SOFTWARE SUMMARY FORM

01.Summary Date: 03/07/94	02. Summary prepared by(Name and Phone) T.J. Ratchford 522-3083	03. Summary Action: New
04. Software Date: 8/15/93	05. Short Title: DITTY	
06. Software Title: DITTY - Calculate total population exposure over a long time period.		07. Internal Software ID: NONE

08. Software Type:	09.Processing Mode:	10. APPLICATION AREA
☐ Automated Data System	☐ Interactive	A. General:
☐ Computer Program	☐ Batch	☐ Scientific/Engineering ☐ Auxiliary Analyses
■ Subroutine/Module	■ Combination	☐ Total System PA
		■ Subsystem PA ☐ Other
		b. Specific:

11. Submitting Organization and Address:	12. Technical Contact(s) and Phone:
CNWRA, SwRI, San Antonio, Texas	R. Janetzke, (210) 522-3318

13. Narrative:

The DITTY code is used by the Total Performance Assessment Code to calculate total population exposure over a long time period.

14. Computer Platform	15. Computer Operating System:	16. Programming Language(s):	17. Number of Source Program Statements:
CRAY/XMP	UNIX	FORTRAN	27,649 lines of code
18. Computer Memory Requirements: UNKNOWN	19. Tape Drives: NONE	20. Disk/Drum Units: N/A	21. Graphics: UNKNOWN

22. Other Operational Requirements

NONE

23. Software Availability:	24. Documentation Availability:
■ Available ☐ Limited ☐ In-House ONLY	■ Available ☐ Inadequate ☐ In-House ONLY

25. Submission Package Status:

Acceptance Criteria: Met ■ Not Met ☐ Software QA Assessment: Successful ■ Unsuccessful ☐

Code Custodian: T. J. Ratchford Date: 3/7/94

CNWRA Form TOP-4-1 (08/93)

DITTY CRAY DIRECTORY LISTING

```
-rw-rw----  1 tjr1          947 Mar  7 06:29 TPA_DIT.DGD
-rw-rw----  1 tjr1         5895 Mar  7 06:29 actin.F
-rw-rw----  1 tjr1          972 Mar  7 06:29 aircon_c.H
-rw-rw----  1 tjr1         2855 Mar  7 06:29 airlin.F
-rw-rw----  1 tjr1         1782 Mar  7 06:29 airpar_c.H
-rw-rw----  1 tjr1         8353 Mar  7 06:29 airrel.in
-rw-rw----  1 tjr1         1701 Mar  7 06:29 anmpar_c.H
-rw-rw----  1 tjr1         5559 Mar  7 06:29 apaths.F
-rw-rw----  1 tjr1         3216 Mar  7 06:29 bchain.F
-rw-rw----  1 tjr1         6639 Mar  7 06:29 bioac1.dat
-rw-rw----  1 tjr1         1620 Mar  7 06:29 biodat_c.H
-rw-rw----  1 tjr1         1857 Mar  7 06:29 biolin.F
-rw-rw----  1 tjr1         1110 Mar  7 06:29 blkorg.F
-rw-rw----  1 tjr1         2705 Mar  7 06:29 block2.F
-rw-rw----  1 tjr1         1420 Mar  7 06:29 blockd.F
-rw-rw----  1 tjr1         1871 Mar  7 06:29 carbon.F
-rw-rw----  1 tjr1         6110 Mar  7 06:29 case2.F
-rw-rw----  1 tjr1         4309 Mar  7 06:29 conset.F
-rw-rw----  1 tjr1         2595 Mar  7 06:29 contrl.F
-rw-rw----  1 tjr1         1134 Mar  7 06:29 day_cmn.H
-rw-rw----  1 tjr1          972 Mar  7 06:29 daypc_cm.H
-rw-rw----  1 tjr1         1053 Mar  7 06:29 decay_cm.H
-rw-rw----  1 tjr1         2349 Mar  7 06:29 dispsn_c.H
-rw-rw----  1 tjr1          437 Mar  7 06:29 ditapoc.in
-rw-rw----  1 tjr1         3677 Mar  7 06:29 ditbyp.F
-rw-rw----  1 tjr1         5002 Mar  7 06:29 ditbyr.F
-rw-rw----  1 tjr1         9437 Mar  7 06:29 ditty.F
-rw-rw----  1 tjr1       165476 Mar  7 06:30 ditty.cpp
-rw-rw----  1 tjr1       194977 Mar  7 06:30 ditty.pre
-rw-rw----  1 tjr1          420 Mar  7 06:30 ditwpoc.in
-rw-rw----  1 tjr1         1620 Mar  7 06:30 dkay_cmn.H
-rw-rw----  1 tjr1         5375 Mar  7 06:30 dosadd.F
-rw-rw----  1 tjr1         1458 Mar  7 06:30 dosfac_c.H
-rw-rw----  1 tjr1        84279 Mar  7 06:30 dsfct30.dat
-rw-rw----  1 tjr1         2673 Mar  7 06:30 edcn_cmn.H
-rw-rw----  1 tjr1         2214 Mar  7 06:30 eovrq.F
-rw-rw----  1 tjr1         1481 Mar  7 06:30 filerr.F
-rw-rw----  1 tjr1         1053 Mar  7 06:30 files_cm.H
-rw-rw----  1 tjr1         2939 Mar  7 06:30 fillup.F
-rw-rw----  1 tjr1         1296 Mar  7 06:30 flags_cm.H
-rw-rw----  1 tjr1         1134 Mar  7 06:30 fodata_c.H
-rw-rw----  1 tjr1         1970 Mar  7 06:30 foolin.F
-rw-rw----  1 tjr1         7020 Mar  7 06:30 ftrans.dat
-rw-rw----  1 tjr1         5268 Mar  7 06:30 graph.F
-rw-rw----  1 tjr1         1215 Mar  7 06:30 grddat_c.H
-rw-rw----  1 tjr1        17119 Mar  7 06:30 grdf.dat
-rw-rw----  1 tjr1         1899 Mar  7 06:30 grdlin.F
-rw-rw----  1 tjr1         1819 Mar  7 06:30 headit.F
-rw-rw----  1 tjr1        24214 Mar  7 06:30 jointfre.in
-rw-rw----  1 tjr1       344250 Mar  7 06:30 lhsoooo.ou
-rw-rw----  1 tjr1          634 Mar  7 06:30 makda2.F
-rw-rw----  1 tjr1         2511 Mar  7 06:30 namlst_c.H
-rw-rw----  1 tjr1         1620 Mar  7 06:30 nucnam_c.H
-rw-rw----  1 tjr1         1587 Mar  7 06:30 nuctst.F
-rw-rw----  1 tjr1         7726 Mar  7 06:30 opnfil.F
-rw-rw----  1 tjr1         5103 Mar  7 06:31 opt_cmn.H
-rw-rw----  1 tjr1         2916 Mar  7 06:31 option_c.H
-rw-rw----  1 tjr1         1377 Mar  7 06:31 orgid_cm.H
-rw-rw----  1 tjr1         1134 Mar  7 06:31 orgmas_c.H
-rw-rw----  1 tjr1         1944 Mar  7 06:31 pathin_c.H
-rw-rw----  1 tjr1         1863 Mar  7 06:31 plot_cmn.H
```

```
-rw-rw----   1 tjr1          1478 Mar   7 06:31 pmac.F
-rw-rw----   1 tjr1           967 Mar   7 06:31 pmeq.F
-rw-rw----   1 tjr1          2818 Mar   7 06:31 pmint.F
-rw-rw----   1 tjr1          1034 Mar   7 06:31 pmset.F
-rw-rw----   1 tjr1          1025 Mar   7 06:31 poppop.in
-rw-rw----   1 tjr1           891 Mar   7 06:31 popu_cmn.H
-rw-rw----   1 tjr1         16982 Mar   7 06:31 qapage.F
-rw-rw----   1 tjr1          2619 Mar   7 06:31 rddat.F
-rw-rw----   1 tjr1          3405 Mar   7 06:31 rdmap.F
-rw-rw----   1 tjr1         15139 Mar   7 06:31 report.F
-rw-rw----   1 tjr1          5427 Mar   7 06:31 result_c.H
-rw-rw----   1 tjr1          4416 Mar   7 06:31 rlibin.F
-rw-rw----   1 tjr1          1134 Mar   7 06:31 rmd2_cmn.H
-rw-rw----   1 tjr1          1620 Mar   7 06:31 rmd_cmn.H
-rw-rw----   1 tjr1         13070 Mar   7 06:31 rmdlib.dat
-rw-rw----   1 tjr1          2993 Mar   7 06:31 setnuc.F
-rw-rw----   1 tjr1          1769 Mar   7 06:31 setup.F
-rw-rw----   1 tjr1          2370 Mar   7 06:31 sigma.F
-rw-rw----   1 tjr1           972 Mar   7 06:31 source_c.H
-rw-rw----   1 tjr1           648 Mar   7 06:31 times2_c.H
-rw-rw----   1 tjr1           810 Mar   7 06:31 titl_cmn.H
-rw-rw----   1 tjr1          2728 Mar   7 06:31 tritium.F
-rw-rw----   1 tjr1          1620 Mar   7 06:31 varybl_c.H
-rw-rw----   1 tjr1         23521 Mar   7 06:31 watrel.in
-rw-rw----   1 tjr1          4457 Mar   7 06:31 wbede2.F
-rw-rw----   1 tjr1          4071 Mar   7 06:31 wbede4.F
-rw-rw----   1 tjr1          6441 Mar   7 06:31 wpaths.F
-rw-rw----   1 tjr1           360 Mar   7 06:31 x_ditty.covr
-rw-rw----   1 tjr1           747 Mar   7 06:32 x_ditty.test
-rw-rw----   1 tjr1           630 Mar   7 06:32 zeroi.F
-rw-rw----   1 tjr1           629 Mar   7 06:32 zeror.F
```

MJR alcliffe 3/7/94

2 of 2

DITTY Fortran Program
Static and Dynamic Analysis

June 28, 1993

Earl S. Marwil
John E. Tolli
Scientific Computing Unit
Idaho National Engineering Laboratory

1. Introduction

This analysis was performed on the Cray version of the software as provided by Southwest Research Institute (SwRI).

One sample problem was supplied along with the source code. The program was analyzed using the Craft (Cross Reference Analysis of Fortran) tool, FORWARN, the Fortran 77 analyzer, and PC-Metric. These tools provide static analysis, coverage analysis, and complexity analysis.

2. References

[1] N.H. Marshall and E.S. Marwil, Cross Reference Analysis of Fortran (CRAFT), EG&G-CATT-9198, EG&G Idaho, Inc., July 1991.
[2] Fortran 77 Analyzer User's Manual, National Bureau of Standards, NBS GCR 81-359, 1981
[3] FORWARN User's Guide, Quibus Enterprises, Inc., July 1991.
[4] PC-Metric User's Guide, SET Laboratories, Inc., 1987.

3. Functions

The DITTY program contains 44 Fortran routines.

DITTY has no alternate entry points.

There is 1 uncalled subroutine ("rdmap").

4. Common Block Irregularities

There are 37 common blocks in the DITTY program.

All common block declarations are consistent.

Common blocks "airpar" and "anmpar" are declared only once (in block data module "blockd"), and could be eliminated from the program. Common block "decay" could also be eliminated since it is declared only once, in block data module "block2". Common blocks "rmd2" and "rmd2c" are declared only once (in "rlibin"), and their

contents are not used except for "nidrmd" in "rmd2"; these two common blocks can be eliminated and "nidrmd" made local to "rlibin".

Common blocks "daypc" and "opt2" can be eliminated from the program since none of their contents are ever used. Also, there are several common blocks which contain some variables that are never used.

There are many instances of a common block being declared in a routine in which none of its elements are otherwise referenced.

5. Interface Irregularities

No notable problems.

6. Local Variable Irregularities

All parameter usage is consistent.

There are several instances of a variable being undefined, unused, or both in a module in which it is declared.

7. Fortran Extensions

Common blocks "source" and "nucnam" have mixed character and noncharacter items.

Modules "actin", "bchain", "blkorg", "block2", "blockd", "report", "wbede2", and "wbede4" all contain REAL*8 declarations. Modules "airlin", "headit", "makda2", "opnfil", "qapage", and "report" all contain INTEGER*2 declarations.

Modules "bchain", "qapage", and "report" all have format statements containing fields which are not separated by a comma.

Module "case2" contains a NAMELIST statement.

Modules "opnfil", "rddat", "rdmap", and "report" all contain some lower case alphabetic characters in their active Fortran.

Modules "opnfil" and "rdmap" have possibly overlapping character string assignment statements.

Module "report" has 3 format statements containing the "x" descriptor without a repeat count.

8. Optimization

The following table summarizes the performance data gathered from execution of the sample problem. Only those routines exercised by the sample problem are shown (see "Coverage Analysis" for a list of routines not exercised by the sample problem, i.e., coverage = 0%). The table lists all program modules in descending order according to CPU time. To optimize code execution time, emphasis should be placed on those modules which appear highest in the listing.

In order to obtain meaningful statistics for performance evaluation, the program should execute for a reasonable amount of time. Note that the execution time for this sample problem is short (< 10 sec) and that the resulting statistics may therefore not accurately reflect program performance for longer runs.

The performance data show that a high percentage of the overall execution time (94.202%) is spent in the first 3 routines listed (DITTY, AIRLIN, BCHAIN). This is due primarily to the following (applies to some or all of the 3 routines):

1) a low percentage of floating point operations which are performed in vector mode (%Vflops is small)

2) a high level of memory conflicts (MC/MR > 1)

3) a high rate of instruction buffer fetches (IBFR > 1)

A detailed optimization analysis effort should focus on these 3 areas.

PERFORMANCE DATA FOR DITTY

ROUTINE NAME	Time	%ExTime	%AccumT	%Vflops	IFact	MC/MR	IBFR
DITTY	3.757	71.483	71.483	0.00036	0.00	2.360	0.193
AIRLIN	0.637	12.115	83.598	0.00000	0.00	0.821	1.016
BCHAIN	0.557	10.604	94.202	28.22436	0.36	1.279	1.172
RLIBIN	0.056	1.058	95.260	0.00000	0.00	1.553	1.058
REPORT	0.045	0.865	96.125	0.00000	0.00	1.301	0.869
QAPAGE	0.033	0.628	96.752	39.05183	0.00	2.099	0.937
GRDLIN	0.030	0.564	97.316	0.00000	0.00	0.688	1.014
ACTIN	0.029	0.555	97.872	0.00000	0.00	0.967	0.919
DITBYR	0.017	0.328	98.200	0.00000	0.00	2.084	0.945
FOOLIN	0.016	0.309	98.509	0.00000	0.00	0.721	0.871
HEADIT	0.015	0.292	98.800	0.00000	0.00	2.009	0.323
BIOLIN	0.015	0.285	99.086	0.00000	0.00	0.357	0.846
CASE2	0.015	0.281	99.367	16.04816	0.00	0.412	0.930
OPNFIL	0.009	0.177	99.543	0.00000	0.00	0.743	0.587
CONSET	0.006	0.122	99.666	0.00000	0.00	3.065	0.021
SETNUC	0.005	0.102	99.768	0.00000	0.00	0.994	0.072
ZEROR	0.005	0.098	99.866	0.00000	0.56	0.106	0.113
DITBYP	0.002	0.046	99.912	0.00000	0.00	2.017	0.900
WBEDE4	0.002	0.034	99.946	82.25590	0.02	1.211	0.319
EOVRQ	0.001	0.018	99.964	85.06788	0.00	0.399	0.433
SIGMA	0.001	0.012	99.976	0.00000	0.03	0.538	0.480
CARBON	0.001	0.010	99.986	0.00000	0.33	3.713	0.823
RDDAT	0.000	0.007	99.992	0.00000	0.00	0.147	1.130
PMINT	0.000	0.004	99.996	0.00000	0.00	0.006	0.021
MAKDA2	0.000	0.003	99.999	0.00000	0.00	1.475	0.728
PMEQ	0.000	0.000	100.000	79.48718	0.00	0.281	0.305
NUCTST	0.000	0.000	100.000	0.00000	0.00	0.026	0.319
ZEROI	0.000	0.000	100.000	0.00000	0.01	1.248	0.345
PMSET	0.000	0.000	100.000	33.33334	0.00	0.000	1.085
Totals (All Traced Routines)	5.255	100.000	100.000	7.43150	0.05	1.956	0.434

```
Key:
      %AccumT - accumulated percentage of total CPU time
      %ExTime - percentage of total CPU time
      %Vflops - percentage of floating point operations due
                to vector floating point operations
      IBFR    - Instruction Buffer Fetch Rate (megafetches/sec)
      IFact   - Inline Factor (total calls to routine /
                average time spent in routine for each call)
      MC      - number of memory conflicts
      MR      - number of memory references
      Time    - total CPU time (sec)
```

9. Coverage Analysis

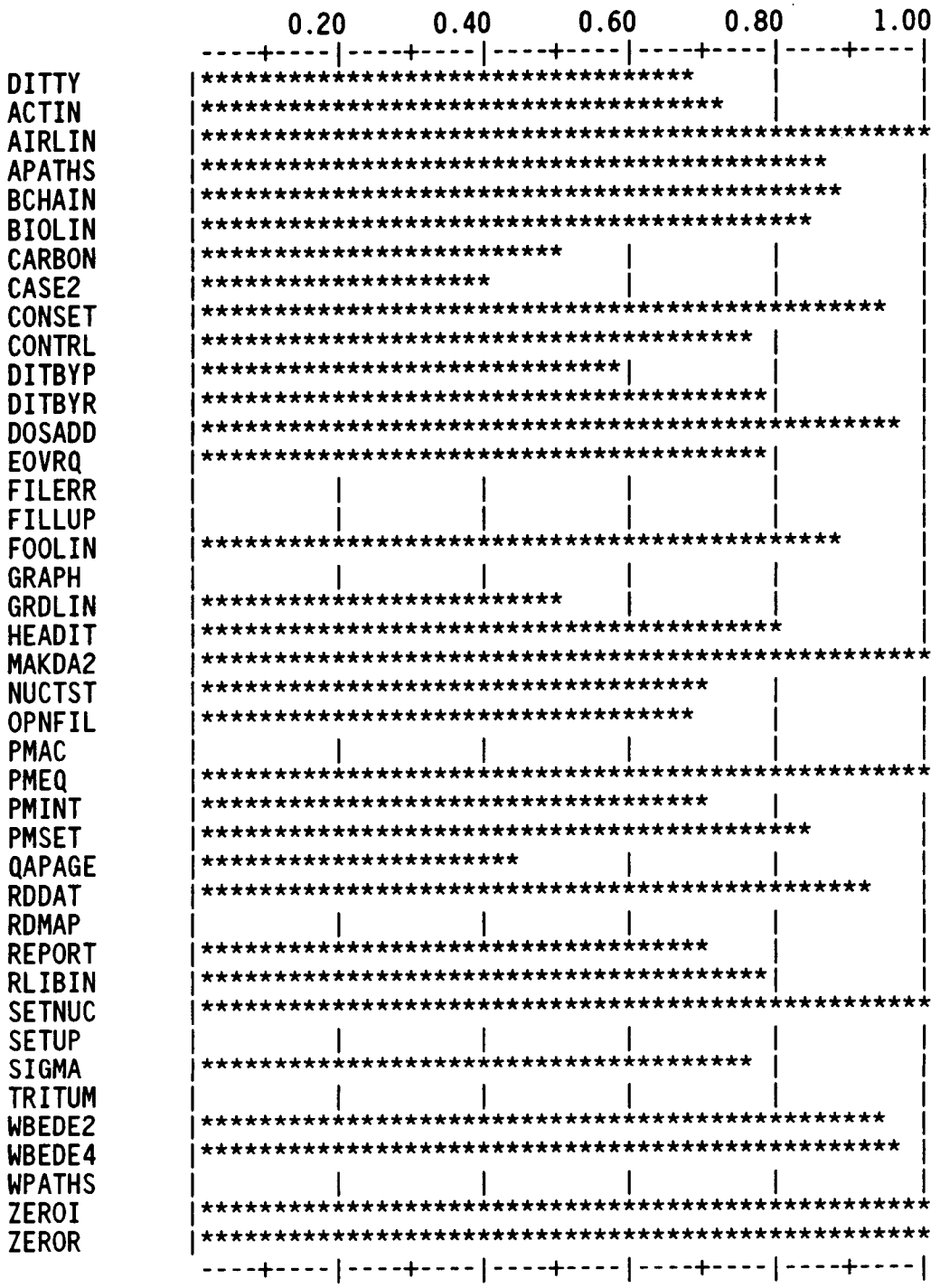
One sample problem was supplied. A coverage analysis shows that this problem yielded a 60% segment coverage of DITTY. Sample problems provided with simulation programs typically achieve 35% to 50% coverage. A statement of software quality cannot be made for routines that have low coverage, i.e., large portions of the code are untested.

Note that 8 routines have 0% coverage. These routines are not tested with the supplied sample problem.

One routine achieves 20%-39% coverage, 4 routines achieve 40%-59% coverage, 12 routines achieve 60%-79% coverage, 10 routines achieve 80%-99% coverage, and 6 routines achieve 100% coverage.

The following table shows the percent coverage for each routine.

Module Name	Number of Segments in module	Number of Segments Executed	Percent Segment Coverage
DITTY	16	11	68.7
ACTIN	54	39	72.2
AIRLIN	27	27	100.0
APATHS	43	37	86.0
BCHAIN	42	37	88.1
BIOLIN	13	11	84.6
CARBON	8	4	50.0
CASE2	68	27	39.7
CONSET	34	32	94.1
CONTRL	24	18	75.0
DITBYP	14	8	57.1
DITBYR	27	21	77.8
DOSADD	33	32	97.0
EOVRQ	23	18	78.3
FILERR	8	0	0.0
FILLUP	30	0	0.0
FOOLIN	9	8	88.9
GRAPH	32	0	0.0
GRDLIN	14	7	50.0
HEADIT	5	4	80.0
MAKDA2	1	1	100.0
NUCTST	10	7	70.0
OPNFIL	92	62	67.4
PMAC	12	0	0.0
PMEQ	5	5	100.0
PMINT	20	14	70.0
PMSET	6	5	83.3
QAPAGE	170	76	44.7
RDDAT	13	12	92.3
RDMAP	38	0	0.0
REPORT	115	81	70.4
RLIBIN	22	17	77.3
SETNUC	21	21	100.0
SETUP	17	0	0.0
SIGMA	8	6	75.0
TRITUM	8	0	0.0
WBEDE2	37	35	94.6
WBEDE4	33	32	97.0
WPATHS	53	0	0.0
ZEROI	3	3	100.0
ZEROR	3	3	100.0
Totals	1211	721	59.5


0.20
0.40
0.60
0.80
1.00
DITTY
ACTIN
AIRLIN
APATHS
BCHAIN
BIOLIN
CARBON
CASE2
CONSET
CONTRL
DITBYP
DITBYR
DOSADD
EOVRQ
FILERR
FILLUP
FOOLIN
GRAPH
GRDLIN
HEADIT
MAKDA2
NUCTST
OPNFIL
PMAC
PMEQ
PMINT
PMSET
QAPAGE
RDDAT
RDMAP
REPORT
RLIBIN
SETNUC
SETUP
SIGMA
TRITUM
WBEDE2
WBEDE4
WPATHS
ZEROI
ZEROR

coverage = 0.	FILERR SETUP	FILLUP TRITUM	GRAPH WPATHS	PMAC	RDMAP
0.20 <= coverage < 0.40	CASE2				
0.40 <= coverage < 0.60	CARBON	DITBYP	GRDLIN	QAPAGE	
0.60 <= coverage < 0.80	DITTY HEADIT RLIBIN	ACTIN NUCTST SIGMA	CONTRL OPNFIL	DITBYR PMINT	EOVRQ REPORT
0.80 <= coverage < 0.85	BIOLIN	PMSET			
0.85 <= coverage < 0.90	APATHS	BCHAIN	FOOLIN		
0.90 <= coverage < 0.95	CONSET	RDDAT	WBEDE2		
0.95 <= coverage < 1.00	DOSADD	WBEDE4			
coverage = 1.00	AIRLIN ZEROR	MAKDA2	PMEQ	SETNUC	ZEROI

Program coverage for this run =0.60

10. Complexity Analysis

Some key metrics are the number of executable statements (sloc), the number of non-blank comments (ncomt), McCabe's extended cyclomatic complexity (vg2), the number of branching statements (cgoto, ugoto, bIF, and lIF), and Halstead's predicted number of errors in (re)writing the code (bhat). Measures are normalized per 100 executable statements for ease of comparison and are listed in the table below.

The branching measures for this code indicate very few unconditional GO TO statements and logical IFs for most program modules. This code appears to be well structured.

Most routines have a good ratio of non-blank comments to source code. The lowest is "bchain" at 26.8%.

McCabe's extended cyclomatic complexity (vg2), normalized per 100 lines of source code, indicates relatively high values. Generally, the routines with the highest complexity are those most likely to have defects. As a guideline, normalized measures of 15 or greater should be considered complex. A software maintenance program should focus on those routines with the highest measures.

Complexity Report by Subprogram for DITTY

Name	loc	sloc	cmnt	ncomt	ncomt /sloc	vg2 /sloc	cgoto	cgoto /sloc	ugoto	ugoto /sloc	bIF	bif /sloc	lIF	lif /sloc	Bhat
DITTY	371	40	375	298	745.0	10.0	0	0.0	1	2.5	0	0.0	2	5.0	1
ACTIN	273	84	162	110	131.0	29.8	0	0.0	2	2.4	9	10.7	7	8.3	2
AIRLIN	217	63	124	77	122.2	17.5	0	0.0	3	4.8	3	4.8	2	3.2	1
APATHS	314	69	207	131	189.9	27.5	0	0.0	0	0.0	10	14.5	0	0.0	1
BCHAIN	125	71	51	19	26.8	25.4	0	0.0	1	1.4	6	8.5	2	2.8	1
BIOLIN	102	23	74	48	208.7	21.7	0	0.0	1	4.3	2	8.7	0	0.0	0
CARBON	100	23	58	34	147.8	21.7	0	0.0	0	0.0	2	8.7	0	0.0	0
CASE2	416	121	267	183	151.2	28.9	0	0.0	1	0.8	14	11.6	7	5.8	2
CONSET	200	82	117	65	79.3	22.0	0	0.0	6	7.3	8	9.8	6	7.3	1
CONTRL	218	47	153	112	238.3	34.0	0	0.0	0	0.0	3	6.4	2	4.3	1
DITBYP	201	43	115	88	204.7	18.6	0	0.0	0	0.0	3	7.0	1	2.3	1
DITBYR	236	75	117	90	120.0	18.7	0	0.0	0	0.0	5	6.7	3	4.0	2
DOSADD	361	74	243	182	245.9	24.3	0	0.0	0	0.0	3	4.1	2	2.7	2
EOVRQ	134	44	83	48	109.1	31.8	0	0.0	0	0.0	4	9.1	2	4.5	1
FILERR	49	13	23	13	100.0	23.1	0	0.0	0	0.0	1	7.7	0	0.0	0
FILLUP	184	68	107	50	73.5	22.1	0	0.0	0	0.0	6	8.8	4	5.9	1
FOOLIN	99	22	69	46	209.1	13.6	0	0.0	1	4.5	1	4.5	0	0.0	0
GRAPH	222	72	116	67	93.1	20.8	0	0.0	0	0.0	5	6.9	1	1.4	1
GRDLIN	92	32	53	38	118.8	18.8	0	0.0	3	9.4	2	6.3	2	6.3	0
HEADIT	89	9	59	37	411.1	11.1	0	0.0	0	0.0	0	0.0	0	0.0	0
MAKDA2	32	4	25	14	350.0	25.0	0	0.0	0	0.0	0	0.0	0	0.0	0
NUCTST	92	15	68	44	293.3	33.3	0	0.0	0	0.0	2	13.3	1	6.7	0
OPNFIL	223	114	104	80	70.2	21.1	0	0.0	5	4.4	7	6.1	15	13.2	2
PMAC	62	21	42	18	85.7	28.6	0	0.0	1	4.8	3	14.3	1	4.8	0
PMEQ	57	8	48	28	350.0	37.5	0	0.0	0	0.0	0	0.0	0	0.0	0
PMINT	109	34	73	37	108.8	32.4	0	0.0	1	2.9	5	14.7	1	2.9	0
PMSET	38	9	29	14	155.6	33.3	0	0.0	0	0.0	1	11.1	0	0.0	0
QAPAGE	826	267	426	243	91.0	31.8	0	0.0	3	1.1	33	12.4	16	6.0	4
RDDAT	140	23	109	93	404.3	8.7	0	0.0	0	0.0	1	4.3	0	0.0	1
RDMAP	282	58	213	176	303.4	27.6	0	0.0	5	8.6	2	3.4	13	22.4	1
REPORT	725	228	334	247	108.3	20.6	0	0.0	1	0.4	20	8.8	8	3.5	4
RLIBIN	149	60	66	48	80.0	15.0	0	0.0	4	6.7	3	5.0	3	5.0	1
SETNUC	200	57	126	66	115.8	17.5	0	0.0	1	1.8	4	7.0	2	3.5	1
SETUP	106	35	66	31	88.6	28.6	0	0.0	0	0.0	2	5.7	0	0.0	0

Name	loc	sloc	cmnt	ncomt	ncomt /sloc	vg2 /sloc	cgoto	cgoto /sloc	ugoto	ugoto /sloc	bIF	bif /sloc	lIF	lif /sloc	Bhat
SIGMA	51	12	29	22	183.3	33.3	0	0.0	0	0.0	1	8.3	1	8.3	0
TRITUM	123	26	66	34	130.8	19.2	0	0.0	0	0.0	2	7.7	0	0.0	1
WBEDE2	301	91	144	110	120.9	18.7	0	0.0	3	3.3	5	5.5	3	3.3	1
WBEDE4	289	80	142	108	135.0	18.8	0	0.0	2	2.5	5	6.3	2	2.5	1
WPATHS	365	85	239	150	176.5	29.4	0	0.0	0	0.0	12	14.1	2	2.4	2
ZEROI	23	5	18	8	160.0	40.0	0	0.0	0	0.0	0	0.0	0	0.0	0
ZEROR	24	5	17	7	140.0	40.0	0	0.0	0	0.0	0	0.0	0	0.0	0

Legend of Metrics in Report

loc -- lines of code
sloc -- number of executable statements
cmnt -- total number of commnts
ncomt -- number of non-blank COMMENT statements
100*ncomt/sloc -- percent, nonblank comments to number of executable statements
100*vg2/sloc -- percent, extended complexity of number of executable statements
cgoto -- number of COMPUTED GO TO statements
100*cgoto/sloc -- percent, computed GOTO's to number of executable statements
ugoto -- number of UNCONDITIONAL GO TO statements
100*ugoto/sloc -- percent, unconditional GOTO's to number of executable statements
bIF -- number of BLOCK IF statements
100*bif/sloc -- percent, Block IF statements to number of executable statements
lIF -- number of LOGICAL IF statements
100*lif/sloc -- percent, logical IF statements to number of executable statements
Bhat -- Halstead's predicted number of errors in writing code

MODIFICATIONS TO DITTY COMPUTER CODE AND ITS LINKAGE TO THE TOTAL SYSTEM PERFORMANCE ASSESSMENT CODE

by B. Sagar and R. Janetzke

INTRODUCTION

A version of the DITTY (Dose In Ten Thousand Years) code was received by the Center for Nuclear Waste Regulatory Analyses (the Center) in the summer of 1991. It was decided through discussions among team members that DITTY will be used as the dose calculation module in Iterative Performance Assessment (IPA) Phase 2. Minor modifications were made to DITTY in the process of linking it to the Total system Performance Assessment (TPA) code. The modifications made, as well as the various links established between DITTY and TPA, are briefly described in the following.

The DITTY computer code was originally developed at the Battelle Pacific Northwest Laboratory specifically for application to high-level waste isolation primarily at the Hanford site. This code estimates the time integral of collective dose over a ten thousand year period for time varying radionuclide releases to the environment which includes surface waters (rivers and lakes), ground waters (wells), and the atmosphere. In the original code, the time frame for the calculation of dose is 10,000 years. This 10,000 year time span is subdivided into 143 periods, each of 70 year duration - 70 years being the average human-life span in which the dose is assumed to be received. The average radionuclide release to the environment in each of these 70 year periods is assumed to be provided as input to the dose code. From this, the population dose is determined for each period. The total radioactive material present during any period is taken as the sum of material released during that period and residual material in the environment from release in previous periods. The original program is designed to run in a stand-alone mode with no other controlling process except the host operating system. Modifications described below were made to adapt the DITTY code so that it would run under the commands of the TPA code.

DESCRIPTION OF MODIFICATIONS TO DITTY

In the Total system Performance Assessment, DITTY will be one of many "consequence modules" that would be executed in a specified sequence. Thus, to be useful as part of the TPA code, DITTY must be able to interact with the TPA - the executive program. This was the primary objective for the modification of DITTY. A few other changes were also made for convenience as detailed below. Modifications are numbered sequentially in the following, for ease of reference and discussion.

The original DITTY code was designed to perform dose calculations for 10,000 years in 143 life spans of 70 years each. To introduce some flexibility, the following modifications were made.

1. A parameter MAXPER (denoting maximum number of periods) is defined to replace the number 143. All arrays whose dimension depend upon the number of 70 year periods used in the calculations are now dimensioned by using the parameter MAXPER. Thus by changing the value of MAXPER, the code can now be compiled to fit calculations for any number of periods. The default value of MAXPER is currently set to 1430 which corresponds to a calculation period of 100,000 years.

2. A new variable MAXT (denoting maximum time for calculations) is introduced where MAXT is the number of years for which calculations are to be performed. It replaces the number 10,000 years in the original code. MAXT is used to check when to stop the calculations. Thus the code can now be used to perform dose calculations for any desired time span. The current default value of MAXT is 10,000 years. As will be explained in the following, the value of MAXT can be provided from the TPA code. Note that while MAXPER is a dimensioning parameter, the variable MAXT is used to decide when to stop calculations.

The length of the life span (i.e., 70 years) is still fixed in the code. In future, this should also be changed by a variable so that it can be modified by the user. However such a modification is expected to be quite involved.

3. A subroutine RDDAT has been created to read the global data file. The global data file contains run parameters which are pertinent to a particular scenario, and which are common to one or more consequence modules. This file may contain the times at which calculations for a particular scenario are to begin and end. This file may also contain the names of the radionuclides that are to be used in calculations. Note that most consequence modules are required to receive some of their data from the global data file. As an interface between a consequence module and the TPA, a subroutine must be added to read such data and place it in appropriate arrays. Unfortunately, we have not yet found a better way for doing this and hence a subroutine must be designed for each consequence module. RDDAT is such a subroutine for DITTY. The time to end the simulations provided to DITTY in the global data file is provided through the variable MAXT described above.

4. A subroutine RDMAP has been added. This subroutine reads data from the Latin Hypercube Sampling (LHS) output file. The map file contains a correspondence (or map) table that tells the DITTY code the locations in the LHS output file where quantities needed by the DITTY code are stored. Note that the DITTY code is designed to function as a deterministic code in its normal mode. However, the TPA code is designed to perform in a Monte Carlo mode for each scenario. All of the statistically defined parameters are sampled once by the LHS module at the beginning and the sampled values are written to a file. Each consequence module then reads the sampled values pertaining to it from this file. Again, we do not yet have an automatic way of passing this information except through a subroutine like RDMAP. Note that for RDMAP (also RDDAT) to work, the user has to make *a priori* (before compiling) decisions as to which parameters are to be sampled.

5. A provision has been added to write a special output file to transfer the calculated dose data to the system code. The system code processes this data to calculate the Complementary Cumulative Distribution Function (CCDF) of dose for either a single scenario or for all scenarios combined.

6. Another feature added to DITTY is the generation of a TECPLOT file for generating a time versus dose plot. TECPLOT is a commercial product available for personal computers and the Silicon Graphics workstation, which facilitates interactive plotting of formatted data files.

7. Subroutine MAKDA2 was modified to accommodate access to the host operating system date and time functions. The date and time functions are used to time stamp the output for run identification. It may be better to affect this by using the preprocessing preFOR utility developed at the Center, but that has not been done yet.

8. Subroutine OPNFIL was modified to open the global and map data files when DITTY is used as a consequence module with the TPA system.

9. To provide an easy mechanism for data entry, the input variables were changed to NAMELIST members. This allows the name of the variable to be specified with the data value associated with it. The input is then more readable and variables absent from the input file are assigned default values.

10. The DITTY program was originally written to allow many cases to be analyzed in one run. This feature was changed to a one case per run mode which will allow the system code to collect and store any intermediate data for CCDF calculations before proceeding to the next vector run.

SUMMARY AND CONCLUSIONS

The PNL dose code DITTY has been adapted for use in IPA Phase 2. This involved making some modifications which included defining one new parameter and one variable. In addition, two subroutines were added to provide the interface with the TPA code. Minor modifications in a few of the existing subroutines were also made.

In the future, for IPA Phase 3, the DITTY code may be further modified to: (1) Use the preFOR utility so that system calls such as for obtaining run time and date for different machines can be automated; (2) Modify the code so that the length of the calculational period can be different from 70 years; and (3) Study in greater detail the mechanistic calculation algorithms in DITTY and replace some of them with more updated ones. One such algorithm may be the one that deals with atmospheric dispersion or organ weighting factors.