the basic concepts. In Sections V and VI, we present two real-life application examples using a U.S. Department of Energy Operating Experience Report [14] and a public-domain statistical data analysis package named Dataplot [48]. Significance of our results, some concluding remarks, and a list of references are given in Sections VII, VIII, and IX, respectively. Two key ANLAP and Dataplot codes, "mpact_concept_nleps.py" and "pedro8.dp," are given in Appendices A & B, respectively.

II. ARTIFICIAL INTELLIGENCE (AI) TOOLS FOR COMPUTATIONAL LINGUISTICS - A BRIEF REVIEW

A common problem associated with data collection in failure event databases, NDE databases, and material property databases, is the proliferation of technical reports written in natural languages. Beginning in the early 1970s, Schank [49, 50] used artificial intelligence (AI) tools to formulate a computer understanding of the natural language. The basic idea in Schank's "conceptual dependency (CD)" representation theory, [50] required the following three assumptions to be valid:

(1) Since it is true that people can understand natural language, it is assumed that it *should be* possible to imitate the human understanding process on a computer, if it is possible to state those processes explicitly.

(2) It is also assumed that there exists a *conceptual base* into which utterances in natural language are mapped during understanding.

(3) The conceptual base is *well-defined enough* such that an initial input into the conceptual base can make possible the prediction the kind of conceptual information that is likely to follow the initial input.

Needless to say, computer speed and memory in the 1970s were not good enough to help Schank and his colleagues develop a tool to parse a technical report such as reference [14]. As computer speed, memory and cost improved exponentially during the last thirty years, two new developments in software languages, SQL [51-53] and Python [54-56], made it possible for database technology to take advantage of the CD theory due to Schank [49, 50]. Since 2002, a natural language toolkit, NLTK, due to Loper and his co-workers [47, 57-58] have become available to facilitate the development of our Automatic Natural Language Abstracting and Processing (ANLAP) tool as described in the next section and its manual [46].

III. DESCRIPTION OF ANLAP

ANLAP is based on a semantic parsing of a natural language input. The method is based on a revision of the original notion of conceptual dependency (CD) as proposed by Schank, et al. [49, 50]. In developing ANLAP, we made extensive use of the Natural Language Tool Kit (NLTK) due to Loper, et al. [47, 57-58].

The process begins with a context free parsing of the input text. The parsing of the input text is accomplished with a statistical parser, based on training data from the Brown and Penn State Treebank corpora [58]. The parsed text is converted to a "chunked" text by a procedure known as a stochastic Parts Parser for noun phrases (see Church [59]), where the parsed

III. DESCRIPTION OF ANLAP (CONT'D)

text is subdivided into Noun and Verb Phrases.

The parsed text is then converted into a semantic one based on Conceptual Dependency. This is accomplished with the aid of a specially constructed dictionary. The construct of the dictionary is illustrated by the example that is discussed below.

The Abstracting and Processing part of the program performs the following three functions :

1. A natural language prompting of user for input.
2. Processing by an expert system based on user's natural language input of rules.
3. Abstracting of unstructured data as demonstrated in examples in this paper.

Here we focus exclusively on the last function above. The Abstracting of unstructured data is performed by first parsing the input text. A question is then formulated by the user who defines the headings to be used in forming a table by searching the unstructured data.

This question is also parsed.

The defined headings in both their keyword and semantic forms are used to search the data in both the keyword and semantic forms.

The semantics is used to disambiguate the presence of the heading words when it is not associated with data.

It was also necessary to string the noun phrases ([NP]) together to define a piece of data as illustrated in the following snippet of text :-

12 percent of the root causes as personnel errors

Chunked

[12 percent][NP] of the [root causes][NP] as [personnel errors][NP].

The semantic-free parsing is now converted into a conceptual dependency form. Its convenient to write this in first order format as shown below in Table 1:

Table 1

Conceptual Dependency

Dictionary phase (look up and disambiguate)

Num	Chunk	Word	type	concept	concept1	concept2	concept3	concept4	sense
3	2	val_5		PP	PTRANS	NUM	NUM	NUM	NUM -1
		val_5 == 12							
4	2	percent		PP	PTRANS	PERCENT	UNIT_OF_MEASURE	COUNT	COMPUTE -1
5		of		PA	ACT	QUALIFIER	INGEST		-1
6	3	the		PA	NUM	THENCE			0
7	3	root		PP	VEGETABLE	BEGINNING	BEGINNING	POINT	LOCATION 1
8	3	cause		PP	MTRANS	CAUSE	BEGINNING	EVENT	0
9	4	as		PA	MTRANS	EQUALLY	SAME	PART	0
10	4	personnel		PP	HUMAN	FORCE	FORCE	ORGANIZATION	INGEST 0
11	4	error		PP	MTRANS	MISTAKE	DEED	TRANSMIT	0

Note-1: The labels "ATRANS," "PTRANS" and "MTRANS" are top level concepts dealing with property, physical and mental concepts, respectively.

Note-2: ACT, PP, and PA are Schank terminology in CD, where ACT is the Actor-Verb, PP is the picture painter-Noun, and PA's are the Picture Aiders, i.e. ADV. ADJ and all the other Prepositions, conjunctions etc.

III. DESCRIPTION OF ANLAP (CONT'D)

The conversion to CD form is now evident from our chunking operations.

We can now write a first order relation for each Chunk as well as the relation between the sequence of chunks as shown in Table 2. We call this Conceptual Analysis and a browse through the Analysis will give a semantic meaning of the text.

Table 2

Conceptual Analysis one relation per Chunk

```
percent <=> val_5
PP <=> PP
PTRANS COMPUTE <=> PTRANS NUM
<=> 4 3
link within the noun phrase (Chunk)
```

```
percent T<=POSS_BY cause
PP T<=POSS_BY PP
PTRANS COMPUTE T<=POSS_BY MTRANS ACT
T<=POSS_BY 4 8
Possession link between the percent and the root cause
phrase
```

```
cause T<=AKA error
PP T<=AKA PP
MTRANS ACT T<=AKA MTRANS TRANSMIT_NOT
T<=AKA 8 11
Similarity link between the root cause and the personnel
error phrase
```

```
error <=> personnel
PP <=> PP
MTRANS TRANSMIT_NOT <=> HUMAN INGEST
<=> 11 10
link within the phrase
```

Search for the heading 'personnel' consists of searching through the conceptual analysis to establish the above sequence of links.

It is worth noting that the performance of ANLAP is greatly enhanced when we decide in advance that we are only interested in looking for very specific kinds of information in texts such as a failure event report, an NDE inspection report, or a material property testing report. With the user's help, ANLAP will first convert the unstructured data of natural language sentences into the structured data of a table with user-defined headings. This method of getting meaning from text is called "Information Extraction" [58, Section 7.1], and the table output format opens the door for us to link up with powerful tools such as SQL [51-53] and Dataplot [48, 60].

IV. ANLAP APPLICATION EXAMPLE - 1

We begin a series of examples to illustrate the application of our code, ANLAP, with a simple example by constructing an English text file, c:\CD_data\test11.txt, as shown in Figure 4.

To run ANLAP, we need to install Python in a Windows-based computer. We also need to create a subdirectory named "c:\CD_data" to manage the input and output files of this exercise. In that subdirectory, we will copy an executable of ANLAP named "mpact_concept_nleps.py" and a Python debugger named "idle.bat". A listing of the subdirectory, "c:\CD_data" will show three files as follows:

```
c:\CD_data\idle.bat
c:\CD_data\mpact_concept_nleps.py
c:\CD_data\test11.txt
```

A total of five steps will be taken to get our ANLAP result:

Step 1: Verify the content of file, test11.txt (Figure 4).

Step 2: Run ANLAP by activating the Python debugger, opening the ANLAP executable, and running the module to view a screen as shown in Figure 5. Enter the name of the input file and the headings of a table-formatted output file. Click OK to begin parsing.

Step 3: View the results of ANLAP parsing (Figure 6).

Step 4: View the name of the output file of ANLAP (Figure 7).

Step 5: Verify the content of the output file (Figure 8).

We observe that ANLAP has conducted an information extraction task on the test file, c:\CD_data\test11.txt, and return a table of data with four headings as specified by the user.

c:\CD_data\test11.txt

In a technical report on hardness measurements of metals, we found the following items of special interest:

"In the year 1999, the measurement for the hardness of steel at a temperature of 40 degrees C was 101.

"In the year 2001, the measurement for the hardness of copper at a temperature of 45 degrees C was 45.

"In the year 2001, the measurement for the hardness of gold at a temperature of 30 degrees C was 20."

We note that the metals were steel, copper and gold.

Figure 4. ANLAP Step 1: User's input of an English text as a test file named c:\CD_data\test11.txt.

IV. ANLAP APPLICATION EXAMPLE - 1 (CONT'D)

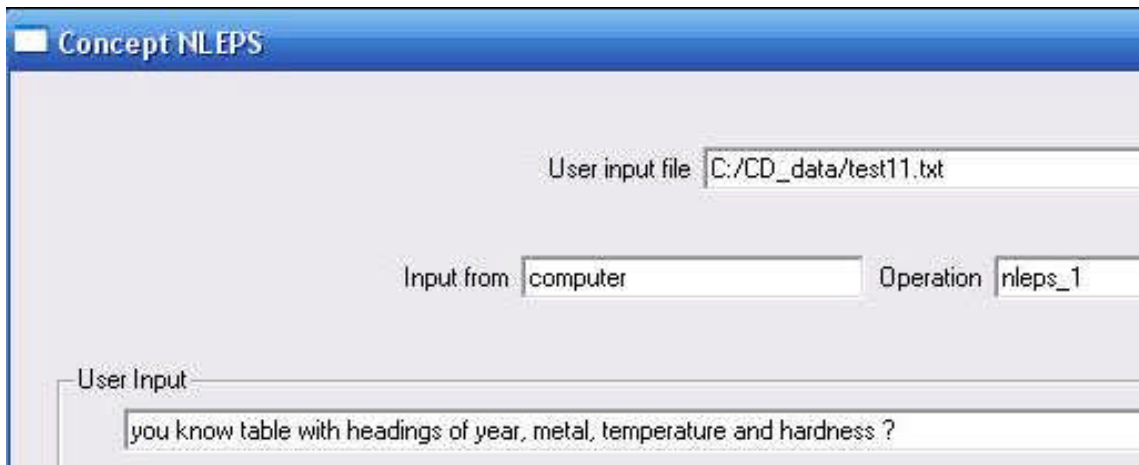


Figure 5. ANLAP Step 2: User specifies an input filename and headings of a table-formatted output.

```

Num Chunk Word type concepter concepter1 concepter2 concepter3 concepter4
0 in PA POSS-BY TYPE -1
1 0 the PA NUM THENCE 0
2 0 year PP MTRANS YEAR YEAR TIME_PERIOD MEASURE 0
3 0 val_0 PP PTRANS NUM NUM NUM NUM -1
val_0 == 1999
4 , PA NUM SEP " " " -1
5 1 the PA NUM THENCE 0
6 1 measurement PP MTRANS MEASURE ACTION ACT 0
7 for PA MTRANS SITUATION-CT_p -1
8 2 the PA NUM THENCE 0
9 2 hardness PP PTRANS SEVERITY DISPOSITION NATURE PROPERTY 4
10 of PA ACT QUALIFIER -1
11 3 steel PP MINERAL STEEL MIXTURE SUBSTANCE PROPERTY 0
12 at PA LOC-IN PLACE PRESENCE FEM 0
13 4 a PA NUM 1 " " " -1
14 4 temperature PP MTRANS TEMPERATURE FUNDAMENTAL_QUANTITY MEASURE PROPERTY 0
15 of PA CONT-BY CONTAIN LOCATION -1
16 5 val_1 PP PTRANS NUM NUM NUM NUM -1
val_1 == 40
17 5 degree PP MTRANS DEGREE DEGREE PROPERTY PROPERTY 0
18 6 is ACT PTRANS EQUAL <-o-<-R SELF CD -1
19 7 val_2 PP PTRANS NUM NUM NUM NUM -1
val_2 == 101
parsing matched followed by match, primMatch match_nums
val_0 T<=SEQ measurement
PP T<=SEQ PP
PTRANS NUM T<=SEQ MTRANS ACT (Additional lines omitted for brevity.)

```

Figure 6. ANLAP Step 3: Conceptual Dependency parsing of User's input file, c:\CD_data\test11.txt.

IV. ANLAP APPLICATION EXAMPLE - 1 (CONT'D)

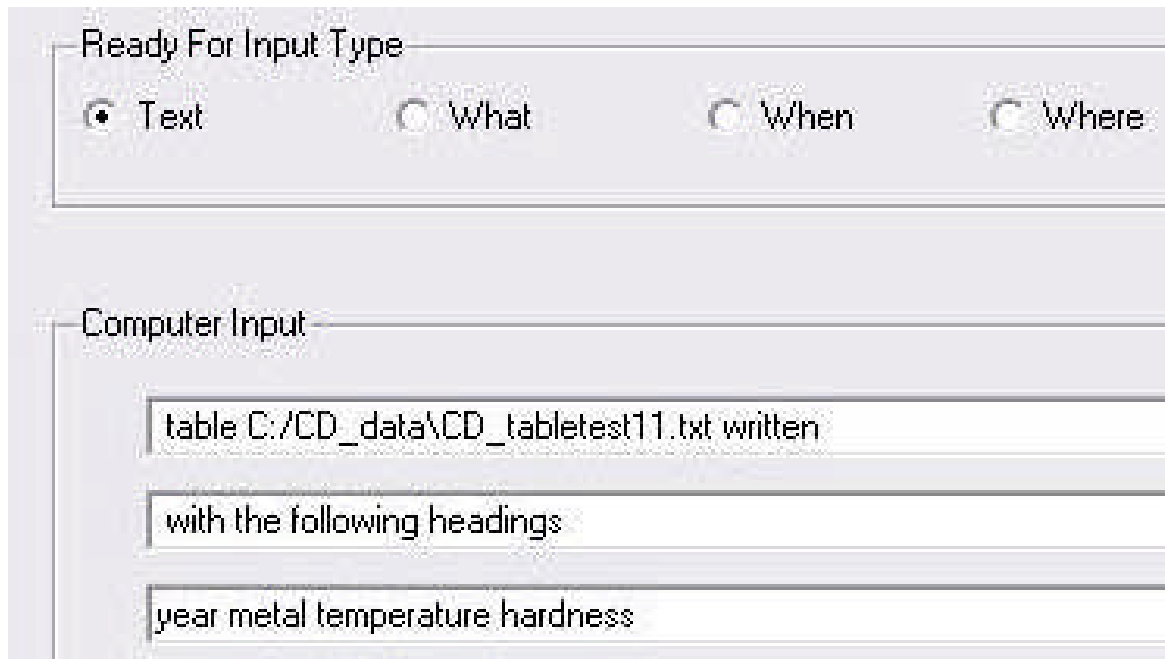


Figure 7. ANLAP Step 4: Display of ANLAP result written in c:\CD_data\CD_tabletest11.txt.

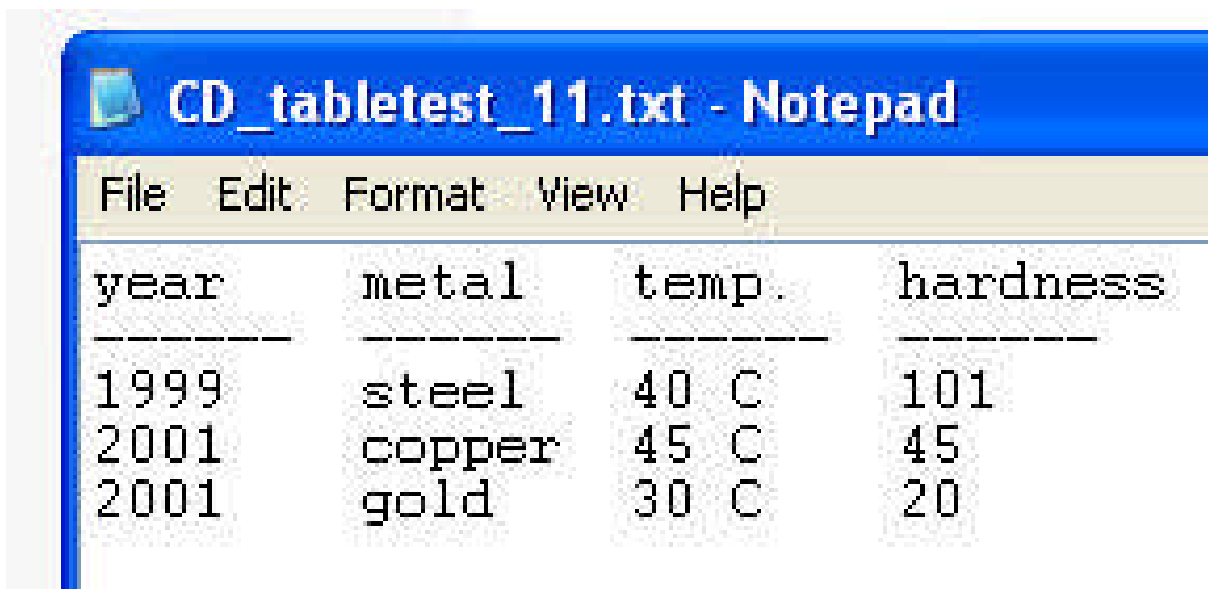


Figure 8. ANLAP Step 5: Print Screen of ANLAP output file, c:\CD_data\CD_tabletest11.txt.

V. ANLAP APPLICATION EXAMPLE - 2

Our next example of applying ANLAP is an exercise in having a computer read a real-life failure event report, extract data from that report, and do some analysis of the extracted data.

For this exercise, we choose to work with a weekly report of the U.S. Department of Energy - Office of Nuclear and Facility Safety entitled "Operating Experience Weekly Summary 98-40, October 2 through October 8, 1998," [14].

A copy of the report cover page is shown in Figure 9, and that of its table of content page in Figure 10. An edited version of the content of Section 1 of that report is given in Figure 11, where we constructed a test file named "c:\CD_data\test11.txt" from the content of that section by suppressing a few paragraphs and adding a fictitious paragraph, shown in red, in order to test the analysis capability of ANLAP.

As described in the previous Example-1, we will first follow the five ANLAP steps to get a table-formatted output file:

Step 1: Verify the content of file, test11.txt (Figure 11).

Step 2: Run ANLAP by activating the Python debugger, opening the ANLAP executable, and running the module to view a screen as shown in Figure 12. Enter the name of the input file and the headings of a table-formatted output file. Click OK to begin parsing.

Step 3: View the results of ANLAP parsing.

Step 4: View the name of the ANLAP output file (Figure 13).

Step 5: Verify the content of the output file (Figure 14).

At this point, we enhance the capability of the Python-based ANLAP by adding a link with a public-domain statistical data analysis and graphics package named Dataplot [48, 60]. More specifically, we first install Dataplot in our computer, then add a second code named "pedro8.dp" in the subdirectory, c:\CD_data, and finally modify ANLAP to have a revised executable code capable of running "pedro8.dp" (see Appendix A for the code modification, and a listing of "pedro8.dp" in Appendix B). The result is given in Figure 15, where we suppress the fictitious data in favor of the real DOE data.

Step 6: Verify output c:\CD_data\fong200.pdf (Figure 15).

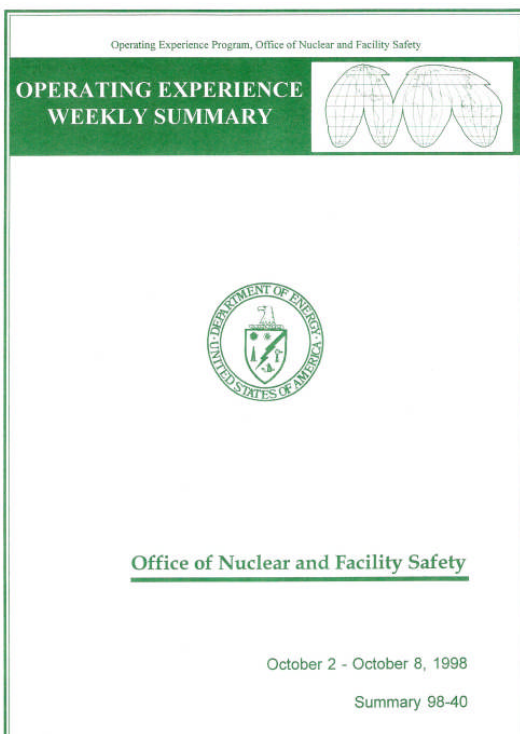


Figure 9. Cover Page of a 1998 DOE weekly report on operating experience of nuclear facilities [14].

Operating Experience Weekly Summary 98-40 October 2 through October 8, 1998

Table of Contents

EVENTS	1
1. SPREAD OF CONTAMINATION AT HANFORD	1
2. RESPIRATOR PROTECTION COUPLING NUTS FAIL	4
3. 480-VOLT ELECTRICAL CABLE SEVERED DURING EXCAVATION ACTIVITY	5
4. CONTAMINATED WASTE DRUM AND WORK AREA AT ROCKY FLATS	8
FINAL REPORTS	11
1. UNPLANNED REACTIVITY ADDITION DURING REACTOR STARTUP	11
2. WORKER MAKES RADIOLOGICAL ENTRIES WITHOUT DOSIMETRY	12
OEAF ACTIVITY	14
DATA ANALYSIS FORUM	14

Figure 10. Table of Content Page of a 1998 DOE weekly report on operating experience [14].

V. ANLAP APPLICATION EXAMPLE - 2 (CONT'D)

c:\CD_data\test11.txt

1. SPREAD OF CONTAMINATION AT HANFORD

On September 28, 1998, at the Hanford site, a health physics technician was performing a routine surveillance and discovered contamination levels of as much as 1,000,000 dpm of predominately strontium-90 in a mobile trailer office/kitchen. Electricians used the trailer as an office and a break area. The technician determined that several items in the trailer were contaminated, including a cutting board, a countertop, a bench-seat, the floor, garbage cans and their contents, door handles, toilet handles, and food wrappers. The technician established a radiological control area and notified the facility manager. Health physics technicians performed additional surveys and determined that several trash dumpsters and an iron workers' shop were also contaminated. They also determined that the trash from one of the dumpsters was collected and dumped at the city of Richland landfill before the technicians had a chance to post it as contaminated. They subsequently determined that two of Hanford's dump trucks and the trash in them are contaminated; the trucks and their contents have been placed under radiological control. Technicians also detected contamination at the landfill and implemented radiological material controls until they can develop and implement removal plans agreed upon with state and local officials. (ORPS Reports RL-PHMC-FSS-1998-0021 and RL-PHMC-WESF-1998-0012)

(The next paragraph is omitted for brevity,)

NFS has reported contamination events in several Weekly Summaries. Some examples follow:

(The next 3 paragraphs are omitted for brevity.)

*(The next 4th paragraph is edited to test the parsing capability of a proprietary computer linguistic software package entitled "**A**utomated **N**atural **L**anguage **A**bstractor and **P**rocessor (**ANLAP**, v.1.0).")*

OEAF engineers searched the ORPS database for events that involved a loss of control of radioactive material or spread of contamination from January 1990 to October 1998 and found 3,494 occurrences. Managers reported 46 percent of the root causes as management problems and 17 percent as radiological/hazardous material problems. Managers also reported 12 percent of the root causes as personnel errors, 11 percent as equipment/material problems, and 14 percent as other.

*(The following **fictitious** paragraph was added to test the parsing capability of ANLAP.)*

A follow-up search by OEAF of the ORPS database for the previous 9-year period (from January 1981 to December 1989) found 2,898 occurrences. Managers reported 34 percent of the root causes as management problems and 27 percent as radiological/hazardous material problems. Managers also reported 14 percent of the root causes as personnel errors, 15 percent as equipment/material problems, and 10 percent as other.

[Source: "DOE Operating Experience Weekly Summary Report 98-40 (Oct. 2 through Oct. 8, 1998)", pp. 1-2, which is available, with word search capability, via the internet at http://tis.eh.doe.gov/web/oeaf/oe_weekly/oe_weekly.html]

Figure 11. ANLAP Step 1: User's Input of an edited page of a 1998 DOE weekly report on operating experience [14].

V. ANLAP APPLICATION EXAMPLE - 2 (CONT'D)

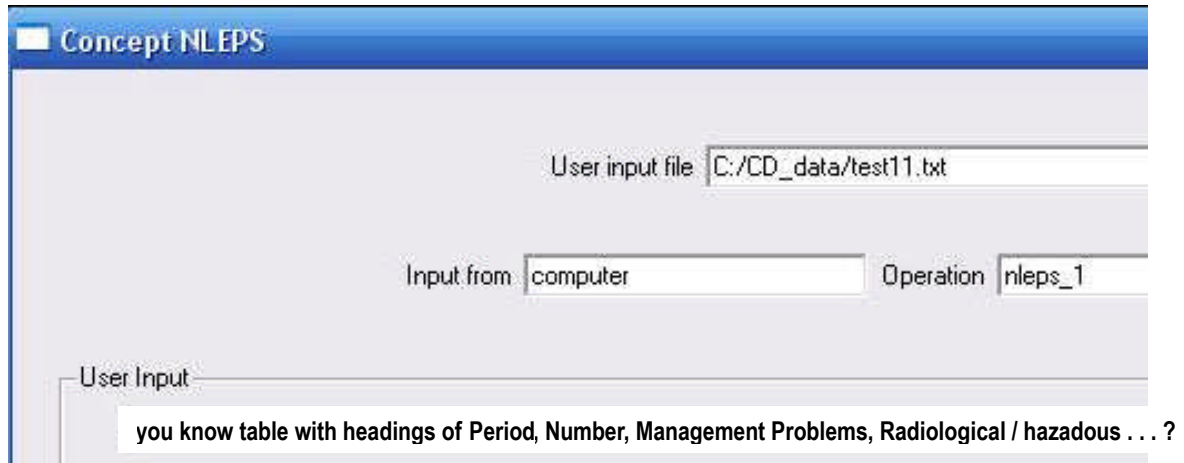


Figure 12. ANLAP Step 2: User specifies an input filename and headings of a table-formatted output.

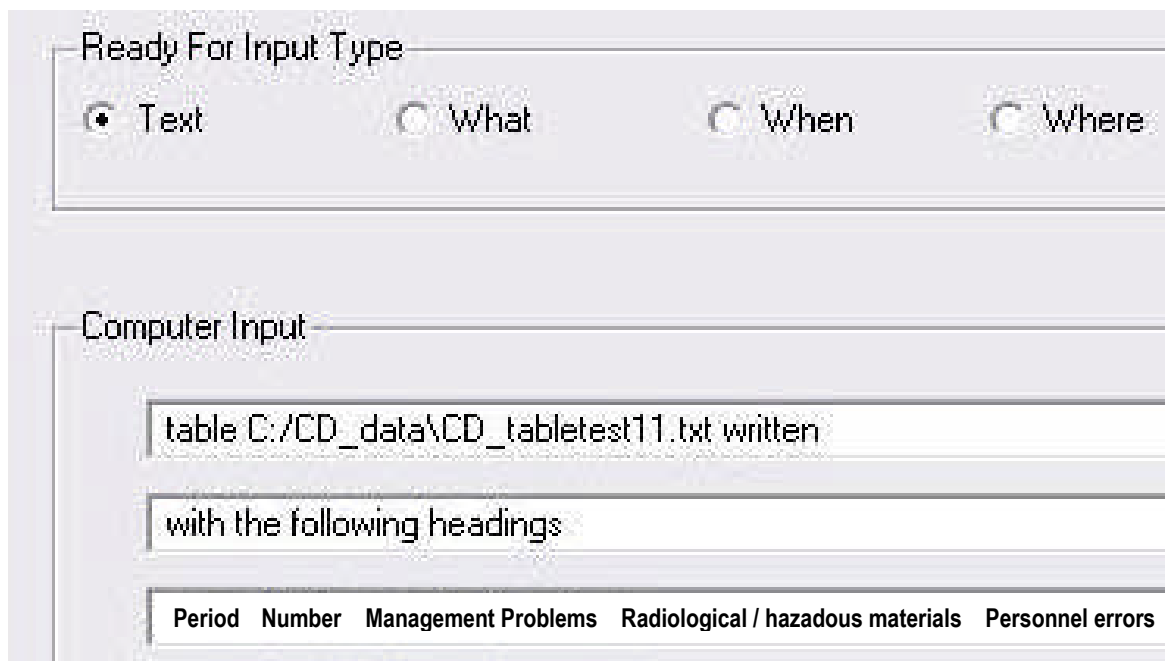


Figure 13. ANLAP Step 4: Display of ANLAP result written in c:\CD_data\CD_tabletest11.txt.

V. ANLAP APPLICATION EXAMPLE - 2 (CONT'D)

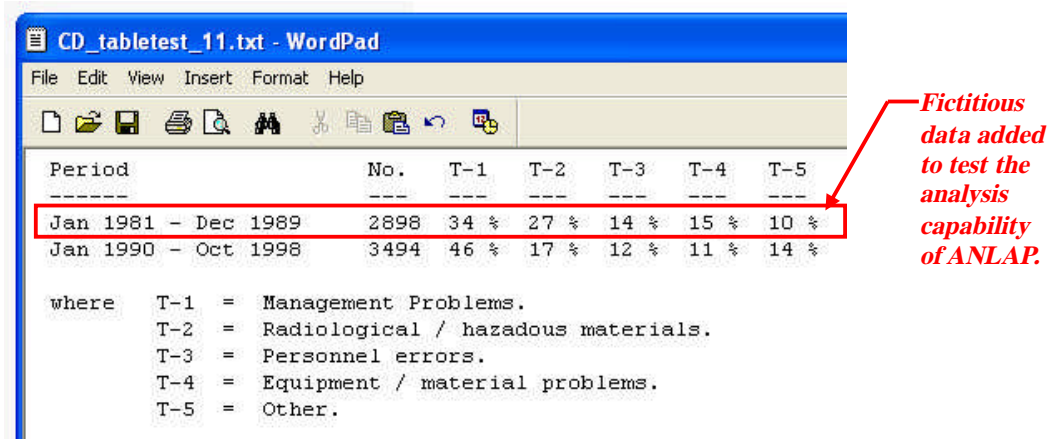


Figure 14. ANLAP Step 5: Printout of the first ANLAP output file, c:\CD_data\CD_tabletest11.txt.

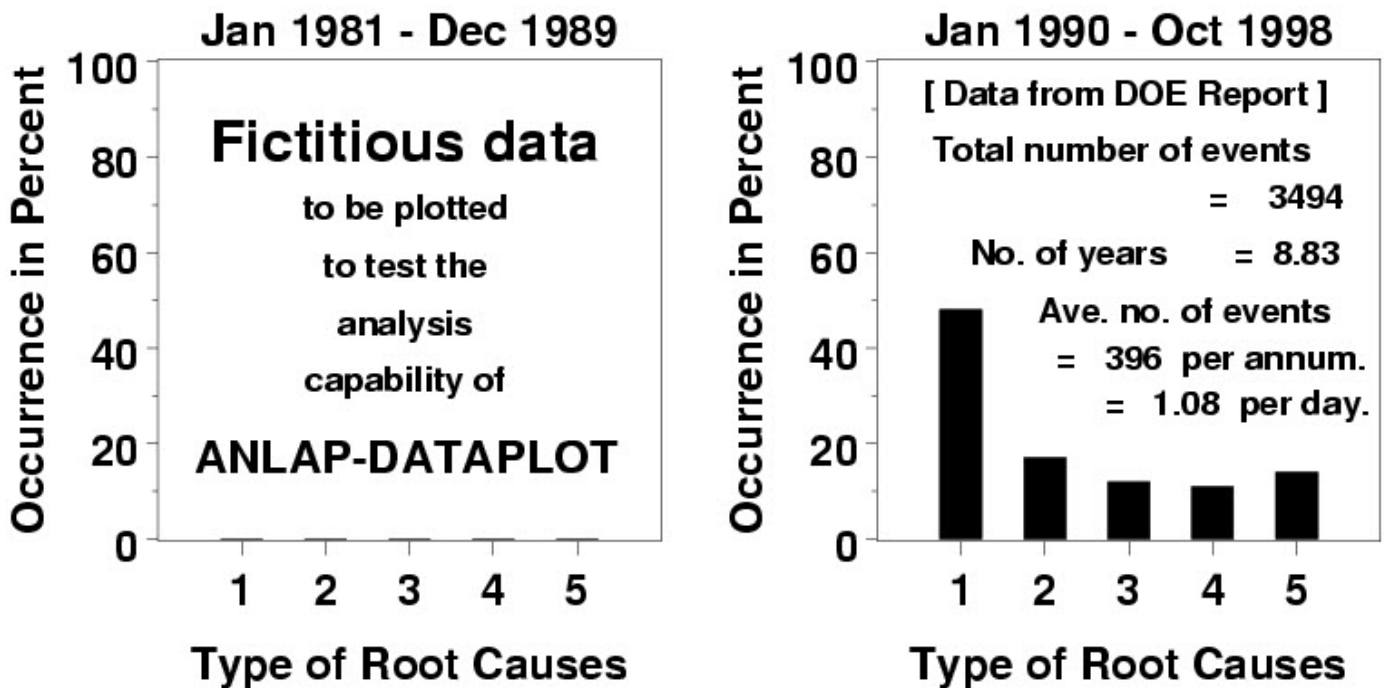


Figure 15. ANLAP Step 6: Printout of the second ANLAP output file, c:\CD_data\fong200.pdf, using "pedro8.dp."

V. ANLAP APPLICATION EXAMPLE - 2 (CONT'D)

A closer look at Figure 15 shows that it is not of report-quality, because there is neither an explanation of the five types of root causes nor an identification of the source of the so-called "data from DOE report." To remedy this situation, we introduce

Step 6a: We modify the Dataplot code, "pedro8.dp," by adding annotation codes for a legend of the five types of root causes and a heading identifying the source of the DOE data. The result is given in Figure 16, where we replace the ANLAP call for "pedro8.dp" with "pedro9.dp" with the creation of a third ANLAP output file, c:\CD_data\fong201.pdf. For brevity, we list all Dataplot codes other than "pedro8.dp" in the website, <http://math.nist.gov/mcsd/software.html>.

To add a second report-quality plot such as a colored pie chart with appropriate annotation for the five types of root causes, we introduce replace "pedro8.dp" with "pedro9pie.dp" to obtain the fourth ANLAP output file as shown below:

Step 6b: Verify output c:\CD_data\fong202.pdf (Figure 17).

So far, we have only one set of real data to work with, i.e., the distribution of root causes for spread of contamination at Hanford covering the period of Jan. 1990 - Oct. 1998. In the next set of two steps, we introduce the analysis capability of the ANLAP-Dataplot link by adding a fictitious set of data as shown in Figure 14:

Step 7a: Working with two sets of data, one fictitious and one real, we obtain a comparative two-bar-chart representation (Figure 18) by replace "pedro8.dp" with "pedro10.dp." Verify the fifth output file, c:\CD_data\fong203.pdf.

Step 7b: Again working with two sets of data, we obtain a 4-plot representation of raw data, box plot, mean plot, and standard of deviation plot (Figure 19) by replacing "pedro8.dp" with "pedro11.dp." Verify the sixth output file, fong204.pdf, in the working directory, c:\CD_data, after ANLAP is modified and re-run.

U. S. Dept. of Energy (DOE) Operating Experience Summary Report 98-40

http://tis.eh.doe.gov/web/oeaf/oe_weekly/oe_weekly.html.

Distribution of Root Causes for Spread of Contamination at Hanford

Period Covered: Jan. 1990 - Oct. 1998

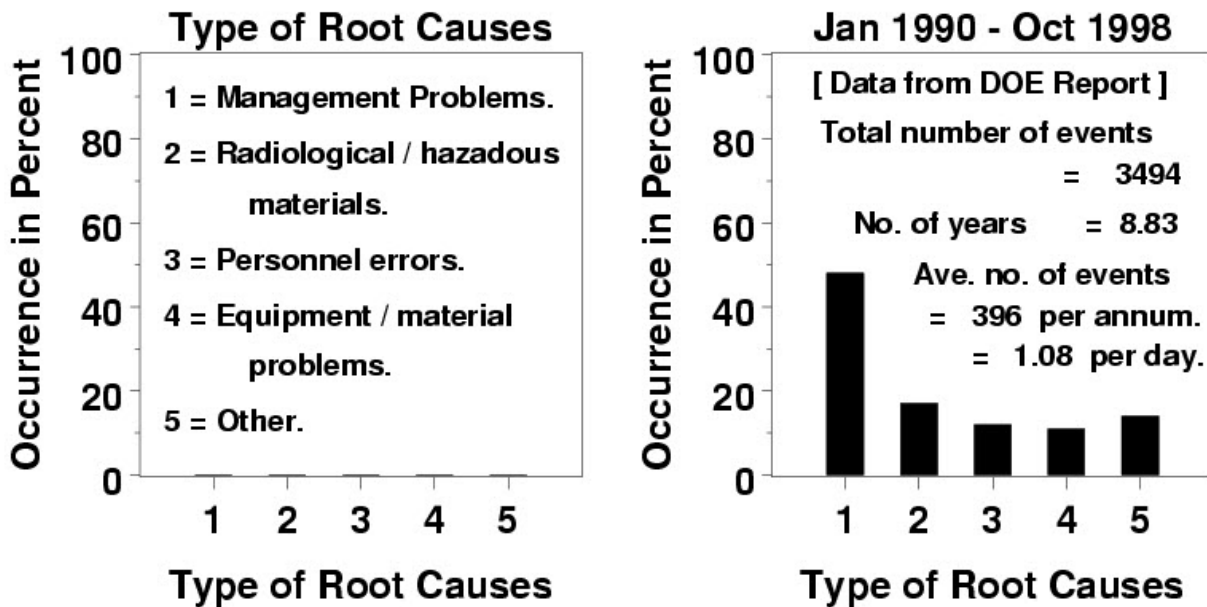


Figure 16. ANLAP Step 6a: Printout of the third ANLAP output file, c:\CD_data\fong201.pdf, using "pedro9.dp."

V. ANLAP APPLICATION EXAMPLE - 2 (CONT'D)

It is interesting to observe that the ANLAP-Datplot link is capable of turning a table-formatted output file (Figure 14) into report-quality graphical plots (bar chart in Figure 16, pie chart in Figure 17, and multi-plot bar charts in Figure 18).

But we can do more with the data if the table of Figure 14 has more than one row, as we purposely introduced a fictitious extra row of data (see Figure 14) to test the statistical analysis capability of ANLAP-Datplot.

In example 1, we have followed 5 steps of ANLAP to produce a table-formatted output file. In this example, we add several more steps to use the ANLAP-Datplot link, namely,

Steps 6a, 6b: For a single row of data, add "pedro9.dp" for a bar chart (Figure 16), and "pedro9pie.dp" for a pie chart (Figure 17).

Steps 7a, 7b: For two rows of data, one fictitious and one real, add "pedro10.dp" for two bar charts, and

"pedro11.dp" for a multiplot of raw data, boxplot, meanplot and standard deviation plot (Figure 19).

Clearly, any Dataplot codes such as "pedro8.dp" can be replaced with another code with more advanced statistical analysis capability so long as the ANLAP output is a text file of a tabular format. One such capability in the form of a Python-Dataplot plug-in was recently introduced by Fong, deWit, Marcal, Filliben, and Heckert [61], where an interactive code with user's input for a two-level factorial design of experiments exercise yields an uncertainty estimate of a large class of simulations based on the finite element method (FEM). Since almost all of the deterministic or probabilistic crack growth and damage models described in Section 1 (Introduction) and Figure 3, are FEM-based, we expect that ANLAP-Dataplot can be further linked to FEM packages such as [62, 63] to estimate the *damage model Uncertainty-M* (e_M), and the *remaining fatigue life Uncertainty-4* (e_4), as defined in Figure 3 and elaborated in a companion paper [45].

The potential value of the ANLAP-Dataplot-FEM link will be illustrated in the next section by an application given in [61].

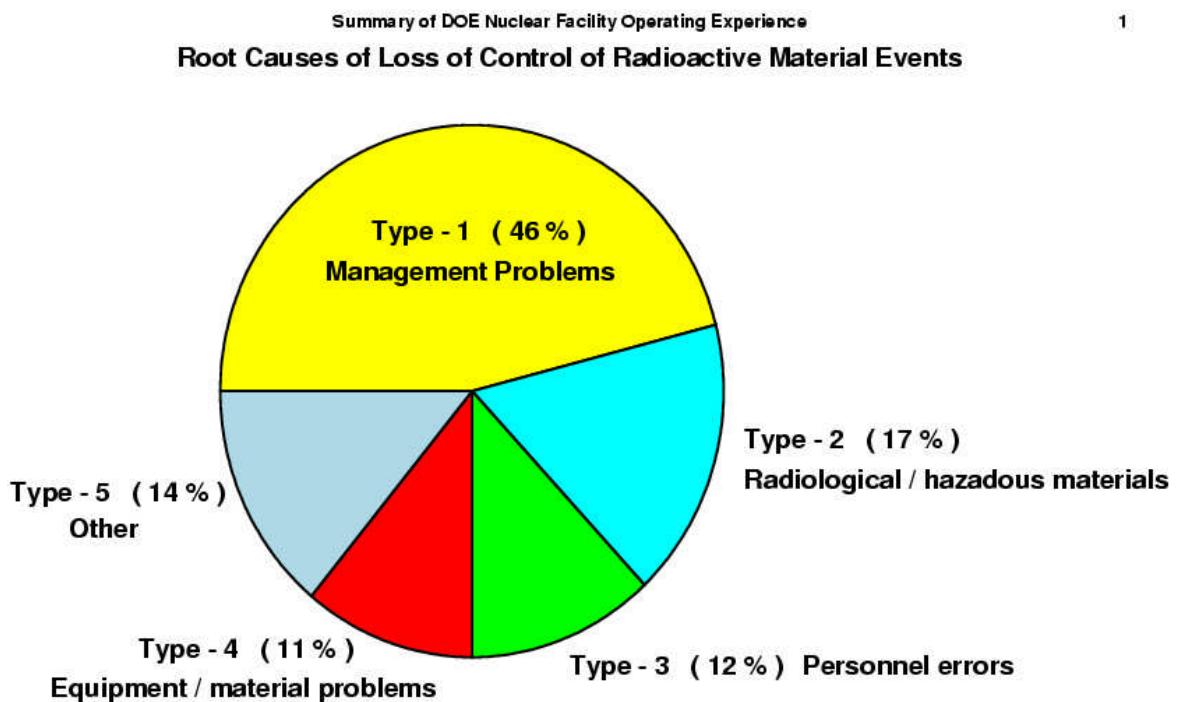


Figure 17. ANLAP Step 6b: Printout of the fourth ANLAP output file, c:\CD_data\fong202.pdf, using "pedro9pie.dp."

V. ANLAP APPLICATION EXAMPLE - 2 (CONT'D)

U. S. Dept. of Energy (DOE) Operating Experience Summary Report 98-40

http://tis.eh.doe.gov/web/oeaf/oe_weekly/oe_weekly.html.

Distribution of Root Causes for Spread of Contamination at Hanford

Period Covered: Jan. 1990 - Oct. 1998

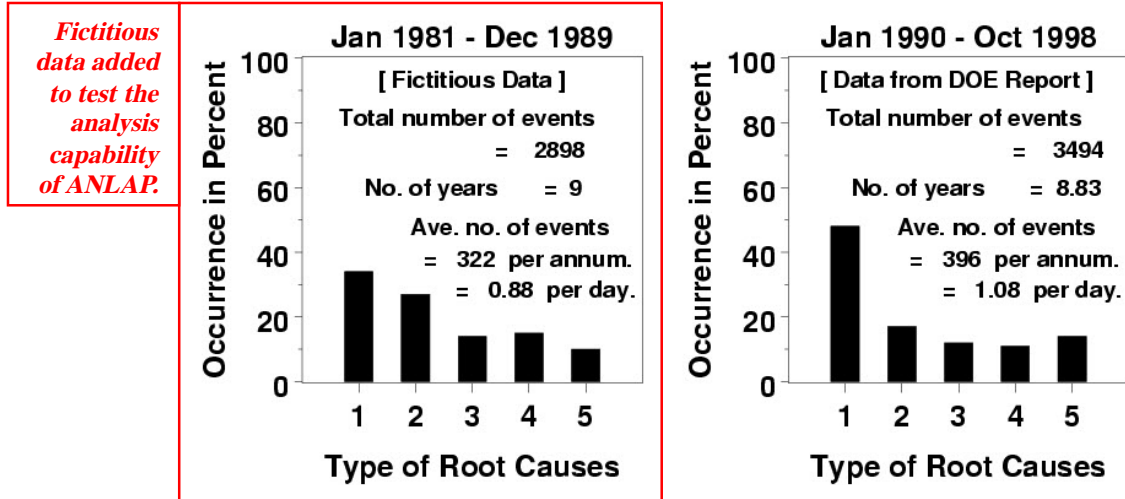


Figure 18. ANLAP Step 7a: Printout of the fifth ANLAP output file, c:\CD_data\fong203.pdf, using "pedro10.dp."

U. S. Dept. of Energy (DOE) Operating Experience Summary Report 98-40

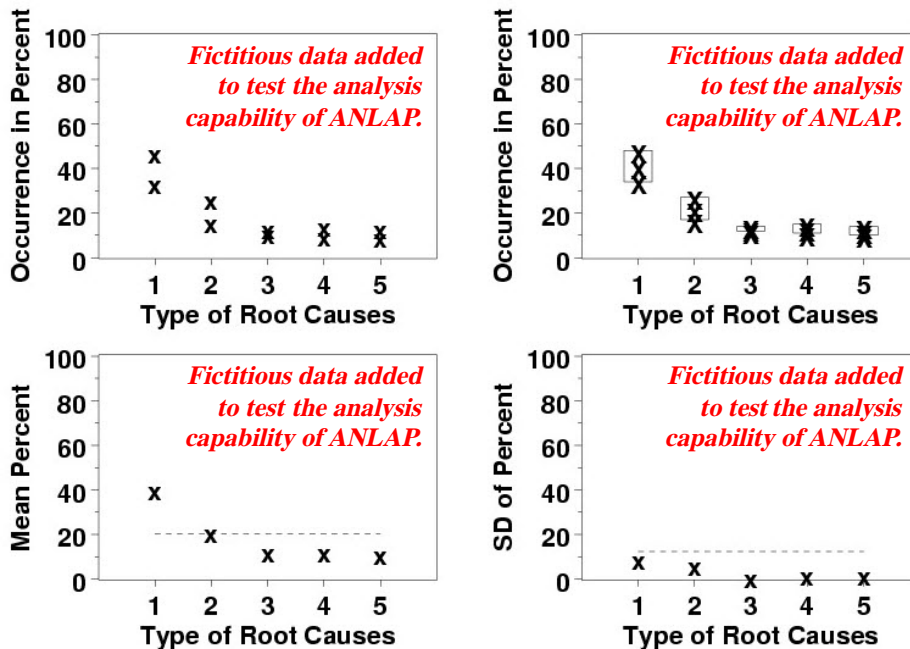


Figure 19. ANLAP Step 7b: Printout of the sixth ANLAP output file, c:\CD_data\fong204.pdf, using "pedro11.dp."

VI. ANLAP APPLICATION EXAMPLE - 3

To illustrate the value of an ANLAP-Dataplot-FEM link, we refer our readers to a recent paper by Fong, et al. [61], where a Python-Dataplot-Abaqus plug-in was designed and implemented to provide uncertainty estimates of Abaqus-FEM simulations. Our work on FEM uncertainty estimation was motivated by a 1996 U. S. Department of Defense directive [64] on a need to verify, validate, and accreditate all defense-related modeling and simulation results, and by pioneering work of Oberkampf [65, 69], Haldar, et al. [66], Roache [67], Kennedy and O'Hagan [68], Lord and Wright [70], and many others. Using a metrological approach based on statistical theory of design of experiments [71], Fong, et al. [72-74] developed an economical method (8 or 16 runs for a complex problem involving 5 to 10 factors) to estimate uncertainty of FEM simulations due to uncertainty in geometric parameters, material property constants, FEM mesh design, FEM mesh type, and FEM platform. In this section, we show in Figures 20 and 21 (for ANLAP Steps 8 and 9, resp.) two examples of this methodology [61, 74] for the FEM simulations of the free vibration of an isotropic elastic cantilever beam, of which we know the exact solution.

Table 3 (Fong, deWit, Marcal, Filliben & Heckert [61])

3-factor (L, h, E) (Length, Thickness, Young's Modulus)	Abaqus [62]	Abaqus [62]
full factorial orthogonal design of experiments (DEX)*	Mesh No. m = 4 (20 x 4 x 8 = 640 elements)	Mesh No. m = 12 (60 x 12 x 24 = 17,280 elem)
(0, 0, 0)	181.052	180.503
(-1, -1, -1)	182.119	181.566
(+1, -1, -1)	170.807	170.286
(-1, +1, -1)	191.692	191.113
(+1, +1, -1)	179.786	179.241
(-1, -1, +1)	182.301	181.748
(+1, -1, +1)	170.977	170.456
(-1, +1, +1)	191.883	191.304
(+1, +1, +1)	179.966	179.421

*DEX denotes Design of EXperiments [71], and is used as an acronym in this paper to avoid a confusion with DOE, which stands for the U.S. Department of Energy.

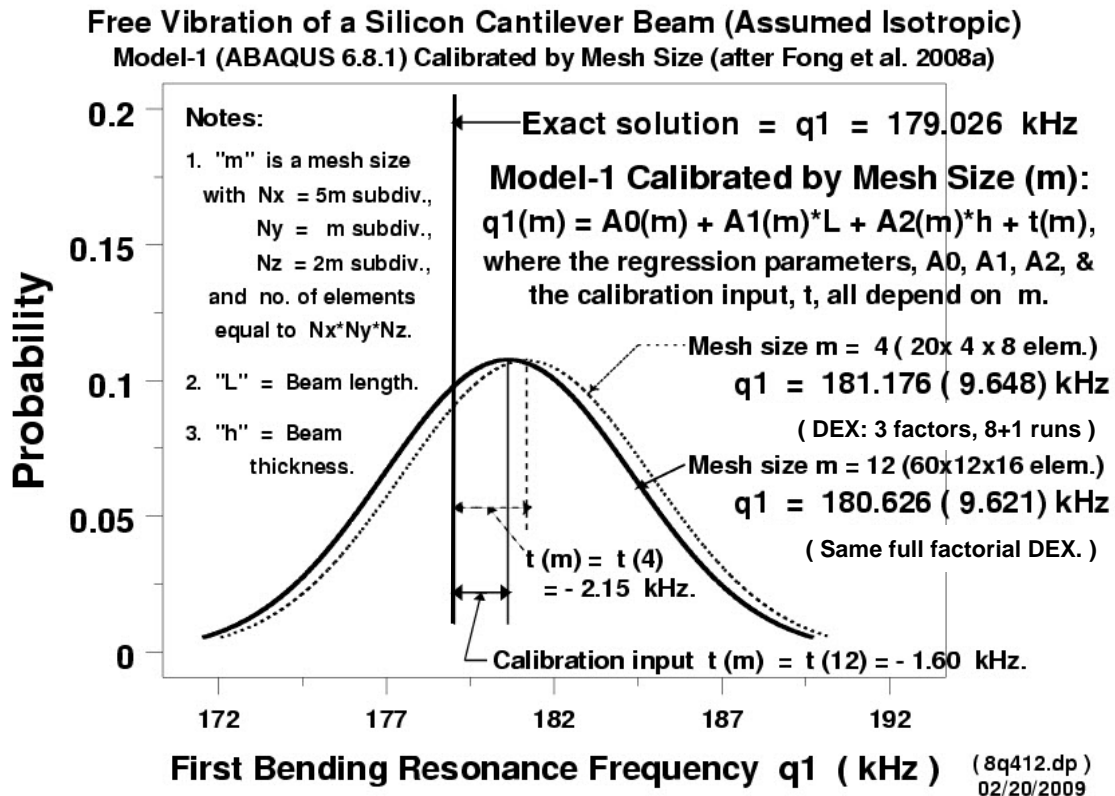


Figure 20. ANLAP Step 8: Plot of ANLAP output file, c:\CD_data\fang301.pdf, based on FEM data in Table 3 [61].

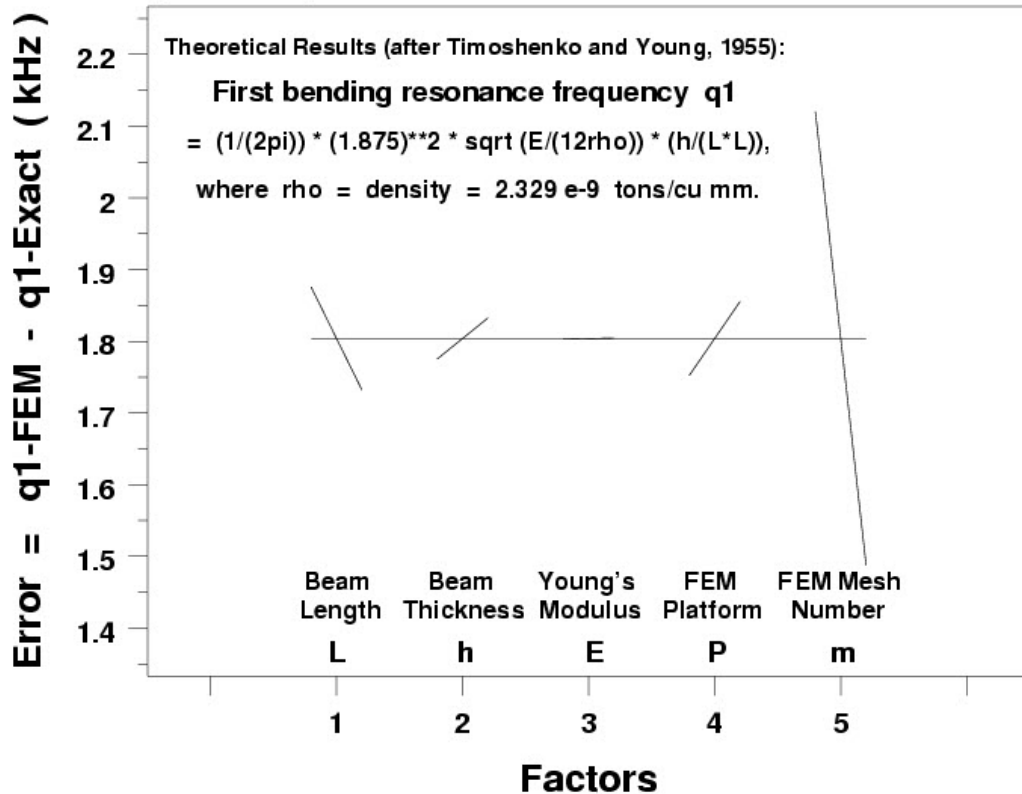
VI. ANLAP APPLICATION EXAMPLE - 3 (CONT'D)

Table 4 (after Fong, deWit, Marcal, Filliben & Heckert [61])

3-factor (L, h, E) (Length, Thickness, Young's Modulus)	Exact	Abaqus [62]	ANSYS [63]	Abaqus [62]	ANSYS [63]
full factorial design of experiments (DEX)	Theoretical Solution	Mesh No. m = 4 (20x4x8 = 640 elem)	Mesh No. m = 4 (640 elem)	Mesh No. m = 12 (60x12x24 = 17,280 elem)	Mesh No. m = 12 (17,280 elem)
(0, 0, 0)	179.026	181.052	181.238	180.503	180.523
(-1, -1, -1)	180.049	182.119	182.306	181.566	181.587
(+1, -1, -1)	168.891	170.807	170.981	170.286	170.305
(-1, +1, -1)	189.553	191.692	191.888	191.113	191.135
(+1, +1, -1)	177.806	179.786	179.970	179.241	179.261
(-1, -1, +1)	180.229	182.301	182.488	181.748	181.768
(+1, -1, +1)	169.060	170.977	171.152	170.456	170.475
(-1, +1, +1)	189.742	191.883	192.080	191.304	191.326
(+1, +1, +1)	177.984	179.966	180.150	179.421	179.441

Free Vibration of a Silicon Cantilever Beam (Assume Isotropic Elastic) 2

Ranking of Most Important Factors from Errors between FEM & Theoretical Results



(fong92a.dp)

02/26/09 21:30 EST

Figure 21. ANLAP Step 9: Plot of ANLAP output file, c:\CD_data\fong302.pdf, based on FEM data in Table 4 [61].

VII. SIGNIFICANCE OF RESULTS

Using an automatic natural language abstracting and processing (ANLAP) tool and its links to a statistical software package (Dataplot) and two finite element codes (Abaqus and ANSYS), we have demonstrated in this paper the power of a new suite of computer codes to *extract* information from failure event reports, NDE and ISI reports, material property test reports, and to *analyze* the tabulated information such that it furnishes the "best" input parameters, with uncertainty estimates, to finite element method-based models of fatigue crack initiation, growth and remaining life assessment.

As computer speed, memory, and cost continue to improve with time, the results of this paper also demonstrate that the artificial intelligence (AI) technology has begun to show promise as a practical tool for "managing" aging plants and structures with more timely and comprehensive diagnosis plus prognosis with uncertainty estimates. In other words, AI tools may minimize chances of human errors when a time-critical operating decision had to be made involving the *mining* of a massive amount of technical reports written in natural languages such as English or Japanese, and a probabilistic modeling of the behavior of a complex system such as a nuclear power plant.

Space limitations in this paper prevented us from including additional examples to illustrate our new results. However, we believe that the set of three examples included in this paper provides an adequate introduction as summarized below:

Example 1: A trivial case study to illustrate 5 key steps of our ANLAP code, which is written in Python 2.3 and fully listed in Appendix A.

Example 2: A real-life application using a page out of a DOE 1998 Nuclear Facility Operating Experience Weekly Report [14] and a statistical data analysis tool named Dataplot [48]. Without the introduction of a set of fictitious data, we will not be able to illustrate the calculation of mean and standard deviation of the raw data given in that DOE report. One of our five Dataplot codes is listed in Appendix B, and the rest is downloadable from <http://math.nist.gov/mcsd/software.html>.

Example 3: An advanced application illustrating how an ANLAP output data sheet can be interpreted with statistical data analysis to yield results including not only mean but also 95 % uncertainty bounds using a statistical design of experiments on the variability of selected parameters of a mathematical model.

Clearly, our results and examples can be significant not only to the nuclear power plant and petro-chemical processing industries but also to a broader class of society's energy and healthcare infrastructure such as oil and gas pipelines, bridges,

aircrafts and biomedical devices, for which safety, reliability, and productivity are always of a major concern.

However, we wish to dampen our enthusiasm with a word of caution. Information retrieval, data mining, and instant segmentation of massive amount of user-guided tables of failure-related statistics are at best a partial solution to the need of an engineer who wishes to expand his or her experience pool in order to make a better operational decision related to safety and reliability of an aging structure. Without adequate training of a user in asking for the right headings of a computer-extracted table and discriminating the "relevant" from the "irrelevant" findings, artificial intelligence (AI) tools as a critical component in the human decision-making system still has a long way to go.

Fortunately, the goal of this paper is limited in its scope to extract failure mechanism information and statistics from failure event reports worldwide in order to better manage the structural integrity of an aging plant. All users are required to be adequately trained, and the emergence of an AI tool to enrich the users with a bigger pool of relevant failure knowledge at a fraction of the cost by conventional means is welcome news indeed.

VIII. CONCLUDING REMARKS

As stated in our introduction (Section I), we have identified four categories of uncertainties and errors associated with fatigue life estimation, and in this paper, we have concentrated our effort in addressing only the first of those four categories, namely, **Uncertainty-1 (e_1)**, which is associated with failure event databases. To see how the other three uncertainty categories are addressed, we refer our readers to the other three companion papers [43, 44, 45] which have been submitted for presentation at the same July 26-30, 2009 ASME PVP Division Conference (Prague, The Czech Republic).

In response to a remark made by Chockie and Gregor [29], as quoted in our Section I, that there is a critical need for assessing the cost and benefits of failure event databases, we would like to refer our readers to the significant work of Chapman and Rushing [75], and an ASTM standards [76] on risk-informed database tools for capital asset protection.

IX. REFERENCES

1. U.S. Nuclear Regulatory Commission, 2008, Fact Sheet on the Three Mile Island Accident. USNRC, Washington, DC, <http://www.nrc.gov/reading-rm/doc-collections/fact-sheets/3mile-isle.html> (2008).
2. Fong, J. T., 1980, "Inservice Data Reporting Standards for Engineering Reliability and Risk Analysis," Nucl. Eng. Design, Vol. 60, No. 1, pp. 159-161 (1980).
3. Fong, J. T., 1986, "Integration of Engineering Analysis and Databases for Critical Decision Making," Computers in Mechanical Engrg. (CIME), Vol. 5 (1), 42-55 (1986).
4. Fong, J. T., and Bernstein, B., 1988, "Building a PC-Based Knowledge Base for Improving NDE Reliability," in Pressure Vessel Technology, Proc. 6th Int'l Conf. On Pressure Vessel Technology, Beijing, China, Sep. 11-15, 1988, C. Liu and R. W. Nichols, eds., Vol. II, pp. 1349-1371. Pergamon (1988).
5. Vo, T. V., Gore, B. F., Eschbach, E. J., and Simonen, F. A., 1989, "Probabilistic Risk Assessment Based Guidance for Piping Inservice Inspection," Nuclear Technology, Vol. 88, pp. 13-20 (1989).
6. Doctor, S. R., and Spanner, J. C., Sr., 1993, "Studies on the Quantification of NDE Reliability," Proc. ASME 1993 PVP Conference, Spec. Pub. PVP-Vol. 257 entitled "Scientific and Engineering Aspects of Nondestructive Evaluation," pp. 39-46 (1993).
7. Bush, S.H., Do, M.J., Slavich, A.L., Chockie, A.D., 1996, Piping Failures in United States Nuclear Power Plants: 1961-1995, SKI 96:20, prepared by the Chockie Group Int'l, Inc., Seattle, WA, for the Swedish Nuclear Power Inspectorate, Stockholm, Sweden, (1996)
8. EPRI TR-106706, 1996, Risk-Informed Inservice Inspection Evaluation Procedure, Electric Power Research Institute, Inc. Palo Alto, California. (1996)
9. ASME Code Case N-577, 1997, Risk-Informed Requirements for Class 1, 2, and 3 Piping, Method A. New York, NY: ASME (1997).
10. Bush, S. H., 1997, "Interpretive Report on Nondestructive Examination Techniques, Procedure for Piping and Heavy Section Vessels," Welding Research Council Bulletin 420, April 1997, Welding Research Council, NY, NY (1997).
11. Zhao, Z., and Haldar, A., 1997, "Reliability-based Structural Fatigue Damage Evaluation and Maintenance Using Non-destructive Inspections," in Uncertainty Modeling in Finite Element, Fatigue and Stability of Systems, A. Haldar, A. Guran, and B. M. Ayyub, Eds., pp. 159-214. World Scientific Publishing Company, River Edge, NJ 07661 (1997).
12. EPRI, 1998, Nuclear Reactor Piping Failures at U.S. Commercial LWRs: 1961-1997, TR-110102, Electric Power Research Institute, Palo Alto, CA (1998)
13. Schuster, G. J., Doctor, S. R., Heasler, P. G., 1998, Characterization of Flaws in U.S. Reactor Pressure Vessels: Density and Distribution of Flaw Indications in PVRUE, NUREG/CR-6471, Vol. 1. U.S. Nuclear Regulatory Commission, Washington, DC (1998).
14. U.S. Department of Energy, 1998, Operating Experience Weekly Summary 98-40, October 2 through October 8, 1998, published by the U. S. Department of Energy, Office of Environment, Safety and Health, and its Office of Nuclear and Facility Safety (NFS) and Office of Operating Experience Analysis and Feedback (OEAF), http://tis.eh.doe.gov/web/oeaf/oe_weekly.html (1998).
15. Khaleel, M. A., and Simonen, F. A., 1999, "Uncertainty Analyses for Calculated Failure Probabilities of Piping Welds," in Proc. ASME PVP Conference, Boston, MA, Aug. 1-5, 1999, PVP Vol. 383, entitled "Pressure Vessel and Piping Codes and Standards, 1999," B. T. Lubin, et al, eds., pp. 77-90. American Society of Mechanical Engineers, New York, NY 10016 (1999).
16. Westinghouse, 1999, Westinghouse Owners Group Application of Risk-Informed Methods to Piping Inservice Inspection Topical Report, WCAP-14572, Revision 1, NP-A Westinghouse Energy Systems, Pittsburgh, PA, (1999)
17. Khaleel, M. A., Simonen, F. A., Phan, H. K., Harris, D. O., and Dedhia, D., 2000, Fatigue Analysis of Components for 60-year Plant Life, NUREG/CR-6674. U. S. Nuclear Regulatory Commission, Washington DC (2000).
18. U.S. Nuclear Regulatory Commission, NUREG-1801, 2001, Vo. 1 Rev. 0, Generic Aging Lessons Learned (GALL) Report, Nuclear Regulatory Commission, Washington DC (2001).
19. API, 2002, API 580 Risk-Based Inspection. Washington, D.C.: American Petroleum Institute (2002).
20. Simonen, F. A., Schuster, G. J., Doctor, S. R., and Dickson, T. L., 2002, "Distributions of Fabrication Flaws in Reactor Pressure Vessels for Structural Integrity Evaluations," Proc. ASME PVP Conf., Vancouver, BC, Canada, Aug. 5-9, 2002, Spec. Pub. PVP-Vol. 443-2, Fatigue, Fracture, and Damage Analysis, Vol. II, pp. 125-131. ASME (2002).
21. Gosselin, S. R., et al, 2005, "Performance Demonstration Based Probability of Detection (POD) Curves for Fatigue Cracks in Piping," paper PVP2005-71027, Proceedings of ASME PVP 2005 Conf., Denver, CO. ASME (2005).
22. Lydell, B.O.Y., 2005, PIPExp/PIPE-2005: Monthly Summary of Database Content (Status as of Dec-2005), RSA-R-2005-01.07, Sigma-Phase Inc. Tucson AZ, (2005)

23. Nyman, R. et al., 2005, "Reliability of Piping System Components Framework for Estimating Failure Parameters from Service Data", SKI Report 97:26 3rd edition, Swedish Nuclear Power Inspectorate, Stockholm, Sweden, (2005)
24. Fong, J. T., and Hedden, O. F., eds., 2007, Engineering Safety, Applied Mechanics, and Nondestructive Evaluation (NDE), Proc. 2007 ASME PVP Symp in honor of Dr. Spencer H. Bush (1920-2005), July 25-26, 2007, San Antonio, TX. Published by Stanford Mechanics Alumni Club, 104 King Farm Blvd, Suite C308, Rockville, MD 20850 (2007).
25. Lydell, B.O.Y., 2007a, "The Probability of Pipe Failure on the Basis of Operating Experience," PVP2007-26281, Proc. 2007 ASME Pressure Vessel and Piping Division Conference, July 22-26, 2007, San Antonio, TX (2007a).
26. Simonen, F. A., Doctor, S. R., Gosselin, S. R., Rudland, D. L., Xu, H., Wilkowski, G. M., and Lydell, B. O. Y., 2007a, Probabilistic Fracture Mechanics Evaluation of Selected Passive Components - Technical Letter Report, a Pacific Northwest National Laboratory Report PNNL-16625 dated May 2007, prepared for the U.S. Nuclear Regulatory Commission, Washington, DC 20555 (2007a).
27. Simonen, F. A., Gosselin, S. R., Lydell, B. O. Y., Rudland, D. L., and Wilkowski, G. M., 2007b, "Application of Failure Event Data to Benchmark Probabilistic Fracture Mechanics Computer Codes," Proceedings of ASME 2007 PVP Conference, San Antonio, TX, July 22-26, Paper No. PVP2007-26373. ASME, New York, NY (2007b).
28. U.S. Nuclear Regulatory Commission, NUREG/CR-6945, 2007, Fabrication Flaw Density and Distribution in Repairs to Reactor Pressure Vessel and Piping Welds, Nuclear Regulatory Commission, Washington DC (2007).
29. Chockie, A. D., and Gregor, F. E., 2008, "The Role of Failure Data in Plant Aging Management and Life Extension," Proc. ASME Pressure Vessels & Piping Conf., July 27-31, 2008, Chicago, IL. Paper No. PVP2008-61552, <http://www.asmeconferences.org/PVP08> (2008).
30. Fong, J. T., and Marcal, P. V., 2008a, "An Intelligent Flaw Monitoring System: From Flaw Size Uncertainty to Fatigue Life Prediction with Confidence Bounds in 24 Hours," Proc. 8th World Congress on Computational Mechanics, Venice, Italy, June 30-July 5, 2008, Ref. 1732, <http://www.iacm-eccomascongress2008.org/> (2008a).
31. Fong, J. T., Hedden, O. F., and Lam, P. S., eds., 2008b, A Symposium Digest on Failure Prevention via Robust Design and Continuous NDE Monitoring, distributed free to July 31, 2008 symposium attendees and contributors at the ASME 2008 Pressure Vessels and Piping Division Conference, July 27-31, 2008, Chicago, IL. For further information, contact: fong@drexel.edu (2008b).
32. Fong, J. T., Hedden, O. F., Filliben, J. J., and Heckert, N. A., 2008c, "A Web-based Data Analysis Methodology for Estimating Reliability of Weld Flaw Detection, Location, and Sizing," Proc. ASME Pressure Vessels & Piping Conf., July 27-31, 2008, Chicago, IL. Paper No. PVP2008-61612, <http://www.asmeconferences.org/PVP08> (2008c).
33. Fong, J. T., Ranson, W. F., III, Vachon, R. I., and Marcal, P. V., 2008d, "Structural Aging Monitoring via Web-based Nondestructive Evaluation Technology," Proc. ASME Pressure Vessels & Piping Conference, July 27-31, 2008, Chicago, IL. Paper No. PVP2008-61607, <http://www.asmeconferences.org/PVP08> (2008d).
34. Khaleel, M. A., and Simonen, F. A., 2008, Evaluations of Structural Failure Probabilities and Candidate Inservice Inspection Programs, a Pacific Northwest National Laboratory Report PNNL-13810 dated Nov. 2008 to appear as a NUREG document of U.S. Nuclear Regulatory Commission, Washington, DC 20555-0001 (2008).
35. Lydell, B.O.Y. and Olsson, A., 2008a, "Reliability Data for Piping Components in Nordic Nuclear Power Plants "R-Book" Project Phase I Rev 1", SKI Report 98:20, Swedish Nuclear Power Inspectorate, Stockholm, Sweden, (2008).
36. Lydell, B., Huerta, A., and Gott, K., 2008b, "Characteristics of Damage & Degradation Mechanisms in Nuclear Power Plant Piping Systems," Proc. ASME Pressure Vessels & Piping Conference, July 27-31, 2008, Chicago, IL. Paper No. PVP2008-61914, <http://www.asmeconferences.org/PVP08> (2008b).
37. Marcal, P. V., and Fong, J. T., 2008, "Continuous NDE Monitoring via Web Technology," Proc. ASME Pressure Vessels & Piping Conference, July 27-31, 2008, Chicago, IL. Paper No. PVP2008-61574, <http://www.asmeconferences.org/PVP08> (2008).
38. Pietrangelo, A. R., 2008a, Congressional testimony before the U.S. Senate Committee on Environment and Public Works, Subcommittee on Clear Air and Nuclear Safety, Washington, DC, July 16, 2008. Nuclear Energy Institute, Washington, DC 20006-3708, http://www.nei.org/newsandevents/speechesandtestimony/2008_speeches_and_testimony/pietrangelo.html (2008a).
39. Pietrangelo, A. R., 2008b, Planning for Success: Reasoned Expectations for New Nuclear Plant Construction, a plenary lecture delivered at the ASME 2008 Pressure Vessels and Piping Conference, July 28, 2008, Chicago, IL. Nuclear Energy Institute, Washington, DC 20006-3708, <http://www.nei.org> (2008b).
40. Teather, D., 2004, "US nuclear industry powers back into life," The Guardian, Tuesday, 13 April 2004. <http://www.guardian.co.uk> (2004).
41. Cooper, M., 2009, "U.S. Infrastructure Is in Dire Straits, Report Says," New York Times, Jan. 28, 2009, page A16 (2009).

42. Allegheny County, 1999, "Jonathan Hulton Bridge," Bridges & Tunnels of Allegheny County & Pittsburgh, PA, <http://pghbridges.com/newkenW/0597-4486/hulton.html> (1999).
43. Fong, J. T., Marcal, P. V., Hedden, O. F., Chao, Y. J., and Lam, P. S., 2009a, "A Web-based Uncertainty Plug-In (WUPI) for Fatigue Life Prediction based on NDE Data and Fracture Mechanics Analysis," to appear in Proc. ASME Pressure Vessels & Piping Conference, July 26-30, 2009, Prague, The Czech Republic, Paper No. PVP2009-77827, <http://www.asmeconferences.org/PVP09> (2009a).
44. Fong, J. T., and Marcal, P. V., 2009b, "A Dataplot-Python-Anlap (DPA) Plug-In for High Temperature Mechanical Property Databases to Facilitate Stochastic Modeling of Fire-Structure Interactions," to appear in Proc. ASME Pressure Vessels & Piping Conference, July 26-30, 2009, Prague, The Czech Republic, Paper No. PVP2009-77867, <http://www.asmeconferences.org/PVP09> (2009b).
45. Fong, J. T., deWit, R., Marcal, P. V., Filliben, J. J., Heckert, N. A., and Gosselin, S. R., 2009c, "Design of a Python-based Plug-In for Benchmarking Probabilistic Fracture Mechanics Computer Codes with Failure Event Data," to appear in Proc. ASME Pressure Vessels & Piping Conference, July 26-30, 2009, Prague, The Czech Republic, Paper No. PVP2009-77974, <http://www.asmeconferences.org/PVP09> (2009c).
46. Marcal, P. V., 2009, ANLAP User's Manual. Published by MPACT Corp., Julian CA, marcalpv@cox.net (2009).
47. Loper, E., and Bird, S., 2002, "NLTK: The Natural Language Toolkit," Proc. ACL Workshop on Effective Tools and Methodologies for Teaching NLP and CL. Somerset, NJ: Association for Computational Linguistics, <http://epydock.sourceforge.net/> (2002).
48. Filliben, J. J., and Heckert, N. A., 2002, Dataplot: A Statistical Data Analysis Software System. A Public Domain Software by NIST, Gaithersburg, MD 20899, <http://www.itl.nist.gov/div898/software/dataplot.html> (2002).
49. Schank, R., 1972, "Conceptual dependency: A theory of natural language understanding," Cognitive Psychology, Vol. 3 (4), pp. 552-631 (1972).
50. Schank, R. C., 1973, "Identification of Conceptualizations Underlying Natural Language," in Computer Models of Thought and Language, R. C. Schank and K. M. Colby, eds., Chapter 5. San Francisco, CA: W. H. Freeman and Company (1973).
51. Chamberlin, D. D., and Boyce, R. F., 1974, "SEQUEL: A Structural English Query Language," Proc. 1974 ACM SIGFIDET Workshop on Data Description, Access and Control, pp. 249-264. Association for Computing Machinery (1974).
52. ANSI/ISO/IEC 9075-2:1999, International Standard (IS), Information Systems -- Database Language -- SQL -- Part 2: Foundation (SQL/Foundation), 1147pp (1999).
53. ISO/IEC 9075-11: 2008, Information technology -- Database languages -- SQL -- Part 11: Information and Definition Schemas (SQL/Schemata), 278pp (2008).
54. van Rossum, G., 1999, Python Tutorial, Feb. 19, 1999, Release 1.5.2. Corp. for National Research Initiative (CNRI), 1895 Preston White Dr., Reston, VA (1999).
55. Hammond, M., and Robinson, A., 2000, Python Programming on Win32, O'Reilly Media, Inc. (2000).
56. Langtangen, H. P., 2008, Python Scripting for Computational Science, third edition. Springer (2008). Blackburn, P., and Bos, J., 2005, Representation and Inference for Natural Language: A First Course in Computational Semantics. CSLI Publications, Center for the Study of Language and Information, Stanford, CA (2005).
57. Bird, S., 2006, "NLTK: The Natural Language Toolkit," Proc. COLING/ACL 2006 Interactive Presentation Sessions, pages 69-72, Sydney, July 2006. Assoc. Computational Linguistics (2006).
58. Bird, S., Klein, E., and Loper, E., 2008, Natural Language Processing. Published by the authors and distributed with the *Natural Language Toolkit* [<http://www.nltk.org/>] Version 0.9.7a, 18 Dec (2008).
59. Church, K. W., "A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text," Proc. Second Conf. on Applied Natural Language Processing, Austin, TX, 1988, pp. 136-143 (1988).
60. Croarkin, C., Guthrie, W., Heckert, N. A., Filliben, J. J., Tobias, P., Prins, J., Zey, C., Hembree, B., and Trutna, eds., 2006, NIST/SEMATECH e-Handbook of Statistical Methods, Chapter 5 on Process Improvement (pp. 1-480), <http://www.itl.nist.gov/div898/handbook/>, first issued, June 1, 2003, and last updated July 18, 2006. Produced jointly by the Statistical Engineering Division of the National Institute of Standards & Technology, Gaithersburg, MD, and the Statistical Methods Group of SEMITECH, Austin, TX. Also available as a NIST Interagency Report in a CD-ROM upon request to alan.heckert@nist.gov (2006).
61. Fong, J. T., deWit, R., Marcal, P. V., Filliben, J. J., and Heckert, N. A., 2009, "A Design-of-Experiments Plug-In for Estimating Uncertainties in Finite Element Simulations," to appear in Proceedings of 2009 International SIMULIA Conference, May 18-21, 2009, London, U.K., Paper ID 9098. Providence, RI: SIMULIA-Dassault Systemes Simulia Corp. (2009).
62. Abaqus, 2008, Abaqus User's Manual, Version 6.8.0. Dassault Systemes Simulia Corp., 166 Valley Street, Providence, RI (2008).

63. ANSYS, 2008, ANSYS User's Manual. Release 11.0. ANSYS, Inc., 275 Technology Dr., Cannonsburg, PA (2008).
64. U.S. Department of Defense, 1996, DOD Directive No. 5000.61: Modeling and Simulation (M&S) Verification, Validation, and Accreditation (VV&A), Defense Modeling and Simulation Office, Office of the Director of Defense Research and Engineering, www.dmsomil/docslib (1996).
65. Oberkampf, W. L., 1994, "A Proposed Framework for Computational Fluid Dynamics Code Calibration/Validation," AIAA Paper 94-2540, 18th AIAA Aerospace Ground Testing Conference., Colorado Spring, CO (1994).
66. Haldar, A., Guran, A., and Ayuub, B. M., eds., 1997, Uncertainty Modeling in Finite Element, Fatigue and Stability of Systems. World Scientific Publishing Co. Pte. Ltd., 1060 Main Street, River Edge, NJ (1997).
67. Roache, P. J., 1998, Verification and Validation in Computational Science and Engineering. Hermosa Publishers, Albuquerque, NM (1998).
68. Kennedy, M. C., and O'Hagan, A., 2001, "Bayesian calibration of computer models," J. Royal Statistical Soc., Serial B, Vol. 63, Part 3, pp. 425-464 (2001).
69. Oberkampf, W. L., Trucano, T. G., and Hirsch, C., 2002, "Verification, Validation, and Predictive Capability in Computational Engineering and Physics," Proc. Workshop on Foundations for V & V in the 21st Century, 22-23 Oct. 2002, John Hopkins Univ./Appl. Phys. Lab., Laurel, Maryland, D. Pace & S. Stevenson, eds., published by Society for Modeling & Simulation International (2002).
70. Lord, G. J., and Wright, L., 2003, Uncertainty Evaluation in Continuous Modeling, Report to the National Measurement System Policy Unit, Department of Trade and Industry, NPL Report CMSC 31/03. Teddington, Middlesex, U.K.: National Physical Laboratory (2003).
71. Box, G. E., Hunter, W. G., and Hunter, J. S., 1978, Statistics for Experimenters: An Introduction to Design, Data Analysis, and Model Building. Wiley (1978).
72. Fong, J. T., Filliben, J. J., deWit, R., Fields, R. J., Bernstein, B., and Marcal, P. V., 2006a, "Uncertainty in Finite Element Modeling and Failure Analysis: A Metrology-Based Approach," ASME Trans., J. Press. Vess. Tech., Vol. 128, pp. 140-147 (2006a).
73. Fong, J. T., Filliben, J. J., deWit, R., and Bernstein, B., 2006b, "Stochastic Finite Element Method (FEM) and Design of Experiments for Pressure Vessel and Piping (PVP) Decision Making," Proc. of 2006 ASME Pressure Vessels and Piping Division Conference, July 23-27, 2006, Vancouver, B. C., Canada, paper no. PVP2006-ICPVT11-93927. New York, NY: American Society of Mechanical Engineers (2006b)..
74. Fong, J. T., Filliben, J. J., Heckert, N. A., and deWit, R., 2008, "Design of Experiments Approach to Verification and Uncertainty Estimation of Simulations Based on Finite Element Method," Proc. Conf. Amer. Soc. for Engineering Education, June 22-25, Pittsburgh, PA, Paper AC 2008-2725 (2008).
75. Chapman, R. E., and Rushing, A. S., 2007, Users Manual for Version 3.0 of the Cost-Effectiveness Tool for Capital Asset Protection, a National Institute of Standards and Technology report, NISTIR 7455. Gaithersburg, MD: National Institute of Standards and Technology, <http://www.nist.gov/> (2007).
76. ASTM Standard E 964-06, 2006, Standard Practice for Measuring Benefit-to-Cost and Savings-to-Investment Ratios for Buildings and Building Systems, issued by ASTM Subcommittee E06.81 on Building Economics, first approved 1983, current edition approved Apr. 1, 2006. ASTM International, 100 Barr Harbor Dr, POB C700, West Conshohocken, PA 19428-2959 (2006).

XIV. ACKNOWLEDGMENT

The authors wish to thank

Howard Baum, Ron Boisvert, Robert Chapman, Robert Cook, Christopher Dabrowski, Nicholas Dagalakis, Roland deWit, Alden Dima, Richard Fields, James Filliben, Elizabeth Fong, Edwin Fuller, John Gross, Alan Heckert, Raghu Kacker, Ma Li, Hung-Kung Liu, Bruce Miller, Anne Plant, Kuldeep Prasad, Ronald Rehm, Emil Simiu, and Sheldon Wiederhorn of U.S. National Institute of Standards and Technology (NIST), Gaithersburg, MD, and **Andrew Dienstfrey** of NIST, Boulder, CO,

H. Norman Abramson (*retired*) of Southwest Research Institute, San Antonio, TX,

Barry Bernstein (*retired*) of Departments of Mathematics and Chemical and Environmental Engineering, Illinois Institute of Technology, Chicago, IL,

Hal Brinson (*retired*) of Department of Mechanical Engineering, University of Houston, Houston, TX,

Spencer Bush (*retired*), **Steven Doctor** and **Fredric Simonen** (*retired*) of Pacific Northwest National Laboratory, Richland, WA,

Borchin Chang, Bradley Layton, Alan Lau, and Jack Zhou of Department of Mechanical Engineering and Mechanics, Drexel University, Philadelphia, PA.

Y. J. (Bill) Chao and **William (Bill) Ranson** of Department of Mechanical Engineering, University of South Carolina, Columbia, SC,

Fu Liang (Bill) Cho of Risk Solver Communications, Inc., Irvine, CA,

Alan Chockie of Chockie Group International, Seattle, WA,

I-Ling Chow (*retired*) of U.S. Department of Energy, Germantown, MD,

Marvin Cohn and **Geoffrey Egan** of Aptech Engineering Services, Sunnyvale, CA,

Steve Gosselin and **Bengt Lydell** of Scandpower Risk Management, Inc., Houston, TX,

Owen Hedden of Codes & Standards Consulting, Fort Worth, TX,

Jon Helton and **William Oberkampf** of Sandia National Laboratory, Albuquerque, NM,

Charles Interrante (*retired*) of U.S. Nuclear Regulatory Commission, Washington, DC,

Mel Kanninen of San Antonio, TX,

Poh-Sang Lam of Savannah River National Laboratory, Aiken, SC,

Andrew MacKeith and **David Winkler** of Dassault Systemes Simulia Corp., Providence, RI,

Paul Mitiguy and **Charles Steele** (*retired*) of Department of Mechanical Engineering, Stanford University, Stanford, CA,

Anthony Pietrangelo of Nuclear Energy Institute, Washington, DC,

Robert Rainsberger of XYZ Scientific Applications, Inc., Livermore, CA,

Robert Reid of RHR Consulting Engineers, Oakmont, PA,

Weiju Ren and **Robert Swindeman** (*retired*) of Oak Ridge National Laboratory, Oak Ridge, TN,

T. Richard (Dick) Robe (*retired*) of College of Engineering, Ohio University, Athens, OH,

Glenn Sinclair of Department of Mechanical Engineering, Louisiana State University, Baton Rouge, LA,

Reginald Vachon of Direct Measurement, Inc., Atlanta, GA, and

Tomasz Wierzbicki of Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA,

for their valuable discussions, comments, and/or technical assistance during the course of our investigation that began by one of our co-authors (**Fong**) thirty years ago shortly after the Three Mile Island accident.

APPENDIX A

A complete listing of the ANLAP 1.0 code, "mpact_concept_nleps.py," which is written in Python 2.3, and runs about 1,000 lines, is listed below and also available for download at <http://math.nist.gov/mcsd/software.html>. Note the ending of this code where a Dataplot code named "pedro8.dp" is called. An updated version of this code in Python 2.5 will appear in ANLAP 2.0.

```
# -----
# Name of file:  mpact_concept_nleps.py (Python Version 2.3) Last Updated: March 21, 2009
# By: Pedro V. Marcal, MPact Corp. Julian CA, and Jeffrey T. Fong, NIST, Gaithersburg, MD
# Ref.: Marcal, Fong, Yamagata, 2009, Proc. ASME PVP Conf, July 26-30, Prague, Paper #77871.
# -----
import re
from qt import *
from mpact_concept_nleps_base import *
import string
import cPickle
import shelve
import os

class ConceptNlepsDialog(Concept_NLEPS):
    def __init__(self,parent = None,name = None,modal = 0,fl = 0):
        Concept_NLEPS.__init__(self,parent,name,modal,fl)
        if not name:
            self.setName("ConceptNlepsDialog")
        self.m_concept = {}
        self.m_parent=parent
        self.m_fileInput=False
        self.m_field=False
        self.m_matchClicked={}
        self.m_n=0
        self.m_L=0
        self.m_lineCount=0
        self.m_lineStoreCount=0
        self.m_flenCount=0
        self.m_undefinedWords=[]
        self.m_maxRating=[0,0]
        self.m_parsedSENTS={}
        self.m_primMatch={}
        self.m_matchPrim={}
        self.m_countSentences=0
        self.m_quotes={}
        self.m_tagged=False
        self.m_inputVariablesCount=0
        self.m_match_ij={}
        self.m_match_nums_ij={}
        self.m_primMatch_ij={}
        self.m_matchPrim_ij={}
        self.m_rcount_ij=[0]
        self.m_locPartsList={}
        self.m_Thunks={}
        self.m_createModel=False
        self.m_InitiateModel=False
        self.m_tcount=[]
        self.m_button=''
        self.m_responseMore=False
        self.connect(self.OKButton,SIGNAL("clicked()"),self.conceptNlepsOK)
        #self.connect(self.CancelButton,SIGNAL("clicked()"),self.CancelButtonPressed)
    def CancelButton_clicked(self) :
        return
    def conceptNlepsOK(self) :
        self.conceptNlepsOKDeb()
    def conceptNlepsOKDeb(self) :
        pass
    def textButton_clicked(self):
        #print "Concept_NLEPS.textButton_clicked(): Not implemented yet"
        self.m_button='text'
        self.UpdateButton()
    def WhatButton_clicked(self):
        print "Concept_NLEPS.WhatButton_clicked(): Not implemented yet"
        self.m_button='what'
        self.UpdateButton()
    def WhenButton_clicked(self):
        print "Concept_NLEPS.WhenButton_clicked(): Not implemented yet"
        self.m_button='when'
        self.UpdateButton()
    def Where_clicked(self):
        print "Concept_NLEPS.Where_clicked(): Not implemented yet"
        self.m_button='where'
        self.UpdateButton()
    def HowButton_clicked(self):
        print "Concept_NLEPS.HowButton_clicked(): Not implemented yet"
        self.m_button='how'
```

```

        self.UpdateButton()
def WhyButton_clicked(self):
    print "Concept_NLEPS.WhyButton_clicked(): Not implemented yet"
    self.m_button='why'
    self.UpdateButton()
def DirectButton_clicked(self):
    #print "Concept_NLEPS.DirectButton_clicked(): Not implemented yet"
    self.m_button='direct'
    self.UpdateButton()
def UpdateButton(self):
    #self.UpdateButtonDeb()
    #def UpdateButtonDeb(self) :
    if not self.m_fileInput :
        self.m_concept["basePath"]=self.inputFileEdit.text().ascii()
        self.m_FileName = self.m_concept["basePath"]
        dir_name=os.path.dirname(self.m_FileName)
        self.m_FileNameSplit=os.path.splitext(self.m_FileName)
        self.m_FileNameSplitPath=os.path.split(self.m_FileNameSplit[0])
        self.m_FilePath=self.m_FileNameSplitPath[1]
        self.m_pick_file=os.path.join(dir_name,'CD_pickle')
    try :
        filer=open(self.m_pick_file,'r')
        self.m_CONCEPTS=cPickle.load(filer)
        self.m_DICTS=cPickle.load(filer)
        self.m_SENTENCES=cPickle.load(filer)
        self.m_NAMES=cPickle.load(filer)
        self.m_SYNS=cPickle.load(filer)
        self.m_synWORDS=cPickle.load(filer)
        self.m_synNONYMS=cPickle.load(filer)
        self.m_GLOSSS=cPickle.load(filer)
        self.m_TRANS=cPickle.load(filer)
        self.m_ANTs=cPickle.load(filer)
        self.m_PARS=cPickle.load(filer)
        self.m_PRIMS=cPickle.load(filer)
        self.m_prevPRIMS=cPickle.load(filer)
        self.m_skipPRIMS=cPickle.load(filer)
        #self.m_comSENSES=cPickle.load(filer)
        self.m_NOUNS=cPickle.load(filer)
        filer.close()
    except :
        pass
    self.m_pick_filer=os.path.join(dir_name,'CD_parsed')
    try :
        book=shelve.open(self.m_pick_filer)
        #book[self.m_FilePath]=self.m_parsedSENTs
        #book.close()
        self.m_book=book
    except :
        pass
    self.m_fileInput=True
    self.m_zero=self.computerEdit_1.text().ascii()
    if not self.m_button=='':
        for self.m_FilePath in self.m_book :
            i=1
            self.m_FilePath='nleps_1'
            self.m_parsedSENTs=self.m_book[self.m_FilePath]
            rang=7
            for i in range(rang) :
                self.SetLineUserUnitToZero()
            for i in self.m_parsedSENTs :
                parsed=self.m_parsedSENTs[i]
                fieldsi=parsed[3]
                line=string.join(fieldsi)
                self.SetLineUserUnit(i,line)
            self.fromEdit.setText(self.__tr('computer'))
            self.operationEdit.setText(self.__tr(self.m_FilePath))
            self.m_more={self.m_FilePath,0}
    else :
        self.m_FileName = self.m_concept["basePath"]
        dir_name=os.path.dirname(self.m_FileName)
        self.m_pick_filer=os.path.join(dir_name,'CD_parsed')
    try :
        book=shelve.open(self.m_pick_filer)
        self.m_book=book
    except :
        pass
    if self.m_button=='text' and not self.m_createModel:
        self.GetUserInput()
        self.m_pick_file=os.path.join(dir_name,'user_input.txt')
        filer=open(self.m_pick_file,'w')
        line=''
        for j in self.m_user_input :
            line+=self.m_user_input[j]
        filer.writelines(line)
        filer.flush()
        filer.close()

```

```

parent=self.m_parent
parent.userFile=True
parent.File=self.m_pick_file
try :
    self.m_book.close()
except :
    pass
parent.slotconceptualDependency()
try :
    book=shelve.open(self.m_pick_filer)
    self.m_book=book
except :
    pass
self.m_FilePath='user_input'
self.m_parsedSENTs=self.m_book[self.m_FilePath]
for i in self.m_parsedSENTs :
    parsed=self.m_parsedSENTs[i]
    fieldsi=parsed[3]
    matchedi=parsed[2]
    match_numsi=parsed[1]
    matchi=parsed[0]
    diali=parsed[4]
    rcounti=parsed[5]
    chunksi=parsed[6]
    inputVariables=parsed[7]
    matchi_ij=parsed[8]
    match_numsi_ij=parsed[9]
    rcounti_ij=parsed[10]
    typeri=parsed[11]
    concepteri=parsed[12]
    concepterli=parsed[13]
    concepter2i=parsed[14]
    concepter3i=parsed[15]
    concepter4i=parsed[16]
    concepter5i=parsed[17]
    primMatchi=parsed[18]
    matchPrimi=parsed[19]
    primMatchi_ij=parsed[20]
    matchPrimi_ij=parsed[21]
    self.m_Thunks=parsed[22]
    self.m_tcount=parsed[23]
    parsedi=parsed
    fleni=len(fieldsi)
    ccount=0
    self.m_chunks=chunksi
    if not concepter2i[0]=='QUESTION' and not self.m_createModel:
        for n in range(rcounti):
            if match_numsi.has_key(n) :
                if match_numsi[n][0]=='<-o' and fieldsi[match_numsi[n][1]]=='create' and
fieldsi[match_numsi[n][2]]=='finite_element' :
                    if match_numsi[n][2]+1 < fleni and fieldsi[match_numsi[n][2]+1]=='model' :
                        self.m_createModel=True
                        # set up rules here
                        dir_name=os.path.dirname(self.m_FileName)
                        filer=os.path.join(dir_name, 'fea_eps.xml')
                        cd_deck = open(filer, 'r')
                        self.m_Lines = cd_deck.readlines()
                        cd_deck.close()
                        m_Lines=[]
                        for lines in self.m_Lines :
                            flen=len(lines)
                            if flen<=1 :
                                continue
                            lines=string.replace(lines, '\n', '')
                            lines=string.replace(lines, '\t', '')
                            fields=string.split(lines, '</>')
                            if len(fields)==2 :
                                if not fields[0]==' ' :
                                    m_Lines.append(fields[0])
                                    m_Lines.append('</' + fields[1])
                            else :
                                if not lines==' ' :
                                    m_Lines.append(lines)
                        self.m_Lines=m_Lines
                        self.m_ActiveRules=[]
                        self.m_Conclusions={}
                        self.m_ConcCount={}
                        self.m_XmlElements={}
                        self.m_ValDB={}
                        # start a new rule
                        self.m_RuleCount=0
                        self.m_RulesDB={}
                        self.m_ActiveElements=[]
                        self.m_GoldenElement=None
                        self.m_PositionLabel={0}
                        self.m_LinesCount=1

```

```

self.m_UnparsedLines=[]
self.m_LastElementOpened=[]
self.m_SameElementCount={}
self.m_ActiveTime=None
self.m_ExpandedActiveRules={}
readOn=True
while readOn :
    lines=self.m_Lines[self.m_LinesCount]
    result=self.IsFirstTokenAStartElement(lines,self.m_PositionLabel[0])
    readOn=result
    if result :
        self.m_LinesCount+=1
    pass
self.m_lastI=-1
self.m_InitiateModel=True
self.PrepareEpsRules()
self.ProcessActiveRules()
if concepter2i[0]=='QUESTION' and not self.m_createModel:
    for n in range(rcounti) :
        if match_nums[i].has_key(n) :
            if match_nums[i][n][0]=='<-o' and match_nums[i][n][1]>match_nums[i][n][2] :
                searchWord=[match_nums[i][n][2]]
                for m in range(rcounti):
                    if match_nums[i].has_key(m) :
                        if match_nums[i][m][0]=='<=>' and match_nums[i][m][1]==searchWord[0] and not
match_nums[i][m][2] <match_nums[i][m][1] :
                            searchWord.append(match_nums[i][m][2])
            if match_nums[i][n][0]=='<>' and fieldsi[match_nums[i][n][1]]=='do_you' :
                for m in range(rcounti):
                    if match_nums[i].has_key(m) :
                        if match_nums[i][m][0]=='<-o' and match_nums[i][m][1]==match_nums[i][n][2] :
                            searchWord=[match_nums[i][m][2]]
                            ii=match_nums[i][m][2]-1
                            if typeri[ii]=='PA' and concepter3i[ii]==concepter3i[ii+1] :
                                searchWord.append(ii)
                for mm in range(rcounti):
                    if match_nums[i].has_key(mm) :
                        if match_nums[i][mm][0]=='<=>' and match_nums[i][mm][1]==searchWord[0] :
                            searchWord.append(match_nums[i][mm][2])
                for mm in range(rcounti):
                    if match_nums[i].has_key(mm) :
                        if match_nums[i][mm][0]=='T<=POSS-BY' and
match_nums[i][mm][1]==searchWord[0] :
                            searchWord.append(match_nums[i][mm][2])

listSearchWord=[]
for k in searchWord :
    listSearchWord.append(fieldsi[k])
searchWords=[]
for filePath in self.m_book :
    if filePath=='nleps_1' or filePath=='test11' or filePath=='user_input' :
        continue
    parsedSENTs=self.m_book[filePath]
    if filePath=='structural_mechanics_5' :
        j=1
    for j in parsedSENTs :
        parsed=parsedSENTs[j]
        try :
            Thunks=parsed[22]
            tcount=parsed[23]
            for k in searchWord :
                listT=[]
                typer1=typeri[k]
                if typer1=='ACT' and fieldsi[k] in Thunks['ACT'] :
                    listT=Thunks['ACT']
                elif typer1=='PP' and fieldsi[k] in Thunks['PP'] :
                    listT=Thunks['PP']
                elif typer1=='PA' and fieldsi[k] in Thunks['PA'] :
                    listT=Thunks['PA']
                tlen=len(listT)
                for m in range(0,tlen,2):
                    if fieldsi[k]==listT[m] :
                        dictWords=[k,filePath,listT[m+1][0],listT[m+1][1]]
                        found=False
                        for dictWor in searchWords :
                            found=True
                            for p in range(4) :
                                if not dictWords[p]==dictWor[p] :
                                    found=False
                            if found :
                                break
                        if not found :
                            searchWords.append(dictWords)
        except :
            continue
# now sort out your response
wlen=len(searchWords)

```

```

response=[]
for src in searchWords :
    out=[src[1],src[2]]
    found=False
    for resp in response :
        found=True
        if not out[0]==resp[0] or not out[1]==resp[1] :
            found=False
            if found :
                break
    if not found :
        response.append(out)
# find second order links
rlen=len(response)
searchWords=[]

for kk in range(rlen) :
    out=response[kk]
    filePath=out[0]
    lineCount=out[1]
    parsedSENTs=self.m_book[filePath]
    parsed=parsedSENTs[lineCount]
    fieldsj=parsed[3]
    match_numsj=parsed[1]
    matchj=parsed[0]
    dialj=parsed[4]
    rcountj=parsed[5]
    typerj=parsed[11]
    searchWordj=[]
    for n in range(rcountj) :
        if match_numsj.has_key(n) :
            if match_numsj[n][0]=='T<=POSS-BY' and fieldsj[match_numsj[n][2]] in listSearchWord :
                searchWordj.append(match_numsj[n][1])
                #break
    for n in range(rcountj) :
        if match_numsj.has_key(n) :
            if match_numsj[n][0]=='<-o' and fieldsj[match_numsj[n][2]] in listSearchWord :
                for m in range(rcountj):
                    if match_numsj.has_key(m) :
                        if match_numsj[m][0]=='<>' and match_numsj[n][1] == match_numsj[m][2] :
                            searchWordj.append(match_numsj[m][1])
                            #break
    listInteg=['one','two','three']
    for n in range(rcountj) :
        if match_numsj.has_key(n) :
            if match_numsj[n][0]=='<-o' :
                searchWordj.append(match_numsj[n][2])
                for m in range(rcountj):
                    if match_numsj.has_key(m) :
                        if match_numsj[m][0]=='<>' and match_numsj[n][2] == match_numsj[m][2] and
typerj[match_numsj[m][1]]=='PA':
                            if fieldsj[match_numsj[m][1]] in listInteg :
                                searchWordj.append(match_numsj[m][1])
                                searchWordj.append(match_numsj[m][1])
                                #break

    for j in parsedSENTs :
        if j==lineCount :
            continue
        parsed=parsedSENTs[j]
        try :
            Thunks=parsed[22]
            tcount=parsed[23]
            foundInteg=1
            for k in searchWordj :
                listT=[]
                typer1=typerj[k]
                fieldsjk=fieldsj[k]
                if fieldsjk in listInteg :
                    if foundInteg==1 :
                        typer1='PP'
                        fieldsjk='first'
                        foundInteg+=1
                    elif foundInteg==2 :
                        typer1='PP'
                        fieldsjk='second'
                if typer1=='ACT' and fieldsjk in Thunks['ACT'] :
                    listT=Thunks['ACT']
                elif typer1=='PP' and fieldsjk in Thunks['PP'] :
                    listT=Thunks['PP']
                elif typer1=='PA' and fieldsjk in Thunks['PA'] :
                    listT=Thunks['PA']
                tlen=len(listT)
                for m in range(0,tlen,2):
                    if fieldsjk==listT[m] :
                        dictWords=[k,filePath,listT[m+1][0],listT[m+1][1]]
                        found=False

```

```

        for dictWor in searchWords :
            found=True
            for p in range(4) :
                if not dictWords[p]==dictWor[p] :
                    found=False
            if found :
                break
            if not found :
                searchWords.append(dictWords)
        except :
            continue
# update response
for src in searchWords :
    out=[src[1],src[2]]
    found=False
    for resp in response :
        found=True
        if not out[0]==resp[0] or not out[1]==resp[1] :
            found=False
        if found :
            break
    if not found :
        response.append(out)
# sort response
responses={}
for resp in response :
    if responses.has_key(resp[0]) :
        listR=responses[resp[0]]
    else :
        listR=[]
        responses[resp[0]]=listR
    listR.append(resp[1])
for res in responses :
    listR=responses[res]
    listR.sort()
listRR=[]
for res in responses :
    listRR.append(res)
listRR.sort()
response=[]
for res in listRR :
    listR=responses[res]
    for lis in listR :
        out=[res,lis]
        response.append(out)
self.m_response=response
self.m_startResponse=0
count=-1
rang=7
for ii in range(rang) :
    self.SetLineUserUnitToZero()
rlen=len(response)
for k in range(self.m_startResponse,rlen) :
    count+=1
    out=response[k]
    self.m_FilePath=out[0]
    self.m_parsedSENTs=self.m_book[self.m_FilePath]
    ii=out[1]
    parsed=self.m_parsedSENTs[ii]
    fields=parsed[3]
    line=string.join(fields)
    self.SetLineUserUnit(count,line)
    if count==rang-2 :
        break
if count==rang-2 :
    self.m_startResponse+=6
    self.m_more=True
    self.fromEdit.setText(self.__tr('computer'))
    self.operationEdit.setText(self.__tr('response'))
    self.m_more=True
    self.SetLineUserUnit(count+1,'more')
    self.m_responseMore=True
    pass
else :
    self.m_more=False
    self.fromEdit.setText(self.__tr('computer'))
    self.operationEdit.setText(self.__tr('end response'))
    self.m_responseMore=True
    pass
if self.m_button=='direct' and not self.m_createModel:
    if self.m_more and not self.m_responseMore:
        path=self.m_more[0]
        ind=self.m_more[1]
        if ind==0 :
            pat=string.split(path,'_')
            inds=string.atoi(pat[1])

```

```

        inds+=1
        line=pat[0]+'_'+inds`
self.m_FilePath=line
if self.m_book.has_key(self.m_FilePath) :
    self.m_parsedSENTs=self.m_book[self.m_FilePath]
    rang=7
    for i in range(rang) :
        self.SetLineUserUnitToZero()
    for i in self.m_parsedSENTs :
        parsed=self.m_parsedSENTs[i]
        fieldsi=parsed[3]
        line=string.join(fieldsi)
        self.SetLineUserUnit(i,line)
    self.fromEdit.setText(self.__tr('computer'))
    self.operationEdit.setText(self.__tr(self.m_FilePath))
    self.m_more=[self.m_FilePath,0]
else :
    self.m_more=None
    rang=7
    for i in range(rang) :
        self.SetLineUserUnitToZero()
    line='No more data from this series' + ' '+self.m_FilePath
    self.SetLineUserUnit(0,line)
    self.fromEdit.setText(self.__tr('computer'))
    self.operationEdit.setText(self.__tr(self.m_FilePath))
if self.m_responseMore:
    response=self.m_response
    count=-1
    rang=7
    for ii in range(rang) :
        self.SetLineUserUnitToZero()
    rlen=len(response)
    for k in range(self.m_startResponse,rlen) :
        count+=1
        out=response[k]
        self.m_FilePath=out[0]
        self.m_parsedSENTs=self.m_book[self.m_FilePath]
        ii=out[1]
        parsed=self.m_parsedSENTs[ii]
        fields=parsed[3]
        line=string.join(fields)
        self.SetLineUserUnit(count,line)
        if count==rang-2 :
            break
    if count==rang-2 :
        self.m_startResponse+=6
        self.m_more=False
        self.fromEdit.setText(self.__tr('computer'))
        self.operationEdit.setText(self.__tr('response'))
        self.SetLineUserUnit(count+1,'more to come')
        self.m_responseMore=True
    else :
        self.m_more=False
        self.fromEdit.setText(self.__tr('computer'))
        self.operationEdit.setText(self.__tr('end response'))
        self.m_responseMore=False
        self.SetLineUserUnit(count+1,'end of output')
if self.m_createModel and not self.m_initiateModel :
    if self.m_button=='text' :
        pass
    if self.m_button=='direct' :
        pass
try :
    #book=shelve.open(self.m_pick_filer)
    #book[self.m_FilePath]=self.m_parsedSENTs
    self.m_book.close()
except :
    pass
return
def ProcessActiveRules(self):
    self.ProcessActiveRulesDeb()
def ProcessActiveRulesDeb(self):
    lastName=None
    response=[]
    inp=[]
    for i in self.m_ActiveRules :
        if i<=self.m_lastI :
            continue
        expRule=self.m_ExpandedActiveRules[i]
        #[rule,var,stat]
        rule=expRule[0]
        var=expRule[1]
        stat=expRule[2]
        name=rule[0]
        sameElementCount=rule[1]
        characters=rule[2]

```

```

variables=rule[3]
if lastName and not lastName==name :
    pass
    #lastName=name
    #self.m_lastI=i
    #break
if expRule[2]==1 :
    # remove from Active if possible
    continue
respCount=0
out=name+' '+ characters
vars=None
if var :
    vars=variables[var]
response.append(out)
inp.append(var)
respCount+=1
lastName=name
if respCount==7 :
    self.m_lastI=i
    break
self.m_startResponse=0
self.m_response=response
count=-1
rang=7
self.SetLineUserUnitToZero()
self.SetLineDirectUnitToZero()
rlen=len(response)
for k in range(self.m_startResponse,rlen) :
    count+=1
    out=response[k]
    self.SetLineUserUnit(count,out)
    out=inp[k]
    self.SetLineDirectUnit(count,out)
    self.m_more=False
    self.fromEdit.setText(self.__tr('computer'))
    self.operationEdit.setText(self.__tr('response'))
    self.m_responseMore=False
    pass
def PrepareEpsRules(self):
    self.PrepareEpsRulesDeb()
def PrepareEpsRulesDeb(self):
    #self.m_ExpandedActiveRules
    for i in self.m_ActiveRules :
        rule=self.m_RulesDB[i]
        name=rule[0]
        sameElementCount=rule[1]
        characters=rule[2]
        variables=rule[3]
        num=string.find(characters,'val_')
        var=None
        if num >=0 :
            fields=string.split(characters,'val_')
            flen=len(fields)
            if flen==2 :
                num1=string.replace(fields[1],' ','')
                var='val_'+num1
                if variables.has_key(var) :
                    vars=variables[var]
                else :
                    vars={}
                    variables[var]=vars
                    vars[sameElementCount]=[]
            stat=-1
            self.m_ExpandedActiveRules[i]=[rule,var,stat]
def IsFirstTokenAStartElement(self,lines,positionLabel) :
    result=self.IsFirstTokenAStartElementDeb(lines,positionLabel)
    return result
def IsFirstTokenAStartElementDeb(self,lines,positionLabel) :
    #self.m_ActiveRules={}
    #self.m_Conclusions={}
    #self.m_ConcCount={}
    #self.m_XmlElements={}
    #self.m_ValDB={}
    # start a new rule
    #self.m_RuleCount=0
    #self.m_rulesDB={}
    #self.m_ActiveElements=[]
    #self.m_GoldenElement=None
    #self.m_PositionLabel=[0]
    #self.m_LineCount
    #self.m_UnparsedLines=[]
    #self.m_LastElementOpened
    #self.m_SameElementCount={}
    result=False
    nameStart=None

```

```

nameEnd=None
characters=None
fields=string.split(lines,'>')
flen=len(fields)
num=string.find(fields[0], '<')
num1=string.find(fields[0], '</')
if num==0 and not num1==0 :
    nameStart=fields[0]+'>'
    num2=string.find(nameStart, '<Time')
    if num2 >=0 :
        if not nameStart in self.m_ActiveElements and self.m_ActiveTime in self.m_ActiveElements :
            result=False
            return result
        else :
            self.m_ActiveTime=nameStart
if not self.m_GoldenElement :
    self.m_GoldenElement=nameStart
    self.m_ActiveElements.append(nameStart)
    parent=[]
    self.m_LastElementOpened.append([nameStart,0,parent])
    self.m_SameElementCount[nameStart]=0
    result=True
    positionLabel+=1
    return result
if nameStart :
    if self.m_SameElementCount.has_key(nameStart) :
        self.m_SameElementCount[nameStart]+=1
    else :
        self.m_SameElementCount[nameStart]=0
    if nameStart not in self.m_ActiveElements :
        self.m_ActiveElements.append(nameStart)
    result=True
if num==0 and num1==0 :
    nameEnd='</'+fields[0]+'>'
    self.m_UnparsedLines.append(nameEnd)
    result=True
    return result
if nameStart :
    parent=[]
    self.m_LastElementOpened.append([nameStart,self.m_SameElementCount[nameStart],parent])
    if not lines==nameStart :
        characters=string.replace(lines,nameStart,'')
        variables={}
        self.m_RulesDB[self.m_RuleCount]=[nameStart,self.m_SameElementCount[nameStart],characters,variables]
        self.m_ActiveRules.append(self.m_RuleCount)
        alen=len(self.m_ActiveElements)
        if alen >=2 :
            nam=self.m_ActiveElements[alen-2]
            lmax=-1
            lmaxPos=-1
            count=-1
            for i in self.m_LastElementOpened :
                count+=1
                elem=i
                if elem[0]==nam :
                    if self.m_SameElementCount[nam] > lmax :
                        lmax=self.m_SameElementCount[nam]
                        lmaxPos=count
            if lmaxPos>=0 :
                elem=self.m_LastElementOpened[lmaxPos]
                elem[2].append([nameStart,self.m_SameElementCount[nameStart]])
            self.m_RuleCount+=1
    return result
def IsLastTokenAEndElement(self,lines,lastStartElement,positionLabel) :
    nameEnd=None
    characters=None
    fields=string.split(lines,'</')
    flen=len(fields)
    num=string.find(fields[flen-1], '>')
    if num>0 :
        nameEnd='<'+fields[flen-1]
        if flen==2 :
            characters=fields[0]
            # assumes no element in line
    else :
        fields=string.split(lines,'/>')
        if len(fields)==2 and fields[1]==' ' :
            nameEnd=lastStartElement
            characters=fields[0]
    return nameEnd,characters
def SetLineUserUnitToZero(self) :
    self.SetLineUserUnitToZeroDeb()
def SetLineUserUnitToZeroDeb(self) :
    for k in range(7) :
        j=k
        subm=''

```

```

    if j == 0:
        self.computerEdit_1.setText(self.__tr(subm))
    elif j == 1:
        self.computerEdit_2.setText(self.__tr(subm))
    elif j == 2:
        self.computerEdit_3.setText(self.__tr(subm))
    elif j == 3:
        self.computerEdit_4.setText(self.__tr(subm))
    elif j == 4:
        self.computerEdit_5.setText(self.__tr(subm))
    elif j == 5:
        self.computerEdit_6.setText(self.__tr(subm))
    elif j == 6:
        self.computerEdit_7.setText(self.__tr(subm))
def SetLineUserUnit(self, j, subm) :
    self.SetLineUserUnitDeb(j, subm)

def SetLineUserUnitDeb(self, j, subm) :
    for k in range(7) :
        if not j==k :
            continue
        if j == 0:
            self.computerEdit_1.setText(self.__tr(subm))
        elif j == 1:
            self.computerEdit_2.setText(self.__tr(subm))
        elif j == 2:
            self.computerEdit_3.setText(self.__tr(subm))
        elif j == 3:
            self.computerEdit_4.setText(self.__tr(subm))
        elif j == 4:
            self.computerEdit_5.setText(self.__tr(subm))
        elif j == 5:
            self.computerEdit_6.setText(self.__tr(subm))
        elif j == 6:
            self.computerEdit_7.setText(self.__tr(subm))
def SetLineDirectUnitToZero(self) :
    self.SetLineDirectUnitToZeroDeb()
def SetLineDirectUnitToZeroDeb(self) :
    for k in range(7) :
        j=k
        subm=''
        if j == 0:
            self.directEdit_1.setText(self.__tr(subm))
        elif j == 1:
            self.directEdit_2.setText(self.__tr(subm))
        elif j == 2:
            self.directEdit_3.setText(self.__tr(subm))
        elif j == 3:
            self.directEdit_4.setText(self.__tr(subm))
        elif j == 4:
            self.directEdit_5.setText(self.__tr(subm))
        elif j == 5:
            self.directEdit_6.setText(self.__tr(subm))
        elif j == 6:
            self.directEdit_7.setText(self.__tr(subm))
def SetLineDirectUnit(self, j, subm) :
    self.SetLineDirectUnitDeb(j, subm)
def SetLineDirectUnitDeb(self, j, subm) :
    for k in range(7) :
        if not j==k :
            continue
        if j == 0:
            self.directEdit_1.setText(self.__tr(subm))
        elif j == 1:
            self.directEdit_2.setText(self.__tr(subm))
        elif j == 2:
            self.directEdit_3.setText(self.__tr(subm))
        elif j == 3:
            self.directEdit_4.setText(self.__tr(subm))
        elif j == 4:
            self.directEdit_5.setText(self.__tr(subm))
        elif j == 5:
            self.directEdit_6.setText(self.__tr(subm))
        elif j == 6:
            self.directEdit_7.setText(self.__tr(subm))
def GetUserDirect(self) :
    self.GetUserDirectDeb()
def GetUserDirectDeb(self) :
    self.m_user_direct={}
    for j in range(7) :
        if j==0 :
            inp=self.directEdit_1.text().ascii()
            if not inp==self.m_zero :
                self.m_user_direct[j]=inp+'\n'
        elif j==1 :
            inp=self.directEdit_2.text().ascii()

```

```

        if not inp==self.m_zero :
            self.m_user_direct[j]=inp+'\n'
    elif j==2 :
        inp=self.directEdit_3.text().ascii()
        if not inp==self.m_zero :
            self.m_user_direct[j]=inp+'\n'
    elif j==3 :
        inp=self.directEdit_4.text().ascii()
        if not inp==self.m_zero :
            self.m_user_direct[j]=inp+'\n'
    elif j==4 :
        inp=self.directEdit_5.text().ascii()
        if not inp==self.m_zero :
            self.m_user_direct[j]=inp+'\n'
    elif j==5 :
        inp=self.directEdit_6.text().ascii()
        if not inp==self.m_zero :
            self.m_user_direct[j]=inp+'\n'
    elif j==6 :
        inp=self.directEdit_7.text().ascii()
        if not inp==self.m_zero :
            self.m_user_direct[j]=inp+'\n'
def GetUserInput(self) :
    self.GetUserInputDeb()
def GetUserInputDeb(self) :
    self.m_user_input={}
    for j in range(5) :
        if j==0 :
            inp=self.user_1.text().ascii()
            if not inp==self.m_zero :
                self.m_user_input[j]=inp+'\n'
        elif j==1 :
            inp=self.user_2.text().ascii()
            if not inp==self.m_zero :
                self.m_user_input[j]=inp+'\n'
        elif j==2 :
            inp=self.user_3.text().ascii()
            if not inp==self.m_zero :
                self.m_user_input[j]=inp+'\n'
        elif j==3 :
            inp=self.user_4.text().ascii()
            if not inp==self.m_zero :
                self.m_user_input[j]=inp+'\n'
        elif j==4 :
            inp=self.user_5.text().ascii()
            if not inp==self.m_zero :
                self.m_user_input[j]=inp+'\n'
def SaveButton_clicked(self):
    self.SaveButton_clickedDeb()
def SaveButton_clickedDeb(self):
    self.m_FileName = self.m_concept["basePath"]
    dir_name=os.path.dirname(self.m_FileName)
    self.m_pick_file=os.path.join(dir_name,'CD_pickle_new')
    try :
        #book[self.m_FilePath]=self.m_parsedSENTS
        #book.close()
        self.m_book.close()
    except :
        pass
def __tr(self,s,c = None):
    return QApplication.translate("Concept_Nleps_Dialog",s,c)
# -----
# We now add a link with Dataplot to run a code named "pedro8.dp" to read the ANLAP output
# table and create postscript files of report-quality plots of raw and processed data
# using two software packages: (1) Dataplot and (2) Ghostview. For more details of
# those two public-domain packages, see http://www.itl.nist.gov/div898/software/dataplot.html
#
# For a complete listing of the Dataplot code, "pedro8.dp," see Appendix B.
# -----
def readInEntry3(self):
    v=self.v.get()
    if v==1 :
        pass
    if v==2 :
        returncode=None
        try :
            returncode= call('C:/Program Files/NIST/DATAPLOT/dataplot < pedro8.dp' )
            pass
            if returncode :
                print 'Failure with return code' ,returncode
        except :
            pass
    i=1
    os.spawnl(os.P_NOWAIT,'C:/Ghostgum/gsview/gsview32.exe', ' C:/0____dataplot/0_test/DPPL1F.DAT')

```

APPENDIX B

In the example described in Sections V, we applied an ANLAP-Dataplot link using five Dataplot codes, "pedro8.dp," "pedro9.dp," "pedro9pie.dp," "pedro10.dp," and "pedro11.dp." The listing of all five such codes is available for download at <http://math.nist.gov/mcsd/software.html>. These Dataplot codes add to ANLAP an analysis capability beyond its artificial intelligence feature of reading a technical report and extracting a table of data in a user-specified format. To give the reader some flavor of the powerful and easy-to-learn statistical analysis and graphics package named Dataplot, we show below the complete text of "pedro8.dp," which is the first of the five codes mentioned in this paper:

```
. -----  
. Date: Mar. 14, 2009      Filename:      pedro8.dp  
.                               Associated Data File:      CD_tabletest_11.txt  
. -----  
. By:  Jeffrey T. Fong, Div. 891, NIST, Gaithersburg MD (fong@nist.gov)  
.                               Re:      Multiplot -- This is a two-plot exercise  
. -----  
dimension 250 rows; column limit 30 80  
. ----- read data from input file  
.                               CD_tabletest_11.txt where  
.                               data begins at line 3  
let kk = 2; let kkl = kk+1  
.                               --- Note: kk = 2 denotes 2 lines to be skipped  
skip kk  
read matrix CD_tabletest_11.txt z; let n = size z1  
.                               --- input file read once  
let zt = matrix transpose z; let m = size zt1; let ml = m+1  
let xm = sequence 1 1 m  
. ----- Plot Preliminaries  
case asis; title case asis; label case asis  
.   
tic offset units data  
xlimits 1 m; xtic offset 1 1  
major xtic mark number m; minor xtic mark number 0  
xllabel displacement 18  
xllabel Type of Root Causes  
x3label off  
.   
ylimits 0 100; ytic offset 0.5 0.5  
major ytic mark number ml; minor ytic mark number 1  
yllabel Occurrence in Percent; yllabel displacement 18  
.   
line blank blank solid; char blank all  
bar on on; bar width 0.5 0.5  
bar fill on off; bar fill color black white  
bar border on on; bar border color black black  
. ----- Multi-plot begins  
. A command, Multiplot rn cn, means rn*cn plots per page with  
. rn rows and cn columns of several plots  
. Example:  rn = 1  cn = 2  for a command, multiplot 1 2, means  
. one row and two columns of plots on a single page with a total  
. of 1*2 = 2 plots, i.e., one left plot and one right plot  
.   
multiplot 1 n; multiplot scale factor 3.2  
multiplot corner coordinates 5 15 100 75  
column limits 1 20  
title offset 2
```

APPENDIX B (CONT'D)

```
. ----- loop for multiplot begins
loop for k = 1 1 n
  let kskip = kk + k - 1
  skip kskip

. ----- read plot sub-heading
  read string CD_tabletest_11.txt s^k
.                                     ---- input file read twice
  if k = 1
    let zt1 = zt1*0.001
.                                     --- fictitious data suppressed
  end if
  title ^s^k
  plot zt^k vs xm
end of loop
end of multiplot

. ----- Annotation begins
.
. ----- Annotation Part 1.  Left box -----
.
justification center
hw 5 2.5
move 28.5 60; text Fictitious data
hw 3 1.5
move 28.5 55; text to be plotted
move 28.5 50; text to test the
move 28.5 45; text analysis
move 28.5 40; text capability of
hw 4 2
move 28.5 33; text ANLAP-DATAPLOT
.
. ----- Annotation Part 2.  Right box -----
.
skip kk1
column limits 21 30
read parameter CD_tabletest_11.txt NOC
.                                     ---- input file read thrice
.
let nyear = 8.83
let Npa = NOC / nyear
let Npd = Npa / 365
let Npa = round(Npa, 0)
let Npd = round(Npd, 2)
.
hw 3 1.5
justification left
move 62 65; text [ Data from DOE Report ]
move 62 60; text Total number of events
move 62 56; text = ^NOC
move 62 51; text No. of years = ^nyear
move 62 46; text Ave. no. of events
move 62 42; text = ^Npa per annum.
move 62 38; text = ^Npd per day.
.
exit
.
. ----- END OF Dataplot CODE, pedro8.dp -----
```