1/236

SOFTWARE RELEASE NOTICE

1. SRN Number: GLGP-SRN-173

2. Project Title: Structural Deformation & Seismicity Code Development	Project No. 20-1402-472
3. SRN Title: 3DStress, Version 1.3	
4. Originator/Requestor: Joe Bangs	Date: 08/07/98

5. Summary of Actions

- ☐ Release of new software
- ■ Release of modified software:
  - ■ Enhancements made
  - ☐ Corrections made
- ☐ Change of access software
- ☑ Software Retirement A. Lemmon M. McKague

6. Persons Authorized Access

Name	Read Only/Read-Write	Addition/Change/Delete
David Ferrill	RW	
John Stamatakos	RW	
Larry McKague	RW	
Philip Justus (NRC)	RO	
Chuck Connor	RO	
Britt Hill	RO	
Ron Martin	RO	

7. Element Manager Approval: A. Lemmon M. McKague Date: 8/11/98

8. Remarks:
Enhancements include significant improvements to the user interface, ability to load coverages in the 2D map and 30 fault viewer, leakage factor calculations, and Mohr Circle displays.

Scientific Notebook #234

SOFTWARE SUMMARY FORM

01. Summary Date: 08/07/98	02. Summary prepared by (Name and phone) Bruce Mabrito, (210) 522-5149	03. Summary Action: New Release
04. Software Date: 08/06/98	05. Short Title: 3DStress Version 1.3	
06. Software Title: 3DStress, Version 1.3		07. Internal Software ID: NONE

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

11. Submitting Organization and Address:	12. Technical Contact(s) and Phone:
CNWRA/SWRI 6220 Culebra Road San Antonio TX 78228	Dave Ferrill, (210) 522-6082

13. Software Application:

3DStress is an interactive tool for analyzing the tendency for faults and fractures to slip or dilate based on a user specified three dimensional stress state.

14. Computer Platform	15. Computer Operating System:	16. Programming Language(s):	17. Number of Source Program Statements:
Silicon Graphics (SGI)	IRIX 5.3 or Higher	C++	44379 lines
18. Computer Memory Requirements: 16 MB minimum	19. Tape Drives: Supplied on CD ROM	20. Disk Units: 6 MB minimum	21. Graphics: Open GL

22. Other Operational Requirements: NONE

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

Software Developer: [signature] Joshua Buckner Date: 8-11-98

8/236

SOUTHWEST RESEARCH INSTITUTE™

INTER-DEPARTMENTAL MEMORANDUM

DATE: June 27, 2000

TO: Wes Patrick

FROM: Louis Rodriguez

SUBJECT: Certificate of Registration No. TXu 924-657
3DStress, Ver. 1.3

Enclosed you will find a copy of the original Certificate of Copyright Registration for the *3DStress, Ver. 1.3* We will retain the original in the legal department.

Please be sure to display the letter "C" enclosed within a circle "©" followed by the name of the copyright claimant and the year of creation whenever you are referring to this computer software program.

Enclosure
cc: David A. Ferrill
Budhi Sagar
Larry McKague

9/236

BAKER BOTTS L.L.P.

1600 SAN JACINTO CENTER
98 SAN JACINTO BLVD.
AUSTIN, TEXAS
78701-4039
512.322.2500
FAX 512.322.2501

AUSTIN
BAKU
DALLAS
HOUSTON
LONDON
MOSCOW
NEW YORK
WASHINGTON

May 14, 2000

Ann C. Livingston
512.322.2634
FAX 512.322.8325
ann.livingston@bakerbotts.com

Louis Rodriguez, Esq.
Deputy General Counsel
Southwest Research Institute
6220 Culebra Road
San Antonio, Texas 78238

Re: U.S. Copyright Registration No. TXu 924-657
Title: 3DStress, Ver. 1.3
Our File: 090936.0366
Author: D. Ferrill

Dear Louis:

I am pleased to enclose the above-referenced original Certificate of Copyright Registration which was issued effective November 29, 1999. This document comprises evidence of valuable property rights of Southwest Research Institute and should be maintained in a safe place.

Notice of copyright registration should be given by displaying the letter "C" enclosed within a circle "©" followed by the name of the copyright claimant and the year of creation.

Since the work covered by this registration is in accordance with the Copyright Law that went into effect on January 1, 1978, as amended by the Berne Convention Implementation Act that went into effect on March 1, 1989, the term of the registration for the work is seventy-five years from the date of publication or one hundred years from the year of creation, whichever expires first. Therefore, assuming that the work remains unpublished, and given that the work was created in 1999, the copyright in the work will expire on December 31, 2099.

If you have any questions concerning the registration, please do not hesitate to contact me.

Very truly yours,

BAKER BOTTS L.L.P.

Ann C. Livingston

ACL/ss
Enclosure

AUS01:206564.1

RECEIVED
MAY 19 2000
Legal & Patent Office

CERTIFICATE OF REGISTRATION

FORM TX 10/236
For a Literary Work
UNITED STATES COPYRIGHT OFFICE
REGISTRATION NUMBER
TXu 924-657

EFFECTIVE DATE OF REGISTRATION
11 (Month) 29 (Day) 99 (Year)

This Certificate issued under the seal of the Copyright Office in accordance with title 17, United States Code, attests that registration has been made for the work identified below. The information on this certificate has been made a part of the Copyright Office records.

Marybeth Peters
REGISTER OF COPYRIGHTS
United States of America

DO NOT WRITE ABOVE THIS LINE. IF YOU NEED MORE SPACE, USE A SEPARATE CONTINUATION SHEET

1 TITLE OF THIS WORK ▼

3DStress, Ver. 1.3

PREVIOUS OR ALTERNATIVE TITLES ▼

PUBLICATION AS A CONTRIBUTION If this work was published as a contribution to a periodical, serial, or collection, give information about the collective work in which the contribution appeared. **Title of Collective Work ▼**

If published in a periodical or serial give: **Volume ▼** **Number ▼** **Issue Date ▼** **On Pages ▼**

2 a NAME OF AUTHOR ▼

Southwest Research Institute

DATES OF BIRTH AND DEATH
Year Born ▼ Year Died ▼

Was this contribution to the work a "work made for hire"?
☒ Yes ☐ No

AUTHOR'S NATIONALITY OR DOMICILE Name of Country
OR Citizen of ▶ _____
Domiciled in ▶ U.S.A.

WAS THIS AUTHOR'S CONTRIBUTION TO THE WORK
Anonymous? ☐ Yes ☒ No
Pseudonymous? ☐ Yes ☒ No
If the answer to either of these questions is "Yes," see detailed instructions.

NAME OF AUTHORSHIP Briefly describe nature of material created by this author in which copyright is claimed. ▼
Entire work

b NAME OF AUTHOR ▼

DATES OF BIRTH AND DEATH
Year Born ▼ Year Died ▼

Was this contribution to the work a "work made for hire"?
☐ Yes ☐ No

AUTHOR'S NATIONALITY OR DOMICILE Name of Country
OR Citizen of ▶ _____
Domiciled in ▶ _____

WAS THIS AUTHOR'S CONTRIBUTION TO THE WORK
Anonymous? ☐ Yes ☐ No
Pseudonymous? ☐ Yes ☐ No
If the answer to either of these questions is "Yes," see detailed instructions.

NAME OF AUTHORSHIP Briefly describe nature of material created by this author in which copyright is claimed. ▼

c NAME OF AUTHOR ▼

DATES OF BIRTH AND DEATH
Year Born ▼ Year Died ▼

Was this contribution to the work a "work made for hire"?
☐ Yes ☐ No

AUTHOR'S NATIONALITY OR DOMICILE Name of Country
OR Citizen of ▶ _____
Domiciled in ▶ _____

WAS THIS AUTHOR'S CONTRIBUTION TO THE WORK
Anonymous? ☐ Yes ☐ No
Pseudonymous? ☐ Yes ☐ No
If the answer to either of these questions is "Yes," see detailed instructions.

NAME OF AUTHORSHIP Briefly describe nature of material created by this author in which copyright is claimed. ▼

NOTE
Under the law, the "author" of a "work made for hire" is generally the employer, not the employee (see instructions). For any part of this work that was made for hire" check "Yes" in the space provided, give the employer or other person for whom the work was prepared) as "Author" of that part, and leave the space for dates of birth and death blank.

3 a YEAR IN WHICH CREATION OF THIS WORK WAS COMPLETED This information must be given in all cases.
1999 ◀ Year

b DATE AND NATION OF FIRST PUBLICATION OF THIS PARTICULAR WORK
Complete this information ONLY if this work has been published.
Month ▶ _____ Day ▶ _____ Year ▶ _____
_____ ◀ Nation

4 COPYRIGHT CLAIMANT(S) Name and address must be given even if the claimant is the same as the author given in space 2. ▼
Southwest Research Institute
P.O. Drawer 28510
San Antonio, Texas 78228-0510

See instructions before completing this space.

TRANSFER If the claimant(s) named her in space 4 is (are) different from the author(s) named in space 2, give a brief statement of how the claimant(s) obtained ownership of the copyright. ▼

DO NOT WRITE HERE OFFICE USE ONLY

APPLICATION RECEIVED
SEP 21 1999

ONE DEPOSIT RECEIVED
SEP 21 1999

TWO DEPOSITS RECEIVED

FUNDS RECEIVED
FUNDS RECEIVED NOV 29 1999

MORE ON BACK ▶ • Complete all applicable spaces (numbers 5-11) on the reverse side of this page. • See detailed instructions. • Sign the form at line 10.

DO NOT WRITE HERE

EXAMINED BY

CHECKED BY

FORM TX

☐ CORRESPONDENCE
Yes

FOR COPYRIGHT OFFICE USE ONLY

DO NOT WRITE ABOVE THIS LINE. IF YOU NEED MORE SPACE, USE A SEPARATE CONTINUATION SHEET

REVIOUS REGISTRATION Has registration for this work, or for an earlier version of this work, already been made in the Copyright Office? 5

] Yes ☒ No If your answer is "Yes," why is another registration being sought? (Check appropriate box) ▼

☐ This is the first published edition of a work previously registered in unpublished form.

☐ This is the first application submitted by this author as copyright claimant.

☐ This is a changed version of the work, as shown by space 6 on this application.

your answer is "Yes," give: **Previous Registration Number** ▼ **Year of Registration** ▼

ERIVATIVE WORK OR COMPILATION Complete both space 6a and 6b for a derivative work; complete only 6b for a compilation. 6

Preexisting Material Identify any preexisting work or works that this work is based on or incorporates. ▼

Material Added to This Work Give a brief, general statement of the material that has been added to this work and in which copyright is claimed. ▼

See instructions before completing this space.

—space deleted— 7

EPRODUCTION FOR USE OF BLIND OR PHYSICALLY HANDICAPPED INDIVIDUALS A signature on this form at space 10 and a check in one the boxes here in space 8 constitutes a non-exclusive grant of permission to the Library of Congress to reproduce and distribute solely for the blind and physically ndicapped and under the conditions and limitations prescribed by the regulations of the Copyright Office: (1) copies of the work identified in space 1 of this plication in Braille (or similar tactile symbols); or (2) phonorecords embodying a fixation of a reading of that work; or (3) both. 8

a ☐ Copies and Phonorecords b ☐ Copies Only c ☐ Phonorecords Only

See instructions.

EPOSIT ACCOUNT If the registration fee is to be charged to a Deposit Account established in the Copyright Office, give name and number of Account. 9

ame

Account Number ▼

ORRESPONDENCE Give name and address to which correspondence about this application should be sent. Name / Address / Apt / City / State / ZIP ▼

ın Livingston
ıker & Botts
0 Trammell Crow Center, 2001 Ross Avenue, Dallas, Texas 75201
Area Code and Telephone Number ▶ (214) 953-6681

Be sure to give your daytime phone ◀ number

ERTIFICATION* I, the undersigned, hereby certify that I am the 10

Check only one ▶
☐ author
☐ other copyright claimant
☐ owner of exclusive right(s)
☒ authorized agent of Southwest Research Institute
Name of author or other copyright claimant, or owner of exclusive right(s) ▲

the work identified in this application and that the statements made me in this application are correct to the best of my knowledge.

/ped or printed name and date ▼ If this application gives a date of publication in space 3, do not sign and submit it before that date.

ın Livingston Date ▶ 9/8/99

Handwritten signature (X) ▼ Ann Livingston

AIL ERTIFI-ATE TO

Name ▼	Ann Livingston Baker & Botts	090936.0366
Number/Street/Apt ▼	800 Trammell Crow Center 2001 Ross Avenue	
City/State/ZIP ▼	Dallas, Texas 75201	

ertificate ill be ailed in indow nvelope

CHECKLIST
- Complete all necessary spaces
- Sign your application in space 10

SEND ALL 3 ELEMENTS IN THE SAME PACKAGE
1. Application form
2. Nonrefundable $20 filing fee in check or money order payable to *Register of Copyrights*
3. Deposit material

MAIL TO
Register of Copyrights
Library of Congress
Washington, D.C. 20559-6000

7 U.S.C. § 506(e): Any person who knowingly makes a false representation of a material fact in the application for copyright registration provided for by section 409, or in any written statement filed in connection h the application, shall be fined not more than $2,500.

CENTER FOR NUCLEAR WASTE REGULATORY ANALYSES

DESIGN VERIFICATION REPORT FOR CNWRA SOFTWARE: 3DStress Version 1.3

August 7, 1998

3DStress (Scientific and Engineering Software) Version 1.3

NOTE: This version of the 3DStress Software contains changes from the previous 1.2 version released November 12, 1996. Software Change Reports (SCRs) and an electronic scientific notebook have been utilized as the change documentation method and are being retained in the 3DStress Version 1.3 folder.

1. **This Design Verification Report is prepared by:** Bruce Mabrito in conjunction with Joshua Buckner and Joe Bangs.

   Full Title of CNWRA scientific and engineering software: 3DStress Version 1.3.

   Demonstration work station: Silicon Graphics Indigo 2 (REDWOOD) in conjunction with the Silicon Graphics Indy (YOSEMITE) in the GIS Room of Building 189.

   Operating System: IRIX 6.2 (A UNIX system).

2. **Software Requirements Description and any changes thereto approved by Element Manager?**

   (YES) NO N/A

   If no, explain:

3. **Software Development Plan (SDP) and any changes have been approved by the Element Manager?**

   (YES) NO N/A

   If no, explain:

4. **Design and Development**
   Module-level testing is documented in either scientific notebooks or in Software Change Reports?

   (YES) NO N/A

   Note: Both SCRs and electronic scientific notebook No. 234 contains module level documentation.

5. **Is the CNWRA scientific and engineering software developed in accordance with the conventions described in the SDP?**

   (YES) NO N/A

   If no, explain:

6. Is the CNWRA software documented internally?

(YES) NO N/A

Does the primary program header contain the following information:

A. Program title, Developed for (Customer), Office/Division/Date/Customer Contact/Telephone number, Software Developer, Telephone number, titles of Associated Documentation/Designator, and the Disclaimer Notice?

(YES) NO N/A

B. Source code module header information provides Program Name, Client Name, Contract Reference, Revision number?

(YES) NO N/A

7. Software designed so that individual runs are uniquely identified by Date, Time, Name of software and version? YES (NO) N/A

Note: There was a conscious decision made when the SCRs were being reviewed to not include this feature in 3DStress Version 1.3. This decision was made by the Element Manager and the P.I., Dr. David Ferrill, stated that he saw no value in this feature considering the type of software program 3DStress is.

8. The physical labeling on the software or the referenced list has Program Name/Title, Module/Name/Title, Module Revision, File Type (i.e. ASCII, OBJ, EXE), Recording Date and Operating System of the Supporting Hardware?

(YES) NO N/A

9. Users' Manual

Is there a Users' Manual for the software?

(YES) NO N/A

If no, explain:

Are there basic instructions for the use of the software?

(YES) NO N/A

If no, explain:

10. Acceptance Testing

Does the acceptance testing demonstrate whether or not requirements in the SDP have been fulfilled?

(YES) NO N/A

If no, explain:

Has acceptance testing been conducted for each intended computer platform and operating system?

(YES) NO N/A

If no, explain: Note: See note above regarding the Silicon Graphics INDIGO 2 workstation. Summaries are in the report produced by Joe Bangs and will be incorporated in this Design Verification Report as added documentation.

Have installation tests been performed on the target platform?

(YES) NO N/A

Note: Tests have been performed on the Silicon Graphics INDY workstation and the REDWOOD server.

11. Configuration Control

Is the Software Summary Form completed and signed?

(YES) NO N/A

If no, explain:

12. Is a software technical description prepared, documenting the essential mathematical and numerical basis?

(YES) NO N/A

If no, explain: The technical description is given in the Users' Manual for 3DStress Version 1.3.

13. Is the source code available (or, is the executable code available in the case of commercial codes)?

(YES) NO N/A

14. Have all the script/make files and executable files been submitted to the Software Custodian?

(YES) NO N/A

8/11/98

Joshua Buckner **Date**

CNWRA 3DStress Software Co-Developer

8/11/98

Bruce Mabrito **Date**

CNWRA Software Custodian

Attachments/
Original to: Software Folder
cc: CNWRA Software Developer
Cognizant EM L. McKague

```
//
//-------------------------  -------------------------  ---------
//    Program Name: 3DStress
//    Program Version: 1.3
//    Release Date: 08-07-98
//    SCR Number: 1 - 13
//-------------------------------------------------------------------
//
//
//    Developed by the Center for Nuclear Waste Regulatory
//    Analyses (CNWRA), Southwest Research Institute (SwRI),
//    San Antonio, Texas, USA.
//    CNWRA Contact: David Ferrill (210) 522-6082
//
//    Copyright 07/20/95 Southwest Research Institute
//                All rights reserved.
//
//    This software is a trade secret owned by Southwest Research
//    Institute, with access limited except as required for use by
//    authorized users.
//
//    This program was developed under sponsorship of the U.S.
//    Nuclear Regulatory Commission, contract number NRC-02-97-009.
//    NRC Office of Nuclear Material Safety and Safeguards
//    NRC Division of Waste Management, Engineering and Geoscience Branch
//
//    This computer code/material was prepared as an account of work
//    performed by the Center for Nuclear Waste Regulatory Analyses (CNWRA)
//    for the Division of Waste Management of the Nuclear Regulatory
//    Commission (NRC), an independent agency of the United States
//    Goverment.  The developer(s) of the code nor any of their sponsors
//    make any warranty, expressed or implied, or assume any legal
//    liability or responsibility for the accuracy, completeness, or
//    usefulness of any information, apparatus, product or process
//    disclosed, or represent that its use would not infringe on
//    privately-owned rights.
//
//    IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW WILL THE SPONSORS
//    OR THOSE WHO HAVE WRITTEN OR MODIFIED THIS CODE, BE LIABLE FOR
//    DAMAGES, INCLUDING ANY LOST PROFITS, LOST MONIES, OR OTHER SPECIAL,
//    INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR
//    INABILITY TO USE (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA
//    BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY THIRD PARTIES OR A
//    FAILURE OF THE PROGRAM TO OPERATE WITH OTHER PROGRAMS) THE PROGRAM,
//    EVEN IF YOU HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES,
//    OR FOR ANY CLAIM BY ANY OTHER PARTY.
//
//  Purpose:
//  Stereo Net Viewing tool.
//
```

```
////////////////////////////,  ////
// Class: MohrClass
//
// Description:
//
//  This class stores the data for
//  and draws the Mohr Graph.  Much
//  of the data processing and error
//  checking is done by simple C
//  functions in the mohrOptionsCB.c++
//  file because of Motif and C++
//  conflicts.
////////////////////////////////////
```

```
////////////////////////////////////////
//  Filename:     mohrOptionCB.hh
//  Author:       Robert Boenau
//  Date:         1-8-97
//
//  Purpose:
//
//
//  $Header: /usr/people/3dstress/3d/src/RCS/mohrOptionCB.hh,v 1.15 1998/07/23 17:
35:01 jbuckner Exp $
//
//  Revision History
//  $Log: mohrOptionCB.hh,v $
//  Revision 1.15  1998/07/23  17:35:01  jbuckner
//  added tighter text-box error-trapping
//
//  Revision 1.14  1998/01/08  17:22:09  jbuckner
//  Added Unit Ifo button callback
//
//  Revision 1.13  1997/08/01  19:19:27  jbuckner
//  Added a new function to take care of the arrowButton
//
//  Revision 1.12  1997/07/14  19:08:42  jbuckner
//  Removed previous stress changing functions and added one and a function to fin
d min, mid, and max sigma indecies
//
//  Revision 1.11  1997/07/08  21:16:52  jbuckner
//  Added a function for the display of raw effective stress ratios
//
//  Revision 1.10  1997/07/08  18:38:42  jbuckner
//  Added a callback function for the info widget
//
//  Revision 1.9  1997/07/02  17:39:42  jbuckner
//  Added functions to handle stress ratio text boxes
//
//  Revision 1.8  1997/07/01  20:40:40  jbuckner
//  Added a fluid scale call back andrenamed the call back for the fluid pressure
text field
//
//  Revision 1.7  1997/07/01  15:48:04  jbuckner
//  Added three new functions to complexity of the mohrScaleChange function when i
n dependent stress mode.  Renamed mohrStressScaleChange to mohrs3Tos1RatioScaleChan
ge and added mohrs2Tos1RatioScaleChange
//
//  Revision 1.6  1997/06/25  19:38:54  jbuckner
//  Added call back for rock data citation window
//
//  Revision 1.5  1997/06/19  18:54:42  jbuckner
//  Added mohrStressScale prototype for Poisson's ratio scale call back
//
//  Revision 1.4  1997/06/18  21:22:35  jbuckner
//  Call back prototype added for stress dependency radio box
//
//  Revision 1.3  1997/02/19  17:45:50  rboenau
//  Callbacks for fluid pressure and effective stress added
//
//  Revision 1.2  1997/02/18  19:06:07  rboenau
//  Normal stress value no longer needed, fixed problem with changing material..sh
ows wrong cValue
//
//  Revision 1.1  1997/02/18  16:30:31  rboenau
//  Initial revision
//
//
////////////////////////////////////////

#ifndef __MOHROPTIONCB_HH__
#define __MOHROPTIONCB_HH__

void mohrChangeLineWidth(Widget, XtPointer, XtPointer);
void recalcSigmas(int min, int mid, int max, double newMax);
```



```
void mohrScaleChange(Widget, XtPointer, XtPointer);
void findMinMidMax(int * min, int * mid, int * max, double values[], int noelts);
void mohrChangeMinMax(Widget, XtPointer, XtPointer);
void mohrToggleButtonChange(Widget, XtPointer, XtPointer);
void mohrToggleRenderMode(Widget, XtPointer, XtPointer);
void mohrToggleStressMode(Widget, XtPointer clientData, XtPointer);
void mohrChanges3Tos1RatioField(Widget, XtPointer, XtPointer);
void mohrChanges2Tos1RatioField(Widget, XtPointer, XtPointer);
void mohrs3Tos1RatioScaleChange(Widget, XtPointer, XtPointer);
void mohrs2Tos1RatioScaleChange(Widget, XtPointer, XtPointer);
void mohrChangeCStr(Widget, XtPointer, XtPointer);
void mohrChangeRockType(Widget, XtPointer, XtPointer);
void mohrChangeRockMaterial(Widget, XtPointer, XtPointer);
void mohrChangeValue(Widget, XtPointer, XtPointer);
void mohrCloseButton(Widget, XtPointer, XtPointer);
void mohrApplyButton(Widget, XtPointer, XtPointer);
void mohrRockInfoButton(Widget, XtPointer, XtPointer);
void mohrUnitInfoButton(Widget, XtPointer, XtPointer);
void mohrApplyOKCB(Widget, XtPointer, XtPointer);
void mohrChangeFluidField(Widget, XtPointer, XtPointer);
void mohrChangeFluidScale(Widget, XtPointer, XtPointer);
void updateEffective();
void updateEffRatios();
void closeBoundsErrorWin(Widget, XtPointer, XtPointer);
void mohrArrowCB(Widget, XtPointer, XtPointer);
int trapAlpha(char * in_str);
#endif
```

25 | 25

```
////////////////////////////////////////
//  Filename:     mohrOptionCB.c++
//  Author:       Robert Boenau
//  Date:         1-8-97
//
//  Purpose:
//
//
//  $Header: /usr/people/3dstress/3d/src/RCS/mohrOptionCB.c++,v 1.36 1998/07/28 21
:18:42 jbuckner Exp $
//
//  Revision History
//  $Log: mohrOptionCB.c++,v $
//  Revision 1.36  1998/07/28  21:18:42  jbuckner
//  fixed parsing of spaces in text box error checking
//
//  Revision 1.35  1998/07/23  17:34:57  jbuckner
//  added tighter text-box error-trapping
//
//  Revision 1.34  1998/06/19  20:57:48  jbuckner
//  fixed the normalization of stresses, Pf, and Tstr
//
//  Revision 1.33  1998/01/08  17:22:00  jbuckner
//  Added Unit Ifo button callback
//
//  Revision 1.32  1997/09/29  21:16:17  jbuckner
//  Changed the precision on the displayed variables
//
//  Revision 1.31  1997/08/05  15:40:05  jbuckner
//  Modified mohrToggleStressMode to unmanage instead of XtSetSensitive(..., False
);
//
//  Revision 1.30  1997/08/04  16:11:37  jbuckner
//  In the function mohrToggleStressMode, changed the XtSetSensitive calls to XtMa
nageChild and XtUnmanageChild calls.
//
//  Revision 1.29  1997/08/01  19:18:04  jbuckner
//  Added a new function to take care of the arrowButton
//
//  Revision 1.28  1997/07/21  19:45:02  jbuckner
//  Added fluid pressure and tensile strength update in the apply button callback
//
//  Revision 1.27  1997/07/15  16:13:24  jbuckner
//  Changed fluid text bounds check to avoid running out of bounds on the sigmas
//
//  Revision 1.26  1997/07/15  15:18:55  jbuckner
//  Fixed long lines, added end-of-function comments, and changed bounds checking
in recalcSigmas
//
//  Revision 1.25  1997/07/14  19:10:43  jbuckner
//  Completely reworked the way dependent stress works.  Dependent stress now oper
ates off strait ratios instead of Mendal's method.
//
//  Revision 1.24  1997/07/09  21:14:20  jbuckner
//  Fixed bug in changeMinMax introduced by adding decimal places to the sigma val
ues
//
//  Revision 1.23  1997/07/08  21:19:38  jbuckner
//  Added a line to updateEffective to update effective ratio text boxes and Added
 function to update effective ratio text boxes
//
//  Revision 1.22  1997/07/08  18:42:30  jbuckner
//  Added prototype for destroyFS, Added a callback for info win, modified updateE
ffective to use new callback
//
//  Revision 1.21  1997/07/02  20:14:05  jbuckner
//  Modified void updateEffective() to be sure that effective stress is within bou
nds of graph
//
//  Revision 1.20  1997/07/02  18:52:13  jbuckner
//  Fixed bug in callbacks for stress ratio text fields
```



```
//
//  Revision 1.19  1997/07/02  17:41:42  jbuckner
//  Added callbacks for stress ratio text fields and updated callbacks for stress
ratio scales
//
//  Revision 1.18  1997/07/01  20:42:54  jbuckner
//  Added function mohrChangeFluidScale and substantialy changed mohrChangeFluidTe
xt
//
//  Revision 1.17  1997/07/01  15:50:30  jbuckner
//  Added three functions to cut complexity of mohrScaleChange and handle to stres
s ratios when stresses are dependent
//
//  Revision 1.16  1997/06/27  21:38:17  jbuckner
//  Fixed Poisson's Ratio problems in several functions
//
//  Revision 1.15  1997/06/27  17:48:19  jbuckner
//  Made use of new mohrObj.attributes->effStress in functions
//
//  Revision 1.14  1997/06/25  19:37:13  jbuckner
//  Added call back for rock data citation window
//
//  Revision 1.13  1997/06/20  19:39:30  jbuckner
//  Converted sigmas to double values and added functionallity to grey out Poisson
's % scale when stresses are independent
//
//  Revision 1.12  1997/06/19  18:58:21  jbuckner
//  Added mohrStressScale callback for the Poisson's ratio scale and added the var
iable ratio to the mohrScaleChange function
//
//  Revision 1.11  1997/06/18  21:34:43  jbuckner
//  modified mohrScaleChange to handel the stress dependency radio box
//
//  Revision 1.10  1997/06/18  21:11:35  jbuckner
//  Added call back function for the new stress dependency radio box
//
//  Revision 1.9  1997/06/16  19:47:43  jbuckner
//  Major changes to mohrScaleChange to fix errors in compensation for internal pr
essure
//
//  Revision 1.8  1997/06/16  14:03:00  jbuckner
//  Fixed indexing problem in mohrScallChange's internal pressure compensation
//
//  Revision 1.7  1997/06/16  13:27:33  jbuckner
//  Fixed an error made in mohrScaleChange's last revision
//
//  Revision 1.6  1997/06/13  19:20:34  jbuckner
//  Modified the mohrScale values to take internal pressures into account in ca
lation of the sigmas
//
//  Revision 1.5  1997/04/03  22:15:07  rboenau
//  Updates display when fluid pressure is changed
//
//  Revision 1.4  1997/02/19  17:45:50  rboenau
//  Callbacks for fluid pressure and effective stress added
//
//  Revision 1.3  1997/02/18  19:59:40  rboenau
//  Added additional rock types
//
//  Revision 1.2  1997/02/18  19:06:07  rboenau
//  Normal stress value no longer needed, fixed problem with changing material..sh
ows wrong cValue
//
//  Revision 1.1  1997/02/18  16:30:25  rboenau
//  Initial revision
//
//
////////////////////////////////////////

// Unix
#include <stdio.h>
```

```
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <iostream.h>

// Motif
#include <Xm/Xm.h>
#include <Xm/Text.h>
#include <Xm/TextF.h>
#include <Xm/Scale.h>

// Application
#include "mohrObj.hh"
#include "mohrOptionObj.hh"
#include "mohrOptionCB.hh"
#include "infoWidget.hh"
#include "cmdObj.hh"

// Prototype added to make destroyFS available to closeBoundsErrorWin:
// this function gets rid of the info window
void   destroyFS(Widget, XtPointer, XtPointer);

void mohrChangeLineWidth(Widget, XtPointer, XtPointer callData){
   XmScaleCallbackStruct *cbs = (XmScaleCallbackStruct *) callData;
   MohrAttributeType *mattr = mohrObj.getAttributes();

   mattr->lineWidth = (int)(cbs->value);
   mohrObj.display();

} // end of mohrChangeLineWidth

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  recalcSigmas (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  double newMax -- The previous maximum sigma value takes
//              this as its new value.
//
//  State Changes:  mohrObj.attributes->sigmas[?], updates the sigmas
//                  with respect to newMax and the dependent stress
//                  ratios.
//
//  Purpose:  This program updates the sigmas according to newMax and
//            the dependent stress ratios.
//
// Last Modified:  14 July 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void recalcSigmas(int min, int mid, int max, double newMax)
{
   MohrAttributeType // mohr graph's attributes
      * mattr = mohrObj.getAttributes();

   if (newMax > mattr->maxScale) // if the new value is larger than possible
      newMax = mattr->maxScale;  // scale it down to maximum

   // ratios only work with positive effective stresses
   if (newMax - mohrObj.fluidPressure <= 0) // effstress = sigma - fluidP.
      newMax = 0.01 + mohrObj.fluidPressure;
   // also, min, mid, and max stresses shouldn't trade places

   mattr->sigmas[max] = newMax;  // now, set the new values according to ratios
   mattr->sigmas[mid] = newMax * mattr->s2TosiRatio;
   mattr->sigmas[min] = newMax * mattr->s3TosiRatio;
}  // end function recalcSigmas

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  mohrScaleChange (not a member function)
//
//  File:  mohrOptionCB.c++
//
```



```
//  Arguments:  Widget, XtPointer clientData, XtPointer callData
//
//  State Changes:  mohrObj.attributes->sigmas[?], updates the sigma
//                  that was changed in the Mohr graph options window
//                  by the user.  It calls recalcSigmas to accomplish
//                  this.  Scales in the Options window are changed.
//
//  Purpose:  This program updates the sigmas according to the settings
//            of the scales in the Mohr graph options window devoted to
//            the sigmas.
//
// Last Modified:  14 July 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void mohrScaleChange(Widget, XtPointer clientData, XtPointer callData)
{
   XmScaleCallbackStruct
      * cbs = (XmScaleCallbackStruct *) callData;

   //  The variable below holds the address to mohrObj's
   //  attributes private member variable.
   MohrAttributeType
      * mohrObjAttrib = mohrObj.getAttributes();

   //  The variable below holds the member variables of the
   //  class responsible for the Mohr Graph's Options window
   MohrOptionType *optionsAttrib = mohrOptionObj.getAttributes();

   //  The variable below, whichSig, is the index of the sigma that
   //  was changed in the Mohr's options window.  Also, newValue holds
   //  the value that the user set to the sigma indicated by whichSig,
   //  and delta holds the change in the sigma value.
   int
      whichSig = (int) clientData,
      max, mid, min;  // indecies of largest, middle, & smallest sigma values

   double
      newValue = (double) cbs->value / 100;

   if(mohrObjAttrib->stressMode == 0)
   {  //stresses should depend on each other
      // Search for the largest and smallest sigmas
      findMinMidMax( & min, & mid, & max, mohrObjAttrib->sigmas, 3 );

      if ( whichSig == max )
         recalcSigmas(min, mid, max, newValue);

      else if ( whichSig == mid ) // user changed middle value
         recalcSigmas(min, mid, max, newValue *
                                     (1 / mohrObjAttrib->s2TosiRatio));

      else  // minimum value changed
         recalcSigmas(min, mid, max, newValue *
                                     (1 / mohrObjAttrib->s3TosiRatio));

      XmScaleSetValue(optionsAttrib->scale[max], mohrObjAttrib->sigmas[max] *
                                                 100);
      XmScaleSetValue(optionsAttrib->scale[mid], mohrObjAttrib->sigmas[mid] *
                                                 100);
      XmScaleSetValue(optionsAttrib->scale[min], mohrObjAttrib->sigmas[min] *
                                                 100);
   }// end if(mohrObjAttrib->stressMode == 0)

   else  // scales should be inDependent
   {
      mohrObjAttrib->sigmas[whichSig] = newValue;
   }  // end else

   updateEffective();
   mohrObj.failure();
   mohrObj.display();
} // end of mohrScaleChange
```

```
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  findMinMidMax (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  int * min, int * mid, int * max, -- addresses to store info in
//              double values[] -- values to get min, mid, max indecies of
//              int noelts -- number of elements in values[]
//
//  State changes:  min, mid, and max are changed to reflect the indecies of
//                  minimum, middle, and maximum values in int values[]
//
//  Purpose:  This function is used to find the indecies of minimum, middle,
//            and maximum values in double values[]
//
// Last Modified:  14 July 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void findMinMidMax(int * min, int * mid, int * max, double values[], int noelts)
{
   int i;   // loop index
   *min = *mid = *max = 0;       // initialize variables

   for(i = 0; i < noelts; i++) // find the max and min
   {
      if (values[i] >= values[(*max)])
         *max = i; // new max has been found
      if (values[i] <= values[(*min)])
         *min = i; // new min has been found
   }

   for(i = 0; i < noelts; i++)  // find mid
   {
      if( (i != *min) && (i != *max) )
         *mid = i; // mid has been found
   }
}  // end function findMinMidMax

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  mohrChanges3Tos1RatioField (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  Widget, XtPointer, XtPointer
//
//  State Changes:  mohrObj.attributes->s3Tos1Ratio, updates the degree
//                  to which the stress are dependent when in dependent
//                  stress mode
//
//  Purpose:  This program updates the allows the user to change Poisson's
//            ratio in the calculation of dependency for dependent stresses
//
// Last Modified:  14 July 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void mohrChanges3Tos1RatioField(Widget, XtPointer, XtPointer)
{
   MohrOptionType // get option window's stats
      * moattr = mohrOptionObj.getAttributes();
   MohrAttributeType // get Mohr graph stats
      * mattr = mohrObj.getAttributes();

   char
      strHolder[100], // holder for transfer of numbers to ASCII
      * val = XmTextFieldGetString(moattr->s3Tos1RatioText); // get new value

   double
      tmpValue;  // holder for new value in numeric form
   int
      min, mid, max;

   findMinMidMax( & min, & mid, & max, mattr->sigmas, 3 );
```



```
   if (strlen(val) == 0) // is field is completely blank
   {  // tell user of mistake
      infoWidget(moattr->mohrDialogForm,"Not allowed to have empty value");
      sprintf(strHolder,"%5.3f", mattr->s3Tos1Ratio);     // reset field
      XmTextFieldSetString(moattr->s3Tos1RatioText, strHolder);
      return;
   }

   tmpValue = atof(val);  // convert new value to numeric form

   if ( ((int) (tmpValue * 100) > mattr->stressScaleMax) ||
        ((int) (tmpValue * 100) < mattr->stressScaleMin) )
   {  // check that new value is within bounds
      infoWidget(moattr->mohrDialogForm,"Ratio out of bounds");
      // dialog box error message
      sprintf(strHolder,"%5.3f", mattr->s3Tos1Ratio);  // reset field
      XmTextFieldSetString(moattr->s3Tos1RatioText, strHolder);
      return;
   }

   if ( tmpValue > mattr->s2Tos1Ratio )
   {  // check that new value is within bounds
      infoWidget(moattr->mohrDialogForm,"Ratio violates sigma orders.");
      // dialog box error message
      sprintf(strHolder,"%5.3f", mattr->s3Tos1Ratio);  // reset field
      XmTextFieldSetString(moattr->s3Tos1RatioText, strHolder);
      return;
   }

   mattr->s3Tos1Ratio = tmpValue; // update variable
   // update scale
   XmScaleSetValue(moattr->s3Tos1RatioScale, (int) (mattr->s3Tos1Ratio * 100));

   recalcSigmas(min, mid, max, mattr->sigmas[max]);
   XmScaleSetValue(moattr->scale[max], mattr->sigmas[max] * 100);
   XmScaleSetValue(moattr->scale[mid], mattr->sigmas[mid] * 100);
   XmScaleSetValue(moattr->scale[min], mattr->sigmas[min] * 100);

   updateEffective();
   mohrObj.failure();
   mohrObj.display();
} // end function  mohrChanges3Tos1RatioField

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  mohrChanges2Tos1RatioField (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  Widget, XtPointer, XtPointer
//
//  State Changes:  mohrObj.attributes->s2Tos1Ratio, updates the degree
//                  to which the stress are dependent when in dependent
//                  stress mode
//
//  Purpose:  This program updates the allows the user to change Poisson's
//            ratio in the calculation of dependency for dependent stresses
//
// Last Modified:  14 July 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void mohrChanges2Tos1RatioField(Widget, XtPointer, XtPointer)
{
   MohrOptionType // get option window's stats
      * moattr = mohrOptionObj.getAttributes();
   MohrAttributeType // get Mohr graph stats
      * mattr = mohrObj.getAttributes();

   char
      strHolder[100], // holder for transfer of numbers to ASCII
      * val = XmTextFieldGetString(moattr->s2Tos1RatioText); // get new value
```

28 | 234

```
    double
       tmpValue;  // holder for new value in numeric form
    int
       min, mid, max;

    findMinMidMax( & min, & mid, & max, mattr->sigmas, 3 );

    if (strlen(val) == 0) // is field is completely blank
    {  // tell user of mistake
       infoWidget(moattr->mohrDialogForm,"Not allowed to have empty value");
       sprintf(strHolder,"%5.3lf", mattr->s2Tos1Ratio);    // reset field
       XmTextFieldSetString(moattr->s2Tos1RatioText, strHolder);
       return;
    }

    tmpValue = atof(val);  // convert new value to numeric form

    if ( ((int) (tmpValue * 100) > mattr->stressScaleMax) ||
         ((int) (tmpValue * 100) < mattr->stressScaleMin) )
    {  // check that new value is within bounds
          infoWidget(moattr->mohrDialogForm,"Ratio out of bounds");
          // dialog box error message
          sprintf(strHolder,"%5.3lf", mattr->s2Tos1Ratio);  // reset field
          XmTextFieldSetString(moattr->s2Tos1RatioText, strHolder);
          return;
    }

    if ( tmpValue < mattr->s3Tos1Ratio )
    {  // check that new value is within bounds
          infoWidget(moattr->mohrDialogForm,"Ratio violates sigma orders.");
          // dialog box error message
          sprintf(strHolder,"%5.3lf", mattr->s2Tos1Ratio);  // reset field
          XmTextFieldSetString(moattr->s2Tos1RatioText, strHolder);
          return;
    }

    mattr->s2Tos1Ratio = tmpValue; // update variable
    // update scale
    XmScaleSetValue(moattr->s2Tos1RatioScale, (int) (mattr->s2Tos1Ratio * 1000));

    recalcSigmas(min, mid, max, mattr->sigmas[max]);
    XmScaleSetValue(moattr->scale[max], mattr->sigmas[max] * 100);
    XmScaleSetValue(moattr->scale[mid], mattr->sigmas[mid] * 100);
    XmScaleSetValue(moattr->scale[min], mattr->sigmas[min] * 100);

    updateEffective();
    mohrObj.failure();
    mohrObj.display();
} // end function  mohrChanges2Tos1RatioField

//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  mohrs3Tos1RatioScaleChange (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  Widget, XtPointer, XtPointer callData
//
//  State Changes:  mohrObj.attributes->s3Tos1Ratio, updates the degree
//                  to which the stress are dependent when in dependent
//                  stress mode
//
//  Purpose:  This program updates the allows the user to change Poisson's
//            ratio in the calculation of dependency for dependent stresses
//
// Last Modified:  14 July 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void mohrs3Tos1RatioScaleChange(Widget, XtPointer, XtPointer callData)
{
   XmScaleCallbackStruct
      * cbs = (XmScaleCallbackStruct *) callData;
```



```
   MohrOptionType // get option window's stats
      * moattr = mohrOptionObj.getAttributes();

   //  The variable below holds the address to mohrObj's
   //  attributes private member variable.
   MohrAttributeType
      * mattr = mohrObj.getAttributes();

   char
      strHolder[100]; // holder for transfer of numbers to ASCII
   int
      min, mid, max;

   findMinMidMax( & min, & mid, & max, mattr->sigmas, 3 );

   // update the ratio
   mattr->s3Tos1Ratio =  (double) (cbs->value / 1000.0);

   if (mattr->s3Tos1Ratio > mattr->s2Tos1Ratio)
   {
      mattr->s3Tos1Ratio = mattr->s2Tos1Ratio - 0.001;
      XmScaleSetValue(moattr->s3Tos1RatioScale,
                      (int) (mattr->s3Tos1Ratio * 1000));
   }

   sprintf(strHolder,"%5.3lf", mattr->s3Tos1Ratio);  // reset field
   XmTextFieldSetString(moattr->s3Tos1RatioText, strHolder);

   recalcSigmas(min, mid, max, mattr->sigmas[max]);
   XmScaleSetValue(moattr->scale[max], mattr->sigmas[max] * 100);
   XmScaleSetValue(moattr->scale[mid], mattr->sigmas[mid] * 100);
   XmScaleSetValue(moattr->scale[min], mattr->sigmas[min] * 100);

   updateEffective();
   mohrObj.failure();
   mohrObj.display();
} // end of mohrs3Tos1RatioScaleChange

//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  mohrs2Tos1RatioScaleChange (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  Widget, XtPointer, XtPointer callData
//
//  State Changes:  mohrObj.attributes->s2Tos1Ratio, updates the degree
//                  to which the stress are dependent when in dependent
//                  stress mode
//
//  Purpose:  This program updates the allows the user to change Poisson's
//            ratio in the calculation of dependency for dependent stresses
//
// Last Modified:  14 July 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void mohrs2Tos1RatioScaleChange(Widget, XtPointer, XtPointer callData)
{
   XmScaleCallbackStruct
      * cbs = (XmScaleCallbackStruct *) callData;

   MohrOptionType // get option window's stats
      * moattr = mohrOptionObj.getAttributes();

   //  The variable below holds the address to mohrObj's
   //  attributes private member variable.
   MohrAttributeType
      * mattr = mohrObj.getAttributes();

   char
      strHolder[100]; // holder for transfer of numbers to ASCII
   int
      min, mid, max;
```

```
   findMinMidMax( & min, & mid, & max, mattr->sigmas, 3 );

   // update the ratio
   mattr->s2Tos1Ratio =  (double) (cbs->value / 1000.0);

   if (mattr->s2Tos1Ratio < mattr->s3Tos1Ratio)
   {
      mattr->s2Tos1Ratio = mattr->s3Tos1Ratio + 0.001;
      XmScaleSetValue(moattr->s2Tos1RatioScale,
                      (int) (mattr->s2Tos1Ratio * 1000));
   }

   sprintf(strHolder,"%5.3lf", mattr->s2Tos1Ratio);  // reset field
   XmTextFieldSetString(moattr->s2Tos1RatioText, strHolder);

   recalcSigmas(min, mid, max, mattr->sigmas[max]);
   XmScaleSetValue(moattr->scale[max], mattr->sigmas[max] * 100);
   XmScaleSetValue(moattr->scale[mid], mattr->sigmas[mid] * 100);
   XmScaleSetValue(moattr->scale[min], mattr->sigmas[min] * 100);

   updateEffective();
   mohrObj.failure();
   mohrObj.display();
}  // end of mohrs2Tos1RatioScaleChange

void mohrChangeMinMax(Widget, XtPointer clientData, XtPointer){
   MohrAttributeType *mattr = mohrObj.getAttributes();
   MohrOptionType *moattr = mohrOptionObj.getAttributes();
   int tmpValue;
   char strHolder[100]; // holder for transfer of numbers to ASCII

   if ((int)clientData == 1) { // change min
      if (XmTextFieldGetString(moattr->minText) != NULL) {
         char *val = XmTextFieldGetString(moattr->minText);
         if ((strlen(val) == 0) || (!trapAlpha(val))){
            infoWidget(moattr->mohrDialogForm,
                       "Not allowed to have empty or alphabetical value");
            sprintf(strHolder,"%d", mattr->minScale);
            XmTextFieldSetString(moattr->minText, strHolder);
            return;
         }
         tmpValue = atoi(val);
         if (tmpValue < -1000)
            tmpValue = -1000;
         mattr->minScale = tmpValue;
      } // end if
   } // end if

   else { // changed max
      if (XmTextFieldGetString(moattr->maxText) != NULL) {
         char *val = XmTextFieldGetString(moattr->maxText);
         if ((strlen(val) == 0) || (!trapAlpha(val))){
            infoWidget(moattr->mohrDialogForm,
                       "Not allowed to have empty or alphabetical value");
            sprintf(strHolder,"%d", mattr->maxScale);
            XmTextFieldSetString(moattr->maxText, strHolder);
            return;
         }
         tmpValue = atoi(val);
         if (tmpValue > 1000)
            tmpValue = 1000;
         mattr->maxScale = tmpValue;
      } // end if
   } // end else

   if (mattr->minScale >= mattr->maxScale){
      infoWidget(moattr->mohrDialogForm,
                 "Min Scale must be less than Max Scale");
      return;
   }
```



```
   // Change value of the sliders and make sure present values lie
   // within the min and max
   for (int i = 0; i < 3; i++) {
      if (mattr->sigmas[i] < mattr->minScale ||
          mattr->sigmas[i] > mattr->maxScale) {
         mattr->sigmas[i] = mattr->minScale;
         XmScaleSetValue(moattr->scale[i], mattr->sigmas[i] * 100);
      } // end if
      XtVaSetValues(moattr->scale[i],
                    XmNmaximum, mattr->maxScale * 100,
                    XmNminimum, mattr->minScale * 100,
                    NULL);
   } // end for int i

   mohrObj.display();
} // end of mohrChangeMinMax

void mohrToggleButtonChange(Widget, XtPointer which, XtPointer){
   MohrAttributeType *mattr = mohrObj.getAttributes();

   if ((int)which == 0) {
      mattr->showAxis = !mattr->showAxis;
   }
   else {
      mattr->showInnerCircles = !mattr->showInnerCircles;
   }

   mohrObj.display();
} // end function mohrToggleButtonChange

void mohrToggleRenderMode(Widget, XtPointer clientData, XtPointer){
   MohrAttributeType *mattr = mohrObj.getAttributes();

   mattr->render = (int)clientData;
   mohrObj.display();
} // end function mohrToggleRenderMode

//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  mohrToggleStressMode (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  Widget, XtPointer clientData, XtPointer callData
//              clientData is the data that represents the action the
//              user took.  Ask Robert Boenau what the hell the rest are.
//
//  State Changes:  mohrObj.attributes->stressMode, changes the stress
//                  mode between dependent stresses and independent
//                  stresses. When stress is set to independent, the stress
//                  dependency ratio scale is greyed out;  when it is set to
//                  dependent, the stress dependency ratio is activated.
//                  When stresses are set to dependent, sigmas are recalculated
//                  to take ratios into account.
//
//  Purpose:  This function switches between the two stress modes
//
// Last Modified:  14 July 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void mohrToggleStressMode(Widget, XtPointer clientData, XtPointer)
{
   // get the attributes of the current Mohr graph for changing
   MohrAttributeType *mohrAttrib = mohrObj.getAttributes();

   // get the attributes of the option window (widgets and such)
   MohrOptionType *optionAttrib = mohrOptionObj.getAttributes();

   // set the stress mode to the appropriate value (0 or 1)
   // stressMode will be 0 for dependent stresses and 1 for independent
   mohrAttrib->stressMode = (int)clientData;

   int
```

30 / 234

```
      min, mid, max;

   // if stress is independent, grey out the ratio scale
   if (mohrAttrib->stressMode == 1)
   {
      XtUnmanageChild(optionAttrib->rowStress);
   }
   else  // if stress is dependent, activate the ratio scale
   {
      findMinMidMax( & min, & mid, & max, mohrAttrib->sigmas, 3 );
      recalcSigmas(min, mid, max, mohrAttrib->sigmas[max]);

      XtManageChild(optionAttrib->colStress2);
      XtManageChild(optionAttrib->rowStress);

      XmScaleSetValue(optionAttrib->scale[0], mohrAttrib->sigmas[0] * 100);
      XmScaleSetValue(optionAttrib->scale[1], mohrAttrib->sigmas[1] * 100);
      XmScaleSetValue(optionAttrib->scale[2], mohrAttrib->sigmas[2] * 100);

      updateEffective();
      mohrObj.failure();
      mohrObj.display();
   }
} // end function mohrToggleStressMode

void mohrChangeRockType(Widget, XtPointer clientData, XtPointer){
   // Change has occured in...intact, verygood....verypoor
   // ie quality

   MohrAttributeType *mattr = mohrObj.getAttributes();
   MohrOptionType *moattr = mohrOptionObj.getAttributes();
   char m[20], s[20];


   mohrObj.typeIndex = (int)clientData;

   mohrObj.mValue = mohrObj.samplesM[mohrObj.typeIndex][mohrObj.materialIndex];
   mohrObj.sValue = mohrObj.samplesS[mohrObj.typeIndex];

   sprintf(m,"%.5lf", mohrObj.mValue);
   sprintf(s,"%.5lf", mohrObj.sValue);

   XmTextSetString(moattr->mText, m);
   XmTextSetString(moattr->sText, s);
   mohrObj.calculate();
   mohrObj.display();
} // end function mohrChangeRockType

void mohrChangeRockMaterial(Widget, XtPointer clientData, XtPointer){
   // Change has occured in...carbonate...etc
   // ie material

   MohrAttributeType *mattr = mohrObj.getAttributes();
   MohrOptionType *moattr = mohrOptionObj.getAttributes();
   char m[20], s[20], c[20];

   XtUnmanageChild(moattr->rockCStr[mohrObj.materialIndex]);

   mohrObj.materialIndex = (int)clientData;

   XtManageChild(moattr->rockCStr[(int)clientData]);
   mohrObj.mValue = mohrObj.samplesM[mohrObj.typeIndex][mohrObj.materialIndex];
   mohrObj.sValue = mohrObj.samplesS[mohrObj.typeIndex];
   mohrObj.cValue =
       mohrObj.samplesCStr[moattr->rockCStrIndex[(int)clientData]];

   sprintf(m,"%.5lf", mohrObj.mValue);
   sprintf(s,"%.5lf", mohrObj.sValue);
   sprintf(c,"%.1lf", mohrObj.cValue);

   XmTextSetString(moattr->mText, m);
```



```
   XmTextSetString(moattr->sText, s);
   XmTextSetString(moattr->cText, c);

   mohrObj.calculate();
   mohrObj.display();
} // end function mohrChangeRockMaterial

void mohrChangeValue(Widget, XtPointer, XtPointer){
   char strHolder[100]; // holder for transfer of numbers to ASCII
   MohrOptionType *moattr = mohrOptionObj.getAttributes();

   char *val = XmTextFieldGetString(moattr->mText);
   char *val2 = XmTextFieldGetString(moattr->sText);
   char *val3 = XmTextFieldGetString(moattr->cText);

   if (strlen(val) == 0 || strlen(val2) == 0
       || strlen(val3) == 0 || !trapAlpha(val)
       || !trapAlpha(val2) || !trapAlpha(val3)) {
      infoWidget(moattr->mohrDialogForm,
         "Not allowed to have empty or alphabetical value");
      sprintf(strHolder,"%5.2lf", mohrObj.mValue);
      XmTextFieldSetString(moattr->mText, strHolder);
      sprintf(strHolder,"%5.2lf", mohrObj.sValue);
      XmTextFieldSetString(moattr->sText, strHolder);
      sprintf(strHolder,"%5.2lf", mohrObj.cValue);
      XmTextFieldSetString(moattr->cText, strHolder);
      return;
   }
   mohrObj.mValue = atof(val);
   mohrObj.sValue = atof(val2);
   mohrObj.cValue = atof(val3);

   free(val);
   free(val2);
   free(val3);
   mohrObj.calculate();
   mohrObj.display();
} // end function mohrChangeValue

void mohrCloseButton(Widget, XtPointer, XtPointer){
   mohrOptionObj.lower();
} // end function mohrCloseButton

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  mohrRockInfoButton (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  Widget, XtPointer, XtPointer
//
//  State Changes:  A dialog window pops up using Robert's little
//                  infoWidget function.  The Mohr Option window is sent
//                  as the parent, and char * text is the text that will
//                  appear in the info window.
//
//  Purpose:  This function cites the book that the Rock Type data came from
//
// Last Modified:  25 June 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void mohrRockInfoButton(Widget, XtPointer, XtPointer)
{
   // Get attributes of option window to pass the dialog form as parent
   MohrOptionType *optionAttrib = mohrOptionObj.getAttributes();

   // text is what will appear in the middle of the window as text!
   char * text = "Rock Type data obtained from:\n\n   Hoek, E. & Brown, E. T.  (19
80) Underground Excavations in Rock.\n      The Institution of Mining and Metallurg
y, London.\n      Pp. 141, 149, 176.\n\n   Hoek, E. & Brown, E. T.  (1988) \"The Ho
ek-Brown Failure Criterion--\n      a 1988 Update.\"  15th Canadian Rock Mechanics
Symposium.\n      The Department of Civil Engineering, University of Toronto.\n
   Pp. 31-38.\n\n   Goodman, Richard E.  (1980) Introduction to Rock Mechanics.\n
```

```
     John Wiley & Sons, New York.  P. 58.\0";

   // option window becomes parent of new info window
   infoWidget(optionAttrib->mohrDialogForm, text);
} // end function mohrRockInfoButton

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  mohrUnitInfoButton (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  Widget, XtPointer, XtPointer
//
//  State Changes:  A dialog window pops up using Robert's little
//                  infoWidget function.  The Mohr Option window is sent
//                  as the parent, and char * text is the text that will
//                  appear in the info window.
//
//  Purpose:  This function informs the user as to the units used in
//            Mohr Graph.
//
// Last Modified:  07 January 1998, J. Buckner (jbuckner@cs.trinity.edu)
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void mohrUnitInfoButton(Widget, XtPointer, XtPointer)
{
   // Get attributes of option window to pass the dialog form as parent
   MohrOptionType *optionAttrib = mohrOptionObj.getAttributes();

   // text is what will appear in the middle of the window as text!
   char * text =
      "All stresses including fluid pressure are expressed in terms of\n MegaPasca
ls.\0";

   // option window becomes parent of new info window
   infoWidget(optionAttrib->mohrDialogForm, text);
} // end function mohrUnitInfoButton

void mohrChangeCStr(Widget w, XtPointer clientData, XtPointer){
   MohrOptionType *moattr = mohrOptionObj.getAttributes();
   char c[20];
   sprintf(c,"%s",XtName(XtParent(w)));

   if (strcmp(c, "rockCStr0") ==0){
      switch ( (int)clientData) {
         case 0:
            moattr->rockCStrIndex[0] = 5;
            mohrObj.cValue = mohrObj.samplesCStr[5];
            break;
         case 1:
            moattr->rockCStrIndex[0] = 6;
            mohrObj.cValue = mohrObj.samplesCStr[6];
            break;
         case 2:
            moattr->rockCStrIndex[0] = 7;
            mohrObj.cValue = mohrObj.samplesCStr[7];
            break;
         case 3:
            moattr->rockCStrIndex[0] = 8;
            mohrObj.cValue = mohrObj.samplesCStr[8];
            break;
         case 4:
            moattr->rockCStrIndex[0] = 9;
            mohrObj.cValue = mohrObj.samplesCStr[9];
            break;
         case 5:
            moattr->rockCStrIndex[0] = 15;
            mohrObj.cValue = mohrObj.samplesCStr[15];
            break;
         case 6:
            moattr->rockCStrIndex[0] = 16;
            mohrObj.cValue = mohrObj.samplesCStr[16];
```



```
            break;
      }
   }
   else if (strcmp(c, "rockCStr1") ==0){
      switch ( (int)clientData) {
         case 0:
            moattr->rockCStrIndex[1] = 3;
            mohrObj.cValue = mohrObj.samplesCStr[3];
            break;
         case 1:
            moattr->rockCStrIndex[1] = 10;
            mohrObj.cValue = mohrObj.samplesCStr[10];
            break;
         case 2:
            moattr->rockCStrIndex[1] = 11;
            mohrObj.cValue = mohrObj.samplesCStr[11];
            break;
      }
   }
   else if (strcmp(c, "rockCStr2") ==0){
      switch ( (int)clientData) {
         case 0:
            moattr->rockCStrIndex[2] = 0;
            mohrObj.cValue = mohrObj.samplesCStr[0];
            break;
         case 1:
            moattr->rockCStrIndex[2] = 1;
            mohrObj.cValue = mohrObj.samplesCStr[1];
            break;
         case 2:
            moattr->rockCStrIndex[2] = 2;
            mohrObj.cValue = mohrObj.samplesCStr[2];
            break;
         case 3:
            moattr->rockCStrIndex[2] = 14;
            mohrObj.cValue = mohrObj.samplesCStr[14];
            break;
         case 4:
            moattr->rockCStrIndex[2] = 4;
            mohrObj.cValue = mohrObj.samplesCStr[4];
            break;
      }
   }
   else if (strcmp(c, "rockCStr3") ==0){
      switch ( (int)clientData) {
         case 0:
            moattr->rockCStrIndex[3] = 20;
            mohrObj.cValue = mohrObj.samplesCStr[20];
            break;
         case 1:
            moattr->rockCStrIndex[3] = 19;
            mohrObj.cValue = mohrObj.samplesCStr[19];
            break;
         case 2:
            moattr->rockCStrIndex[3] = 21;
            mohrObj.cValue = mohrObj.samplesCStr[21];
            break;
         case 3:
            moattr->rockCStrIndex[3] = 22;
            mohrObj.cValue = mohrObj.samplesCStr[22];
            break;
         case 4:
            moattr->rockCStrIndex[3] = 23;
            mohrObj.cValue = mohrObj.samplesCStr[23];
            break;
      }
   }
   else if (strcmp(c, "rockCStr4") ==0){
      switch ( (int)clientData) {
         case 0:
            moattr->rockCStrIndex[4] = 17;
```

32 / 236

```
            mohrObj.cValue = mohrObj.samplesCStr[17];
            break;
         case 1:
            moattr->rockCStrIndex[4] = 18;
            mohrObj.cValue = mohrObj.samplesCStr[18];
            break;
         case 2:
            moattr->rockCStrIndex[4] = 12;
            mohrObj.cValue = mohrObj.samplesCStr[12];
            break;
         case 3:
            moattr->rockCStrIndex[4] = 13;
            mohrObj.cValue = mohrObj.samplesCStr[13];
            break;
      }
   }
   sprintf(c,"%.11f", mohrObj.cValue);

   XmTextSetString(moattr->cText, c);

   mohrObj.calculate();
   mohrObj.display();
} // end of mohrChangeCStr

void mohrApplyButton(Widget, XtPointer, XtPointer){
   MohrAttributeType *mattr = mohrObj.getAttributes();
   MohrOptionType *moattr = mohrOptionObj.getAttributes();

   int sliders[3];
   int hasneg = 0;
   int smallest = 0, largest = 0;
        int normalize = FALSE;
        double norm_factor = 1.0;

   // Get values of the mohr sliders
   for (int i = 0; i < 3; i++)  {
      XmScaleGetValue(moattr->scale[i], &sliders[i]);
      if (mattr->effStress[i] < 0.0)
         hasneg = 1;
      if (sliders[i] < sliders[smallest])
         smallest = i;
      if (mattr->effStress[i] > mattr->effStress[largest])
         largest = i;
   } // end for int i

   // If there is a negative value, want to shift all numbers by
   // subtracting the tensile str.  This is only when none of the
   // numbers are less than the tensile str

   char buf[2000];
      sprintf(buf,
"Warning negative number entered that is\ngreater than computed tensile strength.\
nAssuming tensile strength equal to\nminimum value minus one.");

   // test to see if normalization must take place and by what factor
   if (mattr->effStress[largest] > 100) {
        norm_factor = 100.0 / mattr->effStress[largest];
        normalize = TRUE;
   }

   if (hasneg == 0) {
      cmdObj.setValues(mattr->effStress[0], mattr->effStress[1],
                       mattr->effStress[2], normalize);
      cmdObj.setFluidPress(mohrObj.fluidPressure, normalize, norm_factor);
      cmdObj.setTensileStr(mohrObj.tensileStr, normalize, norm_factor);
   } // end if
   else if (mohrObj.hasTensileStr) {
      if (mohrObj.tensileStr < mattr->effStress[smallest]){
         cmdObj.setValues(
            mattr->effStress[0] - mohrObj.tensileStr,
            mattr->effStress[1] - mohrObj.tensileStr,
```



```
            mattr->effStress[2] - mohrObj.tensileStr, normalize);
            cmdObj.setFluidPress(mohrObj.fluidPressure, normalize, norm_factor);
            cmdObj.setTensileStr(mohrObj.tensileStr, normalize, norm_factor);
      } // end if
      else
         infoWidget(moattr->mohrDialogForm, buf, mohrApplyOKCB);
   } // end else if
   else
      infoWidget(moattr->mohrDialogForm, buf, mohrApplyOKCB);


} // end of mohrApplyButton

void mohrApplyOKCB(Widget w, XtPointer, XtPointer){
   MohrAttributeType *mattr = mohrObj.getAttributes();
   MohrOptionType *moattr = mohrOptionObj.getAttributes();
   int sliders[3];
   int min = 0;

   for (int i = 0; i < 3; i++)  {
      XmScaleGetValue(moattr->scale[i], &sliders[i]);

      if (sliders[i] < sliders[min])
         min = i;
   } // end for int i

   double tensile = mattr->effStress[min] - 1.0;
   cmdObj.setValues(mattr->effStress[0] - tensile,
                    mattr->effStress[1] - tensile,
                    mattr->effStress[2] - tensile);

   cmdObj.setFluidPress(mohrObj.fluidPressure);
   cmdObj.setTensileStr(mohrObj.tensileStr);

   if (w != NULL)
      XtDestroyWidget(w);
} // end of mohrApplyOKCB

void mohrChangeFluidField(Widget, XtPointer, XtPointer){
   MohrOptionType *moattr = mohrOptionObj.getAttributes();
   MohrAttributeType * mohrObjAttrib = mohrObj.getAttributes();
   char strHolder[100]; // holder for transfer of numbers to ASCII
   double tmpValue;

   if (XmTextFieldGetString(moattr->fluidText) != NULL) {
      char *val = XmTextFieldGetString(moattr->fluidText);
      if ((strlen(val) == 0) || !(trapAlpha(val))){
         infoWidget(moattr->mohrDialogForm,
            "Not allowed to have empty or alphabetical value");
         sprintf(strHolder,"%5.2lf", mohrObj.fluidPressure);
         XmTextFieldSetString(moattr->fluidText, strHolder);
         return;
      }
      tmpValue = atof(val);
      if (tmpValue < 0.0) {
         infoWidget(moattr->mohrDialogForm,
                    "Fluid Pressure must be a positive number");
         sprintf(strHolder,"%5.2lf", mohrObj.fluidPressure);
         XmTextFieldSetString(moattr->fluidText, strHolder);
         return;
      }

      if (tmpValue > mohrObjAttrib->maxScale - 1){
         infoWidget(moattr->mohrDialogForm,"Fluid Pressure is out of bounds");
         sprintf(strHolder,"%5.2lf", mohrObj.fluidPressure);
         XmTextFieldSetString(moattr->fluidText, strHolder);
         return;
      }

      mohrObj.fluidPressure = tmpValue;
```

33/236

```
      XmScaleSetValue(moattr->fluidScale,
        (int) (mohrObj.fluidPressure * 100.0));

      updateEffective();
      mohrObj.failure();
      mohrObj.display();
   } // end if

} // end of mohrChangeFluidField

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  mohrChangeFluidScale (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  Widget, XtPointer, XtPointer callData
//
//  State Changes:  mohrObj.fluidPressure, updates the fluidPressure
//                  also calls updateEffective(), mohrObj.failure(),
//                  mohrObj.display()
//
//  Purpose:  This program updates the fluid pressure from its scale
//            in the options window of the Mohr graph
//
// Last Modified:  1 July 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void mohrChangeFluidScale(Widget, XtPointer, XtPointer callData)
{
   XmScaleCallbackStruct
      * cbs = (XmScaleCallbackStruct *) callData;

   //  The variable below holds the address to mohrObj's
   //  attributes private member variable.
   MohrAttributeType
      * mohrObjAttrib = mohrObj.getAttributes();
   MohrOptionType *moattr = mohrOptionObj.getAttributes();

   char strHolder[100]; // holder for transfer of numbers to ASCII

   // update the ratio
   mohrObj.fluidPressure =  (double) ((cbs->value) / 100.0);

   sprintf(strHolder,"%5.2lf", mohrObj.fluidPressure);
   XmTextFieldSetString(moattr->fluidText, strHolder);

   updateEffective();
   mohrObj.failure();
   mohrObj.display();
}  // end of mohrs3Tos1RatioScaleChange

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  updateEffective (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  void
//
//  State Changes:  mohrObj.attributes->effStress are changed and this change
//                  is displayed on screen in the text fields
//
//  Purpose:  This function takes care of calculating effective stress
//
// Last Modified:  14 July 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void updateEffective()
{
   MohrOptionType  // option window attributes
      *moattr = mohrOptionObj.getAttributes();
   MohrAttributeType  // mohr graph attributes
      *mattr = mohrObj.getAttributes();
```



```
   char strs[3][100]; // holder for transfer of numbers to ASCII

   int
      i,    // loop index
      min, mid, max; // hold indeces of smallest, middle, and largest sigmas

   if (mattr->stressMode == 0) // stresses are dependent
   {
      findMinMidMax( & min, & mid, & max, mattr->sigmas, 3 );

      // we are now ready to set the effective stresses
      mattr->effStress[max] = (double)mattr->sigmas[max]
                              - mohrObj.fluidPressure;
      mattr->effStress[mid] = (double)mattr->sigmas[mid]
                              - (mattr->s2Tos1Ratio * mohrObj.fluidPressure);
      mattr->effStress[min] = (double)mattr->sigmas[min]
                              - (mattr->s3Tos1Ratio * mohrObj.fluidPressure);
   } // end if (mattr->stressMode == 0)

   else // stresses are independent so set them strait
   {
      mattr->effStress[0] = (double)mattr->sigmas[0] - mohrObj.fluidPressure;
      mattr->effStress[1] = (double)mattr->sigmas[1] - mohrObj.fluidPressure;
      mattr->effStress[2] = (double)mattr->sigmas[2] - mohrObj.fluidPressure;
   } // end else

   for (i = 0; i < 3; i++)  // set the scales on the screen
   {
      // check bounds
      if (mattr->effStress[i] > mattr->maxScale)
      {
         if(moattr->isInfoWinOpen == FALSE)
         {  // if there is no current error win open
            moattr->isInfoWinOpen = TRUE;
            // open the window and tell the world about it
            infoWidget(moattr->mohrDialogForm,
                       "Warning:  Actual Effective Stress Value out of bounds.\nSe
tting value to an artifical bound value.",
                       closeBoundsErrorWin);
         }
         mattr->effStress[i] = mattr->maxScale;
      }

      if (mattr->effStress[i] < mattr->minScale)
      {
         if(moattr->isInfoWinOpen == FALSE)
         {  // if there is no current error win open
            moattr->isInfoWinOpen = TRUE;
            // open the window and tell the world about it
            infoWidget(moattr->mohrDialogForm,
                       "Warning:  Actual Effective Stress Value out of bounds.\nSe
tting value to an artifical bound value.",
                       closeBoundsErrorWin);
         }
         mattr->effStress[i] = mattr->minScale;
      }
      sprintf(strs[i],"%5.2lf",mattr->effStress[i]);
      XmTextFieldSetString(moattr->effText[i], strs[i]);
   } // end for int i

   updateEffRatios();  // update the effective stress ratio text boxes
} // end of updateEffective

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  updateEffRatios (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  void
//
//  State Changes:  the effective ratio text fields are changed and that
```

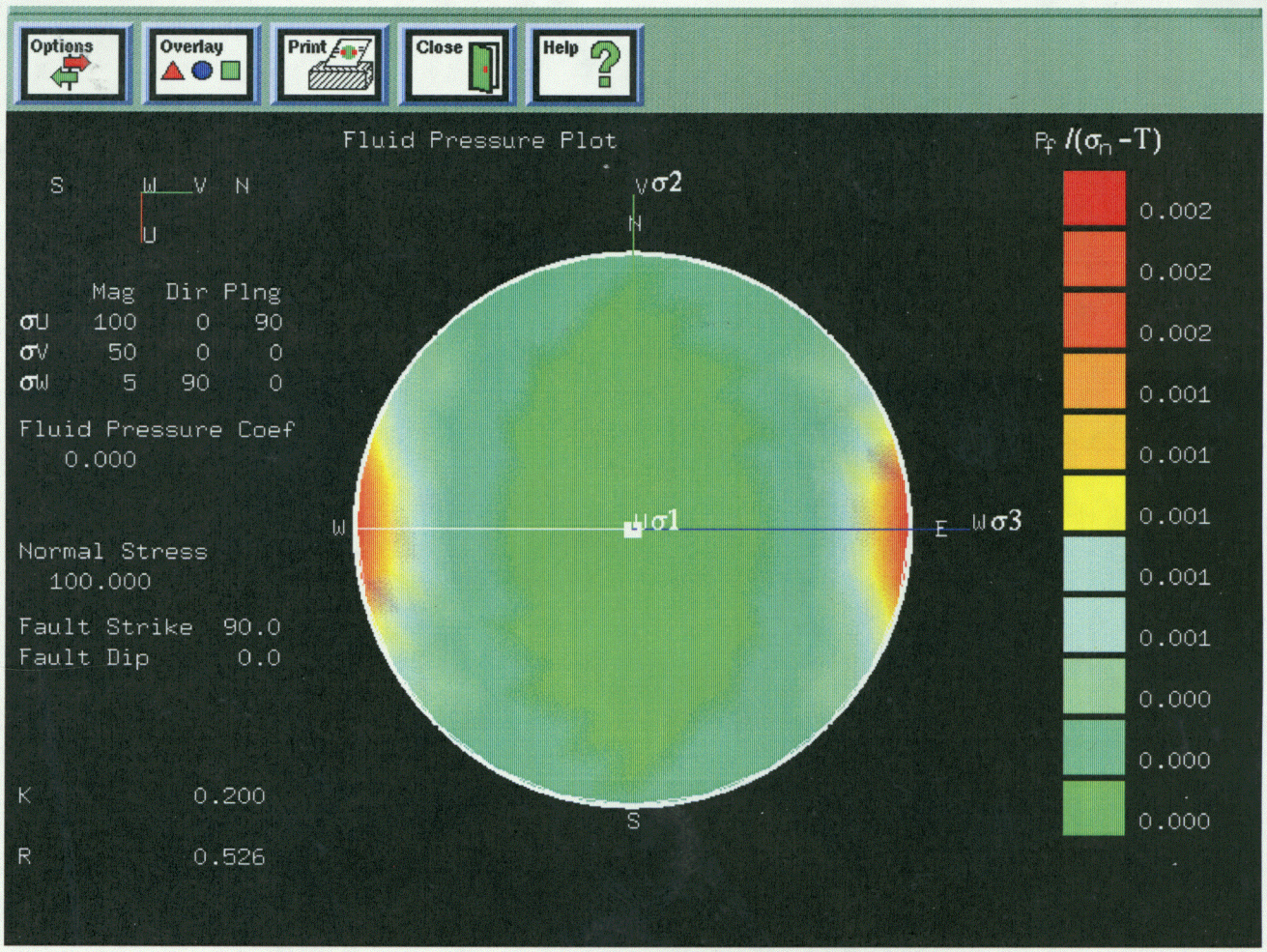
34/226

```
//                 is all
//
//  Purpose:  This function takes care of calculating effective stress
//            ratios
//
// Last Modified:  14 July 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void updateEffRatios()
{
   MohrOptionType  // option window attributes
      *moattr = mohrOptionObj.getAttributes();
   MohrAttributeType  // mohr graph attributes
      *mattr = mohrObj.getAttributes();
   int
      min, mid, max; // hold indeces of smallest, middle, and largest sigma

   double holder; // holder for storing ratios
   char str[100]; // holder for string that goes in text boxes

   findMinMidMax( & min, & mid, & max, mattr->sigmas, 3 );

   if(mattr->effStress[max] != 0.0)   // can't divide by 0.0
   {
      holder = (double) (mattr->effStress[min] / mattr->effStress[max]);
      sprintf(str, "%5.3lf", holder);  // make the number a string
   }
   else  // division by zero is not defined in our system of mathematics
      sprintf(str, "%s", "Undefined\0");
   // update the text box
   XmTextFieldSetString(moattr->effs3Tos1RatioText, str);

   if(mattr->effStress[max] != 0)   // can't divide by 0.0
   {
      holder = (double) (mattr->effStress[mid] / mattr->effStress[max]);
      sprintf(str, "%5.3lf", holder); // make the number a string
   }
   else  // division by zero is not defined in our system of mathematics
      sprintf(str, "%s", "Undefined\0");
   // update the text box
   XmTextFieldSetString(moattr->effs2Tos1RatioText, str);
} // end function updateEffRatios

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  closeBoundsErrorWin (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  Widget w, XtPointer clientData, XtPointer callData
//              these are only passed along to destroyFS
//
//  State Changes:  an info dialog box is destroyed
//
//  Purpose:  This function makes sure that the isInfoWinOpen gets set to
//            flase when the info win is closed
//
// Last Modified:  8 July 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void closeBoundsErrorWin(Widget w, XtPointer clientData, XtPointer callData)
{
   MohrOptionType  // option window attributes
      * moattr = mohrOptionObj.getAttributes();

   moattr->isInfoWinOpen = FALSE; // the info win is about to be closed
   destroyFS(w, clientData, callData); // kill the info win
} // end function closeBoundsErrorWin

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function: mohrArrowCB (not a member function)
//
//  File:  mohrOptionCB.c++
//
```



```
//  Arguments:  Widget, XtPointer, XtPointer
//
//  State Changes:  Makes 1/2 the options window dissappear/reappear
//
//  Purpose:  To conserve screen space
//
// Last Modified:  1 August 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void mohrArrowCB(Widget, XtPointer, XtPointer){
        MohrOptionType *moattr = mohrOptionObj.getAttributes();

        moattr->arrowDir = !(moattr->arrowDir);

        if (moattr->arrowDir){
                XtManageChild(moattr->rockRenderRow);
                XtVaSetValues(moattr->arrowButton, XmNarrowDirection, XmARROW_UP,
NULL);
        }
        else {
                XtUnmanageChild(moattr->rockRenderRow);
                XtVaSetValues(moattr->arrowButton, XmNarrowDirection, XmARROW_DOWN
, NULL);   }
} // end of mohrArrowCB

int trapAlpha(char * in_str)
{
  int i;

  for(i = 0; i < strlen(in_str); i++)
  {
    if ((in_str[i] < '0') || (in_str[i] > '9'))
    {
       if ((in_str[i] != '.') && (in_str[i] != '-') &&
           (in_str[i] != '+') && (in_str[i] != ' '))
        return 0;
    }
  }
  return 1;
}
```

35/230



Example of Typical color display.

SOFTWARE
DEVELOPMENT PLAN

37/236

REVISED SOFTWARE DEVELOPMENT PLAN FOR 3DSTRESS

Prepared for

Nuclear Regulatory Commission
Contract NRC-02-97-009

Prepared by

Joseph H. Bangs

Modifications Made by David A. Ferrill and Joshua Bucknor on 30 December 1999.

December 1999

PI for 3DStress Development.

Center for Nuclear Waste Regulatory Analyses
San Antonio, Texas

Approved by: H. Lawrence McKague **Date:** 12/30/99
H. Lawrence McKague

This was revised in 12/99 To allow for the 3DStress 1.3.2 code To be run on a Sun/Solaris machine.

QA
12/31/99

CONTENTS

CONTENTS (cont'd)

CONTENTS (cont'd)

41/234

FIGURES

42/234

TABLES

1 SCOPE

This document establishes the Software Development Plan (SDP) to be implemented by the Center for Nuclear Waste Regulatory Analyses (CNWRA) for the development and release of the 3DStress version 1.3 software application.. The software will be provided to the government (acquirer) without proprietary restrictions.

1.1 Identification

This SDP applies to software modifications and corrections to be made to version 1.2 of the 3DStress application. The modified code will be identified as 3DStress version 1.3.

1.2 System Overview

The 3DStress application is used by scientists and engineers to study the relationship between static stress fields and geologic faulting. 3DStress utilizes user defined stress fields to compute the likelihood of fault displacement based on the fault orientation. 3DStress provides user input, computation, and data visualization tools to create an interactive environment in which various stress models may be studied and explored efficiently.

3DStress executes on a Silicon Graphics workstation running the IRIX operating system. The application does not communicate or interface with any other computer system or software application.

1.3 Document Overview

This SDP defines the plan for management, development, and software maintenance for the 3DStress software application. This document contains the procedures to address the following program management tasks:

a. Software design practices
b. Software Quality Assurance
c. Software configuration management
d. Software engineering standards
e. Software development process
f. Organizational structure
g. Schedule

These guidelines will ensure the efficient utilization of project resources to deliver a high quality software product in a timely manner.

1.4 Relationship to Other Plans

This SDP is not related to any other plan.

2 REFERENCED DOCUMENTS

The following documents provide guidelines for software development and documentation activities. In the event of conflict between this document and those referenced herein, the contents of this document shall be considered superseding requirements.

CNWRA-TOP-18	1 MAY 98	Development and Control of Scientific and Engineering Software

3 OVERVIEW OF REQUIRED WORK

3.1 General

CNWRA has modified version 1.2 of 3DStress to enhance software performance, provide additional capabilities and correct software defects. All functionality provided by the current version will be duplicated or replaced in the new version. The host hardware platform for 3DStress will be either a Silicon Graphics workstation running version 6.x or 5.x of the IRIX operating system, or a Sun Ultra workstation running version 2.7 or higher of the Solaris operating system.

CNWRA will perform the software requirements analysis, design, development and testing necessary to deliver a reliable software and documentation product at the end of the development project.

3.2 Software Functionality

The 3DStress application calculates either the slip tendency or dilation tendency of one or more geologic faults for a static three dimensional stress field. The application displays various data plots in which colors and 3D surfaces are rendered to convey the computational results to the software user. 3DStress will read data files containing fault geometry information and will save copies of the various display windows for hard copy output or as input to other software applications.

3.3 Software Design and Development

The new version of 3DStress will be designed to meet or exceed the requirements for the existing application version.

All software will be developed in the C++ programming language unless highly specialized coding is required for performance beyond the ability of the commercial compiler. A commercial source control product will be employed to track and coordinate all modifications to the software source code.

Like the existing version, the new 3DStress application will operate in a stand-alone mode requiring operator control for the execution of all software operations. The application will retain the existing man-machine-interface based on the X Windows program environment and the Open GL graphics rendering library.

All new software development and modifications will be done in accordance with CNWRA TOP-18. All existing code being reused will not be unnecessarily modified or documented to CNWRA-TOP-18. Reused code will consist of code used as is or with only minor customization for use with 3DStress version 1.3.

3.4 Hardware Configurations

The 3DStress software application will operate on a single Silicon Graphics computer platform. The software will operate on any SGI equipped with a monitor, keyboard, mouse and removable media drive for software installation. Due to the extensive computational nature of 3DStress, CNWRA recommends the following hardware configuration for acceptable calculation and display performance:

200 MHz Iris Processor or better with floating point coprocessor (on Silicon Graphics platform) or 440 MHz UltraSPARC-IIi processor or better (on Sun platform)
128 Mbytes RAM
High Impact graphics card (on Silicon Graphics platform) or Creater 3D graphics card (on Sun platform)
9 GB hard drive
19" monitor or larger
8 mm Tape or Digital Audio Tape drive
CDROM
Network connection

4 PLANS FOR PERFORMING GENERAL SOFTWARE DEVELOPMENT ACTIVITIES

The following sections outline plans for performing general software development activities for the 3DStress software application.

4.1 Software Development Process

CNWRA will utilize a Grand Design strategy for development of the 3DStress application. The Grand Design approach results in a single software build and is appropriate for this project because:

a. The 3DStress software requirements are well known and documented in the existing 3DStress software documentation.
b. The 3DStress application is not a large development effort and can be accomplished in a short time frame.
c. Once the final software requirements are specified the development schedule will be firm and will not be altered due to changing technical requirements.

4.2 General Plans for Software Development

The following sections define the software development practices and standards to be applied to the 3DStress software development effort.

4.2.1 Software Development Methods

For the same reasons the Grand Design program strategy was selected, the classic life cycle method of software development will be employed. The classic life cycle method involves requirements analysis, design, coding, module testing, integration, system level testing and implementation.

All software will be developed in the C++ programming language unless specialized coding is required for software performance beyond the capability of the C++ compiler. Any deviations from the use of the C++ compiler will be reviewed by the software development team to determine the impact on related software modules.

CNWRA will base the C++ software design and implementation of the following development approaches:

1. Information Hiding - Decomposition of a system into units, such that each is characterized by its knowledge of a design decision which it hides from all others. The design decision may relate to either a routine or data. Access to hidden data or routines will be controlled through well defined interfaces with limited update privileges.
2. Encapsulation - Related data and data processing/manipulation processes will be organized or structured as classes reflecting 3DStress component organization, interfaces and processing. Subclasses will be derived from parent classes until the child class represents a unit process or interface in enough detail to express the class behavior as data variables and member functions.

4.2.2 Standards for Software Products

The following sections define the standards to be followed in developing the software requirements, design, coding, test procedures and documenting test results for the 3DStress development project.

4.2.2.1 Software Design Standards

Software design is the process by which requirements are translated into software representations using structured analysis techniques. A preliminary software design will define modifications to existing or additional 3DStress computational and/or display capabilities. These capabilities will be mapped to software classes by functional and data access requirements. A subsequent refinement of the design will lead to detailed class definitions optimized for efficient software operation.

Throughout the design process, the quality of the evolving design will be reviewed by the software developer with the software or project manager. The software team will adhere to the following design quality criteria:

1. The design will be modular and logically partitioned into components that perform distinct functions.
2. The design will contain distinct classes reflecting the modular design of the software.
3. The design will lead to software modules that exhibit independent functional characteristics
4. The design will strive to simplify user interfaces.
5. The design will incorporate the concept of abstraction, enabling the designer to simplify and reuse software components.

4.2.2.2 Software Coding Standards

Coding will translate the software design into C++ language software files and will begin after the detailed design has been completed and reviewed with the project and element managers. CNWRA will code the software to have the following characteristics:

1. Ease of code to design translation
2. Maximum compiler efficiency
3. Maximum use of development tools
4. Maintainability

CNWRA will employ a coding style that stresses simplicity and clarity. This approach will be applied to data declaration, statement construction, and data input and output. This coding philosophy will enhance the software readability while simplifying the test/debug/implement portion of the software development cycle.

4.2.2.2.1 Headings

Each software unit will begin with a unit header that explains the following:

1. Description and purpose of the operation
2. General unit design

3. Initialization
4. Global interactions (if any)
5. Error conditions and handling

4.2.2.2.2 Comments

Source code will be explained with comments. Comments will explain the intended operation, logic and possible error conditions associated with code sections. General comments will precede code sections while detailed comments will be interspersed in the code. Comments will be written for readers with moderate software comprehension.

4.2.2.2.3 Variable Naming

Names of variables and classes will be descriptive and indicative of program activity. Names shall avoid the use of abbreviations, mnemonics and jargon within the constraint of size limitations. Comments will be used to explain the role of all non-trivial variables and classes at the time of declaration. Names will adhere to consistent formats across all code modules. This will include the use of capitalization, under scores and unique letter combinations to identify specific classes of variables.

4.2.2.2.4 Restrictions

Previous versions of the 3DStress application software were developed using the C/C++ programming language and incorporated function calls to various operating system, X Windows, and Open GL libraries bundled with each Silicon Graphics hardware platform. No restrictions will be placed on the use of additional software libraries except that the use of all third party library products will be documented in the source code and software version or release description.

Also, the software team will restrict its use of multiple inheritance to only those version 1.2 base classes already utilizing multiple inheritance. The use of multiple inheritance is discouraged and the software developers will also attempt to avoid mixing traditional C function calls with their equivalent C++ counterparts.

4.2.2.2.5 Complexity

Code aggregates will be limited to the level that a software programmer can understand them without in-depth study. Individual coding statements will be simple and direct and will not be convoluted for esoteric or marginal efficiency gains. Individual source code statements will be simplified by avoiding complicated conditional statements, tests on negative test conditions, and unnecessary nesting in loops or conditions. Source code statements will use parenthesis to clarify statement content. Related code such as loops, blocks and cases will be grouped and commented as a functional entity.

4.2.2.3 Software Test Standards

Software testing accounts for a large percentage of technical effort in the software development process. The objective of software testing is to identify errors. To fulfill this objective, CNWRA will utilize a series of steps in testing the software first at the unit level, and then progressing to the integration and system levels.

Unit level tests will concentrate on functional verification of software modules prior to incorporation into the program structure. Unit testing makes heavy use of white box testing techniques to exercise specific paths in a module's control structure to maximize error detection. After unit testing, modules are assembled to form the complete software package.

Integration testing addresses reliability issues associated with program verification and construction. Software modules must work in concert to provide program functionality. Integration testing reveals errors in module interactions and deficiencies in meeting functional requirements. After successful integration testing, a set of high order system tests are conducted.

System validation testing will demonstrate traceability to software requirements, and will provide assurance that software meets functional, behavioral and performance requirements. The validated software will then be installed in an operational environment to demonstrate system performance.

4.2.3 Reusable Software Products

The following sections outline the approach for incorporating reusable software products and developing new reusable software for the 3DStress software application.

4.2.3.1 Incorporating Reusable Software Products

CNWRA will investigate several potential sources of reusable software for the 3DStress application. Version 1.2 of 3DStress is written in the C++ programming language and several software modules will be incorporated directly into version 1.3. Wherever possible, existing software will be analyzed to determine if software modifications are necessary to enhance the reusability of the source code in future software versions.

A second source of reusable software will be in the form of commercial device drivers, function and class libraries, operating system resources and documentation generators. Wherever possible and advisable, CNWRA will identify commercial products for incorporation or utilization in the development of the new version of 3DStress.

4.2.3.2 Developing Reusable Software Products

This development project will apply good software development techniques in developing the new 3DStress version. By combining good software development practices with the use of C++, which lends itself to reuse through class inheritance, the 3DStress project will result in some software that is reusable. However, it is not the goal of this project to develop reusable software at the expense of software efficiency or simplicity.

4.2.4 Handling of Critical Requirements

The following sections outline the approach for handling critical requirements for the 3DStress project.

4.2.4.1 Safety Assurance

3DStress software activities do not require safety assurances. This paragraph has been tailored out.

4.2.4.2 Security Assurance

The 3DStress application does not contain any security related procedures or data. This activity is tailored out.

4.2.4.3 Privacy Assurance

The 3DStress application will not contain any privacy related procedures or data. This activity is tailored out.

4.2.4.4 Assurance of Other Critical Requirements

Requirements deemed critical by the technical directive will be presented by the acquirer and will be incorporated into this plan as appropriate.

4.2.5 Computer Hardware Resource Utilization

The 3DStress application will be developed, tested and executed on existing CNWRA Silicon Graphics and Sun workstations. No additional hardware resources are required for this development effort.

4.2.6 Recording Rationale

Software development activities will be documented in Software Development Files (SDFs) maintained by individual software developers. These files will contain engineering assumptions as well as standard software development information. Rationale will be recorded and submitted to the project manager at the conclusion of the development effort. Key decisions and rationale will be discussed during technical and management reviews throughout the development project.

4.2.7 Access for Acquirer Review

Throughout the project performance period, the CNWRA project team will be available for telephone discussions regarding the development effort. All development activities will take place in the CNWRA GIS laboratory, that is accessible to acquirer personnel.

5 PLANS FOR PERFORMING DETAILED SOFTWARE DEVELOPMENT ACTIVITIES

The following sections outline the detailed Software Development Activities for the 3DStress project.

5.1 Project Planning and Oversight

The following sections describe the approach to be employed for project planning and oversight of the 3DStress development project.

5.1.1 Software Development Planning

This document contains the pertinent information related to software development planning. The project team through the Project Manager may make recommendations for improvements or changes to the SDP. The Project Manager will determine the impact on schedule and cost and, if appropriate for the program, present the SDP modifications to the acquirer for approval and contract modifications.

5.1.2 Software Test Planning

Based on the results of the Software Requirements analysis, a Software Test Plan (STP) will be developed for qualification testing of the 3DStress application. This plan will describe the software test environment, the test(s) to be performed, and the test schedule. Test results will be recorded in the SDFs and will be available for acquirer review.

5.1.3 System Test Planning

The 3DStress application is a single build computer software configuration item (CSCI) that interacts directly with the software user. System testing is not separable from CSCI testing and will thus be conducted with CSCI testing. System level tests will be defined in the STP.

5.1.4 Software Installation Planning

The 3DStress application will be delivered on removable media. The installation procedure and scripts will be designed, documented, and built to simplify the installation procedure. No hardware modifications are anticipated for the migration from version 1.2 to 1.3. Recipients of version 1.3 will be responsible for internally coordinating local software installations.

5.1.5 Software Transition Planning

CNWRA will document any version specific requirements associated with the new release of 3DStress. Recipients of the new version will be responsible for internally coordinating any file translations necessary to support version 1.3.

5.1.6 Following and Updating Plans, Including the Intervals for Management Review

CNWRA will conduct the development and testing of 3DStress version 1.3 in accordance with the SDP and STP. The software development team will meet periodically with the Project Manager to verify the software process is adhering to these plans. At this time, no changes to the plans are anticipated. However, should a plan need modification, the Project Manager will present the changes to the acquirer for approval and coordinate their implementation with the development team.

5.2 Establishing a Software Development Environment

The following sections outline the approach for establishing, controlling and maintaining the software environment for the 3DStress project.

5.2.1 Software Engineering Environment

Silicon Graphics and Sun software development environments will be established for this project in the CNWRA GIS Laboratory, Bldg. 189 at the CNWRA facility. All printed materials, vendor CD's, floppies and tapes will be stored in the GIS Laboratory. Intermediate disk backups will be made to tape and will also be stored in the GIS Laboratory.

5.2.2 Software Test Environment

CNWRA maintains four SGI and three Sun workstations for software development and GIS activities. Software development and testing will be conducted on these computers.

5.2.3 Software Development Library

The lead software developer will serve as the software librarian and will have primary control over the software development library. Because the software development team is small, all team members will have access to the library in the absence of the librarian. The librarian will establish a working library on the development Silicon Graphics workstation. Both libraries will be subdivided into a subdirectory structure designed to contain deliverable documents, software units, SDFs, and commercial software products.

5.2.4 Software Development Files

Informal Software Development Files (SDFs) will be created for the 3DStress software units. The SDFs will be created prior to the initiation of detailed design and shall be maintained for the duration of the project. They will be made available to the product evaluation team, quality assurance, and acquirer representatives as requested. The SDF may reference information in other project documents as necessary. All schedule and status data will be in other project documents. SDFs will be created and maintained by the programming staff under the direction of the Project Manager.

The SDFs will be maintained predominantly in electronic form. The electronic form will be a combination of plain ASCII and word processor files. If necessary paper submissions will be included in a binder and referenced in the electronic form.

SDFs will generally include the following information:

1. Record Sheet - The contents of the SDF are listed by item name and electronic name and location. The engineer responsible for the SDF is identified with the due date, completion date, originator sign-off, and reviewer sign-off.
2. Requirements Specification - All requirements that the CSU must satisfy are listed by reference to the applicable sections of the Software Requirements Specification.
3. Interface Description - Global variables/constants, calling sequences, and input/output formats are defined or referenced.
4. Preliminary Design - Preliminary design description.
5. Software Test Information - All test cases and test procedures are defined or referenced. Concurrent with code walk-through, the reviewer will verify that the test plan fully tests capabilities, interfaces, and design constraints.
6. Source Code Organization Description - A description of the location and directory structure of the CSU source code as well as commercial products used in the CSU.
7. Test Results - At all levels, records of test results are maintained by test case identifier, tester, date, and the revision status of test drivers, tools, database, and code tested. Significant differences between expected and actual results will be explained..
8. Software Problem Reports - SPR forms shall be used to document problems encountered in software and software documentation.
9. Notes - All explanatory materials relevant to the CSU are maintained in the section. Formal deviation and waivers are also kept in this section.
10. Reviewers Comments - Reviewers comments on the other sections of the SDF are kept in this section.

5.2.5 Non-deliverable Software

Where necessary, CNWRA will develop simulators to test software component functionality. The end item software will not utilize these test fixtures and therefore will not be delivered, controlled or documented to the software release standards.

5.3 System Requirements Analysis

The following sections describe the approach CNWRA will follow in developing the software system design for the 3DStress application.

5.3.1 Analysis of User Input

3DStress operates as a stand-alone software application requiring user directives to control application execution. Additional features planned for version 1.3 will be analyzed to design an optimal user interface environment. The primary user interface design criteria will be ease of operator control and the effective display of computational results. At this time, no additional user input devices are anticipated beyond the traditional input devices (keyboard, mouse) attached to standard SGI workstations.

5.3.2 Operational Concept

3DStress is designed to be an interactive software application utilized by research and scientific staff at irregular intervals. The software is not intended to become an integral part of day-to-day operations. Therefore an operational concept description will not be written for this application. This activity has been tailored out.

5.3.3 System Requirements

Based on experience with 3DStress version 1.2 and planned modifications for version 1.3, no system modifications are necessary for this release version.

5.4 System Design

Version 1.3 represents an incremental change to 3DStress version 1.2. The existing version 1.2 system design will be utilized in version 1.3.

5.5 Software Requirements Analysis

This version of 3DStress is intended to be a functional replacement of the current application. CNWRA will review and analyze the requirements for the new version to determine the operational concepts and software specific requirements of the new CSCI. Software technology areas needed to implement the new version will be evaluated with respect to technologies utilized in the current software revision. New requirements will be categorized as new technology or extensions of existing capabilities. The results of this analysis will be documented in the Software Requirements Description (SRD) as a reference for testing and validating the new software version.

5.6 Software Design

The following sections describe the approach CNWRA will follow in preparing the software design for 3DStress, version 1.3. The results of the software design process will be documented in the software development files maintained by the development team members.

5.6.1 CSCI-wide Design Decisions

CNWRA will analyze the SRD to refine the existing concept of data and event management within the current 3DStress application. The software team will prioritize event management and data processing tasks according to their impact on overall system performance and functionality. From this prioritization, the team will evaluate various models for allocating hardware and software resources during software execution. Any modifications to the current CSCI event and data management concepts will be documented in the SDFs.

5.6.2 CSCI Architectural Design

Using the high-level resource allocation model, the software team will design an internal CSCI architecture to implement the major functional requirements specified in the SRD. This design will define any new or modified classes of data and functionality necessary to implement the new software capabilities.

5.6.3 CSCI Detailed Design

CNWRA will refine the architectural design into individual software units by designing algorithmic approaches for implementing specific software requirements defined in the SRD. Algorithm development will focus on meeting or exceeding performance and functional specifications while adhering to the previously defined communication, processing and event management framework.

5.7 Software Implementation and Unit Testing

The following sections describe the approach to be followed for software implementation and unit testing for the 3DStress application.

5.7.1 Software Implementation

The software will be developed within the coding techniques described above in Section 4.2.1. All software will be developed in the C++ programming language unless highly specialized coding is required for performance beyond the ability of the C++ compiler to produce efficient binary executables. Any deviations from the use of C++ will be approved by the Project Manager.

No relational databases are required for the 3DStress application. All system configuration and geologic fault information will be maintained in plain ASCII text files or publicly defined binary file formats. ASCII file formats are generally preferred for all files except very large data files where ASCII storage is impractical.

5.7.2 Preparing for Unit Testing

Unit testing will be designed to verify the new software meets the detailed software design in the SDFs. CNWRA will develop test cases using by calculating outputs from known or controlled inputs for each major software unit. Controlled test cases will be computed using independent computations not relying on the newly developed software. Information regarding the test case computations and results will be documented in the SDFs.

5.7.3 Performing Unit Testing

As major software units are completed, the developer will conduct unit testing with test cases to verify the expected results.

5.7.4 Revision and Retesting

As needed, the developer will revise and retest software units to ensure compliance with the functionality described in the SDFs and SRD.

5.7.5 Analyzing and Recording Unit Test Results

Each iteration of testing and revision through the successful completion of the test case will be documented in the applicable SDF. Persistent test failure by a software unit will be analyzed to determine if failures are derived from an inadequate design, insufficient documentation, or improper coding practices.

5.8 Unit Integration and Testing

The following sections describe the approach to be followed for unit integration and testing for the 3DStress project.

5.8.1 Preparing for Unit Integration and Testing

Unit integration testing will be performed at the major software component level. CNWRA will develop test cases and data in terms of inputs and expected outputs for the control subsystem. Information regarding the test cases, procedures and results will be stored in the SDFs.

5.8.2 Performing Unit Integration and Testing

As major software units are ready for testing, the development team will conduct component level testing with test cases and verify the expected outputs.

5.8.3 Revision and Retesting

As needed, the developer will revise and retest software components to ensure compliance with the functionality described in the SDD and SRS.

5.8.4 Analyzing and Recording Unit Integration and Test Results

Each iteration of testing and revision through the successful completion of the test case will be documented in the applicable SDF. Persistent test failure by a software component will be analyzed to determine if failures are derived from an inadequate design, insufficient documentation, or fundamental errors in the CSCI architectural design.

5.9 CSCI Qualification Testing

The intent of the CSCI qualification testing for the 3DStress application is to verify that the new controller is functionally equivalent to the previous version and provides the additional capabilities described in the SRD.

The following sections describe the approach to be followed for CSCI qualification testing for the 3DStress application.

5.9.1 Independence in CSCI Qualification Testing

The Project Manager will assign an individual from the CNWRA , CNWRA QA or SwRI staff who has not participated in the design or development of the 3DStress to conduct formal testing of the CSCI.

5.9.2 Testing on the Target Computer System

All CSCI qualification testing will be performed on Silicon Graphics and Sun workstations available to the CNWRA.

5.9.3 Preparing for CSCI Qualification Testing

Upon completion of the CSCI software, the software will be entered into the Configuration Management (CM) system as the 3DStress product baseline.

The Software Test Plan (STP) and Software Test Procedures (STPr) will be given to the independent software tester in preparation for the dry run CSCI qualification test.

5.9.4 Dry Run of CSCI Qualification Testing

The software developer will dry run the test procedures to ensure that they are complete, accurate and are ready for witnessed testing. The results of the test will be recorded in the appropriate SDFs. Software Problem Reports will be prepared for problems uncovered during the testing process.

5.9.5 Performing CSCI Qualification Testing

A witnessed qualification test will be conducted at CNWRA to verify and demonstrate that the 3DStress application meets the system and software requirements stated in the SRD. If additional revision and retesting is required, the appropriate portions of the CSCI qualification test will be rerun after completion of the revisions.

5.9.6 Revision and Retesting

Revisions, based on SPR (corrective action) processing, and retesting will be accomplished prior to final approval of the qualification testing. Where necessary SDFs will be updated to reflect the revisions and retesting.

5.9.7 Analyzing and Recording CSCI Qualification Test Results

When the CSCI qualification testing is competed, the test results will be recorded in a Software Test Report. If revisions to the 3DStress application were made during the qualification test process, the qualified software will be resubmitted to CM as the new baseline version.

5.10 CSCI/HWCI Integration and Acceptance Testing

The 3DStress application is a single build CSCI designed to run on Silicon Graphics or Sun workstations. 3DStress will be tested on CNWRA SGI machines that were not used in the software development process.

CNWRA has prepared a standard installation test case that shall be run after software installation to verify correct software installation. This acceptance test procedure is documented in the 3DStress on-line help manual.

5.11 System Qualification Testing

The 3DStress application does not interface other computer hardware or software systems. This paragraph has been tailored out.

5.12 Preparing for Software Use

The following sections describe the approach to be followed for preparing the 3DStress application for distribution to existing users.

5.12.1 Preparing the Executable Software

CNWRA will prepare the executable software for delivery to the user community. This preparation will include script and data files, executables, shared object libraries, configuration files, and any other software files required to operate the application. These files will be stored on standard removable storage media such as CDs or tapes.

5.12.2 Preparing Version Descriptions for User Sites

CNWRA will prepare a Software Release Notice for delivery with the 3DStress application to identify and describe the released software version for tracking and control purposes.

5.12.3 Preparing User Manuals

The following sections outline the preparation of the user manuals for the 3DStress application.

5.12.3.1 Software Users Manual

CNWRA will prepare a Software User Manual (SUM) which describes the installation and operation of the 3DStress application. The SUM will describe all user input and activities required to control and review the computational results generated by 3DStress.

5.12.3.2 Software Input/Output Manual

A separate Software Input/Output manual will not be prepared for this application. This activity has been tailored out.

5.12.3.3 Software Centers Operators Manual

A separate Software Centers Operators Manual will not be prepared for this application. This activity has been tailored out.

5.12.3.4 Computer Operation Manuals

A separate Computer Operation Manuals will not be prepared for this application. This activity has been tailored out.

5.12.4 Installation at User Sites

CNWRA will support NRC on-site installation of 3DStress as needed. Support for other 3DStress users will be arranged on a case by case basis.

5.13 Preparing for Software Transition

CNWRA will retain rights to the 3DStress application executable and support files. No software transitions to another organization are planned at this time. This activity has been tailored out.

5.14 Software Configuration Management

Software configuration management (CM) is the process by which baselined documents and source code are identified and changes are identified and recorded. All deliverable documents and source code will be placed under CM.

3DStress CM will be the responsibility of the Project Manager. The Project Manager will determine when source and documents are to be submitted to CM and will control the release, modification and resubmission of these materials to CM. The following sections define the CM process that will be followed by the 3DStress Project Manager.

5.14.1 Configuration Identification

Two software products will be placed under CM for the 3DStress project: software source code and application executables produced during the project. Each release of 3DStress will have a unique version number.

5.14.2 Configuration Control

On initial and subsequent release to CM of software products, the Project Manager will follow the procedure listed below:

1. The Project Manager will prepare a Software Release Notice (SRN) form, refer to CNWRA-TOP-18 for the appropriate format.

2. If this is the final delivery to the acquirer, the Project Manager will make sufficient copies of the deliverable material as required by the acquirer.
3. The Project Manager will provide the CNWRA QA group a copy of all products to be placed in CM

5.14.3 Configuration Status Accounting

The 3DStress Project Manager will prepare and maintain records of the configuration status of all software documentation and the 3DStress CSCI that have been placed under configuration control. These records will be maintained for the life of the 3DStress application. The records will contain the current version/revision/release of each entity, changes to the entity since being placed under CM, and the status of open SPRs affecting the entity.

5.14.4 Configuration Audits

The 3DStress Project Manager will make configuration management records available on a non-update basis for audit by the CNWRA or acquirer representatives.

5.14.5 Packaging, Storage, Handling, and Delivery

The software and documentation will be stored in paper and electronic form. Documentation will be stored on 3.5" floppy disks or CDs in WordPerfect for Windows 8 or later format. End item software will be delivered on 3.5" floppy disks, CD-ROM or 8mm tape in plain ASCII text file format.

5.15 Software Product Evaluation

The 3DStress application will be demonstrated for potential clients but no plans exist to distribute evaluation copies of the software. This activity has been tailored out.

5.16 Software Quality Assurance

CNWRA will follow a two-fold approach to building a quality product for the 3DStress application:

> Quality Development - define and follow good software development practices throughout the development effort. For the 3DStress project, CNWRA will use internal development staff for planning, coding and testing who will adhere to the plans and implementation procedures outline in this SDP.
>
> Quality Assurance - ongoing verification that the process are being followed by the development team. CNWRA will utilize the CNWRA QA department for review, evaluation and recommendations.

CNWRA QA (CQA) will monitor the software development process to verify the procedures and practices identified in this plan are being utilized in the 3DStress development. Evaluations will be informal and deviations from this development plan will be brought to the attention of the Project Manager. Continued deviation from the development plan will require notification of CNWRA management to discuss

corrective actions or initiate an update of the software development plan to reflect changes in the project scope.

5.17 Corrective Action

The corrective action process is uniform for any software unit requiring correction. The formal corrective action process becomes effective once the 3DStress control subsystem CSCI enters the CM system. All corrective actions (CA) are initiated with a Software Problem Report (SPR). The SPR form to be used for this project is described in Section 5.17.1. Document or software comments from the contracting agency or IQA are not required to be submitted on the SPR form, other formats are acceptable. Proposed enhancements to the system may also be initiated through the use of the SPR.

5.17.1 Problem/Change Reports

An example SPR form is shown in Figure 5-1. To accommodate lengthy explanations or supporting material, attachments to the forms may be referenced in the appropriate fields. All SPRs are maintained in the project file by the Project Manager for the duration of the project and will be made available to acquirer representatives upon request.

5.17.2 Corrective Action System

The corrective action system centers around the submission of the SPR. SPR processing will generally adhere to the following sequence:

1. Problem identification and report submission. An SPR can be generated by any project member or software user who detects a problem or recognizes a required enhancement to a baselined document or software program.

Software Problem/Change Report

Project:	Originator:	Date:	Number:
Problem/Change Name:		Priority	
Affected Software Element:			
Description:			
Analysis:			
Modifications by:		Date:	Version:
Implementation:			

Figure 5-1. Sample software problem report

2. Logging. Following receipt of an SPR (or equivalent), the requested CA is entered into a CA log sheet. This log sheet facilitates tracking and reporting of all CA's issued during the life of the project. The Project Manager will then assign the CA to a software engineer for analysis.
3. Analysis. Analysis will be performed by the assigned engineer to determine the category of the CA: software, documentation, design, user, of requirement problem. The analysis also needs to determine what priority level should be assigned to the problem. Analysis of problems that lead to modification of software need further documentation, including test cases in order to assure that the problem has indeed been resolved.
4. Approval. After analysis the Project Manager will decide if a software or document change is necessary. The Project Manager is also responsible for final determination of the category, priority, and type of action required.
5. Implementation. During implementation, the affected products are "checked out" from the appropriate library and corrections made. Appropriate unit tests and integration tests must be determined and performed.
6. Release. Once the corrections have been made, they must be verified/tested at the appropriate software development level and/or CSCI testing depending on the level and type of change. The corrected products are reinserted into the baseline, and the products returned to configuration control. Following this they are ready for release to the acquirer.

5.18 Progress Reporting

During the 3DStress development, brief monthly project reports in the form of the Program Manager's Periodic Report will be produced by CNWRA.

5.19 Other Software Development Activities

The following sections describe the approach to be followed for other software development activities for the 3DStress application development project.

5.19.1 Risk Management, Including Known Risks and Corresponding Strategies

Areas of technical risk will be investigated as early in the development cycle as possible to allow adjustments in software design if required.

Schedule and cost problems are normally identified by use of CNWRA Project Manager data sheets. This control is currently being used in all projects. In addition, Project Managers hold timely project review meetings with all key project personnel to discuss, review, and solve schedule, cost, and technical problems.

The initial step in risk mitigation is identification of the risk, its potential impact on the project performance, and likelihood of developing into a problem. Risks are identified by careful review of all project aspects by analysis of the WBS. Risks are then tracked through the project until task completion to monitor their impact on cost, schedule and technical performance.

During each reporting period, work projections are made for the next reporting period, and costs are estimated. Progress for both performance and cost is evaluated against these projections. When progress does

not match projections, discussions are initiated within the project staff, and then with division management to resolve the problems, i.e., mitigate risks.

5.19.2 Software Management Indicators

The Project Manager will monitor the software management indicators listed below on an ongoing basis against the proposed project schedule and milestones.

1. Requirements volatility: total number of requirements and requirements changes over time.
2. Software staffing: planned and actual staffing levels over time.
3. Software complexity: complexity of each software unit.
4. Software progress: planned and actual number of software units designed implemented, unit tested, and integrated over time.
5. Milestone performance: planned and actual dates of key project milestones.

5.19.3 Security and Privacy

The 3DStress software application does not contain any extraordinary security or privacy issues (Section 4.2.4.2 & 4.2.4.3). This activity is tailored out.

5.19.4 Subcontractor Management

CNWRA does not plan to utilize subcontractors on the 3DStress development project. This activity has been tailored out.

5.19.5 Interface with Software Independent Verification and Validation Agents

CNWRA will utilize in-house staff for review of software quality issues and internal staff for verification and validation. This activity has been tailored out.

5.19.6 Coordination with Associate Developers

The 3DStress application will be developed using only internal staff. No other associated developers will be used. This activity has been tailored out.

5.19.7 Improvement of Project Processes

The Project Manager will periodically assess the processes used on the project to determine the suitability and effectiveness. Based on these assessments, the Project Manager will identify any necessary and beneficial improvements to the process, and identify these changes to the acquirer in the form of proposed updates to this Software Development Plan. All proposed changes will have acquirer approval prior to implementation.

5.19.8 Other Activities Not Covered Elsewhere in the Plan

CNWRA plans to utilize the consulting services of Dr. Alan Morris, University of Texas at San Antonio during the development of the 3DStress application. Dr. Morris is one of the original developers of the 3DStress algorithms and will be utilized as a resource for software requirements development and software validation.

6 SCHEDULES AND ACTIVITY NETWORK

Table 6-1 presents an overview of the significant milestones that will be completed during this project.

Table 6-1. Schedule of software development activities

Activity	Planned/Actual Completion Date
Software release v.1.1	August 2, 1996
Software release v.1.2	November 12, 1996
Software Requirement Document (v.1.3)	August 5, 1997
Software Planning Document (v.1.3)	July 13, 1998
Acceptance Testing	July 15,1998
Verification Testing (v.1.3)	July 15, 1998
Software Test Report	July 17,1998
User Guide to NRC (v.1.3)	June 29,1998
Software release (v.1.3)	August 12,1998
3DStress v.1.3 to NRC	August 14, 1998

7 PROJECT ORGANIZATION AND RESOURCES

The following sections describe the project organization and resources to applied to the 3DStress application development.

7.1 Project Organization

On a functional basis, CNWRA conducts programs under the Project Manager concept. The Project Manager is delegated authority for overall technical direction and administrative supervision of the project. The Project Manager reports directly to the Element Manager, who in turn reports directly to the Technical Director. This structure permits ready access to higher management to quickly resolve any problems which might arise. The quick access to management allows for close schedule coordination on projects of an interdivisional nature. Thus, once a project team is formed, the Project Manager has vertical line authority over team members for the duration of the project.

The support staff of CNWRA, including such functions as accounting, contract administration, purchasing, computer processing, report reproduction, library, and security, are at the disposal of the Project Manager. The direct availability of the support staff leads to effective project management and eliminates delays which might be experienced in a less flexible system.

CNWRA Quality Assurance reports directly to the CNWRA President. CNWRA QA performs audits of the software development process. CNWRA QA ensures conformance to contractual requirements and determines the adequacy and effectiveness of project activities.

Management controls are imposed by CNWRA to ensure progress and eventual delivery of end items in accordance with the agreed upon schedule and cost. Scheduling control is maintained by short interval updating of the approved schedule. As a minimum, bar chart project schedules with clearly defined milestones are prepared. The charts are divided into appropriate phases, tasks and, if needed, subtasks. These charts and work breakdown structures (WBS) are entered into a computer to facilitate monitoring and updating.

Cost status reports are prepared and distributed to project managers every two weeks at the close of the normal pay period. Labor data for these reports are obtained from individual time sheets which all employees are required to complete daily. Charges are listed by project number, as well as phase or task numbers. Itemized labor, materials, travel, reproduction services, and overhead for the preceding two-week accounting period are given, including commitments made which have not yet resulted in expenditures. Also, the balance of project funds available is noted. Every four weeks, a computerized summary of the two preceding biweekly reports is prepared and given to individual project managers.

The Project Manager has full responsibility for all software products created and/or utilized by the project. The data items, documentation reports, drawings, and manuals constitute project team activity paralleling the hardware and software development activities. The same team members performing the hardware and software tasks will also provide direct input and analysis for all data supplied on this contract.

7.2 Project Resources

The following sections describe the resources that CNWRA will apply to the 3DStress application development project.

7.2.1 Personnel

The software development team will be composed of software analysts experienced in the development of Silicon Graphics Open GL software. Approximately 0.2 FTE's and one summer employee will be committed to the software development team. The Project Manager is responsible for coordinating the activities of the software team project evaluations with the CQA department. Project management tasks will require approximately 0.2 FTE's.

7.2.2 Facilities

CNWRA will establish a development and testing environment for this project in the CNWRA GIS Laboratory, Bldg. 189. The environment will consist of one Silicon Graphics development workstation and three additional Silicon Graphics systems for testing and evaluation.

7.2.3 Acquirer Furnished Equipment, Data, and Documentation

No acquirer furnished software or equipment is required for this project

8 NOTES

8.1 Acronyms

CM	Configuration Management
CNWRA	Center for Nuclear Waste Regulatory Analyses
CSCI	Computer Software Configuration Item
HWCI	Hardware Configuration Item
IAW	In accordance with
CQA	CNWRA Quality Assurance
SDD	Software Design Description
SDF	Software Development File
SPR	Software Problem Report
SQA	Software Quality Assurance
SRD	Software Requirements Description
STP	Software Test Plan
STPr	Software Test Procedures
SUM	Software User's Manual
SVD	Software Version Description
SwRI	Southwest Research Institute
WBS	Work breakdown structure

8.2 Definitions

Acquirer

An organization that procures software products for itself or another organization.

Approval

Written notification by an authorized representative of the acquirer that a developer's plans, design, or other aspects of the project appear to be sound and can be used as the basis for further work. Such approval does not shift responsibility from the developer to meet contractual requirements.

Architecture

The organizational structure of a system or CSCI, identifying the components, their interfaces, and a concept of execution among them.

Associate Developer

An organization that is neither prime contractor nor subcontractor to the developer, but who has a development role on the same or related system or project.

Behavioral Design

The design of how an overall system or CSCI will behave, from a user's point of view, in meeting its requirements, ignoring the internal implementation of the system or CSCI. This design contrasts with architectural design, which identifies the internal components of the system or CSCI, and with the detailed design of those components.

Build

(1) A version of software that meets a specified subset of the requirements that the completed software will meet. (2) The period of time during which such a version is developed. Note: The relationship of the terms "build" and "version" is up to the developer; for example, it may take several versions to reach a build, a build may be released in several parallel versions (such as to different sites), or the terms may be used as synonyms.

Computer Hardware

Devices capable of accepting and storing computer data, executing a systematic sequence of operations on computer data, or producing control outputs. Such devices can perform substantial interpretation, computation, communication, control, or other logical functions.

Computer program

A combination of computer instructions and data definitions that enable computer hardware to perform computational or control functions.

Computer Software Configuration Item (CSCI)

An aggregation of software that satisfies an end use function and is designated for separate configuration management by the acquirer. CSCIs are selected based on tradeoffs among software function, size, host or target computers, developer, support concept, plans for reuse, criticality, interface considerations, need to be separately controlled, and other functions.

Configuration Item

An aggregation of hardware, software, or both that satisfies an end use function and is designated for separate configuration management by the acquirer.

Database

A collection of related data stored in one or more computerized files in a manner that can be accessed by users or computer programs.

Deliverable software product

A software product that is required by the contract to be delivered to the acquirer or other designated recipient.

Design

Those characteristics if a system or CSCI that are selected by the developer in response to the requirements. Some will match the requirement; others will be elaborations of requirements, such as definitions of all error messages in response to a requirement to display error messages; other will be implementation related, such as decisions about what software units and logic to use to satisfy the requirements.

Developer

An organization that develops software products ("develops" may include new development, modification, reuse, reengineering, maintenance, or any other activity that results in software products). The developer may be a contractor or a Government agency.

Document/documentation

A collection of data, regardless of the medium on which it is recorded, that generally has permanence and can be read by humans or machines.

Evaluation

The process of determining whether an item or activity meets specified criteria.

Firmware

The combination of a hardware device and computer instructions and/or computer data that reside as read-only software on the hardware device.

Hardware Configuration Item (HWCI)

An aggregation of hardware that satisfies an end use function and is designated for separate configuration management by the acquirer.

Interface

In software development, a relationship among two or more entities (such as CSCI-CSCI, CSCI-HWCI, CSCI-user, or software unit-software unit) in which the entities share, provide, or exchange data. An interface is not a CSCI, software unit, or other system component; it is a relationship among them.

Joint review

A process or meeting involving representatives of both the acquirer and the developer, during which project status, software products, and/or project issues are examined and discussed.

Non-deliverable software product

A software product that is not required by the contract to be delivered to the acquirer or other designated recipient.

Process

An organized set of activities performed for a given purpose; for example, the software development process.

Qualification testing

Testing performed to demonstrate to the acquirer that a CSCI or a system meets the specified requirements.

Reengineering

The process of examining and altering an existing system to reconstitute it in a new form. May include reverse engineering (analyzing a system and producing a representation at a higher level of abstraction, such as design from code), restructuring (transforming a system from one representation to another at the same level of abstraction), redocumentation (analyzing a system and producing user or support documentation), forward engineering (using software products derived from an existing system, together with new requirements, to produce a new system), retargeting (transforming a system to install it on a different target system), and translation (transforming source code from one language to another or from one version of a language to another).

Requirement

(1) A characteristic that a system or CSCI must possess in order to be acceptable to the acquirer. (2) A mandatory statement in this standard or another portion of the contract.

Reusable software product

A software product developed for one use but having other uses, or one developed specifically to be usable on multiple projects or in multiple roles on one project. Examples include, but are not limited to, commercial off-the-shelf software products, acquirer furnished software products, software products in reuse libraries, and pre-existing developer software products. Each use may include all or part of the software product and may involve its modification. This term can be applied to software product (for example, requirements, architectures, etc.), not just to software itself.

Software

Computer programs and database. Note: Although some definitions of software include documentation, MIL-STD-498 limits the definition to computer programs and databases in accordance with Defense Federal Acquisition Regulation Supplement 227.401.

Software development

A set of activities that results in software products. Software development may include new development, modification, reuse, reengineering, maintenance, or any other activities that result in software products.

Software development file

A repository for material pertinent to the development of a particular body of software. Contents typically include (either directly or by reference) considerations, rationale, and constraints related to requirements analysis, design, and implementation; developer-internal test information; and schedule and status information.

Software development library (SDL)

A controlled collection of software, documentation, other intermediate and final software products, and associated tools and procedures used to facilitate the orderly development and subsequent support of software.

Software development process

An organized set of activities performed to translate user needs into software products.

Software engineering

In general usage, a synonym for software development. As used in this standard, a subset of software development consisting of all activities except qualification testing. The standard makes this distinction for the sole purpose of giving separate names to the software engineering and software test environments.

Software engineering environment

The facilities, hardware, software, firmware, procedures, and documentation needed to perform software engineering. Elements may include but are not limited to computer-aided software engineering (CASE) tools, compilers, assemblers, linkers, loaders, operating systems, debuggers, simulators, emulators, documentation tools, and database management systems.

Software product

Software or associated information created, modified, or incorporated to satisfy a contract. Examples include plans, requirements, design, code, databases, test information, and manuals.

Software quality

The ability of software to satisfy its specified requirements.

Software support

The set of activities that takes place to ensure that software installed for operational use continues to perform as intended and fulfill its intended role in system operation. Software support includes software maintenance, aid to users, and related activities.

Software system

A system consisting solely of software and possibly the computer equipment on which the software operates.

Software test environment

The facilities, hardware, software, firmware, procedures, and documentation needed to perform qualification, and possibly other testing of software. Elements may include but are not limited to simulators, code analyzers, test case generators, and path analyzers, and may also include elements used in the software engineering environment.

Software transition

The set of activities that enables responsibility for software development to pass from one organization, usually the organization that performs initial software development, to another, usually the organization that will perform software support.

Software component/unit

An element in the design of a CSCI; for example, a major subdivision of a CSCI, a component of that subdivision, a class, object, module, function, routine, or database. Software components may occur at different levels of a hierarchy and may consist of other software units. Software units in the design may or may not have a one-to-one relationship with the code and data entities (routines, procedures, databases, data files, etc.) that implement them or with the computer files containing those entities.

SOFTWARE REQUIREMENTS DESCRIPTION

76/236

SOFTWARE REQUIREMENTS DESCRIPTION 3DSTRESS VERSION 1.3

by

Robert T. Boneau

Center for Nuclear Waste Regulatory Analyses
Southwest Research Institute
San Antonio, Texas

August 5, 1997

Approved by:

H. Lawrence McKague 8/18/97

H. Lawrence McKague, Element Manager
Geology and Geophysics

77/236

CONTENTS

78/236

1 INTRODUCTION

3DStress is a software application that computes the tendency for faults and fractures to slip or dilate. Slip tendency is the ratio of the shear stress to the normal stress on a fault surface. Dilation tendency is the likelihood for a fault or extension fracture to dilate based on the three-dimensional (3-D) stress conditions and is computed from the normal stress and the principal stresses. The input 3-D stress orientations and magnitudes may be interactively modified through a user interface. Faults and fractures displayed by 3DStress are colored based on the computed slip or dilation tendency. In addition to slip and dilation tendency, 3DStress computes the expected slip direction by finding the maximum shear stress for the fault surface.

Paragraphs that begin with the label "Version 1.3" summarize features that will be included in 3DStress version 1.3. These features were not included in the previous release.

79/236

2 SOFTWARE FUNCTIONS

3DStress performs three primary tasks. First, 3DStress provides a user interface for interactive control of the input stress orientations and magnitudes. Second, 3DStress computes slip tendency, dilation tendency, and slip direction from the input stress parameters and fault surface orientation. Third, 3DStress displays 2-D and 3-D representations of faults and fracture surfaces colored by slip or dilation tendency.

Version 1.3 - The following features will be added to 3DStress version 1.3.

- Provide Mohr circle and failure envelope.
- Build 2-D and 3-D fault coverages in the map viewer and 3D viewer windows.
- Display map of stress azimuths.
- Save and load current stress conditions to a file.
- Include a leakage factor calculation mode.

3 TECHNICAL BASIS AND MATHEMATICAL MODEL

3DStress is founded on the principals of fault kinematics. These principals state that the input principal stresses can be resolved into a normal stress and shear stress acting on a fault surface. The normal stress is perpendicular to the fault surface, while the shear stress lies in the plane of the fault surface. The greater the ratio of the shear stress to the normal stress, the greater the slip tendency. Friction characteristics and rock material properties are not modeled by 3DStress.

The input principal stresses are labeled as follows:

σ_1 = *maximum principal stress*
σ_2 = *intermediate principal stress*
σ_3 = *minimum principal stress*

Where: $\sigma 1 > \sigma 2 > \sigma 3$

The equation for computing slip tendency is given below.

$$T_s = \textit{slip tendency} = \frac{\tau_s}{\sigma_n}$$

Where: τ_s = *shear stress*
σ_n = *normal stress*

The equation for computing dilation tendency is given below.

$$T_d = \textit{dilation tendency} = \frac{(\sigma_1 - \sigma_n)}{(\sigma_1 - \sigma_3)}$$

Where: σ_n = *normal stress*
σ_1 = *maximum principal stress*
σ_3 = *minimum principal stress*

Versions 1.3 - The equation for computing leakage factor is given below.

$$\textit{leakage factor} = \frac{P_f}{(\sigma_n + T_0)}$$

Where: σ_n = *normal stress*
P_f = *fluid pressure*
T_0 = *tensile strength*

81/236

4 DATA FLOW AND USER INTERFACES

In order to compute slip or dilation tendency, two sets of input data are required. First, the input principal stress orientations and magnitudes are needed. Second, the fault surface orientation is required. From these inputs, the stresses normal and shear to the fault surface are computed. Finally, slip or dilation tendency is computed from the principal, normal, and shear stresses.

The user interface enables the user to input the principal stress orientations and magnitudes and to select a particular fault surface orientation. In addition, the user may select a 2-D or 3-D fault coverage that is displayed and colored by slip or dilation tendency.

82/236

5 PROGRAMMING LANGUAGE

3DStress is written in the C++ programming language using an object oriented design. The program utilizes the OpenGL graphics and Motif libraries supplied on Silicon Graphics workstations. The OpenGL libraries provide 2-D and 3-D graphics rendering capabilities. The Motif libraries are used to create the graphical user interface to the program.

83/236

6 HARDWARE PLATFORMS

3DStress executes on Silicon Graphics workstations. The program is compatible with the IRIX 5.3 operating system.

Version 1.3 - 3DStress will be ported to the Sun Ultra platform.

84/236

7 GRAPHICS OUTPUT DEVICES

Screen displays of 3DStress may be saved and printed using utilities such as scrsave, snapshot, imgworks, and showcase. These utilities are provided by Silicon Graphics on their workstations. The user may store the graphics window displays to raster files through a user interface menu.

85/234

8 SUMMARY

3DStress in an interactive tool for computing and displaying the slip and dilation tendency for faults and fractures. The input stress orientations and magnitudes are controlled by a user interface. The slip or dilation tendency and expected slip direction are computed for the fault surface orientation using the input stress conditions. The 2-D and 3-D fault representations displayed by 3DStress are colored by slip or dilation tendency. The program executes on Silicon Graphics workstations.

90/236

SOFTWARE REQUIREMENTS DESCRIPTION (SRD)
3DSTRESS VERSION 1.3

APRIL 16, 1997

1 INTRODUCTION

3DStress is a software application that computes the tendency for faults and fractures to slip or dilate. Slip tendency is the ratio of the shear stress divided by the normal stress on a fault surface. Dilation tendency is the likelihood for a fault or extension fracture to dilate based on the three-dimensional (3-D) stress condititons and is computed from the normal stress and the principal stresses. The input 3-D stress orientations and magnitudes may be interactively modified through a user interface. Faults and fractures displayed by 3DStress are colored based on the computed slip or dilation tendency. In addition, to slip and dilation tendency, 3DStress computed the expected slip direction by finding the maximum shear stress for the fault surface.

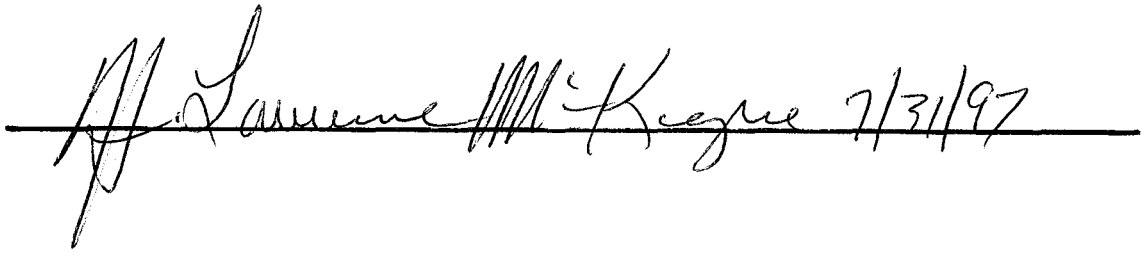
Paragraphs that begin with the label "Version 1.3" summarize features that will be included in 3DStress version 1.3. These features were not included in the previous release.

2 SOFTWARE FUNCTIONS

3DStress performs three primary tasks. First, 3DStress provides a user interface for interactive control of the input stress orientations and magnitudes. Second, 3DStress computes slip tendency, dilation tendency, and slip direction from the input stress parameters and fault surface orientation. And third, 3DStress displays 2-D and 3-D representations of faults and fracture surfaces colored by slip or dilation tendency.

Version 1.3 - The following features wil be added to 3DStress version 1.3.

* Mohr circle and failure envelope.
* Build 2-D and 3-D fault coverages in the map viewer and 3D viewer windows.
* Display map of stress azmimuths.
* Save and load current stress conditions to a file.
* Leakage Factor calculation mode.



3 TECHNICAL BASIS AND MATHEMATICAL MODEL

3DStress is founded on the principals of fault kinematics. These principals state that the input principal stresses can be resolved into a normal stress and shear stress acting on a fault surface. The normal stress is perpendicular to the fault surface while the shear stress lies in the plane of the fault surface. The greater the ratio of the shear stress to the normal stress, the greater the slip tendency. Friction characteristics and rock material properties are not modeled by 3DStress.

The input principal stress are labeled as follows:

σ_1 = *maximum principal stress*
σ_2 = *intermediate principal stress*
σ_3 = *minimum principal stress*

Where: $\sigma_1 > \sigma_2 > \sigma_3$

The equation for computing slip tendency is given below.

$$T_S = \textit{slip tendency} = \frac{\tau_s}{\sigma_n}$$

Where: τ_s = *shear stress*
σ_n = *normal stress*

The equation for computing dilation tendency is given below.

$$T_d = \textit{dilation tendency} = \frac{(\sigma_1 - \sigma_n)}{(\sigma_1 - \sigma_3)}$$

Where: σ_n = *normal stress*
σ_1 = *maximum principal stress*
σ_3 = *minimum principal stress*

Versions 1.3 - The equation for computing leakage factor is given below.

$$\textit{leakage factor} = \frac{P_f}{(\sigma_n + T_0)}$$

Where: σ_n = *normal stress*
P_f = *fluid pressure*
T_0 = *tensile strength*

92/236

4 DATA FLOW AND USER INTERFACES

In order to compute slip or dilation tendency, two sets of input data are required. First, the input principal stresses orientations and magnitudes are needed. Second, the fault surface orientation is required. From these inputs, the stresses normal and shear to the fault surface are computed. Finally, slip or dilation tendency is computed from the principal, normal, and shear stresses.

The user interface enables the user to input the principal stresses orientations and magnitudes and to select a particular fault surace orientation. In addition, the user may select a 2-D or 3-D fault coverage that is displayed and colored by slip or dilation tendency.

5 PROGRAMMING LANGUAGE

3DStress is written in the C++ programming language using an object oriented design. The program utilizes the OpenGL graphics and Motif libraries supplied on Silicon Graphics workstations. The OpenGL libraries provide 2-D and 3-D graphics rendering capabilities. The Motif libraries are used to create the graphical user interface to the program.

6 HARDWARE PLATFORMS

3DStress executes on Silicon Graphics workstations. The program is compatible with the IRIX 5.3 operating system.

Version 1.3 - 3DStress ported to the Sun Ultra platform.

7 GRAPHICS OUTPUT DEVICES

Screen displays of 3DStress may be saved and printed using utilites such as scrsave, snapshot, imgworks, and showcase. These utilities are provided by Silicon Graphics on their workstations. The user may store the graphics window displays to raster files through a user interface menu.

8 SUMMARY

3DStress in an interactive tool for computing and displaying the slip and dilation tendency for faults and fractures. The input stress orientations and magnitudes are controled by a user interface. The slip or dilation tendency and expected slip direction are computed for the fault surface orientation using the input stress conditions. The 2-D and 3-D fault representations displayed by 3DStress are colored by slip or dilation tendency. The program executes on Silicon Graphics workstations.

Software Validation Test Report

96|236

SOFTWARE TEST REPORT FOR 3DSTRESS

Prepared for

Nuclear Regulatory Commission
Contract NRC-02-97-009

Prepared by

Josh Buckner
Joseph H. Bangs

Center for Nuclear Waste Regulatory Analyses
San Antonio, Texas

Approved by: H. Lawrence McKague **Date:** 8/7/98
H. Lawrence McKague

9 BSm 9/20/2000

July 1988

97/236

CONTENTS

TABLES

1 SCOPE

This report documents the results of software validation testing performed on Version 1.3 of the 3DStress software application. This testing took place at the Center for Nuclear Waste Regulatory Analyses (CNWRA) located at Southwest Research Institute (SwRI) on July 15–16, 1998. The report includes an assessment of the software testing and lists the defects identified by the test procedures.

1.1 IDENTIFICATION

This Software Test Report (STR) applies to the software modifications applied to version 1.2 of 3DStress, resulting in version 1.3. Software capabilities and functions implemented in version 1.2 and older were not retested during this qualification test.

1.2 SYSTEM OVERVIEW

3DStress executes on Silicon Graphics workstations (SGI) operating under version 5.3 or later of the IRIX operating system. 3DStress does not communicate with other computer systems or software applications. 3DStress operates entirely under local user control and does not include facilities for automatic process scheduling. The user analyzes results of software processing via X windows displays containing graphical and text-based data displays. Hard copy outputs from the application may be generated from screen dumps of the display windows. Tabular data files containing 2-D and 3-D fault geometry information are read by 3DStress to generate base map and 3-D representations of the faults. 3DStress does not output tabular data files for analysis by other software applications.

2 REFERENCED DOCUMENTS

The following documents are referenced in this test report and provide additional information regarding the implementation of 3DStress version 1.3.

CNWRA-TOP-18	5 MAY 98	Development and Control of Scientific and Engineering Software
3DStress-SRD	17 APR 97	3DStress Software Requirements Description
3DStress-SVTP	15 JULY 98	3DStress Software Validation Test Plan

3 TESTING

The software validation test was performed in accordance with the executed SVTP included in Appendix A. The software testers did not deviate from the test procedures except when mandated by software failures. Software Problem Reports (SPRs) generated over the course of the test are included in Appendix B. Each SPR will be tracked until the defect is corrected or the functional capability is removed from the 3DStress Software Requirements Description.

3.1 RESULTS

Version 1.3 of 3DStress generally performed as expected. However, two serious errors occurred during the test. The Software Requirements Description (SRD) specifies that 3DStress will be used to build 2-D and 3-D fault coverages in the map and 3DViewer windows, respectively. Tests on the fault building capability revealed that no provision for saving 2-D coverages has been included in the Map display. The 3DView software does provide a file saving capability, but other errors in the polygon builder prevent the user from entering polygon coordinates into the 3DStress application. The requirement to build 2-D and 3-D fault coverages in 3DStress is not yet completed.

The second error, unplanned terminations (crashes) of the 3DStress application, occurred three times during the software testing. The software crashes were not repeatable and are thus likely to be related to incorrect memory management operations. One crash required the system super-user to remotely log into the test machine and reset the console display software.

Other, less significant, errors occurred during the test and are also listed in the Software Problem Report table below. The significance of the SPR is indicated by the priority level. Priority 1 SPR's correspond to software defects that prevent the user from completing a specified task while Priority 5 SPR's represent minor user inconveniences. Time estimates are included in the table for the developer to diagnose, and correct each software defect.

Table 3-1. Software trouble report log for version 1.3 of the 3DStress application

Date Opened	SVTP Section	SPR #	Priority	Problem Summary	Repair Time (hours)	Date Closed
7/15/98	4.1.1	1	5	Incomplete error trapping of user parameters in the Mohr's Circle Options window. Alpha characters entered in numeric field were not trapped.	3	
7/15/98	4.1.3	2	4	Printing a window in the Mohr Circle, Map, 3DView and other windows was not successful if the display window was covered by another window or if the window was not completely visible on the monitor. Also the image captured by the software excluded the top portion of the desired display window.	40	

Date Opened	SVTP Section	SPR #	Priority	Problem Summary	Repair Time (hours)	Date Closed
7/15/98	4.2.2	3	4	When adding a new line segment point in the Map display, the edit point in the window does not accurately track the movements of the mouse.	16	
7/15/98	4.2.2	4	1	After entering several new line segment points in the Map window display, the user attempted to view the new data with the Browse function. 3DStress crashed when the Browse function was invoked. After restarting 3DStress, the problem could not be duplicated with the same Edit/Browse processing sequence.	40	
7/15/98	4.2.2	5	1	There is no capability to store fault control points entered in the Map display. User edits cannot be saved to disk.	16	
7/16/98	4.2.6	6	3	2-D faults added to a map display are not incorporated in the calculation of the Rose diagram.	16	
7/16/98	4.2.7	7	1	Clicking the Map Help caused 3DStress to crash and "locked-up" the console monitor, preventing user inputs. Correction of this problem required the system root user to remotely login into the test machine and restart the Xwindows window manager application. The problem was not duplicated in several attempts after the software was restarted.	40	
7/16/98	4.2.8	8	5	The Rose plot diagram did not close when the Map display was closed.	8	

Date Opened	SVTP Section	SPR #	Priority	Problem Summary	Repair Time (hours)	Date Closed
7/16/98	4.2.9	9	4	When adding multiple 3-D faults to the 3-D fault viewer, the faults must be generally within the same region. Otherwise the 3DViewer display grid will not plot any data on the plot grid. This is confusing to the user but does not cause the loss of data.	40	
7/16/98	4.2.10	10	4	When adding polygons to the 3DView display, the edit point does not move proportionally to the distance traveled by the mouse pointer. This may be an artifact of the 3-D display, but it is confusing to the user.	16	
7/16/98	4.2.10	11	1	Triangles added to the 3DView are erased when the End Triangle button is clicked. By closing the Options window without clicking the End Triangle, the user can save one polygon to the fault file being edited. The user cannot effectively save fault polygon information in the 3DViewer.	40	
7/16/98	4.2.10	12	1	When attempting to save a fault polygon file, we received an Alert message (1154) from the operating system that the system swap space had run out. 3DStress then crashed. After restarting the application, we able to save the same 3-D fault file we were processing prior to the software crash.	40	

Date Opened	SVTP Section	SPR #	Priority	Problem Summary	Repair Time (hours)	Date Closed
7/16/98	4.4.2	13	4	While testing the Mohr Circle and Slider capabilities to save/load stress magnitude information on disk, we received a 3DStress error message that the previously save stress magnitude file had an erroneous entry for fluid pressure. 3DStress is not error trapping the fluid pressure value when it is saved to disk, but is error trapping the fluid pressure when loading from disk. The error trapping routines should be consistent in both directions.	8	

3.2 IMPACT OF TEST ENVIRONMENT

The validation testing was performed on a SGI Indigo CPU (Yosemite) running under the IRIX 6.2 operating system. The errors encountered during the software testing appear to be related to software defects and not the host operating system configuration. If time allows, this test could be run on an SGI platform running under IRIX 5.3, but this is a low priority since the IRIX 6.x operating systems have been available for a number of years and usage of IRIX 5.3 is not common.

4 RECOMMENDED IMPROVEMENTS

This section does not document software defects, but suggests user interface modifications for future releases of 3DStress.

3DStress contains a powerful set of stress analysis and visualization tools accessible through sets of window displays. The user may potentially open numerous windows, cluttering the monitor display area. This creates a disorganized visual effect that detracts from the overall software presentation. 3DStress would benefit from incorporating controls to prevent unrelated windows from being displayed simultaneously.

3DStress relies on external software applications (Arc/Info, EarthVision, etc.) to create fault and symbol coverages. 3DStress should support vendor specific file formats when possible to simplify the data import/export process. Direct data communication with other applications may not be desirable until a specific workflow methodology is identified by the user community.

Showcase provides an efficient means of displaying Help file information, however, future ports of 3DStress may preclude the use of this SGI specific application. It is recommended that the Help files be converted to an HTML or PDF format to enhance portability and simplify maintenance.

Wherever possible, the Options windows should be simplified to reduce the amount of information the user must enter/update. If this is not practical, "fly-by" annotations should be used to used to assist the user in understanding the significance of the parameter. Context sensitive help would be an alternative to the "fly-by" annotations.

APPENDIX A

Software Validation Test Plan

SOFTWARE VALIDATION TEST PLAN FOR 3DSTRESS

Prepared for

Nuclear Regulatory Commission
Contract NRC-02-97-009

Prepared by

Joseph H. Bangs

Center for Nuclear Waste Regulatory Analyses
San Antonio, Texas

CONTENTS

TABLES

1 SCOPE

This document establishes the Software Validation Test Plan (SVTP) for validating the installation and functionality of the 3DStress (version 1.3) software application developed by Southwest Research Institute (SwRI).

1.1 IDENTIFICATION

This SVTP applies to the all source code incorporated into the version 1.3 release of the 3DStress software application (3DStress). This version of 3DStress is an upgrade to a previous release of the application. This SVTP documents test procedures for validating new software capabilities and verification that existing capabilities were not adversely affected by the software modifications/additions.

1.2 SYSTEM OVERVIEW

3DStress executes on Silicon Graphics workstations (SGI) operating under version 5.3 or later of the IRIX operating system. 3DStress does not communicate with other computer systems or software applications. 3DStress operates entirely under local user control and does not include facilities for automatic process scheduling. The user analyzes results of software processing via X windows displays containing graphical and text-based data displays. Hard copy outputs from the application may be generated from screen dumps of the display windows. Tabular data files containing 2-D and 3-D fault geometry information are read by 3DStress to generate base map and 3-D representations of the faults. 3DStress does not output tabular data files for analysis by other software applications.

2 REFERENCED DOCUMENTS

The following documents are referenced or were used as the basis for this SVTP.

CNWRA-TOP-18	5 MAY 98	Development and Control of Scientific and Engineering Software
3DStress-SRD	17 APR 97	3DStress Software Requirements Description
3DStress-SDP	3 JULY 98	3DStress Software Development Plan

3 TEST ORGANIZATION AND EXECUTION

The test procedures to be completed during this software validation process have been designed to demonstrate the new version of 3DStress satisfies the requirements specified in the Software Requirements Description (SRD) and has not introduced defects in existing capabilities. Table 3-1 summarizes the software capabilities required in this release of 3DStress. Each software requirement is cross referenced from the SRD to the corresponding test procedure defined in this SVTP. Each test procedure in this SVTP includes a synopsis of the software function being tested, user specified parameters and/or input files, and test protocol.

Table 3-1. Software Requirements Description to Software Validation Test Plan cross reference

Requirement	SRD Section	SVTP Section
Mohr's circle and failure envelope	Section 2	4.1
Build 2-D and 3-D fault coverages in the map viewer and 3D viewer windows.	Section 2	4.2
Display map of stress azimuths	Section 2	4.3
Save and load current stress conditions to a file	Section 2	4.4
Leakage Factor calculation mode	Section 2	4.5

This SVTP is intended to be a self contained document made up of individual test procedures. Each test procedure will address a specific software function requirement. The validation process will address individual module or component testing first, followed by tests in which data or commands are passed between processes or display windows.

The executed test procedures will be attached to the test report as an appendix. Software testing should be conducted by an independent tester not directly involved with the software development tasks. The software developer should be available to assist the tester with program execution questions. The software tester shall date and initial all test procedures regardless of the test outcome.

If software defects are identified during the test process, the problem will be noted in this document and also be fully documented in a software problem report. The extent and significance of the defect will be analyzed by the testers to determine if software testing should continue or be postponed until the defect has been corrected.

Software testers are encouraged to exercise software controls beyond the test procedures described in this document. Unexpected test results encountered in these tests should be carefully documented by describing the program state in which the error occurred as well as the error condition itself.

4 TEST PROCEDURES

The following sections present the validation test procedures for the 3DStress application. Unless noted otherwise, all tests will be conducted on one SGI computer. All testing should be completed in one continuous session. Software (or system) restarts are discouraged unless specified by the test protocol.

4.1 MOHR'S CIRCLE

The Mohr's Circle display and associated control windows provides a capability for the user to evaluate the relationship between 3-dimensional stresses and rock properties. This sequence of tests will validate the accuracy of the stress calculations, interactions with other components of 3DStress, general parameter entry and support function operations.

4.1.1 Mohr's Circle Options

SRD Traceability: Section 2

Summary: Verify that parameters entered in the Options parameter window result in accurate Mohr's Circle plot.

Protocol:

1. Open the Mohr's Circle display.
2. Select the Options menu item.
3. Select the Independent Stresses radio button. Enter a set of stress magnitudes with the slider bars. Manually verify the resulting Mohr's Circle and material failure point computed by 3DStress.

Inputs: $\sigma u = -18.18$ $\sigma W = 66.67$ $\sigma V = 48.48$ RockType is Carbonate

Plot low σ_3 is σ_u mid σ_2 is σ_v H: σ_1 is σ_w Ratios were correct

Output Display Verified: Yes σ_3 to σ_1 Ratio = -0.273 σ_2 to σ_1 ratio = 0.727

Plot was displayed correctly

4. Select the Dependent Stresses radio button. Enter a set of stress magnitudes with the slider bars. Manually verify the resulting Mohr's Circle and material failure point computed by 3DStress.

Inputs: $\sigma_u = 7.09$ $\sigma_v = 31.45$ $\sigma_w = 60.59$ σ_3 to σ_1 Ratio set at 0.117 σ_2 to σ_1 ratio set to 0.519

Plot was displayed correctly

Output Display Verified: Yes Ratios were correct RockType is carbonate

5. Modify the Rock Type selection in the Mohr Option window. Manually verify the program output and material failure point on the Mohr's Circle display.

Inputs: Rock Type is Arenaceous Independent Stresses $\sigma_u = -3.03$ $\sigma_w = 60.59$ $\sigma_v = 31.45$ → failure of material was reported by 3DStress

Output Display Verified:

Equations were verified using "The Hoek-Brown Failure Criteron - a 1988 update" 15th Canadian Rock mechanics Symposium Univ. of Toronto Equation 7

manual calculations

$h = 1.081517$ eq. 7

$\theta = 30.3651°$ eq. 6

$\phi_i' = 1.3156$ eq. 5

$m = 15$ $s = 1.$ $\sigma_c = 73.8$

JB: 7/15/98 The manual calculation was very tedious & time consuming. ~~The material failure calculations will be used to verify the~~ I will compare the program against the published results in reference paper 3

6. Repeat steps 3, 4, & 5 using several variations of input parameters. Manually verify the program output and material failure point on the Mohr's Circle display.

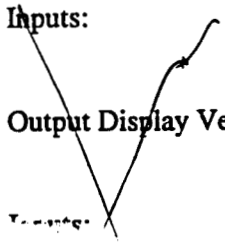
Inputs: $S = 0.1$
$\mu = 3.5$
$\sigma_c = 100$ MPA

Test Values taken from

Output Display Verified: $\sigma' = 0$ (OK)
$\tau = 5^+$

Inputs:

Output Display Verified: $\sigma' = 5$ (OK)
$\tau = \sim 11^+$



$\sigma' = 100$ (OK)
$\tau = \sim 66$

Output Display Verified:

Inputs:

Output Display Verified:

7. Open the Tendency Plot window. Enter a new set of stresses in the Mohr Option window. Click the Apply button and verify the stresses are accurately displayed in the Tendency Plot window.

Inputs: $\sigma_u = 24.24$ $\sigma_w = 18.187$ Mohr Options
$\sigma_v = 81.82$
Independent Stresses

Dependent Stresses
Mohr $\sigma_u = 57.58$
$\sigma_v = 71.08$
$\sigma_w = 32.91$

Output Display Verified:

Tendency Plot Display
$\sigma_u = 24$
$\sigma_v = 81$
$\sigma_w = 18$
OK

Tendency Plot
$\sigma_u = 57$
$\sigma_v = 71$
$\sigma_w = 32$
OK

JB 7/15/98

8. Enter erroneous values in the parameter boxes. Verify correct error trapping by the application.

Inputs: Sigma Range
Max Value was left Blank - Trapped correctly
negative fluid pressure - Trapped ok
Max scale < min scale - Trapped OK
Alpha character — Did not trap correctly

SPR #1

Errors Trapped Correctly:
Most errors trapped OK. Alpha characters entered in numeric fields were not trapped

Tested by: JB (Initials) Date: 7/15/98
JB 7/16/98 Parts 5-6

4.1.2 Mohr's Circle Reset

SRD Traceability: Section 2

Summary: Verify that 3DStress will reset the Mohr's circle display to the default display when this option is executed.

Protocol:

1. Open the Mohr's Circle display.
2. Corrupt the Mohr's Circle display.
3. Click the Reset button and verify the plot display is reset to the default position.

Inputs used to corrupt the plot and parameters:
plot was moved off display, could not recover plot easily using manual & mouse controls

Reset verified: Yes

Tested by: JB (Initials) Date: 7/15/98

4.1.3 Mohr's Circle Print

SRD Traceability: Section 2

Summary: Verify that 3DStress will create an image of the Mohr's Circle display suitable for printing.

Protocol:

1. Open the Mohr's Circle display.
2. Select the Print menu item.
3. Specify an output filename for the image file. /pscr0/mohr.capture.rgb
4. Examine the image file created by 3DStress.

SPR #7

Was the image file created correctly? 1. No, only a portion of the screen was captured, and not the mohr's circle plot ~~Mohr's circle~~ was covered by another window

2. Repeated test with window exposed on monitor
/pscr0/mohr.capture2.rgb

3. Repeated test w window size changed and plot shifted in window
/pscr0/mohr.capture3.rgb

Tested by: TB (Initials) Date: 2/15/99

4.1.4 Mohr's Circle Help

SRD Traceability: Section 2

Summary: Verify that 3DStress will display a context sensitive help file for the Mohr's Circle display.

Protocol:

1. Open the Mohr's Circle display.
2. Select the Help menu item.
3. Verify the help file is opened and displays the correct help information.

Was the help information displayed correctly?

Yes, the mohr graph help window was displayed correctly

Tested by: JB (Initials) Date: 7/15/98

4.1.5 Mohr's Circle Close

SRD Traceability: Section 2

Summary: Verify that 3DStress will close all windows associated with the Mohr's Circle display.

Protocol:

1. Open the Mohr's Circle display.
2. Click the Options menu item.
3. Click the Close menu item on the Mohr's Circle display window.
4. Verify all windows associated with the Mohr's circle display are closed.

Were the windows closed correctly?
CloseHelp command closed all mohr circle windows except the help window because the Help window is displayed by a separate application – ShowCase. This will lead to porting problems to other platforms that cannot execute the SGI ShowCase application. No SPR generated here

Tested by: JB (Initials) Date: 7/15/98

4.2 FAULT COVERAGES

2-D and 3-D fault coverages may be created with 3DStress. 2-D coverages are built in the Map display while 3-D coverages are assembled in the 3DView display window. The following tests verify the functionality of the Map and 3DView windows for manipulating fault coverages.

4.2.1 Map Load

SRD Traceability: Section 2

Summary: Verify the capability of 3DStress to load and display 2-D fault map coverages.

Protocol:

1. Open the Map display.
2. Click the Load menu item and select an existing map file to display.
3. Verify the map file contents are displayed in the window with the correct scale information.
4. Repeat step 3 for another map file. Verify the original map is replaced by the map selection.

Were the map files displayed correctly?
Loaded first map file first - successfully
Loaded a couple non .lin files ⇒ the bad file errors were trapped correctly

Tested by: JB (Initials) Date: 7/15/98

4.2.2 Map ~~Options~~ Builder

SRD Traceability: Section 2

Summary: Verify the user capability to create new 2-D map coverages.

Protocol:

1. Open the Map display. Builder
2. Click the ~~Options~~ menu item.
3. Add several lines, each containing several control points, to the map display.
4. Save the new map file.
5. Load a different map file to ensure the new map file is purged from memory.
6. Load the newly created map file.
7. Verify the new map file is correctly loaded and displayed.

Was the new map file created, saved, loaded and displayed correctly?

When Adding line control points the point on the screen does scale correctly to the mouse position – SPR# 3

Attempted to Browse data in the options menu – program crashed SPR#

Points added to the Map display cannot be saved SPR#5

Tested by: TB (Initials) Date: 7/15/98

4.2.3 MAP RESET

SRD Traceability: Section 2

Summary: Verify the user capability to reset a corrupted 2-D map display.

Protocol:

1. Open the Map display.
2. Click the Load menu item.
3. Manipulate the map in a manner that corrupts the display.
4. Click the Reset menu item.
5. Verify the Map display returns to its default display presentation.

Was the corrupted map display properly reset?

Yes, the plot display was reset correctly

Tested by: TB (Initials) Date: 7/16/98

12/236

4.2.4 Map Print

SRD Traceability: Section 2

Summary: Verify the user capability to create an image of a 2-D map display suitable for printing.

Protocol:

1. Open the Map display.
2. Select the Print menu item.
3. Specify an output filename for the image file. /pscr0/ map.capture.rgb
4. Examine the image file created by 3DStress.

Was the image file created correctly?

No, same problem as noted in SPR #2

Tested by: JB (Initials) Date: 7/16/98

4.2.5 Map Coverage

SRD Traceability: Section 2

Summary: Verify the user capability to load and display symbol coverages on the 2-D map display.

Protocol:

1. Open the Map display.
2. Load a 2-D map file into the map display.
3. Select the Coverage menu item.
4. Load a symbol file to be displayed as a coverage over the 2-D fault map.
5. Verify the symbol coverage has been displayed correctly.

Was the coverage displayed correctly?
load world stress vectors and world geography correctly

Tested by: JB (Initials) Date: 7/16/98

4.2.6 Map Rose Diagram

SRD Traceability: Section 2

Summary: Verify the user capability to 2-D fault data and display a Rose diagram.

Protocol:

1. Open the Map display.
2. Load a 2-D map file into the map display. Loaded Fritzel-geo ~~file~~ fault file
3. Select the Rose menu item.
4. Verify the Rose diagram accurately portrays the faults trace azimuth distributions.
5. Repeat steps 2–4 for a different fault map file. loaded Dduran fault file

Were the Rose diagrams displayed correctly?

yes the display was plotted correctly for existing maps, but when we added several faults with the ~~map builder~~ the Rose diagram was not updated (SPR #6)

Tested by: JB (Initials) Date: 7/16/98

4.2.7 Map Help

SRD Traceability: Section 2

Summary: Verify that 3DStress will display a context sensitive help file for the Map display.

Protocol:

1. Open the Map display.
2. Select the Help menu item.
3. Verify the help file is opened and displays the correct help information.

Was the help information displayed correctly?

Killed the Window manager when the Map Help was requested Had to login from another machine as root to kill and restart the X window manager. After the restart the problem could not be recreated. (SPR#7)

Tested by: JTB (Initials) Date: 7/16/98

125/234

4.2.8 Map Close

SRD Traceability: Section 2

Summary: Verify that 3DStress will close all windows associated with the Map display.

Protocol:

1. Open the Map display.
2. Click the Options menu item. add the Rose Window
3. Click the Close menu item on the Map display window.
4. Verify all windows associated with the Map display are closed.

Were the windows closed correctly?

No the Rose Plot window did not close when the map window was closed SPR #8

Tested by: JB (Initials) **Date:** 7/16/98

4.2.9 3DView Load

SRD Traceability: Section 2

Summary: Verify the capability of 3DStress to load and display 3-D fault map coverages.

Protocol:

1. Open the 3DView display.
2. Click the Load menu item and select an existing 3-D fault file to display.
3. Verify the fault file contents are displayed in the window with the correct scale information.
4. Repeat step 3 for another 3-D fault file. Verify the original fault display is replaced by the new selection.

Were the fault files displayed correctly? Yes the faults are all saved in memory until they are explicitly removed in the option window the addition of faults having different coordinate systems leads to unusual plots where none of the faults will be displayed. (SPR 9)

Tested by: JB (Initials) Date: 7/16/98

127/234

4.2.10 3DView Builder

SRD Traceability: Section 2

Summary: Verify the user capability to create new 3-D fault coverages.

Protocol:

1. Open the 3DView display.
2. Click the Builder menu item.
3. Add several polygons, each containing several control points, to the 3-D display.
4. Save the new fault file.
5. Load a different fault file to ensure the new fault file is purged from memory.
6. Load the new fault file.
7. Verify the new fault file is correctly loaded and displayed.

Was the new fault file created, saved, loaded and displayed correctly?

The control over triangle control points was not scaled correctly to the mouse movement. (SPR #10)

When the End Triangle button is clicked, all triangles added by the user are erased (SPR #11)

This behavior occurred w and wout a fault file being displayed

Tested by: JB (Initials) Date: 7/16/98

→ However, if the Option window is closed while editing the polygons the polys remain plotted on the display. We attempted to save the file in /pscr/ghost/Dance2.flt and received an alert that the system was out of (1154) logical swap space. The program aborted with a ~~core dump~~ 85MB (SPR #12) the fault file was not saved.

4.2.11 3DView Reset

SRD Traceability: Section 2

Summary: Verify the user capability to reset a corrupted 3-D fault display.

Protocol:

1. Open the 3DView display.
2. Click the Load menu item.
3. Manipulate the fault view in a manner that corrupts the display.
4. Click the Reset menu item.
5. Verify the fault view returns to its default display presentation.

Was the corrupted display properly reset?

Yes the display was reset correctly

Tested by: SB (Initials) Date: 7/16/98

4.2.12 3DView Print

SRD Traceability: Section 2

Summary: Verify the user capability to create an image of a 3-D fault display suitable for printing.

Protocol:

1. Open the 3DView display.
2. Select the Print menu item.
3. Specify an output filename for the image file. /pscr0/view.capture.rgb
4. Examine the image file created by 3DStress. /pscr0/view.capture2.rgb

Was the image file created correctly?

Some problem ~~about~~ with window capture that has been documented in SPR #2

Tested by: JB (Initials) Date: 7/16/98

4.2.13 3DView Coverage

SRD Traceability: Section 2

Summary: Verify the user capability to load and display symbol coverages on the 3-D fault display.

Protocol:

1. Open the 3DView display.
2. Load a 3-D fault file into the 3DView display.
3. Select the Coverage menu item.
4. Load a symbol file to be displayed as a coverage on the 3-D fault display.
5. Verify the symbol coverage has been displayed correctly.

Was the coverage displayed correctly?

The coverage was displayed correctly

Tested by: JTB (Initials) Date: 7/16/98

131/236

4.2.14 3DView Help

SRD Traceability: Section 2

Summary: Verify that 3DStress will display a context sensitive help file for the 3DView display.

Protocol:

1. Open the 3DView display.
2. Select the Help menu item.
3. Verify the help file is opened and displays the correct help information.

Was the help information displayed correctly?

3D FaultViewer help file was displayed correctly

Tested by: JTB (Initials) Date: 7/16/98

4.2.15 3DView Close

SRD Traceability: Section 2

Summary: Verify that 3DStress will close all windows associated with the 3DView display.

Protocol:

1. Open the 3DView display.
2. Click the Builder menu item.
3. Click the Close menu item on the 3DView display window.
4. Verify all windows associated with the 3DView display are closed.

Were the windows closed correctly?

All windows were closed correctly

Tested by: JB (Initials) Date: 7/16/98

133/234

4.3 STRESS AZIMUTH MAP DISPLAY

The stress azimuth map display is used to plot distributed stress data as a map coverage.

SRD Traceability: Section 2

Summary: Verify the capability to load stress data from a symbol coverage and plot the stress data against a map backdrop.

Protocol:

1. Open the Map display. Coverage
2. Click the ~~Options~~ display menu item.
3. Click the Load button and specify a symbol file for loading. Select Sym Radio button
4. Verify the symbol file is plotted correctly on the map display.
5. Modify the stress coverage data and plot the modified file.
6. Verify the modified stress values were plotted correctly.

Were the stress coverage values loaded and plotted correctly?

yes, but very small symbols plotted erratically due to the screen resolution

Tested by: JB (Initials) Date: 7/16/98

4.4 STRESS CONDITION FILE INPUT AND OUTPUT

The 3DStress user enters stress field data in the Sliders and Mohr's Circle displays. The following tests will validate the capability to save and load the stress values in a disk data file.

4.4.1 Sliders

SRD Traceability: Section 2

Summary: Verify the capability of the 3DStress Sliders window to perform file input and output with the stress field data.

Protocol:

1. Open the Sliders display.
2. Modify the stress magnitudes.
3. Save the modified data to disk.
4. Change the stress data a second time to purge the previous values from the program memory.
5. Load the stress values stored in step 3.
6. Verify the stress values correspond to the expected values.

Were the stress values stored and loaded correctly?

Saved
$\sigma_u = 50$
$\sigma_v = 10$
$\sigma_w = 20$

/pscr0/stress1.mag

Changed
$\sigma_u = 1$
$\sigma_v = 30$
$\sigma_w = 40$

Loaded
$\sigma_u = 50$
$\sigma_v = 10$
$\sigma_w = 20$

Tested by: JB (Initials) Date: 7/16/98

4.4.2 Mohr's Circle

SRD Traceability: Section 2

Summary: Verify the capability of the 3DStress Mohr's Circle window to perform file input and output with the stress field data.

Protocol:

1. Open the Mohr's Circle display.
2. Open the Options menu item.
3. Modify the stress magnitudes.
4. Save the modified data to disk.
5. Change the stress data a second time to purge the previous values from the program memory.
6. Load the stress values stored in step 3.
7. Verify the stress values correspond to the expected values.

Were the stress values stored and loaded correctly?

Save: $\sigma_u = 5$
$\sigma_v = 10$
$\sigma_w = 15$

Change $\sigma_u = 100.00$
$\sigma_v = 50.0$
$\sigma_w = 1.0$

/pscro/stress2.mag
Load
Received
"Invalid magnitude Line 5"
message because there was
a ~~difference in the Fluid Pr~~
0.0 value in the Fluid Pressur

SPR # 13

Tested by: JB (Initials) Date: 7/16/98

4.5 LEAKAGE FACTOR CALCULATIONS

SRD Traceability: Section 2

Summary: Verify the 3DStress capability to compute Leakage Factors based on a user specified stress field.

Protocol:

1. Open the Plot display.
2. Click the Options menu item on the Main menu bar.
3. Click on the Leakage Factor radio button in the Compute section of the window.
4. Click the Sliders menu item on the Main menu bar.
5. Manually compute several data points on the Plot display to verify the Leakage Factor computation.

Input Stress values: $\sigma_u = 66$, $\sigma_v = 57$, $\sigma_w = 74$

Answer: 0.348 = FP/(σ_n − T)

Output Display Verified: FP = 27.273, σ_n = 37.299, T = −21.053

27.27/(57.299 − −21.053) = .3481 (Correct)

6. Change the stress data and repeat step 5.

Input Stress values: $\sigma_u = 59$, $\sigma_v = 69$, $\sigma_w = 65$

FP = 50, T = 30, σ_n = 88.483

.5077 = 50/(68.483 − −30) correct

Output Display Verified:

Were the Leakage Factor calculations performed correctly?

Yes

ed by: JB (Initials) Date: 1/16/98

APPENDIX B

Software Problem Report

Project: **3DStress Ver 1.3**	Originator: Joe Bongo	Date: 1/15/98	Problem Number: 1
Problem Name: Incorrect error trapping		Problem Priority: very low - 5	
Affected Software Element: ~~Error~~ Map Option Window			
Problem Description: Incorrect user entries - alpha characters in numeric fields were not trapped. However, ~~prob~~ program did not crash			
Problem Analysis:			
Problem Corrector:	Correction Date:	Version:	
Solution Description:			

139/236

Software Problem Report

Project: **3DStress Ver 1.3**	Originator: JB	Date: 7/15/98	Problem Number: 2
Problem Name: Mohr Circle Print		Problem Priority low - 4	
Affected Software Element: Printing			
Problem Description: If a window being printed is covered by another window the image file will include unwanted portions of the top window. The full window area is not being copied to the image file — the top of the image is clipped.			
Problem Analysis:			
Problem Corrector:		Correction Date:	Version:
Solution Description:			

Software Problem Report

Project: **3DStress Ver 1.3**	Originator: JB	Date: 7/15/98	Problem Number: 3
Problem Name: Map Builder Mouse Control		Problem Priority ~~Medium~~ 4	

Affected Software Element:
Map / Builder / Add Line Feature

Problem Description:
Fix When adding a new point to a line, the ~~dot~~ point being edited does not track the mouse pointer

Problem Analysis:

Problem Corrector:	Correction Date:	Version:

Solution Description:

Software Problem Report

Project: **3DStress Ver 1.3**	Originator: JB	Date: 7/15/98	Problem Number: 54
Problem Name: Options Map ~~Builder~~ Data Browse		Problem Priority: High 1	
Affected Software Element: Map/~~Builder~~/Options/Browse			
Problem Description: After entering several new data points in the Map Builder, User attempted to Browse the data. 3D stress crashed. Intermittent problem that could not be duplicated reliably.			
Problem Analysis:			
Problem Corrector:		Correction Date:	Version:
Solution Description:			

142/236

Software Problem Report

Project: **3DStress Ver 1.3**	Originator: JB	Date: 7/15/98	Problem Number: 5
Problem Name: Map Builder Data Save		Problem Priority: High 1	
Affected Software Element: Map/Builder			
Problem Description: There is no capability to save the data points entered by the user in the Map Builder display window			
Problem Analysis:			
Problem Corrector:		Correction Date:	Version:
Solution Description:			

Software Problem Report

Project: **3DStress Ver 1.3**	Originator: TB	Date: 7/16/98	Problem Number: 6
Problem Name: Rose Diagram of Added faults		Problem Priority 3	
Affected Software Element: Map/Rose/Builder			
Problem Description: Faults that are added to a map display are not used in the calculation of the Rose diagram.			
Problem Analysis:			
Problem Corrector:		Correction Date:	Version:
Solution Description:			

144/236

Software Problem Report

Project: **3DStress Ver 1.3**	Originator: JB	Date: 7/16/98	Problem Number: 7
Problem Name: Map Help Crash		Problem Priority #1	
Affected Software Element: Map Help			
Problem Description: After playing w the map functions, the Help button was clicked. This action locked up the console and required intervention from another machine by the Root superuser. After the console was restarted the Map Help button could not be made to recreate this bug.			
Problem Analysis:			
Problem Corrector:	Correction Date:	Version:	
Solution Description:			

145/234

Software Problem Report

Project: **3DStress Ver 1.3**	Originator: JB	Date: 7/16/98	Problem Number: 8
Problem Name: Map Close		Problem Priority 5	
Affected Software Element: Map			
Problem Description: The Rose plot window did not close when the Map Window was closed.			
Problem Analysis:			
Problem Corrector:		Correction Date:	Version:
Solution Description:			

Software Problem Report

Project: **3DStress Ver 1.3**	Originator: JB	Date: 7/16/98	Problem Number: 9
Problem Name: 3DView Coordinate System		Problem Priority 3	
Affected Software Element: 3DView Fault Viewer			
Problem Description: Adding faults to the display having different coord. system or ranges leads to a blank grid display with no user feedback about ~~where~~ why the previously displayed faults are no longer visible.			
Problem Analysis:			
Problem Corrector:		Correction Date:	Version:
Solution Description:			

Software Problem Report

Project: **3DStress Ver 1.3**	Originator: JB	Date: 7/16/98	Problem Number: 10
Problem Name: 3DView BuilderAddTriangle		Problem Priority 4	
Affected Software Element: 3DView			
Problem Description: When adding or moving triangle control points, the control point does not move in a proportionally scaled amount WRT the mouse motion			
Problem Analysis:			
Problem Corrector:		Correction Date:	Version:
Solution Description:			

Software Problem Report

Project: **3DStress Ver 1.3**	Originator: JB	Date: 7/16/98	Problem Number: 11
Problem Name: 3D View Builder-AddTriangle		Problem Priority 1	
Affected Software Element: 3DViews Builder			
Problem Description: Triangles added to the 3DView are erased when the End Triangle button is clicked. No user edits can be saved.			
Problem Analysis:			
Problem Corrector:		Correction Date:	Version:
Solution Description:			

Software Problem Report

Project: **3DStress Ver 1.3**	Originator: JB	Date: 7/16/98	Problem Number: 12
Problem Name: 3DViewer Builder/Save		Problem Priority 1	
Affected Software Element: 3DView			
Problem Description: We attempted to save a fault polygon file. Received an Alert 1154 from the OS that swap space had run out. 3DStress then crashed. We restarted the program and saved the modified file successfully. Note we were able to add a polygon in the saved file but the End Triangle button erased it when Builder was started			
Problem Analysis: Crashy problems may be the result of not freeing memory as the program executed → could be reason why program crashes are not repeatable.			
Problem Corrector:		Correction Date:	Version:
Solution Description:			

150/236

Software Problem Report

Project: **3DStress Ver 1.3**	Originator: JB	Date: 7/16/98	Problem Number: 13
Problem Name: ~~Mag~~ Saving ~~Fluid~~ Mag File		Problem Priority 2	
Affected Software Element: Plot Class			
Problem Description: Whenever a 0.0 Fluid pressure is saved, it could not be reloaded by the program.			
Problem Analysis: There needs to be an error trapping routine in saving the magnitude files to prevent the bad Fluid pressure from being saved in the magnitude file			
Problem Corrector:		Correction Date:	Version:
Solution Description:			

APPENDIX C

Mohr's Circle Options - Additional Tests

SRD Traceability: Section 2

Summary: **Verify that parameters entered in the Options parameter window result in accurate Mohr's Circle plot.**

Protocol:

1. **Open the Mohr's Circle display.**
2. **Select the Options menu item.**
3. **Modify the Rock Type selection in the Mohr Option window. Manually verify the program output and material failure point on the Mohr's Circle display.**

Inputs: R_{type} = Arenaceous $m = 15$ $s = 1.0$ $c = 73.8$

$\sigma_u = 3.03$ $\sigma_U = 50$ $\sigma_w = 100$ $\sigma' = 4.9$ $\tau = 17.$ plot values to be verified

Output Display Verified:

verified

Inputs: R_{type} = Carbonate $m = 7$ $\sigma' = 2.22$ $\tau = 24$

Lockport Dolomite $s = 1$

$c = 90.3$

$\sigma_u = 2.22$ $\sigma_U = 50$ $\sigma_w = 100$

Output Display Verified:

Verified

Inputs: Lithified Argillaceous $m = 10$ $\sigma' = -3.54$

Herkemack Siltstone $s = 1$

$c = 122.7$ $\tau = 17.$

$\sigma_u = -3.54$ $\sigma_U = 50$ $\sigma_w = 100$

Output Display Verified:

Verified

4. **Modify the Rock integrity parameter (Intact, very good...etc) in the Mohr Option window. Manually verify the program output and material failure point on the Mohr's Circle display.**

Inputs: Lith. Arg. Hack. Siltstone Good

$m = 2.965$
$S = 0.0205$
$C = 122.7$

$\sigma' = 18.17$
$\tau \approx 22$

Output Display Verified:

Verified

Inputs: Lith Arg. Hackensack Siltstone Poor

$m = .639$
$S = .00019$
$C = 122.7$

$\sigma' = 42.35$
$\tau \approx 21.5$

Output Display Verified:

Verified

Inputs:

no test

Output Display Verified:

5. Modify the Rock Type and then select a different Uniaxial Compressive Strength item in the Mohr Option window. Manually verify the program output and material failure point on the Mohr Graph display.

Inputs: Lith. Argill
Poor
Flaming Gorge
m = .639
S = .00019
C = 35.2
$\sigma' = 62.5$
$\tau = 18$

Output Display Verified:
Verified

Inputs:
m = .639
S = .00019
C = 53.6 ← user modified value
$\sigma' = 56.13$
$\tau = 18$

Output Display Verified:
Verified

Tested by: JB **Date:** 7/27/98
(Initials)

APPENDIX D

Software Problem Report Update on 3DSTRESS Version 1.3

Thirteen SPRs were generated by the validation testing and were documented in this STR. Larry McKague, David Ferrill and Josh Buckner reviewed the STR and selected five SPRs (1, 5, 6, 11 and 13) for correction in this release of 3DSTRESS. Four SPRs (3, 8, 9 and 10) were deemed to be either software features or not requiring correction and were closed. The four remaining SPRs (2, 4, 7, and 12) related to printing problems or the periodic abnormal termination of 3DSTRESS. Due to schedule constraints, these SPRs will remain open and be addressed in a future software release.

Mr. Buckner modified 3DSTRESS to address the five selected SPRs. These SPRs were then retested and successfully validated on July 29, 1998. The SPR log included in this appendix package reflects the status of the software retest and the four open SPRs.

The CD-ROM accompanying this report contains both the installation and archive versions of 3DSTRESS version 1.3. The installation file contains all binaries, scripts and help files required to successfully install and run 3DSTRESS version 1.3. This installation file was successfully transferred to and executed on CNWRA's "Redwood" Silicon Graphics workstation on July 31, 1998. The archive file contains all source code, header and other related files necessary to recompile or modify 3DSTRESS.

Based on this testing and documentation effort, 3DSTRESS is ready for outside distribution and entry into the CNWRA configuration management system.

3DSTRESS
Software Problem Report and Change Log

Date Opened	SVTP Section	SPR #	Change/Problem Summary	Status
7/15/98	4.1.1	1	Incomplete error trapping of user parameters in the Mohr's Circle Options window. Alpha characters entered in numeric field were not trapped.	Closed 7/29/98 Ver. 1.3
7/15/98	4.1.3	2	Printing a window in the Mohr Circle, Map, 3DView and other windows was not successful if the display window was covered by another window or if the window was not completely visible on the monitor. Also the image captured by the software excluded the top portion of the desired display window.	
7/15/98	4.2.2	3	When adding a new line segment point in the Map display, the edit point in the window does not accurately track the movements of the mouse.	Closed 7/29/98 Ver. 1.3
7/15/98	4.2.2	4	After entering several new line segment points in the Map window display, the user attempted to view the new data with the Browse function. 3DStress crashed when the Browse function was invoked. After restarting 3DStress, the problem could not be duplicated with the same Edit/Browse processing sequence.	
7/15/98	4.2.2	5	There is no capability to store fault control points entered in the Map display. User edits cannot be saved to disk.	Closed 7/29/98 Ver. 1.3
7/16/98	4.2.6	6	2-D faults added to a map display are not incorporated in the calculation of the Rose diagram.	Closed 7/29/98 Ver. 1.3
7/16/98	4.2.7	7	Clicking the Map Help caused 3DStress to crash and "locked-up" the console monitor, preventing user inputs. Correction of this problem required the system root user to remotely login into the test machine and restart the Xwindows window manager application. The problem was not duplicated in several attempts after the software was restarted.	
7/16/98	4.2.8	8	The Rose plot diagram did not close when the Map display was closed.	Closed 7/29/98 Ver. 1.3

Date Opened	SVTP Section	SPR #	Change/Problem Summary	Status
7/16/98	4.2.9	9	When adding multiple 3-D faults to the 3-D fault viewer, the faults must be generally within the same region. Otherwise the 3DViewer display grid will not plot any data on the plot grid. This is confusing to the user but does not cause the loss of data.	Closed 7/29/98 Ver. 1.3
7/16/98	4.2.10	10	When adding polygons to the 3DView display, the edit point does not move proportionally to the distance traveled by the mouse pointer. This may be an artifact of the 3-D display, but it is confusing to the user.	Closed 7/29/98 Ver. 1.3
7/16/98	4.2.10	11	Triangles added to the 3DView are erased when the End Triangle button is clicked. By closing the Options window without clicking the End Triangle, the user can save one polygon to the fault file being edited. The user cannot effectively save fault polygon information in the 3DViewer.	Closed 7/29/98 Ver. 1.3
7/16/98	4.2.10	12	When attempting to save a fault polygon file, we received an Alert message (1154) from the operating system that the system swap space had run out. 3DStress then crashed. After restarting the application, we able to save the same 3-D fault file we were processing prior to the software crash.	
7/16/98	4.4.2	13	While testing the Mohr Circle and Slider capabilities to save/load stress magnitude information on disk, we received a 3DStress error message that the previously save stress magnitude file had an erroneous entry for fluid pressure. 3DStress is not error trapping the fluid pressure value when it is saved to disk, but is error trapping the fluid pressure when loading from disk. The error trapping routines should be consistent in both directions.	Closed 7/29/98 Ver. 1.3

SOFTWARE
DEVELOPMENT PLAN

198/236

SOFTWARE DEVELOPMENT PLAN FOR 3DSTRESS

Prepared for

Nuclear Regulatory Commission
Contract NRC-02-97-009

Prepared by

Joseph H. Bangs

Center for Nuclear Waste Regulatory Analyses
San Antonio, Texas

Approved by: H. Lawrence McKague **Date:** 8/7/98
H. Lawrence McKague

199/236

CONTENTS

CONTENTS (cont'd)

Section Page

CONTENTS (cont'd)

Section Page

FIGURES

TABLES

1 SCOPE

This document establishes the Software Development Plan (SDP) to be implemented by the Center for Nuclear Waste Regulatory Analyses (CNWRA) for the development and release of the 3DStress version 1.3 software application.. The software will be provided to the government (acquirer) without proprietary restrictions.

1.1 Identification

This SDP applies to software modifications and corrections to be made to version 1.2 of the 3DStress application. The modified code will be identified as 3DStress version 1.3.

1.2 System Overview

The 3DStress application is used by scientists and engineers to study the relationship between static stress fields and geologic faulting. 3DStress utilizes user defined stress fields to compute the likelihood of fault displacement based on the fault orientation. 3DStress provides user input, computation, and data visualization tools to create an interactive environment in which various stress models may be studied and explored efficiently.

3DStress executes on a Silicon Graphics workstation running the IRIX operating system. The application does not communicate or interface with any other computer system or software application.

1.3 Document Overview

This SDP defines the plan for management, development, and software maintenance for the 3DStress software application. This document contains the procedures to address the following program management tasks:

a. Software design practices
b. Software Quality Assurance
c. Software configuration management
d. Software engineering standards
e. Software development process
f. Organizational structure
g. Schedule

These guidelines will ensure the efficient utilization of project resources to deliver a high quality software product in a timely manner.

1.4 Relationship to Other Plans

This SDP is not related to any other plan.

2 REFERENCED DOCUMENTS

The following documents provide guidelines for software development and documentation activities. In the event of conflict between this document and those referenced herein, the contents of this document shall be considered superseding requirements.

CNWRA-TOP-18	1 MAY 98	Development and Control of Scientific and Engineering Software

3 OVERVIEW OF REQUIRED WORK

3.1 General

CNWRA has modified version 1.2 of 3DStress to enhance software performance, provide additional capabilities and correct software defects. All functionality provided by the current version will be duplicated or replaced in the new version. The host hardware platform for 3DStress will remain a Silicon Graphics workstation running version 6.x or 5.x of the IRIX operating system.

CNWRA will perform the software requirements analysis, design, development and testing necessary to deliver a reliable software and documentation product at the end of the development project.

3.2 Software Functionality

The 3DStress application calculates either the slip tendency or dilation tendency of one or more geologic faults for a static three dimensional stress field. The application displays various data plots in which colors and 3D surfaces are rendered to convey the computational results to the software user. 3DStress will read data files containing fault geometry information and will save copies of the various display windows for hard copy output or as input to other software applications.

3.3 Software Design and Development

The new version of 3DStress will be designed to meet or exceed the requirements for the existing application version.

All software will be developed in the C++ programming language unless highly specialized coding is required for performance beyond the ability of the commercial compiler. A commercial source control product will be employed to track and coordinate all modifications to the software source code.

Like the existing version, the new 3DStress application will operate in a stand-alone mode requiring operator control for the execution of all software operations. The application will retain the existing man-machine-interface based on the X Windows program environment and the Open GL graphics rendering library.

All new software development and modifications will be done in accordance with CNWRA TOP-18. All existing code being reused will not be unnecessarily modified or documented to CNWRA-TOP-18. Reused code will consist of code used as is or with only minor customization for use with 3DStress version 1.3.

3.4 Hardware Configurations

The 3DStress software application will operate on a single Silicon Graphics computer platform. The software will operate on any SGI equipped with a monitor, keyboard, mouse and removable media drive for software installation. Due to the extensive computational nature of 3DStress, CNWRA recommends the following hardware configuration for acceptable calculation and display performance:

200 MHz Iris Processor or better with floating point coprocessor
128 Mbytes RAM
High Impact graphics board

9 GB hard drive
19" monitor or larger
8 mm Tape or Digital Audio Tape drive
CDROM
Network connection

4 PLANS FOR PERFORMING GENERAL SOFTWARE DEVELOPMENT ACTIVITIES

The following sections outline plans for performing general software development activities for the 3DStress software application.

4.1 Software Development Process

CNWRA will utilize a Grand Design strategy for development of the 3DStress application. The Grand Design approach results in a single software build and is appropriate for this project because:

a. The 3DStress software requirements are well known and documented in the existing 3DStress software documentation.
b. The 3DStress application is not a large development effort and can be accomplished in a short time frame.
c. Once the final software requirements are specified the development schedule will be firm and will not be altered due to changing technical requirements.

4.2 General Plans for Software Development

The following sections define the software development practices and standards to be applied to the 3DStress software development effort.

4.2.1 Software Development Methods

For the same reasons the Grand Design program strategy was selected, the classic life cycle method of software development will be employed. The classic life cycle method involves requirements analysis, design, coding, module testing, integration, system level testing and implementation.

All software will be developed in the C++ programming language unless specialized coding is required for software performance beyond the capability of the C++ compiler. Any deviations from the use of the C++ compiler will be reviewed by the software development team to determine the impact on related software modules.

CNWRA will base the C++ software design and implementation of the following development approaches:

1. Information Hiding - Decomposition of a system into units, such that each is characterized by its knowledge of a design decision which it hides from all others. The design decision may relate to either a routine or data. Access to hidden data or routines will be controlled through well defined interfaces with limited update privileges.
2. Encapsulation - Related data and data processing/manipulation processes will be organized or structured as classes reflecting 3DStress component organization, interfaces and processing. Subclasses will be derived from parent classes until the child class represents a unit process or interface in enough detail to express the class behavior as data variables and

member functions.

4.2.2 Standards for Software Products

The following sections define the standards to be followed in developing the software requirements, design, coding, test procedures and documenting test results for the 3DStress development project.

4.2.2.1 Software Design Standards

Software design is the process by which requirements are translated into software representations using structured analysis techniques. A preliminary software design will define modifications to existing or additional 3DStress computational and/or display capabilities. These capabilities will be mapped to software classes by functional and data access requirements. A subsequent refinement of the design will lead to detailed class definitions optimized for efficient software operation.

Throughout the design process, the quality of the evolving design will be reviewed by the software developer with the software or project manager. The software team will adhere to the following design quality criteria:

1. The design will be modular and logically partitioned into components that perform distinct functions.
2. The design will contain distinct classes reflecting the modular design of the software.
3. The design will lead to software modules that exhibit independent functional characteristics
4. The design will strive to simplify user interfaces.
5. The design will incorporate the concept of abstraction, enabling the designer to simplify and reuse software components.

4.2.2.2 Software Coding Standards

Coding will translate the software design into C++ language software files and will begin after the detailed design has been completed and reviewed with the project and element managers. CNWRA will code the software to have the following characteristics:

1. Ease of code to design translation
2. Maximum compiler efficiency
3. Maximum use of development tools
4. Maintainability

CNWRA will employ a coding style that stresses simplicity and clarity. This approach will be applied to data declaration, statement construction, and data input and output. This coding philosophy will enhance the software readability while simplifying the test/debug/implement portion of the software development cycle.

4.2.2.2.1 Headings

Each software unit will begin with a unit header that explains the following:

1. Description and purpose of the operation
2. General unit design
3. Initialization
4. Global interactions (if any)
5. Error conditions and handling

4.2.2.2.2 Comments

Source code will be explained with comments. Comments will explain the intended operation, logic and possible error conditions associated with code sections. General comments will precede code sections while detailed comments will be interspersed in the code. Comments will be written for readers with moderate software comprehension.

4.2.2.2.3 Variable Naming

Names of variables and classes will be descriptive and indicative of program activity. Names shall avoid the use of abbreviations, mnemonics and jargon within the constraint of size limitations. Comments will be used to explain the role of all non-trivial variables and classes at the time of declaration. Names will adhere to consistent formats across all code modules. This will include the use of capitalization, under scores and unique letter combinations to identify specific classes of variables.

4.2.2.2.4 Restrictions

Previous versions of the 3DStress application software were developed using the C/C++ programming language and incorporated function calls to various operating system, X Windows, and Open GL libraries bundled with each Silicon Graphics hardware platform. No restrictions will be placed on the use of additional software libraries except that the use of all third party library products will be documented in the source code and software version or release description.

Also, the software team will restrict its use of multiple inheritance to only those version 1.2 base classes already utilizing multiple inheritance. The use of multiple inheritance is discouraged and the software developers will also attempt to avoid mixing traditional C function calls with their equivalent C++ counterparts.

4.2.2.2.5 Complexity

Code aggregates will be limited to the level that a software programmer can understand them without in-depth study. Individual coding statements will be simple and direct and will not be convoluted for esoteric or marginal efficiency gains. Individual source code statements will be simplified by avoiding complicated conditional statements, tests on negative test conditions, and unnecessary nesting in loops or conditions. Source code statements will use parenthesis to clarify statement content. Related code such as loops, blocks and cases will be grouped and commented as a functional entity.

4.2.2.3 Software Test Standards

Software testing accounts for a large percentage of technical effort in the software development process. The objective of software testing is to identify errors. To fulfill this objective, CNWRA will utilize

a series of steps in testing the software first at the unit level, and then progressing to the integration and system levels.

Unit level tests will concentrate on functional verification of software modules prior to incorporation into the program structure. Unit testing makes heavy use of white box testing techniques to exercise specific paths in a module's control structure to maximize error detection. After unit testing, modules are assembled to form the complete software package.

Integration testing addresses reliability issues associated with program verification and construction. Software modules must work in concert to provide program functionality. Integration testing reveals errors in module interactions and deficiencies in meeting functional requirements. After successful integration testing, a set of high order system tests are conducted.

System validation testing will demonstrate traceability to software requirements, and will provide assurance that software meets functional, behavioral and performance requirements. The validated software will then be installed in an operational environment to demonstrate system performance.

4.2.3 Reusable Software Products

The following sections outline the approach for incorporating reusable software products and developing new reusable software for the 3DStress software application.

4.2.3.1 Incorporating Reusable Software Products

CNWRA will investigate several potential sources of reusable software for the 3DStress application. Version 1.2 of 3DStress is written in the C++ programming language and several software modules will be incorporated directly into version 1.3. Wherever possible, existing software will be analyzed to determine if software modifications are necessary to enhance the reusability of the source code in future software versions.

A second source of reusable software will be in the form of commercial device drivers, function and class libraries, operating system resources and documentation generators. Wherever possible and advisable, CNWRA will identify commercial products for incorporation or utilization in the development of the new version of 3DStress.

4.2.3.2 Developing Reusable Software Products

This development project will apply good software development techniques in developing the new 3DStress version. By combining good software development practices with the use of C++, which lends itself to reuse through class inheritance, the 3DStress project will result in some software that is reusable. However, it is not the goal of this project to develop reusable software at the expense of software efficiency or simplicity.

4.2.4 Handling of Critical Requirements

The following sections outline the approach for handling critical requirements for the 3DStress project.

4.2.4.1 Safety Assurance

3DStress software activities do not require safety assurances. This paragraph has been tailored out.

4.2.4.2 Security Assurance

The 3DStress application does not contain any security related procedures or data. This activity is tailored out.

4.2.4.3 Privacy Assurance

The 3DStress application will not contain any privacy related procedures or data. This activity is tailored out.

4.2.4.4 Assurance of Other Critical Requirements

Requirements deemed critical by the technical directive will be presented by the acquirer and will be incorporated into this plan as appropriate.

4.2.5 Computer Hardware Resource Utilization

The 3DStress application will be developed, tested and executed on existing CNWRA Silicon Graphics workstations. No additional hardware resources are required for this development effort.

4.2.6 Recording Rationale

Software development activities will be documented in Software Development Files (SDFs) maintained by individual software developers. These files will contain engineering assumptions as well as standard software development information. Rationale will be recorded and submitted to the project manager at the conclusion of the development effort. Key decisions and rationale will be discussed during technical and management reviews throughout the development project.

4.2.7 Access for Acquirer Review

Throughout the project performance period, the CNWRA project team will be available for telephone discussions regarding the development effort. All development activities will take place in the CNWRA GIS laboratory, that is accessible to acquirer personnel.

5 PLANS FOR PERFORMING DETAILED SOFTWARE DEVELOPMENT ACTIVITIES

The following sections outline the detailed Software Development Activities for the 3DStress project.

5.1 Project Planning and Oversight

The following sections describe the approach to be employed for project planning and oversight of the 3DStress development project.

5.1.1 Software Development Planning

This document contains the pertinent information related to software development planning. The project team through the Project Manager may make recommendations for improvements or changes to the SDP. The Project Manager will determine the impact on schedule and cost and, if appropriate for the program, present the SDP modifications to the acquirer for approval and contract modifications.

5.1.2 Software Test Planning

Based on the results of the Software Requirements analysis, a Software Test Plan (STP) will be developed for qualification testing of the 3DStress application. This plan will describe the software test environment, the test(s) to be performed, and the test schedule. Test results will be recorded in the SDFs and will be available for acquirer review.

5.1.3 System Test Planning

The 3DStress application is a single build computer software configuration item (CSCI) that interacts directly with the software user. System testing is not separable from CSCI testing and will thus be conducted with CSCI testing. System level tests will be defined in the STP.

5.1.4 Software Installation Planning

The 3DStress application will be delivered on removable media. The installation procedure and scripts will be designed, documented, and built to simplify the installation procedure. No hardware modifications are anticipated for the migration from version 1.2 to 1.3. Recipients of version 1.3 will be responsible for internally coordinating local software installations.

5.1.5 Software Transition Planning

CNWRA will document any version specific requirements associated with the new release of 3DStress. Recipients of the new version will be responsible for internally coordinating any file translations necessary to support version 1.3.

5.1.6 Following and Updating Plans, Including the Intervals for Management Review

CNWRA will conduct the development and testing of 3DStress version 1.3 in accordance with the SDP and STP. The software development team will meet periodically with the Project Manager to verify the software process is adhering to these plans. At this time, no changes to the plans are anticipated. However, should a plan need modification, the Project Manager will present the changes to the acquirer for approval and coordinate their implementation with the development team.

5.2 Establishing a Software Development Environment

The following sections outline the approach for establishing, controlling and maintaining the software environment for the 3DStress project.

5.2.1 Software Engineering Environment

A single software development environment will be established for this project in the CNWRA GIS Laboratory, Bldg. 189 at the CNWRA facility. One Silicon Graphics workstation will be used for all software development processes. All printed materials, vendor CD's, floppies and tapes will be stored in the GIS Laboratory. Intermediate disk backups will be made to tape and will also be stored in the GIS Laboratory.

5.2.2 Software Test Environment

CNWRA maintains four SGI workstations for software development and GIS activities. One SGI will be used for software development while the other SGI machines will be available for software and installation testing.

5.2.3 Software Development Library

The lead software developer will serve as the software librarian and will have primary control over the software development library. Because the software development team is small, all team members will have access to the library in the absence of the librarian. The librarian will establish a working library on the development Silicon Graphics workstation. Both libraries will be subdivided into a subdirectory structure designed to contain deliverable documents, software units, SDFs, and commercial software products.

5.2.4 Software Development Files

Informal Software Development Files (SDFs) will be created for the 3DStress software units. The SDFs will be created prior to the initiation of detailed design and shall be maintained for the duration of the project. They will be made available to the product evaluation team, quality assurance, and acquirer representatives as requested. The SDF may reference information in other project documents as necessary. All schedule and status data will be in other project documents. SDFs will be created and maintained by the programming staff under the direction of the Project Manager.

The SDFs will be maintained predominantly in electronic form. The electronic form will be a combination of plain ASCII and word processor files. If necessary paper submissions will be included in a binder and referenced in the electronic form.

SDFs will generally include the following information:

1. Record Sheet - The contents of the SDF are listed by item name and electronic name and location. The engineer responsible for the SDF is identified with the due date, completion date, originator sign-off, and reviewer sign-off.
2. Requirements Specification - All requirements that the CSU must satisfy are listed by reference to the applicable sections of the Software Requirements Specification.
3. Interface Description - Global variables/constants, calling sequences, and input/output formats are defined or referenced.
4. Preliminary Design - Preliminary design description.
5. Software Test Information - All test cases and test procedures are defined or referenced. Concurrent with code walk-through, the reviewer will verify that the test plan fully tests capabilities, interfaces, and design constraints.
6. Source Code Organization Description - A description of the location and directory structure of the CSU source code as well as commercial products used in the CSU.
7. Test Results - At all levels, records of test results are maintained by test case identifier, tester, date, and the revision status of test drivers, tools, database, and code tested. Significant differences between expected and actual results will be explained..
8. Software Problem Reports - SPR forms shall be used to document problems encountered in software and software documentation.
9. Notes - All explanatory materials relevant to the CSU are maintained in the section. Formal deviation and waivers are also kept in this section.
10. Reviewers Comments - Reviewers comments on the other sections of the SDF are kept in this section.

5.2.5 Non-deliverable Software

Where necessary, CNWRA will develop simulators to test software component functionality. The end item software will not utilize these test fixtures and therefore will not be delivered, controlled or documented to the software release standards.

5.3 System Requirements Analysis

The following sections describe the approach CNWRA will follow in developing the software system design for the 3DStress application.

5.3.1 Analysis of User Input

3DStress operates as a stand-alone software application requiring user directives to control application execution. Additional features planned for version 1.3 will be analyzed to design an optimal user interface environment. The primary user interface design criteria will be ease of operator control and the effective display of computational results. At this time, no additional user input devices are anticipated beyond the traditional input devices (keyboard, mouse) attached to standard SGI workstations.

5.3.2 Operational Concept

3DStress is designed to be an interactive software application utilized by research and scientific staff at irregular intervals. The software is not intended to become an integral part of day-to-day operations. Therefore an operational concept description will not be written for this application. This activity has been tailored out.

5.3.3 System Requirements

Based on experience with 3DStress version 1.2 and planned modifications for version 1.3, no system modifications are necessary for this release version.

5.4 System Design

Version 1.3 represents an incremental change to 3DStress version 1.2. The existing version 1.2 system design will be utilized in version 1.3.

5.5 Software Requirements Analysis

This version of 3DStress is intended to be a functional replacement of the current application. CNWRA will review and analyze the requirements for the new version to determine the operational concepts and software specific requirements of the new CSCI. Software technology areas needed to implement the new version will be evaluated with respect to technologies utilized in the current software revision. New requirements will be categorized as new technology or extensions of existing capabilities. The results of this analysis will be documented in the Software Requirements Description (SRD) as a reference for testing and validating the new software version.

5.6 Software Design

The following sections describe the approach CNWRA will follow in preparing the software design for 3DStress, version 1.3. The results of the software design process will be documented in the software development files maintained by the development team members.

5.6.1 CSCI-wide Design Decisions

CNWRA will analyze the SRD to refine the existing concept of data and event management within the current 3DStress application. The software team will prioritize event management and data processing tasks according to their impact on overall system performance and functionality. From this prioritization, the team will evaluate various models for allocating hardware and software resources during software execution. Any modifications to the current CSCI event and data management concepts will be documented in the SDFs.

5.6.2 CSCI Architectural Design

Using the high-level resource allocation model, the software team will design an internal CSCI

architecture to implement the major functional requirements specified in the SRD. This design will define any new or modified classes of data and functionality necessary to implement the new software capabilities.

5.6.3 CSCI Detailed Design

CNWRA will refine the architectural design into individual software units by designing algorithmic approaches for implementing specific software requirements defined in the SRD. Algorithm development will focus on meeting or exceeding performance and functional specifications while adhering to the previously defined communication, processing and event management framework.

5.7 Software Implementation and Unit Testing

The following sections describe the approach to be followed for software implementation and unit testing for the 3DStress application.

5.7.1 Software Implementation

The software will be developed within the coding techniques described above in Section 4.2.1. All software will be developed in the C++ programming language unless highly specialized coding is required for performance beyond the ability of the C++ compiler to produce efficient binary executables. Any deviations from the use of C++ will be approved by the Project Manager.

No relational databases are required for the 3DStress application. All system configuration and geologic fault information will be maintained in plain ASCII text files or publicly defined binary file formats. ASCII file formats are generally preferred for all files except very large data files where ASCII storage is impractical.

5.7.2 Preparing for Unit Testing

Unit testing will be designed to verify the new software meets the detailed software design in the SDFs. CNWRA will develop test cases using by calculating outputs from known or controlled inputs for each major software unit. Controlled test cases will be computed using independent computations not relying on the newly developed software. Information regarding the test case computations and results will be documented in the SDFs.

5.7.3 Performing Unit Testing

As major software units are completed, the developer will conduct unit testing with test cases to verify the expected results.

5.7.4 Revision and Retesting

As needed, the developer will revise and retest software units to ensure compliance with the functionality described in the SDFs and SRD.

5.7.5 Analyzing and Recording Unit Test Results

Each iteration of testing and revision through the successful completion of the test case will be documented in the applicable SDF. Persistent test failure by a software unit will be analyzed to determine if failures are derived from an inadequate design, insufficient documentation, or improper coding practices.

5.8 Unit Integration and Testing

The following sections describe the approach to be followed for unit integration and testing for the 3DStress project.

5.8.1 Preparing for Unit Integration and Testing

Unit integration testing will be performed at the major software component level. CNWRA will develop test cases and data in terms of inputs and expected outputs for the control subsystem. Information regarding the test cases, procedures and results will be stored in the SDFs.

5.8.2 Performing Unit Integration and Testing

As major software units are ready for testing, the development team will conduct component level testing with test cases and verify the expected outputs.

5.8.3 Revision and Retesting

As needed, the developer will revise and retest software components to ensure compliance with the functionality described in the SDD and SRS.

5.8.4 Analyzing and Recording Unit Integration and Test Results

Each iteration of testing and revision through the successful completion of the test case will be documented in the applicable SDF. Persistent test failure by a software component will be analyzed to determine if failures are derived from an inadequate design, insufficient documentation, or fundamental errors in the CSCI architectural design.

5.9 CSCI Qualification Testing

The intent of the CSCI qualification testing for the 3DStress application is to verify that the new controller is functionally equivalent to the previous version and provides the additional capabilities described in the SRD.

The following sections describe the approach to be followed for CSCI qualification testing for the 3DStress application.

5.9.1 Independence in CSCI Qualification Testing

The Project Manager will assign an individual from the CNWRA , CNWRA QA or SwRI staff who has not participated in the design or development of the 3DStress to conduct formal testing of the CSCI.

5.9.2 Testing on the Target Computer System

All CSCI qualification testing will be performed on Silicon Graphics workstations available to the CNWRA.

5.9.3 Preparing for CSCI Qualification Testing

Upon completion of the CSCI software, the software will be entered into the Configuration Management (CM) system as the 3DStress product baseline.

The Software Test Plan (STP) and Software Test Procedures (STPr) will be given to the independent software tester in preparation for the dry run CSCI qualification test.

5.9.4 Dry Run of CSCI Qualification Testing

The software developer will dry run the test procedures to ensure that they are complete, accurate and are ready for witnessed testing. The results of the test will be recorded in the appropriate SDFs. Software Problem Reports will be prepared for problems uncovered during the testing process.

5.9.5 Performing CSCI Qualification Testing

A witnessed qualification test will be conducted at CNWRA to verify and demonstrate that the 3DStress application meets the system and software requirements stated in the SRD. If additional revision and retesting is required, the appropriate portions of the CSCI qualification test will be rerun after completion of the revisions.

5.9.6 Revision and Retesting

Revisions, based on SPR (corrective action) processing, and retesting will be accomplished prior to final approval of the qualification testing. Where necessary SDFs will be updated to reflect the revisions and retesting.

5.9.7 Analyzing and Recording CSCI Qualification Test Results

When the CSCI qualification testing is competed, the test results will be recorded in a Software Test Report. If revisions to the 3DStress application were made during the qualification test process, the qualified software will be resubmitted to CM as the new baseline version.

5.10 CSCI/HWCI Integration and Acceptance Testing

The 3DStress application is a single build CSCI designed to run on Silicon Graphics workstation. CNWRA possesses four different Silicon Graphics workstation configurations. 3DStress will be tested on CNWRA SGI machines that were not used in the software development process.

CNWRA has prepared a standard installation test case that shall be run after software installation to verify correct software installation. This acceptance test procedure is documented in the 3DStress on-line help manual.

5.11 System Qualification Testing

The 3DStress application does not interface other computer hardware or software systems. This paragraph has been tailored out.

5.12 Preparing for Software Use

The following sections describe the approach to be followed for preparing the 3DStress application for distribution to existing users.

5.12.1 Preparing the Executable Software

CNWRA will prepare the executable software for delivery to the user community. This preparation will include script and data files, executables, shared object libraries, configuration files, and any other software files required to operate the application. These files will be stored on standard removable storage media such as CDs or tapes.

5.12.2 Preparing Version Descriptions for User Sites

CNWRA will prepare a Software Release Notice for delivery with the 3DStress application to identify and describe the released software version for tracking and control purposes.

5.12.3 Preparing User Manuals

The following sections outline the preparation of the user manuals for the 3DStress application.

5.12.3.1 Software Users Manual

CNWRA will prepare a Software User Manual (SUM) which describes the installation and operation of the 3DStress application. The SUM will describe all user input and activities required to control and review the computational results generated by 3DStress.

5.12.3.2 Software Input/Output Manual

A separate Software Input/Output manual will not be prepared for this application. This activity has been tailored out.

5.12.3.3 Software Centers Operators Manual

A separate Software Centers Operators Manual will not be prepared for this application. This activity has been tailored out.

5.12.3.4 Computer Operation Manuals

A separate Computer Operation Manuals will not be prepared for this application. This activity has been tailored out.

5.12.4 Installation at User Sites

CNWRA will support NRC on-site installation of 3DStress as needed. Support for other 3DStress users will be arranged on a case by case basis.

5.13 Preparing for Software Transition

CNWRA will retain rights to the 3DStress application executable and support files. No software transitions to another organization are planned at this time. This activity has been tailored out.

5.14 Software Configuration Management

Software configuration management (CM) is the process by which baselined documents and source code are identified and changes are identified and recorded. All deliverable documents and source code will be placed under CM.

3DStress CM will be the responsibility of the Project Manager. The Project Manager will determine when source and documents are to be submitted to CM and will control the release, modification and resubmission of these materials to CM. The following sections define the CM process that will be followed by the 3DStress Project Manager.

5.14.1 Configuration Identification

Two software products will be placed under CM for the 3DStress project: software source code and application executables produced during the project.

The 3DStress application is a single build CSCI and will have a single version number. Any commercial software products incorporated in the control subsystem software will have vendor version numbers that will be documented in the Software Release Notice, but these numbers will only be used internally and will not be reflected in the CNWRA assigned version number. The CNWRA software version numbering will follow the following format:

12.34

Where: 12 is a one or two digit version number which identifies the major software version, starting with the number 1. Changes to basic functionality or major enhancements will cause this number to be incremented.

.34 is a one character separator (.) and two digit revision number. This number will start with 00 and will be incremented as minor software changes are made in response to Software Problem Reports (SPRs).

5.14.2 Configuration Control

On initial and subsequent release to CM of software products, the Project Manager will follow the procedure listed below:

1. The Project Manager will prepare a Software Release Notice (SRN) form, refer to CNWRA-TOP-18 for the appropriate format.
2. If this is the final delivery to the acquirer, the Project Manager will make sufficient copies of the deliverable material as required by the acquirer.
3. The Project Manager will provide the CNWRA QA group a copy of all products to be placed in CM

5.14.3 Configuration Status Accounting

The 3DStress Project Manager will prepare and maintain records of the configuration status of all software documentation and the 3DStress CSCI that have been placed under configuration control. These records will be maintained for the life of the 3DStress application. The records will contain the current version/revision/release of each entity, changes to the entity since being placed under CM, and the status of open SPRs affecting the entity.

5.14.4 Configuration Audits

The 3DStress Project Manager will make configuration management records available on a non-update basis for audit by the CNWRA or acquirer representatives.

5.14.5 Packaging, Storage, Handling, and Delivery

The software and documentation will be stored in paper and electronic form. Documentation will be stored on 3.5" floppy disks or CDs in WordPerfect for Windows 8 or later format. End item software will be delivered on 3.5" floppy disks, CD-ROM or 8mm tape in plain ASCII text file format.

5.15 Software Product Evaluation

The 3DStress application will be demonstrated for potential clients but no plans exist to distribute evaluation copies of the software. This activity has been tailored out.

5.16 Software Quality Assurance

CNWRA will follow a two-fold approach to building a quality product for the 3DStress application:

Quality Development - define and follow good software development practices throughout the development effort. For the 3DStress project, CNWRA will use internal development staff for planning, coding and testing who will adhere to the plans and implementation procedures outline in this SDP.

Quality Assurance - ongoing verification that the process are being followed by the development team. CNWRA will utilize the CNWRA QA department for review, evaluation and recommendations.

CNWRA QA (CQA) will monitor the software development process to verify the procedures and practices identified in this plan are being utilized in the 3DStress development. Evaluations will be informal and deviations from this development plan will be brought to the attention of the Project Manager. Continued deviation from the development plan will require notification of CNWRA management to discuss corrective actions or initiate an update of the software development plan to reflect changes in the project scope.

5.17 Corrective Action

The corrective action process is uniform for any software unit requiring correction. The formal corrective action process becomes effective once the 3DStress control subsystem CSCI enters the CM system. All corrective actions (CA) are initiated with a Software Problem Report (SPR). The SPR form to be used for this project is described in Section 5.17.1. Document or software comments from the contracting agency or IQA are not required to be submitted on the SPR form, other formats are acceptable. Proposed enhancements to the system may also be initiated through the use of the SPR.

5.17.1 Problem/Change Reports

An example SPR form is shown in Figure 5-1. To accommodate lengthy explanations or supporting material, attachments to the forms may be referenced in the appropriate fields. All SPRs are maintained in the project file by the Project Manager for the duration of the project and will be made available to acquirer representatives upon request.

5.17.2 Corrective Action System

The corrective action system centers around the submission of the SPR. SPR processing will generally adhere to the following sequence:

1. Problem identification and report submission. An SPR can be generated by any project member or software user who detects a problem or recognizes a required enhancement to a baselined document or software program.

Blank Page

Software Problem/Change Report

Project:	Originator:	Date:	Number:
Problem/Change Name:		Priority	
Affected Software Element:			
Description:			
Analysis:			
Modifications by:		Date:	Version:
Implementation:			

Figure 5-1. Sample software problem report

2. Logging. Following receipt of an SPR (or equivalent), the requested CA is entered into a CA log sheet. This log sheet facilitates tracking and reporting of all CA's issued during the life of the project. The Project Manager will then assign the CA to a software engineer for analysis.
3. Analysis. Analysis will be performed by the assigned engineer to determine the category of the CA: software, documentation, design, user, of requirement problem. The analysis also needs to determine what priority level should be assigned to the problem. Analysis of problems that lead to modification of software need further documentation, including test cases in order to assure that the problem has indeed been resolved.
4. Approval. After analysis the Project Manager will decide if a software or document change is necessary. The Project Manager is also responsible for final determination of the category, priority, and type of action required.
5. Implementation. During implementation, the affected products are "checked out" from the appropriate library and corrections made. Appropriate unit tests and integration tests must be determined and performed.
6. Release. Once the corrections have been made, they must be verified/tested at the appropriate software development level and/or CSCI testing depending on the level and type of change. The corrected products are reinserted into the baseline, and the products returned to configuration control. Following this they are ready for release to the acquirer.

5.18 Progress Reporting

During the 3DStress development, brief monthly project reports in the form of the Program Manager's Periodic Report will be produced by CNWRA.

5.19 Other Software Development Activities

The following sections describe the approach to be followed for other software development activities for the 3DStress application development project.

5.19.1 Risk Management, Including Known Risks and Corresponding Strategies

Areas of technical risk will be investigated as early in the development cycle as possible to allow adjustments in software design if required.

Schedule and cost problems are normally identified by use of CNWRA Project Manager data sheets. This control is currently being used in all projects. In addition, Project Managers hold timely project review meetings with all key project personnel to discuss, review, and solve schedule, cost, and technical problems.

The initial step in risk mitigation is identification of the risk, its potential impact on the project performance, and likelihood of developing into a problem. Risks are identified by careful review of all project aspects by analysis of the WBS. Risks are then tracked through the project until task completion to monitor their impact on cost, schedule and technical performance.

During each reporting period, work projections are made for the next reporting period, and costs are estimated. Progress for both performance and cost is evaluated against these projections. When progress does

not match projections, discussions are initiated within the project staff, and then with division management to resolve the problems, i.e., mitigate risks.

5.19.2 Software Management Indicators

The Project Manager will monitor the software management indicators listed below on an ongoing basis against the proposed project schedule and milestones.

1. Requirements volatility: total number of requirements and requirements changes over time.
2. Software staffing: planned and actual staffing levels over time.
3. Software complexity: complexity of each software unit.
4. Software progress: planned and actual number of software units designed implemented, unit tested, and integrated over time.
5. Milestone performance: planned and actual dates of key project milestones.

5.19.3 Security and Privacy

The 3DStress software application does not contain any extraordinary security or privacy issues (Section 4.2.4.2 & 4.2.4.3). This activity is tailored out.

5.19.4 Subcontractor Management

CNWRA does not plan to utilize subcontractors on the 3DStress development project. This activity has been tailored out.

5.19.5 Interface with Software Independent Verification and Validation Agents

CNWRA will utilize in-house staff for review of software quality issues and internal staff for verification and validation. This activity has been tailored out.

5.19.6 Coordination with Associate Developers

The 3DStress application will be developed using only internal staff. No other associated developers will be used. This activity has been tailored out.

5.19.7 Improvement of Project Processes

The Project Manager will periodically assess the processes used on the project to determine the suitability and effectiveness. Based on these assessments, the Project Manager will identify any necessary and beneficial improvements to the process, and identify these changes to the acquirer in the form of proposed updates to this Software Development Plan. All proposed changes will have acquirer approval prior to implementation.

5.19.8 Other Activities Not Covered Elsewhere in the Plan

CNWRA plans to utilize the consulting services of Dr. Alan Morris, University of Texas at San Antonio during the development of the 3DStress application. Dr. Morris is one of the original developers of the 3DStress algorithms and will be utilized as a resource for software requirements development and software validation.

6 SCHEDULES AND ACTIVITY NETWORK

Table 6-1 presents an overview of the significant milestones that will be completed during this project.

Table 6-1. Schedule of software development activities

Activity	Planned/Actual Completion Date
Software release v.1.1	August 2, 1996
Software release v.1.2	November 12, 1996
Software Requirement Document (v.1.3)	August 5, 1997
Software Planning Document (v.1.3)	July 13, 1998
Acceptance Testing	July 15,1998
Verification Testing (v.1.3)	July 15, 1998
Software Test Report	July 17,1998
User Guide to NRC (v.1.3)	June 29,1998
Software release (v.1.3)	August 12,1998
3DStress v.1.3 to NRC	August 14, 1998

7 PROJECT ORGANIZATION AND RESOURCES

The following sections describe the project organization and resources to applied to the 3DStress application development.

7.1 Project Organization

On a functional basis, CNWRA conducts programs under the Project Manager concept. The Project Manager is delegated authority for overall technical direction and administrative supervision of the project. The Project Manager reports directly to the Element Manager, who in turn reports directly to the Technical Director. This structure permits ready access to higher management to quickly resolve any problems which might arise. The quick access to management allows for close schedule coordination on projects of an interdivisional nature. Thus, once a project team is formed, the Project Manager has vertical line authority over team members for the duration of the project.

The support staff of CNWRA, including such functions as accounting, contract administration, purchasing, computer processing, report reproduction, library, and security, are at the disposal of the Project Manager. The direct availability of the support staff leads to effective project management and eliminates delays which might be experienced in a less flexible system.

CNWRA Quality Assurance reports directly to the CNWRA President. CNWRA QA performs audits of the software development process. CNWRA QA ensures conformance to contractual requirements and determines the adequacy and effectiveness of project activities.

Management controls are imposed by CNWRA to ensure progress and eventual delivery of end items in accordance with the agreed upon schedule and cost. Scheduling control is maintained by short interval updating of the approved schedule. As a minimum, bar chart project schedules with clearly defined milestones are prepared. The charts are divided into appropriate phases, tasks and, if needed, subtasks. These charts and work breakdown structures (WBS) are entered into a computer to facilitate monitoring and updating.

Cost status reports are prepared and distributed to project managers every two weeks at the close of the normal pay period. Labor data for these reports are obtained from individual time sheets which all employees are required to complete daily. Charges are listed by project number, as well as phase or task numbers. Itemized labor, materials, travel, reproduction services, and overhead for the preceding two-week accounting period are given, including commitments made which have not yet resulted in expenditures. Also, the balance of project funds available is noted. Every four weeks, a computerized summary of the two preceding biweekly reports is prepared and given to individual project managers.

The Project Manager has full responsibility for all software products created and/or utilized by the project. The data items, documentation reports, drawings, and manuals constitute project team activity paralleling the hardware and software development activities. The same team members performing the hardware and software tasks will also provide direct input and analysis for all data supplied on this contract.

7.2 Project Resources

The following sections describe the resources that CNWRA will apply to the 3DStress application development project.

7.2.1 Personnel

The software development team will be composed of software analysts experienced in the development of Silicon Graphics Open GL software. Approximately 0.2 FTE's and one summer employee will be committed to the software development team. The Project Manager is responsible for coordinating the activities of the software team project evaluations with the CQA department. Project management tasks will require approximately 0.2 FTE's.

7.2.2 Facilities

CNWRA will establish a development and testing environment for this project in the CNWRA GIS Laboratory, Bldg. 189. The environment will consist of one Silicon Graphics development workstation and three additional Silicon Graphics systems for testing and evaluation.

7.2.3 Acquirer Furnished Equipment, Data, and Documentation

No acquirer furnished software or equipment is required for this project

8 NOTES

8.1 Acronyms

CM	Configuration Management
CNWRA	Center for Nuclear Waste Regulatory Analyses
CSCI	Computer Software Configuration Item
HWCI	Hardware Configuration Item
IAW	In accordance with
CQA	CNWRA Quality Assurance
SDD	Software Design Description
SDF	Software Development File
SPR	Software Problem Report
SQA	Software Quality Assurance
SRD	Software Requirements Description
STP	Software Test Plan
STPr	Software Test Procedures
SUM	Software User's Manual
SVD	Software Version Description
SwRI	Southwest Research Institute
WBS	Work breakdown structure

8.2 Definitions

Acquirer

An organization that procures software products for itself or another organization.

Approval

Written notification by an authorized representative of the acquirer that a developer's plans, design, or other aspects of the project appear to be sound and can be used as the basis for further work. Such approval does not shift responsibility from the developer to meet contractual requirements.

Architecture

The organizational structure of a system or CSCI, identifying the components, their interfaces, and a concept of execution among them.

Associate Developer

An organization that is neither prime contractor nor subcontractor to the developer, but who has a development role on the same or related system or project.

Behavioral Design

The design of how an overall system or CSCI will behave, from a user's point of view, in meeting its requirements, ignoring the internal implementation of the system or CSCI. This design contrasts with architectural design, which identifies the internal components of the system or CSCI, and with the detailed design of those components.

Build

(1) A version of software that meets a specified subset of the requirements that the completed software will meet. (2) The period of time during which such a version is developed. Note: The relationship of the terms "build" and "version" is up to the developer; for example, it may take several versions to reach a build, a build may be released in several parallel versions (such as to different sites), or the terms may be used as synonyms.

Computer Hardware

Devices capable of accepting and storing computer data, executing a systematic sequence of operations on computer data, or producing control outputs. Such devices can perform substantial interpretation, computation, communication, control, or other logical functions.

Computer program

A combination of computer instructions and data definitions that enable computer hardware to perform computational or control functions.

Computer Software Configuration Item (CSCI)

An aggregation of software that satisfies an end use function and is designated for separate configuration management by the acquirer. CSCIs are selected based on tradeoffs among software function, size, host or target computers, developer, support concept, plans for reuse, criticality, interface considerations, need to be separately controlled, and other functions.

Configuration Item

An aggregation of hardware, software, or both that satisfies an end use function and is designated for separate configuration management by the acquirer.

Database

A collection of related data stored in one or more computerized files in a manner that can be accessed by users or computer programs.

Deliverable software product

A software product that is required by the contract to be delivered to the acquirer or other designated recipient.

Design

Those characteristics if a system or CSCI that are selected by the developer in response to the requirements. Some will match the requirement; others will be elaborations of requirements, such as definitions of all error messages in response to a requirement to display error messages; other will be implementation related, such as decisions about what software units and logic to use to satisfy the requirements.

Developer

An organization that develops software products ("develops" may include new development, modification, reuse, reengineering, maintenance, or any other activity that results in software products). The developer may be a contractor or a Government agency.

Document/documentation

A collection of data, regardless of the medium on which it is recorded, that generally has permanence and can be read by humans or machines.

Evaluation

The process of determining whether an item or activity meets specified criteria.

Firmware

The combination of a hardware device and computer instructions and/or computer data that reside as read-only software on the hardware device.

Hardware Configuration Item (HWCI)

An aggregation of hardware that satisfies an end use function and is designated for separate configuration management by the acquirer.

Interface

In software development, a relationship among two or more entities (such as CSCI-CSCI, CSCI-HWCI, CSCI-user, or software unit-software unit) in which the entities share, provide, or exchange data. An interface is not a CSCI, software unit, or other system component; it is a relationship among them.

Joint review

A process or meeting involving representatives of both the acquirer and the developer, during which project status, software products, and/or project issues are examined and discussed.

Non-deliverable software product

A software product that is not required by the contract to be delivered to the acquirer or other designated recipient.

Process

An organized set of activities performed for a given purpose; for example, the software development process.

Qualification testing

Testing performed to demonstrate to the acquirer that a CSCI or a system meets the specified requirements.

Reengineering

The process of examining and altering an existing system to reconstitute it in a new form. May include reverse engineering (analyzing a system and producing a representation at a higher level of abstraction, such as design from code), restructuring (transforming a system from one representation to another at the same level of abstraction), redocumentation (analyzing a system and producing user or support documentation), forward engineering (using software products derived from an existing system, together with new requirements, to produce a new system), retargeting (transforming a system to install it on a different target system), and translation (transforming source code from one language to another or from one version of a language to another).

Requirement

(1) A characteristic that a system or CSCI must possess in order to be acceptable to the acquirer. (2) A mandatory statement in this standard or another portion of the contract.

Reusable software product

A software product developed for one use but having other uses, or one developed specifically to be usable on multiple projects or in multiple roles on one project. Examples include, but are not limited to, commercial off-the-shelf software products, acquirer furnished software products, software products in reuse libraries, and pre-existing developer software products. Each use may include all or part of the software product and may involve its modification. This term can be applied to software product (for example, requirements, architectures, etc.), not just to software itself.

Software

Computer programs and database. Note: Although some definitions of software include documentation, MIL-STD-498 limits the definition to computer programs and databases in accordance with Defense Federal Acquisition Regulation Supplement 227.401.

Software development

A set of activities that results in software products. Software development may include new development, modification, reuse, reengineering, maintenance, or any other activities that result in software products.

Software development file

A repository for material pertinent to the development of a particular body of software. Contents typically include (either directly or by reference) considerations, rationale, and constraints related to requirements analysis, design, and implementation; developer-internal test information; and schedule and status information.

Software development library (SDL)

A controlled collection of software, documentation, other intermediate and final software products, and associated tools and procedures used to facilitate the orderly development and subsequent support of software.

Software development process

An organized set of activities performed to translate user needs into software products.

Software engineering

In general usage, a synonym for software development. As used in this standard, a subset of software development consisting of all activities except qualification testing. The standard makes this distinction for the sole purpose of giving separate names to the software engineering and software test environments.

Software engineering environment

The facilities, hardware, software, firmware, procedures, and documentation needed to perform software engineering. Elements may include but are not limited to computer-aided software engineering (CASE) tools, compilers, assemblers, linkers, loaders, operating systems, debuggers, simulators, emulators, documentation tools, and database management systems.

Software product

Software or associated information created, modified, or incorporated to satisfy a contract. Examples include plans, requirements, design, code, databases, test information, and manuals.

Software quality

The ability of software to satisfy its specified requirements.

Software support

The set of activities that takes place to ensure that software installed for operational use continues to perform as intended and fulfill its intended role in system operation. Software support includes software maintenance, aid to users, and related activities.

Software system

A system consisting solely of software and possibly the computer equipment on which the software operates.

Software test environment

The facilities, hardware, software, firmware, procedures, and documentation needed to perform qualification, and possibly other testing of software. Elements may include but are not limited to simulators, code analyzers, test case generators, and path analyzers, and may also include elements used in the software engineering environment.

Software transition

The set of activities that enables responsibility for software development to pass from one organization, usually the organization that performs initial software development, to another, usually the organization that will perform software support.

Software component/unit

An element in the design of a CSCI; for example, a major subdivision of a CSCI, a component of that subdivision, a class, object, module, function, routine, or database. Software components may occur at different levels of a hierarchy and may consist of other software units. Software units in the design may or may not have a one-to-one relationship with the code and data entities (routines, procedures, databases, data files, etc.) that implement them or with the computer files containing those entities.