

FINAL

**An Evaluation of the Software through Pictures/T Tool (StP/T)
for the Software Support Activity (SSA)**

November 28, 1994

Prepared By: Intermetrics, Inc.
607 Louis Drive
Warminster, PA 18974

Prepared For: Naval Air Warfare Center - Aircraft Division
Warminster, PA 18974

Contract Number: N62269-90-C-0412

CDRL B00J

Virgil Banowetz

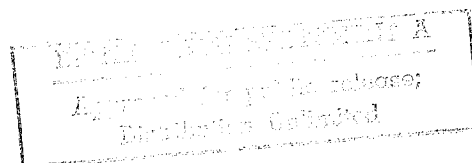
Virgil Banowetz Nov 28, 1994
Originator Date

Steve Nicolay

Stephen C. Nicolay Nov 28, 1994
Reviewer Date

DTIC QUALITY INSPECTED 1

19941214 032



FINAL

An Evaluation of the Software through Pictures/T Tool (StP/T)
for the Software Support Activity (SSA)

November 28, 1994

Prepared By: Intermetrics, Inc.
607 Louis Drive
Warminster, PA 18974

Prepared For: Naval Air Warfare Center - Aircraft Division
Warminster, PA 18974

Contract Number: N62269-90-C-0412

CDRL B00J

Accession For	
NTIS	CRA&I ☒
DTIC	TAB ☐
Unannounced	☐
Justification	<i>per ltr</i>
By _____	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

Virgil Banowetz

Virgil R Banowetz Nov 28, 1994
Originator Date

Steve Nicolay

Stephen C. Nicolay Nov 28, 1994
Reviewer Date



TABLE OF CONTENTS

1. Introduction	3
1.1 Background	3
1.2 Purpose of StP/T Study	5
1.3 Scope	5
1.4 Test Evaluation Approach	5
1.5 Report Overview	6
2. Applicable Documents	8
3. Tool Assessment Environment	9
3.1 Evaluation Exercise Environment.....	9
3.2 Display User interface	9
3.3. CASE Tools Evaluated.....	9
3.4. SSA Project Evaluation	10
4. StP/T Examples	11
4.1 StP/T Example 1, Calculate Weight Computer Experiment	11
4.2 StP/T Example 2, ASQ-212 Tactical Mission Software Experiment.....	11
5. Evaluation Findings	12
5.1 CASE Tool Product Continuity Issues	12
5.2 ASQ-212 Project Findings	12
5.3 StP Diagram Changes Needed to Support T.....	13
6. T Tool Problems and Limitations	15
6.1 Domain Increment Limitation	15
6.2 Inability to Generate Appropriate Test Cases for Characteristic Data	15
6.3 Inability to Generate Appropriate Test Cases for Array Data.	16
6.4 Miscellaneous T Problems	16
7. StP/Extract Limitation	17
7.1 No Non-primitive Extractions for Processes	17
7.2 StP/EXTRACT 4.2D Type Definition Error	17
8. StP Problems Encountered.....	19
9. Training Requirements for StP/Extract and StP/T	20
10. Test Documentation Preparation with StP/T Results.	21
11. SSA Software Tools for the Future	22

12. Conclusions And Recommendations	24
12.2 T Tool Evaluation Conclusions	26
12.2.1 T Stand Alone Application	26
12.2.2 StP/T Adaptation to Existing Programs.....	26
12.2.3 StP/T Application for New Programs	27
12.3 Overall Recommendations	27
13. Acronyms	29

APPENDICES

APPENDIX A. StP/Extract/T Installation, Setup, and Operation Procedures	A-1
APPENDIX B. Diagnostic Messages, Errors and Countermeasures	B-1
APPENDIX C. Questions and Answer Tips	C-1
APPENDIX D. Weight Computer Program Illustrations	D-1
APPENDIX E. ASQ-212 Samples	E-1
APPENDIX F. Documentation Errors	F-1

FIGURES

Figure 1-1.	StP/Extract/T Data Flow	4
Figure 11-1.	StP Product Support for T	23
Figure 12-1.	Weaknesses in StP/T Software Development	25
Figure A3-1.	Example of ToolInfo file for StP/Extract	A-5
Figure A3-2.	Example of TCONFIG file for T	A-6
Figure D1-1.	Example of a simple context diagram	D-3
Figure D1-2.	Verbose 0 level Data Flow Diagram (DFD) decomposition	D-4
Figure D1-3.	Terse 0 level DFD decomposition	D-5
Figure D1-4.	Preferred 0 level DFD decomposition	D-6
Figure D1-5.	Level 2 DFD	D-9
Figure D2-1.	Context Diagram Data DSD	D-13
Figure D2-2.	Local Data DSD	D-16
Figure D3-1.	Top level System Design Structure Chart Diagram (SCD)	D-19
Figure D3-2.	Second level System Design SCD	D-21
Figure D4-1.	System Analysis SDF for volume computer	D-24
Figure D4-2.	System Analysis SDF for mass computer	D-25
Figure D4-3.	System Analysis SDF for weight computer	D-26
Figure D4-4.	System Design SDF for top level weight computer (Main)	D-27
Figure D4-5.	System Design SDF for volume computer	D-28
Figure D4-6.	System Design SDF for mid level weight computer	D-29
Figure D4-7.	System Design SDF for mass computer	D-30
Figure D4-8.	System Design SDF for low level weight computer (by gravity) .	D-31
Figure E1-1.	Context Diagram for Navigation Steering	E-3
Figure E1-2.	Level 0 DFD	E-4
Figure E1-3.	Level 1 DFD	E-5
Figure E1-4.	Data Structure Diagrams for the components of process 1.1	E-7
Figure E1-5.	SDF file for process 1.1 (Determine Steering Mode)	E-27

TABLES

Table D4-1.	Figures for listings of SDF files for SA and SD	D-23
Table D5-1.	T output files for weight computer	D-33
Table E2-1.	T output files for a Navigation-Steering Process	E-28

EXECUTIVE SUMMARY

This report is the result of a technical evaluation of an automated test case generation tool, T Tool ("T"). The focus is on using T for Navy Software Support Activities (SSAs).

T is used to generate test cases based on input and output specifications derived from code, design, or even higher level software requirements. When T is supported by a CASE tool, the input and output specifications of the software under test are extracted by this tool which selects key information (units, range of values, etc.) about each process (requirement) or module (design, code) and creates a unique Software Description File (SDF). The SDF has all the information required to initially or regressively test each program element. T cannot execute tests, so after the SDF has been created, conduct of the tests can be performed using traditional application testbed/unit test methods or, ideally, using an automatic text execution tool which is compatible with the SDF output.

The Navy's P-3 Update III ASQ-212 program was used as a testbed for evaluating T primarily because the ASQ-212 program developer used a CASE tool which is integrated with T. The Integrated Development Environments, Inc. (IDE), Software Through Pictures (StP) Structured Engineering CASE tool was used for the requirements phase of the development. This product is one of at least two CASE tools presently integrated with T.

The T evaluation initially focused on evaluating its potential use as part of an on going SSA effort like the ASQ-212 program. Attempts were made to utilize T within the StP Structured Environment (SE) products used during the ASQ-212 requirements phase, followed by experiments using T with the StP Ada Design Environment (ADE) product set utilized during the design phase of the program.

Additionally, to determine the effectiveness of using T when integrated at the start of a new project, a sample program was developed which utilized the StP SE product line for the requirements (Structured Analysis (StP/SA)) and design (Structured Design (StP/SD)) phases. StP diagrams were produced with all T requirements included from the initiation of this sample program.

The results of this study yielded several important findings.

1. With the correct inputs, T performed as promised. However, in order for new programs to benefit from test tools like T, a commitment needs to be made from

program startup to utilize an integrated set of CASE tools which supports T. Without this foresight, there are two possible problems:

- The CASE tool may not generate the inputs required to run T, or
- Different CASE tools may be used for different parts of the life cycle (for example, requirements analysis and design). This approach would likely result in invalid test cases being generated by T.

The ASQ-212 project exhibited both of the above problems, neither of which was detrimental to ASQ-212 software itself, but they did limit the test readiness of the ASQ-212 as well as this evaluation of T. Unfortunately, one or both of these conditions is probably present on most Navy SSA programs. While manually "reverse engineering" CASE tool data to produce outputs which properly run on T is achievable, this extra effort can be very time consuming and costly.

New automated reverse engineering tools in the near future may alleviate this problem. These tools are being designed to take existing code which contains the necessary data specifications to support automated test case generation and regenerate a design baseline. This capability, along with the creation of new automated test execution tools which can operate on the T's test case outputs, would allow an SSA's software test program to be more efficient and achieve a greater level of product quality and reliability through more repeatable regression tests.

2. Automated test case generation tools cannot *totally* replace the current methods used to identify proper software test cases. Test tool limitations when dealing with some unique test requirements such as tests of data arrays or characteristic data will still require some level of manual test case definition. However, these manual test cases should account for only a small portion of the overall software test cases defined for a typical, large software program like ASQ-212.

In the absence of additional studies which validate fully integrated approaches to software development and testing, the procurement of the T Tool to generate test cases for an established Navy SSA can be recommended, but only with the above reservations.

1. INTRODUCTION

1.1 Background

This study was conducted by the Naval Air Warfare Center Aircraft Division (NAWCADWAR) to evaluate the Interactive Development Environments (IDE) software product known as the T tool. The evaluation was initiated at the request of NAVAIRSYSCOM AIR-54661 to determine the applicability of these tools to the SSA environment. The conduct of the evaluation spanned the time period from March 1994 through September 1994. A "Plan for Testing the T tool for SSA" was presented to and approved by AIR-54661 in April, 1994.

The T tool is a specification based test case generation tool. This means that a specification which describes the inputs and outputs of a process or module is the only basis for producing the test cases. T was previously evaluated by Christine Youngblut of the Institute for Defense Analysis (item h in Applicable Documents) who reported that StP/T is currently being used by various government organizations including the Naval Avionics Center, the Jet Propulsion Laboratories, Naval Coastal Systems Center and US. Army Forts Monmouth and Sill.

As illustrated in Figure 1-1, T can be used as a stand alone operation or as a product more tightly integrated into the overall software development process. For the T stand alone operation, the user types or edits specification data into a unique Software Description File (SDF) for input to the T tool. This SDF is a text file formatted in Semantic Transfer Language (STL), a superset of the IEEE 1175 standard language for software specifications. In modern Ada software developments which employ Computer Aided Software Engineering (CASE) tools such as Software Through Pictures (StP) for the Structured Environment (SE), StP supporting the Object Management Technique (StP/OMT), or Teamwork, a T interface is used to "Extract" the CASE tools own internal specification data and produce the SDF for T.

The CASE tool selected for this T tool evaluation was StP/SE (Structured Environment). StP/SE is currently being used for the requirements portion of the Navy's P3 Update III AN/ASQ-212 operational software development at NAWCADWAR. The SE product line of StP is also known as StP/STRAND and StP/ISE (Integrated Structured Environment). StP/SE can be used for both requirements analysis and design and has its "Extract" capability embedded in the StP graphical user interface for both functions. The StP Extract capability produces the SDF which is used by T to create test cases. The actual execution of test cases is traditionally implemented by a test harness, i.e, application testbed, and may include automated test execution tools to expedite repetitive and regression tests.

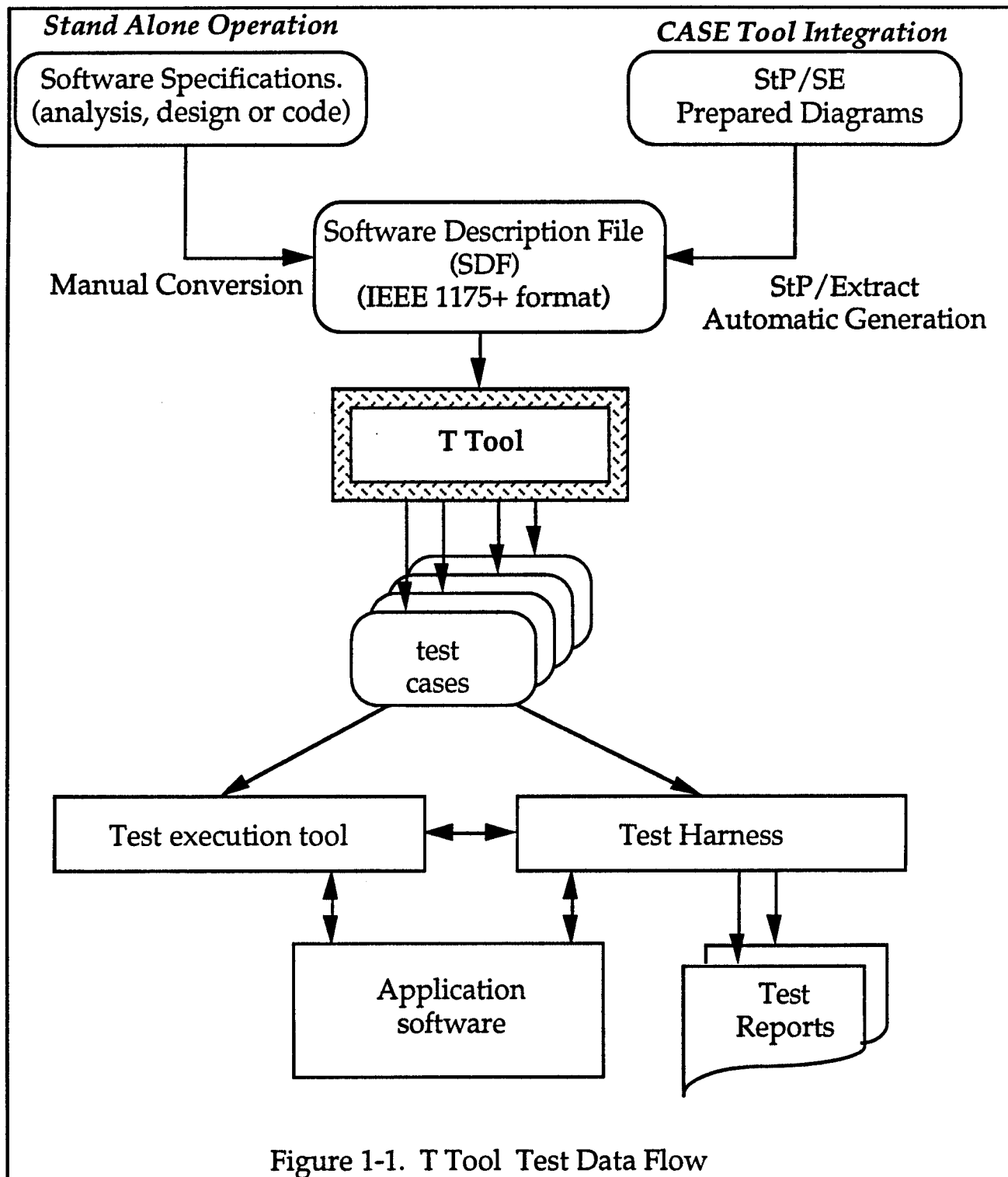


Figure 1-1. T Tool Test Data Flow

1.2 Purpose of StP/T Study

An efficient and effective test process is essential to assure software quality over the operational software life cycle. Much of SSA maintenance involves regression testing. Tools which make these tests faster, more thorough, more repeatable, and less costly, would be highly desirable for any SSA organization. The purpose of this study is to evaluate the capabilities of T and its potential utilization by Navy Fleet Software Support Activities (SSAs) to maintain Navy operational software.

1.3 Scope

This study offers a limited evaluation of the T tool capabilities, its ease of use, and adaptability within an existing SSA. Four evaluation experiments were conducted to evaluate the StP/T tool:

- a. T as a stand alone tool with the SDF data created manually from specifications defined in the ASQ-212 Software Requirements Specification (SRS). This experiment was performed to assess the general capabilities of T and the effort required to manually create the SDF.
- b. T supported by StP for SDF data which was created automatically from Systems Analysis (SA) diagrams recreated from the ASQ-212 SRS and enhanced to test ready form. This experiment was performed to evaluate the automatic SDF creation features of the StP/Extract interface with T.
- c. T supported by StP for SDF data which was created automatically from Systems Analysis (SA) diagrams from the actual ASQ-212 StP-based SRS diagrams enhanced to test ready form. This experiment was performed to evaluate the feasibility of using T as part of an established software product baseline.
- d. T supported by StP for SDF data which was created automatically from Systems Design (SD) diagrams. This experiment was conducted to illustrate how well SD would support T.

1.4 Test Evaluation Approach

This study was approached in four phases:

- a. Planning. The evaluator prepared a "Plan for Testing the T tool for SSA" which outlined the objectives and schedule.
- b. Training. A one week course was purchased by NAWCADWAR to train key personnel on T and StP/SE.
- c. T Tool Evaluation. The initial approach to evaluating the T tool focused on utilizing the current ASQ-212 software baseline which employs StP products

within their requirements analysis and software design process. The goal was to determine the T tool compatibility with the current StP based software and its potential use within the current ASQ-212 software life cycle process. When incompatibilities were encountered with creating the T tool SDF data directly from the ASQ-212 requirements and/or design data, the evaluation effort was redirected toward evaluating the tool based on a software program "model" specifically designed for the purpose of assessing the T tool capabilities to support automated test case generation.

d. Report Evaluation Results. The preparation and review of this document.

1.5 Report Overview

This evaluation report is organized into the following sections:

Section 1 provides the introduction to this report .

Section 2 lists the documents used and/or referenced for this study.

Section 3 describes the laboratory evaluation environment.

Section 4 describes the two main program examples used to conduct the StP/T study.

Section 5 summarizes the findings from the exercises of section 4.

Section 6 describes the problems encountered with T in the stand alone mode of operation.

Section 7 describes the problems encountered with T when integrated with StP.

Section 8 describes the problems encountered with StP independent of T.

Section 9 summarizes the training recommended for using T as part of the software test process.

Section 10 summarizes the test report generation capabilities of T.

Section 11 describes the T interface test tools projected to become available in the near future.

Section 12 presents the conclusions and recommendations from this study.

Section 13 offers a list of acronyms associated with this study report.

The appendices provide information to aide any ongoing evaluation of StP/Extract/T. They include procedures used for this study, problems encountered and their solution or work around, diagnostic aids, etc. as listed below:

Appendix A describes the installation, setup, and operation procedures for the software evaluated.

Appendix B provides diagnostic messages for errors and problems encountered in this study. Each item includes the appropriate advice and/or work around.

Appendix C is a list of tips in question and answer format for the operation of T and StP/Extract .

Appendix D documents the weight computer program experiment.

Appendix E documents the ASQ-212 sample experiment used for this study.

Appendix F lists IDE documentation errors encountered in this study.

2. APPLICABLE DOCUMENTS

- a. *T Users Guide*, Interactive Development Environments (IDE)
- b. *StP ISE 4.2D Users Guide*, IDE
- c. *StP ISE 4.2D Reference Manual*, IDE
- d. *Automated Software Testing*, Version 0.6, IDE
- e. *Application Note: Software through Pictures (StP) and T Integration*, IDE
- f. *T Installation Guide*, Installing T3.1 - Sun SPARC]
- g. *StP ISE Customizing the Environment* , IDE
- h. *An Examination of Selected Software Testing Tools: 1992*, Christine Youngblut, IDA Paper P-2769, Institute for Defense Analysis, December, 1992
- i. *Strategies for Real-Time System Specification*, Hatley and Pirbhai, Dorset House, 1988
- j. *Designing Large Real-Time Systems with Ada*, Nielsen and Shumate, McGraw-Hill, 1988.
- k. "Plan for Testing the T Tool for SSA", NAWCADWAR report, April, 1994.

3. TOOL ASSESSMENT ENVIRONMENT

3.1 Evaluation Exercise Environment

The StP/T evaluation exercises were performed on a Sun4 processor running with the SUN OS 4.1.3_U1 operating system.

3.2 Display User interface

The display user interface was the Openlook window manager. The following two display processors were used interchangeably at the NAWCADWAR VP Software Production Facility (SPF) :

- a. C shell on the Solaris operating system on SPARC station 10
- b. C shell on X terminal display driven by a SUN OS 4.1.3_U1 operating system on an IBM workstation.

3.3. CASE Tools Evaluated

The version of T evaluated in this study is T3.1. This tool has three functions:

- Tverify tests the SDF for errors.
- Tdefine generates the test cases.
- Tprepare formats the test cases for test execution.

Since test execution is not part of this study, only the first two above functions were evaluated.

The version of StP which was compatible with T and was evaluated in this study was the Integrated Structured Environment (ISE) version 4.2D . This product line is also known as StP/SE and Structured Analysis and Design (STRAND). Only StP functions relating to T were evaluated:

- Data Flow Diagram (DFD), the basis for Requirements/Systems Analysis (SA)
- Structure Chart Diagram (SCD), the basis for System Design (SD)
- Data Structure Diagram (DSD), supporting DFDs and SCDs
- Extract. This product is an interface utility embedded in the StP/SE graphical user interface to create Software Description Files (SDFs).

3.4. SSA Project Evaluation

The Navy P-3 Update III ASQ-212 program was selected as the representative SSA software project used to evaluate StP/SE and T. StP/SE was used by the ASQ-212 developer during the Systems Analysis phase for the purpose of preparing the Software Requirements Specification (SRS). Instead of using the System Design (SD) feature of the StP/SE for the design phase, another StP product, Ada Development Environment (StP/ADE aka StP/OOSD/Ada) was used for the design effort. The StP/ADE was used in lieu of StP/SE SD because (1) StP/ADE supports the generation of Ada compilable data specifications which was an essential component of the Software Design Document (SDD) and (2) ADE provided a more object oriented development methodology than SD.

4. StP/T EXAMPLES

4.1 StP/T Example 1, Calculate Weight Computer Experiment

The purpose of the calculate weight computer program experiment was to demonstrate a very simple end to end application of StP Structured Analysis, Structured Design, StP/Extract, and T for test case generation. This demonstration of StP/T capability allowed the evaluator to focus only on the tools since the application itself was trivial. To use the Calculate Weight Computer program, the user must enter five numbers (length, width, height, density, and gravity) and the system multiplies them and returns their product, the weight of a rectangular solid object. To illustrate multiple levels of diagrams, the multiplications are decomposed into components using intermediate data with familiar names such as volume and mass. The diagrams, extracted SDF files, and T results are shown in Appendix D.

The test cases generated were adequate to thoroughly exercise this simple model with both normal and abnormal data. For this example, the data required to make the StP diagrams test ready were not significantly more than that desired to do a complete unit level specification design.

4.2 StP/T Example 2, ASQ-212 Tactical Mission Software Experiment.

The purpose of the ASQ-212 software experiment was to demonstrate how and under what conditions T can be used to support testing of the current ASQ-212 software baseline which was developed with StP/SE and StP/ADE without previous consideration for the use of the T tool. The diagrams developed for the ASQ-212 software are of two types, (1) StP/SE Structured Analysis to support the Software Requirements Specification (SRS) and (2) OOSD/Ada to support creation of the Software Design Document (SDD). A few SE Structured Design diagrams also support the SDD but were not adequate to support T. The SRS and SDD are document specifications supporting the Department of Defense (DoD) 2167A standard. Sample Systems Analysis diagrams supporting the ASQ-212 navigation model experiment, the extracted SDF files, and the T results are shown in Appendix E.

The test cases generated were adequate to thoroughly exercise this unit with both normal and abnormal data. Input types for the module included data records, enumeration types, and boolean types. There were no input data arrays for this module. The data required to make the StP diagrams test ready was significantly more than that provided by the system analysis diagrams.

5. EVALUATION FINDINGS

5.1 CASE Tool Product Continuity Issues

When the StP Structured Design (SD) process follows Structured Analysis (SA), it is generally desirable to reuse the Data Structure Diagrams (DSDs) from SA for the SD. This provides some continuity between requirements and design. Similarly and more significantly, the Ada specifications supporting the Ada code should be based on the design DSDs to permit and encourage compatibility of StP based test cases with the code. In the absence of this continuity, high level specification based testing will have only limited utility. The present StP/T tools have a number of weaknesses which contribute to the SA/SD/Code continuity problem .

- a. A design tool which incorporates data structure diagrams like StP should use these data structures to generate code for data specifications. However, a flaw in the StP/SE SD process makes it unusable for the Ada code generation . Data specifications which are generated by SD are incomplete and, as a result, cannot be successfully compiled. The continuity from StP design to the actual Ada code is lost in StP/SE due to this incomplete SD capability. The code specifications are likely to differ from the design, thus preventing the generation of meaningful StP/T test cases.
- b. The StP Ada Development Environment (ADE) product line is a tool specifically created for Ada design. However, it is incompatible with StP/SE due to the fact that ADE uses a different methodology known as Object Oriented Structured Design (OOSD). When StP/SE SA is used for requirements analysis and StP/ADE is used for the design, the continuity from requirements to design is lost. This discontinuity stems from the incompatibility with the Data Structure Diagrams between the two StP product lines. When ADE is used, T cannot generate valid CASE based test cases. Specifically inhibited are
 - (1) requirements based testing (the code is unlikely to match the requirements data), and
 - (2) design based testing (StP/ADE has no T interface).

5.2 ASQ-212 Project Findings

The use of the Navy ASQ-212 project as a test baseline for evaluating T illustrates some problems which are probably typical of any Navy SSA which did not consider the use of automated test tools as part of the overall test program from project conception.

As part of the requirements based testing experiment using T, the SRS and SDD for the Navigation Steering component were compared to determine compatibility. At the time of this tool evaluation, the ASQ-212 program is nearing the formal Navy test phase. As often encountered with many large, complex software developments, the requirement and design documentation has lagged behind the code. Also, to reduce cost and technical risk, the DoD documentation standards were tailored to eliminate

data duplication and simplify the overall configuration management effort. At the time of this T tool evaluation, the SDD had no Data Structure (missing from Appendix E). It was also missing from the electronic version of the diagrams. Furthermore, the data names used in the SDD diagrams have little resemblance to the data names in the data flow diagrams used for the SRS. Because of this lack of continuity, attempting to use T for StP SA/requirements based testing was found to be unproductive. Section 11 offers a discussion of some possible test alternatives for using T following after the initial program development phase.

5.3 StP Diagram Changes Needed to Support T

The StP diagrams from the current ASQ-212 program would require considerable changes to support automated StP/T based testing. In the Navigation module experiment, the System Analysis diagrams required extensive changes in the following areas to make them T tool compatible:

- a. Corrections for diagram error detected by StP
- b. Dataitem type revisions in Data Structure Diagrams (DSD) annotations.
- c. Datatype definition additions and revisions in DSD annotations
- d. DSD revisions of literal types to selection/enumeration diagrams.

In a typical software development effort, errors and inconsistencies in StP diagrams produced early in a project are likely to remain well into the test preparation stage which occurs later in the development. Even though StP detects some inconsistencies, or incompleteness, these errors may not get resolved prior to the test preparation phase. The current ASQ-212 Tactical Mission Software (TMS) Navigation SA diagrams have some errors detected by the StP "check data dictionary" function. The resolution of these errors may or may not be necessary to produce test cases, but the errors suggest that the errant diagrams are incomplete or out of date. At the very least, the presense of these errors raises questions about the applicability of testing at this stage.

Primitive dataitems (those not decomposed into components) must be properly typed in the DSD annotations or they will be flagged as undefined by StP. However, T requires that only specific types be used. These types must be either (1) a datatype defined in a Data Structure Diagram (DSD) or (2) one of the datatypes defined in an include file such as stp.std or tsdl.std (which already have the values/domain defined). In many cases the mapping of types provided to the correct types will not be easy. For example, in the ASQ-212 SA diagrams, the type "numeric" is often used. For T to operate properly from extracted data, these "numeric" types must be replaced by the true type based on the actual design/code. These new types must be defined in the DSDs or in text files *included* by the SDF.

Datatypes defined in the diagrams (or in a user defined *include* file similar to stp.std) for the primitive dataitems must include the value(s) or domain (Minimum, Maximum, and Increment) needed for testing. The domain values are used to generate selections

for values for a numeric (integer or real) dataitem. Also, the base type for these datatypes is restricted by T to integer, real, boolean, string, or character. These base type names must be in lower case to be recognized by the StP/Extract program.

When the type "literal" as used in an ASQ-212 SA diagram, it must typically be changed to a selection structure with the values provided for the literal dataitem used to name the selections. This supports the Ada *enumeration* type which is used by StP/T and Ada. Also some uses of the string and literal types might need to be changed to boolean types depending on the code.

The above special typing has not, for the most part, been required of the developer for creating the ASQ-212 program SA and SD StP diagrams, thus, it will not in most cases be available for T utilization. To date, testing has been separated from the requirements and design activities which utilize the StP tool set. A code tester wishing to use the diagrams for generation of test cases with T currently has to manually enter this test data into the diagrams based on a review of the code.

It is anticipated that the ASQ-212 SA diagram documentation will "catch up" to the actual code at final delivery. Even so, if T were to be utilized for requirements based testing in future revisions of this program, a conscious effort would have to be made to "raise" the level of the data specifications in the SA diagrams to a T tool compatible state.

6. T TOOL PROBLEMS AND LIMITATIONS

During the evaluation of T using the Calculate Weight and ASQ-212 software experiments, several shortcomings were identified with StP/T which affected the generation of test cases. These limitations are provided in the following paragraphs.

6.1 Domain Increment Limitation

The domain specification is part of the data input specification used by T for picking sample integer or real data for test cases. It consists of a minimum, maximum, and increment. However, the domain increment specification may not be adequate. It is used by T to generate test cases with values for items which are above and below the minimum and maximum by this amount. However, by using the same increment about both minimum and maximum, we may not produce the best debug cases. If the maximum is very large, say 1E10, and the increment is 1, an increment of 1 part per 1E10 about the maximum may be too small to produce a unique debug result (different from the maximum case). This same increment applied about a small minimum (say 0) may produce a debug result too different from the minimum result. Separate low and high increments should be allowed in T and StP to produce acceptable debug test cases. This recommendation is being considered by the T tool developer, IDE.

6.2 Inability to Generate Appropriate Test Cases for Characteristic Data

Characteristic data are special values which are needed for proper testing. Acoustic signature characteristics, for example, must be tested with a set of comparison data which both matches and fails to match the arrays of records of characteristics in a complex pattern. T, however, cannot produce test cases for characteristic data except for the simple case where the data item is essentially an enumeration variable with characteristic values provided. Here, individual values of variables are defined for testing purposes within the SDF. However, this does not assure that the values selected will be combined with other variables in a desired characteristic way for testing. The means for dealing with this problem are as follows:

- Manually add test cases to supplement the automatic test cases for characteristic data.
- Represent the characteristic data in arrays via the test harness as described in the paragraph below.
- The TSDL syntax could be enhanced by the vendor to permit both "has value range" and "has values" clauses for the same variable. This would permit independent characteristic values along with a domain specified range of values for the same variable type. However, it would not test characteristic data combinations.

6.3 Inability to Generate Appropriate Test Cases for Array Data.

Like the characteristic data limitation above, T cannot generate adequate test cases to represent Ada data arrays. Data to represent these lists of data must be prepared manually. These supplementary array data will need to be incorporated into the test harness for test execution. No tool enhancement is anticipated for this limitation.

6.4 Miscellaneous T Problems

There is no index section to the T Users Guide, no annotated list of T produced error messages, and no T Reference Manual. Also, there is no documented list of keywords (types etc.) for the T Software Description Language (TSDL). In the absence of the list of keywords, the StP user may use these keywords for data names. This can confuse the tester or at least be an annoyance since it causes nuisance warning messages when Tverify is run.

Diagnosis of problems is somewhat disorderly due to lack of documentation, especially a good index. The support service at IDE is very helpful and typically responds to email questions within a few days.

7. StP/EXTRACT LIMITATION

During this study, several limitations were discovered in using the StP/Extract tool to create the SDF required for T execution. The below paragraphs discuss these limitations.

7.1 No Non-primitive Extractions for Processes

The 4.2D version of StP does not permit extraction of SDF files for DFD processes (SA) which are not primitive. Test cases are only produced for the lowest level of requirement process. For example, the volume/density/gravity multiplier process in Figure D1-4 will not have an extracted SDF since it is not primitive. In Figure D1-5, this process is decomposed into two component processes which can have SDFs extracted. Higher level processes (including the entire model) cannot have SDFs extracted and therefore are not testable without manual preparation of an SDF. This defect is corrected in StP version 5.0.

7.2 StP/EXTRACT 4.2D Type Definition Error

The TSDL language provides for the use of text files separate from the SDF for the purpose of defining commonly used types for T dataitems. This is done by an INCLUDE statement within the SDF. The files stp.std and tsdl.std are examples which are typically included. These files are provided as part of the T package. For the stand alone T operation, the include statements in the SDF can be written to use type definition files such as these. However, the extracted SDF files from StP version 4.2D have an error which makes use of these type/include files cumbersome. Only stp.std defined types are usable without a manual edit of the extracted SDF file. For the 4.2D extract, a type in a user defined file or tsdl.std is usable in the dataitem annotations only if you manually edit the extracted SDF file to eliminate the spurious "undefined" statement associated with this type. This edit is necessary due to a 4.2D extraction program bug. Only StP DSD defined types, stp.std defined types, and the five base types described in the Application Notes are recognized by StP Extract. The use of any other type for a dataitem generates an "undefined" statement whether the type (1) is defined in tsdl.std, (2) is defined in a user defined include file, (3) is misspelled, or (4) is otherwise undefined.

For an include file defined type (cases 1 or 2), the spurious statement in the SDF causes an error when running Tverify. Since the type is defined in an included file, it results in a duplication error (two specifications for the type, one saying it is undefined and one defining it).

T will not create test cases of data items using a misspelled/undefined type (cases 3 and 4). The presence of the string "undefined" in the SDF is an error flag... an indicator of an action item.

Aside from the above StP/Extract bug, it should be noted that the use of a customized include file requires only a simple edit of the extracted SDF file. Initially, the extracted SDF only includes the delivered stp.std which in turn includes tsdl.std.

According to Bob Poston at IDE (the T creator), the problem has been fixed for StP/SE 5.0. Include/type files like stp.std can be created or modified and types therein are not flagged as undefined by the extraction program. For the 4.2D user, application types are best defined in the StP diagrams rather than in include/type files.

8. StP PROBLEMS ENCOUNTERED

There is an occasional problem with the project and system/model function of StP. For illustration, the StP/T application software package includes an elevator example which has a problem loading. StP generates an error when it is brought up with the defaults set as follows:

```
system=elevator  
projdir=/tools/cots/sunos.4.1.x/T3.1/t_int
```

The error is "Project not found".

Resolution: Unresolved. Problem has not appeared in other areas. Reported to IDE Tech Support 8/25/94.

9. TRAINING REQUIREMENTS FOR StP/EXTRACT AND StP/T

The recommended training for using StP and T has typically been one to two weeks of dedicated on-site classes. The costs for this is in the order of \$10,000 per week for a class of up to four people, not counting the cost of the student's time. However, after someone is trained at a site, the use of these personnel and available documentation is sufficient to train others on site. Formal in-house training might be limited to one hour per day for a month supplemented by hands on experience.

The most difficult part of learning T is the language of the SDF, the T Software Description Language (TSDL). The easiest way to learn TSDL is to use StP/SE to prepare a sample program and generate the SDF. The TSDL easily becomes an understandable language representing the software process or module specifications.

10. TEST DOCUMENTATION PREPARATION WITH StP/T RESULTS.

Test documentation under DOD-STD-2167A which can be provided by StP/T includes the Software Development Files (SDFs) and portions of the Software Test Descriptions (STD) and the Software Test Report (STR). The SDFs document the Computer Software Unit (CSU) tests. These SDFs can be produced by the input and output of StP/T and the test harness. The STD documents the Formal Qualification Tests (FQT) of the Computer Software Components (CSC) and the Computer Software Configuration Item (CSCI). Much of the STD report can be created by the Tdesign report file, samples.rpt, generated for each design module designated as the CSCI or a CSC. This report lists sample values for each variable for a test unit. In this way samples.rpt encapsulates the totality of the test cases without actually itemizing them individually.

To supplement the test cases for special characteristic data, the sample values produced by T can be controlled by editing the T input. For T stand alone operation, edit the Software Description File. The samples associated with a numeric datatype can be defined by entering values in a "has values ...,..." clause for that datatype. This is appropriate when specific (characteristic) values within the range are significant. The "has value {minimum, maximum, resolution}" clause is used when there are no characteristic values to test and the full range of values should control the samples. For StP/T testing, the above can be accomplished by using the "Value" annotation instead of "Domain" annotation for the datatype in the StP Data Structure diagrams.

To supplement the test cases to provide a variation on array data, the tester must provide auxiliary data (not provided by T) to support the test harness. This input will supplement the SDFs and the STD.

In addition to the above, the STD will need to be supported (electronically only for large systems) by the individual test cases and their expected results using another Tdesign report file, summary.rpt. The expected results must be manually edited into this file since there is no automaton for expected values.

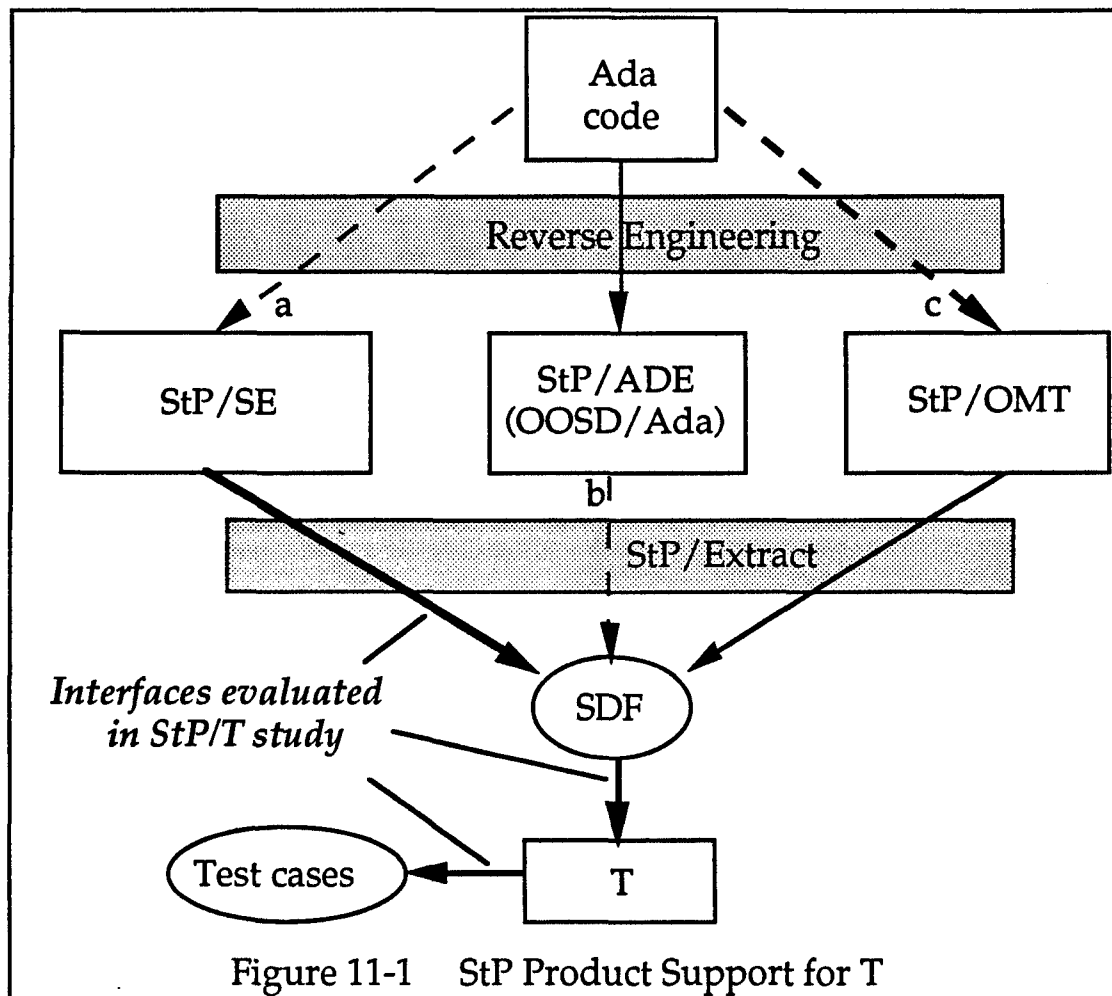
The STR can be automated to some extent by using the Tdesign report file, summary.rpt, supplemented by test execution reports and differencing reports.

11. SSA SOFTWARE TOOLS FOR THE FUTURE

Upgrades to both T and StP are now available. The features of the new 4.0 T version are reportedly easier to use than the present version (T3.1). The StP upgrade from 4.2D to 5.0 has resolved the problem described in Section 7.2. The NAWCADWAR has plans to upgrade StP to 5.0 early in 1995.

Projecting software tool development for the next year or so, IDE has three approaches under development or consideration to provide automated means for using T to test Ada code. Figure 11-1 shows the available (solid line) and potential (dashed line) interfaces connecting Ada code with T. One interface effort is currently under development by IDE and is represented by the heavy dashed line. Excluding the manual and non IDE interfaces, the potential developments connecting Ada code with T are as follows:

- a. The Ada based Design Approach for Real Time Systems (ADARTS), a product of the Software Productivity Consortium (SPC), and IDE are considering a capability to *reverse engineer* Ada code to StP/SE diagrams (Presently the IDE reverse engineering capability produces only StP/ADE, i.e., OOSD diagrams). Since SE format is compatible with T, test cases could be generated. ADARTS is currently under development as a methodology and protocol for enhancing software development productivity.
- b. IDE is also considering a more direct (*extract*) T interface from StP/ADE design diagrams. StP/ADE would extract SDF files from these diagrams for use by T to generate test cases. (similar to the present StP/SE/Extract).
- c. Another recently released StP product line is Object Modeling Techniques (OMT). StP/OMT supports a very popular design methodology known as Object Oriented Modeling. It has recently been given a T interface (Extract). The facility to reverse engineer from Ada code is under development and is expected to be released in mid 1995. OMT is expected to replace OOSD as a design methodology on new projects. This product was not purchased for evaluation, but is advertised to produce a link between Ada code and the T tool.



12. CONCLUSIONS AND RECOMMENDATIONS

12.1 The Need for Vertical Integration

The degree of vertical integration between the requirements, design, and code phases will significantly effect the degree of data correspondence between them and the usability of both requirements based and design based test cases. Where the code units do not correspond to the requirement processes, i.e, Data Flow Diagrams (DFDs), requirements based test cases will be nonfunctional. Where the code units do not correspond to the design modules, i.e., Structure Chart Diagrams (SCDs), design based test cases will be nonfunctional . Where the data structures used by the code are not based on the StP Data Structure Diagrams (DSDs), all StP/Extract based test cases generated from these diagrams will be nonfunctional .

Automatic specification generation is a desirable StP feature which can lead to (or at least encourage) this data correspondence between design and code. The generated data specifications from the DSDs defined by StP should be compileable and usable for the code effort. StP/SE does NOT generate compileable Ada specifications.

Figure 12-1 illustrates the critical interfaces in a typical StP/SE development under the 2167A standard. The traditional outputs of the Systems Analysis and System Design phases are documents, specifically the Software Requirements Specification (SRS) and the System Design Document (SDD), respectively. If a software development program is to benefit from the utility of test tools like T, this is no longer sufficient. A vertically integrated approach is essential to assure test tool compatibilities and efficient data reuse within the project.

The StP Data Structure Editor has a specification generation facility which creates files to represent the data specification. However, the resulting Ada specifications are incomplete and not compileable. Also, one might expect that the minimum and maximum in the domain specifications used for testing could also be used for the type declarations in the Ada specification. This, however, is not the case for the current version 4.2D of StP. Since the domain data is not usable for anything but testing, one can reasonably expect that this test information will only be incorporated into the diagram by the developer if the developer plans to use T for his own internal testing or if contractually required.

The issue of SA (or even SD) inadequacy due to dissimilar code is expected to be far from unique to the ASQ-212 program . For StP/T to be useful, a discipline will need to be followed to assure StP data sufficient for generation of automated test data. The most sensible source for StP/T data is the design diagrams since these are a big step closer to the code which will be tested. The maintenance of design processes (Structure Chart Diagrams and Data Flow Diagrams) to correspond to code throughout the life cycle is important for both testing and software maintenance.

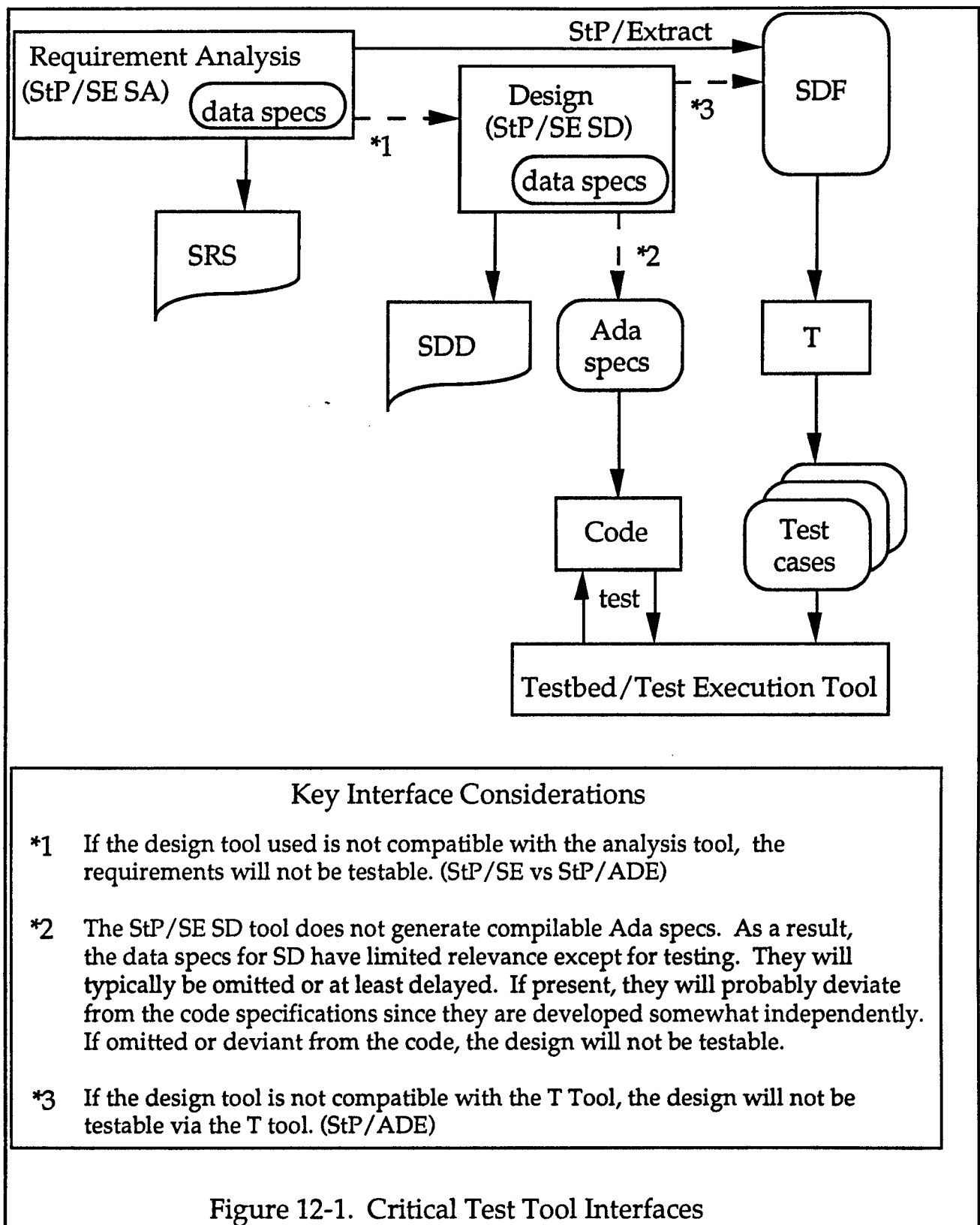


Figure 12-1. Critical Test Tool Interfaces

The maintenance of requirement processes (Data Flow Diagrams) ONLY to correspond to design/code throughout the life cycle may also be important. In *Strategies for Real-Time System Specification*, Hatley and Pirbhai stress the importance of iterative development throughout the life cycle. The requirements are not finished when the design phase starts. With the arrival of modern software test tools, it is essential to reduce the gap between the requirements and design baselines throughout the code evolution phase.

12.2 T Tool Evaluation Conclusions

12.2.1 T Stand Alone Application

The use of T in the stand alone operation has merit for unit testing, CSC testing, CSCI testing, and integration testing, even though its limitations in dealing with arrays, the increment problem, and characteristic data will require case by case resolution at the test execution level. The T Software Description Language (TSDL) could originate from the design or the Ada specification code. Data, procedures, and functions could be taken from specifications and edited into TSDL form to build the SDF file manually. This method has been used successfully by the Longbow Guidance Section Test Bench Program of the US. Army for test case generation. The test support software group for this program estimated their cost savings to be 89% for test case generation.

T in the stand alone operation is easy to use. While it is initially somewhat difficult to understand the TSDL syntax, it is an English like language which can be learned by a combination of formal training, reading, and hacking. T performs as described in the T user documentation.

12.2.2 StP/T Adaptation to Existing Programs

The problems encountered while trying to evaluate T with the P-3 Update III ASQ-212 software are probably typical of problems which might be encountered when trying to adapt T to other existing programs. The combination of mixing StP product lines, tailoring documentation requirements, and utilizing a development methodology which does not support a tight coupling of specification data between development phases can limit a straightforward retrofit implementation of test tools like T. However, this does not mean that an existing project cannot take advantage of automated testing after initial program development.

Augmenting an SSA's test program with automated test case generation tools and, possibly, automated test execution tools can have a positive impact on the overall quality and efficiency of the software test program for software updates in the future. IDE is in the process of developing a reverse engineering tool which will take previously developed Ada code and generate the appropriate OMT design diagrams which will support test case generation using T. This would permit an existing Ada program like the ASQ-212 software to take advantage of automated test tools by reverse

engineering the applicable code into T compatible OMT design structures for test case generation.

12.2.3 StP/T Application for New Programs

T used along with StP could be a valuable test aid when a developer is required to enter the appropriate type and value/domain information in the data annotations, and prepare diagrams which will support both code development AND testing. The early integration of test data in diagrams for use by StP/T is a viable method for enhancing and more tightly integrating the software development and test processes.

A very important requirement for specification based testing is that the code match the specifications. If the design specification is the basis for test, the Navy must require software designers to use a software design tool which (1) can generate usable Ada specifications for the code *and* (2) can interface with a test case generation tool such as T. This ties the design, code, and test data together which reduces redundancy and makes automatic test case generation feasible.

The StP/SE product line evaluated as support for the T tool does not adequately support the above proposed integrated approach because its data structures do not translate to usable Ada specifications. Further studies are proposed for evaluating newer, more advanced tools such as StP/OMT which will have both a T interface and an Ada reengineering capability.

12.3 Overall Recommendations

Even in the absence of a validated, fully integrated CASE tool environment for software development and testing, the use of the T tool for an SSA still looks promising. At this time, its use for existing programs can be expected to be in the stand alone mode with little or no support from available StP diagrams. However, T's full utility cannot be assessed without additional hands on experience with an actual real time program which has an operational testbed and an automatic test execution tool. A follow on evaluation of T with a test execution tool such as Software Test Works by Software Research, Inc. , is recommended to build upon this initial evaluation of T for SSA utilization.

For the front end interface with T, the capabilities of OMT need to be evaluated including reverse engineering (when available), design, code creation, and extract.

For new software programs which are considering Ada CASE tools, the following capabilities should be heavily considered in the CASE tool selection decision in order to use the T tool:

- Is the tool set vertically integrated?
- Does the design tool generate compileable Ada specifications?

- Does the tool incorporate test ready data which can be extracted for testing?
- Is there a reverse engineering capability?

With T and the proper CASE tool support for a new software development, the required test effort can be made more effective and less costly.

13. ACRONYMS

ADARTS	Ada based Design Approach for Real Time Systems
ADE	Ada Development Environment, a StP tool implementing OOSD notation
ASQ	The designation for the VP 212 project hardware upgrade
ASW	Anti - Submarine Warfare
aka	also known as
CASE	Computer Aided Software Engineering
CFD	Control Flow Diagram (Supporting SA for SE methodology)
CSC	Computer Software Components
CSCI	Computer Software Configuration Items
CSE	Control Specification Editor (an StP capability)
CSU	Computer Software Unit
DBMS	Database Management System
DED	Data Entity Editor (an StP capability)
DFD	Data Flow Diagram (Supporting SA for SE methodology)
DFE	Data Flow editor (an StP capability for a DFD)
DoD	Department of Defense.
DPS	Data Preparation System (an StP capability)
DSD	Data Structure Diagram (Supporting SA/SD for SE)
DSE	Data Structure Editor (an StP capability for a DSD)
FQT	Formal Qualification Tests
FTP	Fly To Point.
IDA	Institute for Defense Analysis, Alexandria, VA
IDE	Interactive Development Environments, Inc. (owner of StP and T)
IEEE	Institute of Electrical and Electronics Engineers
ISE	Integrated Structured Environment (Same as SE)
NAWC	Naval Air Warfare Center.
OMT	Object Modeling Technique, a new StP product line built on the Core
OOSD	Object Oriented Structured Design , a StP product line
PAT	Process Activation Table (Supporting SA methodology)
RFP	Request for proposal.
RT	Real Time
RTD	Run Time Data
RTM	Requirements Traceability Management (A StP associated tool)
SA	Structured Analysis. The StP SE (STRAND) methodology for requirements
SCD	Structure Chart Diagram (Supporting SD methodology)
SCE	Structure Chart Editor (an StP capability for SCDs)
SD	Structured Design. The StP SE (STRAND) methodology for design
SDD	Software Design Document
SDF	Software Description File, a StP/T creation from IDE
SDF	Software Development Files (2167A term)
SE	Structured Environment, the designation of the StP STRAND ,i.e., ISE product line
SPARC	Scalable Processor Architecture (Sun).

SPC	Software Productivity Consortium
SPF	Software Production Facility at NAWCADWAR
SRS	Software Requirements Specification
SSA	US Navy fleet operational Software Support Activity
STD	Software Test Descriptions
STE	State Transition Editor (an StP capability)
STL	Semantic Transfer Language, IEEE standard 1175, 1991
STR	Software Test Report
STRAND	Structured Analysis and Design. The StP/SE product line, aka ISE
StP	Software through Pictures, a set of product lines from IDE, Inc.
TMS	Tactical Mission Software
TSDL	T Software Description Language, a superset of STL

APPENDICES

The appendices which follow provide information to aide any ongoing evaluation of StP/T. They include procedures used for the study, problems encountered, their solution or work around, diagnostic aids, etc.

APPENDICES

The appendices which follow provide information to aide any ongoing evaluation of StP/T. They include procedures used for the study, problems encountered, their solution or work around, diagnostic aids, etc.

APPENDIX A. StP/Extract/T Installation, Setup, and Operation Procedures

A1. StP/Extract Installation

Refer to the "StP Installation Guide" provided with the StP delivery for the 4.2D installation. NAWC has no formal installation instructions for the associated StP Extract software at this time. All extract installation files have been put in the directory /staff/imet/banowetz/S TP/ISE-T-integ/ which includes the file S TPmenu.spec and two sub directories t_int and ann_tmpl. These constitute the interim StP/Extract installation until formal installation of these files onto protected directories. The three are referenced in the users StP/Extract ToolInfo file as follows:

The S TPmenu.spec file corresponds to the S TPMenuSpec variable.
The tint sub directory must be included in the PATH definition.
The ann_tmpl sub directory must be used to set the annot_path StP variable.

A2. T Installation

Refer to the "T Installation Guide" provided with the StP/T delivery. The T installation directory is

/tools/cots/sunos.4.1.x/T3.1

which is available if logged into guinan (or crusher or troi if man (help) pages are not needed).

A3. StP/Extract User Directory Setup

Create a local directory for your customized StP related information. In this directory, create a customized StP/Extract "ToolInfo" file. A formal approach to this would be to start with a copy of the delivered ToolInfo in

/tools/IDE/libsun4/

and delete all lines not to be customized. Then add a final line which includes (by reference) the original file in its original location with the line

ToolInfo =/tools/IDE/libsun4/ToolInfo

This reference will supplement the StP ToolInfo data not in the modified file. Further instructions for customizing this file can be found in the StP Reference Manual.

As a shortcut for the above, copy the ToolInfo file used by another user of StP/Extract and customize at least the following items:

projdir [default project for StP]
system [default system/model for StP]
S TPMenuSpec [the file for menus supplemented for Extract]
PATH [to include the StP/Extract software installation directory]
annot_path [to the directory containing all StP annotation
templates including Extract annotations.]

An example of a user ToolInfo file for StP/Extract is ToolInfo.ISE-Tbase in /staff/imet/banowetz/t_test/ which is displayed in Figure A3-1.

Set up an alias (in your alias setup script, i.e., .cshrc) to set the environmental variable ToolInfo to the above ToolInfo file and run the script. This is to easily switch to the StP/Extract capability from the StP/T capability. For example:

```
alias setSTPE "setenv ToolInfo <full path to customized file> "
```

Run the alias setup script to activate the alias.

Set up a StP project directory which will be the parent directory of your StP system models. Each of these system models will eventually have its own file structure of subdirectories containing a database, the StP diagram directories, and SDF directories.

A4. T User Directory Setup

The following steps will set up your account for running T:

1. Create a local directory for your customized T related information.
2. Copy the tconfig.ini file from the T installation directory
/tools/cots/sunos.4.1.x/T3.1
to this directory. This file is customized as described in step 5 below. A typical "TCONFIG" file is shown in Figure A3-2.
3. Set up an alias (in your alias setup script) to (1) set the environmental variable TCONFIG to the TCONFIG file above, and (2) set the environmental variable ToolInfo to the delivered ToolInfo file in the T installation directory . This is to easily switch to the StP/T capability from the StP/Extract capability. This alias must be written in one line, for example, by appending the following two lines:
alias setT "setenv TCONFIG <full path to customized file> ;
setenv ToolInfo /tools/cots/sunos.4.1.x/T3.1/ToolInfo"
4. Run the alias setup script to activate the alias.
5. To customize the "TCONFIG"file, use the T help facility in T or StP to list the HELP information on configuration:

From T:

After the T directory setup above is complete, enter

`t3 -h`

and respond to the menu to obtain Customizing Defaults information.

Edit your TCONFIG file accordingly.

From StP:

After the StP directory setup above is complete, enter

`StP &`

From the Main Menu, select "T integration" and "T3 help".

Execute and enter a "c" selection for Customizing Defaults.

Edit your TCONFIG file accordingly.

A5. StP and Extract Operation

If man (help) pages are needed, log into guinan. In the man pages are not needed, crusher or troi can be used as processors to run StP.

Execute the StP/Extract setup alias prepared above.

Change the default project and or system/model in the ToolInfo file if desired.

Run the StP in the background by entering

`StP&`

Refer to the StP Users Guide of version 4.2D for operation instructions.

A6. T Operation

Log into crusher of any system with direct access to the T installation directory.

Set the TCONFIG and ToolInfo environmental variables by executing the StP/T setup alias prepared above in T User Directory Setup.

To run Tverify, go to the directory where the SDF file is and enter

`t3 -v<sdf>`

where <sdf> is the name of the SDF file (not preceded by a space).

To run Tdesign from the data in TestUnitDir as defined in your TCONFIG file, enter

`t3 -d`

To run both Tverify and Tdesign, go to the directory where the SDF file is and enter

`t3 -v<sdf> -d`

Refer to the T Users Guide for operation instructions. Also use the help text in T itself (t3 -h). The file catalog.txt in the T installation directory is a valuable aide for TSDL syntax matters.

```

ToolInfo.ISE-Tbase  Mon Sep 26 17:06:13 EDT 1994
% ToolInfo.ISE-Tbase in /staff/imet/banowetz/t_test
% Used by .stp_t to build ToolInfo.ISE-T to support StP/T
% Do NOT edit ToolInfo.ISE-T directly.

stp_product=/tools/IDE/templates/ide_stp_product.dat
%-----
% Project/System environment
%-----
% Initially project directory is the current directory
projdir=/staff/imet/banowetz/STP
% projdir=/tools/cots/sunos.4.1.x/T3.1/t_int
% does not work !!!!!!!
% Initially project database is enabled;
pdb_enabled=1
% This is the same as specified by the -s argument on the command line
% or the ToolInfo attribute system
system=vb212
% alt values above are init_system, and apartment.  The above value is
% used to edit by .stp_t
%-----

% The specification file for the StP main menu
%
% Customized STPmenu.spec file for the StP/T integration
%
STPMenuSpec=/staff/imet/banowetz/STP/ISE-T-integ/STPmenu.spec
%
%
% StP-binaries -> PATH
PATH=./staff/imet/banowetz/STP/ISE-T-integ/t_int:/tools/IDE/binsun4:/bin:/usr/bin:
%
%
% Customized annotations for the T Integration
%
annot_path=/staff/imet/banowetz/STP/ISE-T-integ/ann_tmpl
%
%
% establishes the path to T
%
%
Tdir=/tools/cots/sunos.4.1.x/T3.1
%
% T demo license
%
%nlm_license_dat=/tools/cots/sunos.4.1.x/T3.1/license.dat
STP_DEMO_LIC_DAT=7A0-E66-307-A6F-133-07A-0E6-630-0E3-80E-38D
%
%
ToolInfo =/tools/IDE/libsun4/ToolInfo

ToolInfo.ISE-Tbase

```

Figure A3-1. Example of ToolInfo file for StP/Extract

```

##### ***** tconfig.ini n Thu Nov 10 12:40:27 EST 1994
# T3 Configuration File
# tconfig.ini   A user customized selections file for running T

[BASE]
                                # following two entries for INSTALLATION
TPath = /tools/cots/sunos.4.1.x/T3.1      # full path to installed T product
# TPath = /staff/imet/banowetz/STP/ISE-T-integ/t_int  # try this for extraction
# Does not work for extraction and T fails.)

Tldb = /tools/cots/sunos.4.1.x/T3.1/tldb      # full pathname of T language data l

TestUnitDir    = /staff/imet/banowetz/t_test/UNITDIR      # path to a default testun:

MessageLevel   = 3      # default highest message level reported (0,1,2,3)

[VERIFY]
ListExtra      = no      # list extra sdf information in verify.rpt
                                # n,no,NO,... or y,yes,YES...

[DESIGN]
Preconditioned = yes      # require preconditions to be TRUE in test design
                                # n,no,NO,... or y,yes,YES...
SummaryLineSpacing = 1    # summary report, 1=single space, 2=double space

[PREPARE]
TestCaseDir    = /staff/imet/banowetz/t_test/PREPDIR
    # path to a default testcase directory for prepared test case files

TestCaseSubsetAction = all    # default test case subset of actions to prepare
TestCaseSubsetState  = all    # default test case subset of states to prepare
TestCaseSubsetIndex  = all    # default test case subset of indexes to prepare

                                # path to default case descr. file for prepare
CaseDescriptFile = /staff/imet/banowetz/t_test/tcf.cdf

[END]
##### ***** tconfig.ini n Thu Nov 10 12:40:28 EST 1994

```

Figure A3-2. Example of TCONFIG file for T

APPENDIX B. Diagnostic Messages, Errors and Countermeasures

B1. T Diagnostic Messages, Errors and Countermeasures

B2. StP Diagnostic Messages, Errors and Countermeasures

B2.1 StP/E startup problems

B2.1.1 Setup error

B2.1.2 Overload

B2.1.3 Segmentation Fault StP (core dumped)

B2.2 DFD problems

B2.3 Control Spec problems

B3. StP Extract Problems

B3.1 Extract SA problems

B3.2 Extract SD problems

B3.3 Extract RT problems

B3.4 General extract problems

B1. T Diagnostic Messages, Errors and Countermeasures

Problem: Tverify fails with Segmentation fault (core dumped)

Resolution: Failure to define the environmental variable TCONFIG variable for T execution

Problem: T stand alone fails with the following in StP execution window:

FLM: "T" checkout failed

FLM error: no such feature exists

error 9900: internal error

No T license is currently available

Resolution: The ToolInfo environmental variable is defined for StP instead of for T. Use the alias set up in Appendix A3.

B2. StP/SE Diagnostic Messages, Errors and Countermeasures

B2.1 StP/E startup problems

B2.1.1 Setup error

Problem: StP/Extract fails at startup with the following in active window:

ToolSpecification file:: No such file or directory

S TPmenu.spec

Setup command/argument panels failed

Resolution: The ToolInfo environmental variable is defined for StP/T instead of StP/Extract? Use the alias set up in Appendix A4.

B2.1.2 Overload

Problem: Starting up StP, the window used to start up StP has the following message:

sh: getwd: can't stat .

after the usual:

StP: Parsing Specification File

StP: Setup Control Panels

Other related problems:

The START DBMS switch is present.

The DFD edit on an existing file comes up void.

Resolution: Switch to another processor for StP operations. This problem may be related to high load.

B2.1.3 Segmentation Fault StP (core dumped)

Problem: Starting up StP, the window used to start up StP hangs after the following message:

StP: Parsing Specification File

StP hangs up its execution window. After entering CTRL C the response is

[2] Segmentation fault StP (core dumped)

Resolution: Your UNIX path variable may not be set up to use the Strand 4.2D version of StP or your ToolInfo file may not be compatible with the StP version selected based on you path.

B2.2 DFD problems

Problem: The generate StP/DD command produces an error message:

StP/DD allnames locked --- retries left 15

Resolution. The cause of this lockup is unknown. As a work around, quit all StP processes and restart.

B2.3 Control Spec problems

Lines of information in the Process Activation Table (PATs) lines must be deleted using the line delete command. Otherwise, a "Bus error" can be expected when doing the associated RT extraction. The error is caused by removing a row of PAT data by deleting the components rather than the entire row as must be done. To delete the row correctly, the two horizontal parallel bars must replace the cursor above the line to be deleted in column 0 (row numbers) when the delete is selected.

B3. StP Extract Problems

The Extract program and scripts are integrated into StP/SE. Problems relating to their use for System Analysis (SA), System Design (SD), and Real Time (RT) are described below.

B3.1 Extract SA problems

Problem: Only primitive processes can be extracted for test.

Resolution: Upgrade to StP version 5.0 with the associated extract software.

B3.2 Extract SD problems

N/A

B3.3 Extract RT problems

The extraction of RT data from the Control Specification Process Activation Table (PAT) is not usable by T. It is only a translation tool which makes the PAT language more readable.

B3.4 General extract problems

The SDF file produced by the extraction is sometimes void. This seems to be a unix file problem caused by entering the new SDF directory too soon. A new directory is created for each SDF extract product. Do not get into this new directory until after the "Completed extraction" line appears in the cmdtool window.

For testing purposes, StP annotations must include min, max, and increment data for all numeric data. This must be added even though this information is not required for intermediate data from the point of view of the SRS, or function testers. It is only needed from the point of view of unit testing.

APPENDIX C. Questions and Answer Tips

- C1. StP tips
- C2. StP Extract Tips
- C3. T tips

C1. StP tips

Question: What are the installation parameters and constraints on the installation of StP at NAWC?

Answer: StP was installed on guinan. It is executable on either guinan or crusher but the man pages are available only on guinan.

Question: What are the StP diagram types, their purposes, and corresponding data storage directories?

Answer: The Diagram types are listed below numbered by StP 4.2D User's Guide Chapter.

5. DF -- Data Flow. Requirements/SRS (SA model for T). Data in dfe_files.
6. DS -- Data Structure. Supporting SA, SD, T Data in dse_files.
7. ER -- Entity Relationship. Supporting SA, SD, (alternative to DSE) Data in ere_files
8. SC -- Structure Chart design/SDD (SD model for T) Data in sce_files.
9. CF -- control flow (included in DFE by option box) Data in dfe_files.
10. CS -- control specification. (Competes with STE/11)
Evoked by pushing on cspec symbol and selecting CSE and then one of the 7 media of expression OR by selecting the CSE editor directly. Only the Process Activation Tables (PATs) are used by T. The six other forms or control specifications are not used by T3. Associated files are in csp_files . (There is no cse_files directory)
11. ST -- State Transition (compare to 18. Competes with CSE/10).
Evoked by pushing on cspec symbol and selecting STE. Data in ste_files
12. PIC -- picture editor. Data in pct_files. Not connected to DD.
13. OA -- Object Annotation (activate by right mouse button) Data embedded in parent diagram repository.
16. DPS -- Document Preparation System. Data in doc_files and ted_files
18. TD -- Transition Diagram (missing from main menu ????) Data in tde_files.
(not used in ASQ-212/tms/nav) Not connected to DD.

Question: Other than the directories supporting StP above, what additional data directories are created by StP?

Answer:

t_files: The depository of SDF files from the StP/T extraction
obj_files unknown purpose (not used in ASQ-212/tms/nav)
src_files unknown purpose (not used in ASQ-212/tms/nav)

Question: How can you create links between modules in the SCE without arrows on the ends like on page 8-17 of the StP Users Guide?

Answer: Toggle on arrow symbol in options area when SCE group is active. You may have to re-edit the diagram.

Question: How do you define the scenario number in the annotation?

Answer: Select a data flow item in a DFD for annotation editing and add the note type "Scenario". Then enter a note key value consistent with other data flows in the same group you wish to define.

Question: Must a data store be used for both input and output.

Answer: Yes, if not it will be flagged as an error. If both input and output are not part of the requirements, then the database should be set up as external.

Question: How do you resolve the "Undefined Data" problem?

Answer: Items marked as "Undefined Data" by the check diagram command are primitive data items which do not have a type defined in the data dictionary. Even selection items must have the type defined but any value such as N/A is sufficient. The extraction program takes the selections items as string values for the parent dataitem. Also be sure to generate the StP/DD for each diagram.

Question: How do you delete a StP entity:

Answer: The following items which appear in the data dictionary can be deleted as follows:

Diagram - Main Menu - Diagram Utilities, Delete Diagram (Avoid using a unix delete ... you will have to cleanup using the data browser)

Object - Main Menu - Project Database, Delete Named Object

Question: What happens when the ToolInfo environment variable is not defined and pointing to the proper file?

Answer: Error Message:

ToolSpecification file:: No such file or directory

Question: Are Control flows permitted to flow into any DFD/CFD process?

Answer: Primitive processes (those with a process spec) cannot accept control flow input. The control flow for this process should just flow into an anchor point to indicate that it is present but is really of interest only at a higher decomposed level. The process specification at the highest level possible should indicate which processes are activated under what condition. Use the Process Activation Table (PAT) form if you wish to extract a TSDL form interpretation of the process specification.

Question: What is the value of SDF files from RT (Real Time) Extract?

Answer: These files provide a verbose interpretation of the Process Activation Table (PAT) of a control specification to clarify its meaning. It is useful as a training device but has little relevance to StP/T.

Question: What can be done when there appear to be mysterious, spurious constraints to changes in the annotations or diagrams.

Answer: Use the DD browser to remove obsolete objects and regenerate the data dictionary.

Question: What is the cause of the error message below produced by Check DD?

ERROR: Unconnected Offpage connector:

Answer: There are spurious unused anchor points in a DFD. Be sure that the anchor points are not part of a control flow by displaying both data and control flow and remove all unnecessary anchor points.

Question: What data dictionary checking utilities are available outside of the menu driven DD checks?

Answer: The StP data can be checked beyond the check_dd function by the script:

/cp2044/fss/a47/local/rjohnsons_srs_tools/gen_dd_all.sh

with help file:

/cp2044/fss/a47/local/rjohnsons_srs_tools/gen_dd_all.doc

and supporting file:

/cp2044/fss/a47/local/rjohnsons_srs_tools/gen_dd_all.trl

Question: How do you enter minimum, maximum, and increment when adding annotation for a data item?

Answer: Select Annotation edit for an object (either a data item or type).

In the Note Type mode, select Add and add Domain as a note type. With Domain selected, select Edit. Data item which switches from Note Type to Note Item mode. Select Add and add minimum, maximum, and increment (as needed). Move the cursor to the Value area to change and enter the desired values.

Question: Where are the variables defined in the StP ToolInfo file described or documented?

Answer: The StP ISE Customizing the Environment document describes the StP ToolInfo variables.

Question: Is there an annotated list of error messages generated by StP?

Answer: There is a list of error messages in file in the StP installation directory (/tools/IDE/libsun4/msg_file) but it is NOT annotated.

C2. StP Extract Tips

Question: What types of changes to the diagrams for T extraction are of a global nature?

Answer:

The T extract program requires all base types used in the diagrams to be spelled in lower case. (WARNING: the base types may not contain the most desirable domain data.) For the ASQ-212 data, the case is typically mixed.

Question: What tools are available for global changes to the data structure diagrams ?

Answer: DSD files can be changed quickly by the UNIX sed command by setting up an edit file such as the one below to change the base type names to lower case:

```

1,$s$.T Boolean$.T boolean$
1,$s$.T String$.T string$
1,$s$.T Real$.T real$
1,$s$.T Integer$.T integer$

```

By using the above text in a file named after the -f parameter of the sed command, a DSD file can be edited. This would have to be done to both .dse and .str files.

Changes such as the above could be done to all DSD diagrams for a model by using a perl script (perl is not yet installed at NAWC). Then the changes to all dse diagram files can be incorporated into the data dictionary by the following c shell script:

```

foreach j (`cd /<system directory>/dse_files; ls *.dse`)
    echo ""
    echo "Generating Structure for DSE diagram: $j"
    dsstr -v $j
end
foreach j (`cd /<system directory>/dse_files; ls *.str`)
    echo ""
    echo "Generating data dictionary from DSE diagram $j"
    struct_dd -v $j
end

```

Question: What diagrams and data are used for the extraction for T?

Answer: There are extraction process for diagrams of the following types:

DFD, Data Flow Diagrams in directory dfe_files: (All primitive processes including and under a selected process)

DFD, Same as above but only processes with a specified scenario number)

SCD, Structure Chart Diagram in directory sce_files for a selected module.

RTD. Run Time Data. Selected Process Specification (Process Activation Tables only) in directory csp_files

DSD, Data Structure Diagrams in directory dse_files: As needed for definitions for above DFDs and SCDs

Question: Which StP requirements and design diagrams are NOT incorporated into the T extraction?

Answer:

DED, Data Entity Diagrams in directory ere_files

(An alternative data description format to DSD.

CFD, Control Flow Diagrams in directory dfe_files: (This information is ignored)

Question: What documentation is available for the StP/T extraction?

Answer:

- a. Appendix A of the soft spiral bound class handout "Automated Software Testing, Version 0.6"
- b. Application Note: Software through Pictures (StP) and T Integration

Question: What types of dataitems in the SDF are meaningful to T.

Answer: The Application Note for StP/T indicates that the base types are real, integer, string, boolean, or character. However, dataitems should NOT normally be typed directly to these base types (except for boolean) if the diagrams are to be used for testing since these datatypes do not have reasonably usable domains. Instead, they should be typed to higher level datatypes within the StP diagrams which themselves have these base types (but with reasonable, customized domains). Alternatively, they can be typed to a suitable type in the Stp.std or tsdl.std file included by the SDF file. (However, version 4.2D of the StP extract program has a bug. It does not recognize types in the tsdl.std file as special and generates spurious lines in the SDF labeling these types as undefined.)

Question: What constraints are there on domain values for the numeric types:

Answer:

integer: no decimal point

real: decimal point required. However, the decimal point need not be preceded and followed by a digit as in Ada

Question: How can test cases for the entire system be extracted?

Answer: Extractions from the Structure Chart Editor (SCE) design diagrams provide test cases for all modules which have inputs. However, the input must not be from an anchor point. Use an external symbol to represent the input to the main module if test cases are desired.

Extraction of the top level (context diagram) (DFD) process is not possible for StP version 4.2D. However, any non-primitive SA process including the top level one can be extracted using StP version 5.0.

C3. T tips

Question: How do you setup a command window to run T at the VP facility.

Answer:

The processor for running T must be "crusher" or one which has access to its file directory.

The TCONFIG environment variable must point to your TCONFIG file, the file specifying the selections for the T execution. Do this by
setenv TCONFIG <filename>

The ToolInfo environment variable must point to your T ToolInfo file, the file specifying the locations of the Menu data for the T execution. Do this by
setenv ToolInfo <filename>

The above two files can be customized from a sample provided with the T delivery.

Question: Where are the variables defined in the StP/T ToolInfo files described or documented?

Answer: The StP ISE Customizing the Environment document describes the StP ToolInfo variables.

Question: What are the reserved words and keywords of the TSDL language?

Answer: The TSDL language is one of the largest languages around but has NO reserved words. There ARE a large number of keywords which have meaning in context but are not restricted to use by the user for variable names. However, their use can be confusing so a warning is generated by Tverify. Bob Poston, the creator of T, recalled that a list of keywords was formerly available and possibly called keywords.txt. However, this was not part of the T tool delivery. The rnames.exe utility appears to offer a capability to list of keywords but it is not documented.

Question: Is it possible to change the Stp.std used when running T.

Answer: The extraction program generates a SDF file with an include of <sdf.std>. This Stp.std file is in the same directory as the T executables based on your TCONFIG and ToolInfo files. However, to change the file, create your own version in your own directory and modify all SDF files to change the reference to that file. Replace the <sdf.std> reference by the full path name within double quotes.

Question: What is the source of the low bound and high bound values in the samples.rpt file?

Answer: The SDF file provides this by datatype definitions in one of two ways :

In a "has values" clause, the first item listed becomes the "low" bound whether it be real, integer, or string. It does not necessarily mean low in the algebraic sense. Similarly, the last value listed becomes the "high" bound.

In a "has value range..." clause, the minimum becomes the low bound and the maximum becomes the high bound.

APPENDIX D. Weight Computer Program Illustration

- D1. Requirement Specification Data Flow Diagrams (DFDs) for Structured Analysis
- D2. Data Structure Diagrams (DSDs)
- D3. Structure Chart Diagrams (SCDs) for Structured Design (SD)
- D4. Software Description File (SDF) Listings
- D5. Tverify/Tdesign Reports for Main Process

D1. Requirement Specification Data Flow Diagrams (DFDs) for Structured Analysis

The starting point for defining requirements in the StP world is a special type of DFD called the context diagram . This diagram has one or more characteristic symbol, the rectangle. A rectangle in the context diagram represents an external element, i.e., an elements outside the program being described. In the very simple model shown in Figure D1-1, the user is the external element. This external might also be called the display/keyboard.

For all DFDs, a circle represents a process. A process circle is numbered and the character following the process number indicates whether the process is decomposed (broken down into components) or not decomposed (primitive). Those with a "*" following the number are decomposed processes . Those with a "p" following the number are primitive processes. These processes are described in a process specification. These process specifications are automatically generated except for the description part which is provided by the StP user for the process.

Process 0 of the context diagram (also called the top level diagram) is decomposed into a diagram called level 0. This diagram focuses on the internal flow instead of the external flow to process 0. This decomposition is illustrated in three different ways in Figures D1-2 through D1-4. These three decompositions are labeled "verbose", "terse", and "preferred" respectively because they illustrate the alternatives in diagram detail available to the user. The three are essentially equivalent and produce comparable SDF files from the StP/T extraction. However, an additional, rather trivial, SDF file was created for the verbose split_record process in Figure D1-2. (No listings specifically for the verbose and terse diagrams are shown beyond the DFDs.)

A data flow coming to or from an unlabeled point (known as an *anchor* point) in a DFD represents communication with an external. Data flow coming to or from a numbered but otherwise unlabeled circle in a DFD represents communication with a process defined in a higher level diagram.

In the preferred level 0 diagram (Figure D1-4), there are two processes numbered 1 and 2. Since process 1 is primitive, there is no level 1 diagram . However, process 2 is further decomposed into a level 2 diagram in Figure D1-5 to illustrate multiple levels of decomposition. The listings which follow the DFDs list the data flow items, annotations for all processes, and the process specifications for those which are primitive. The process specification is nothing more than text to describe what a primitive process is required to do.

Project: /staff/inet/banowitz/STP/
System: init_system
Diagram: top

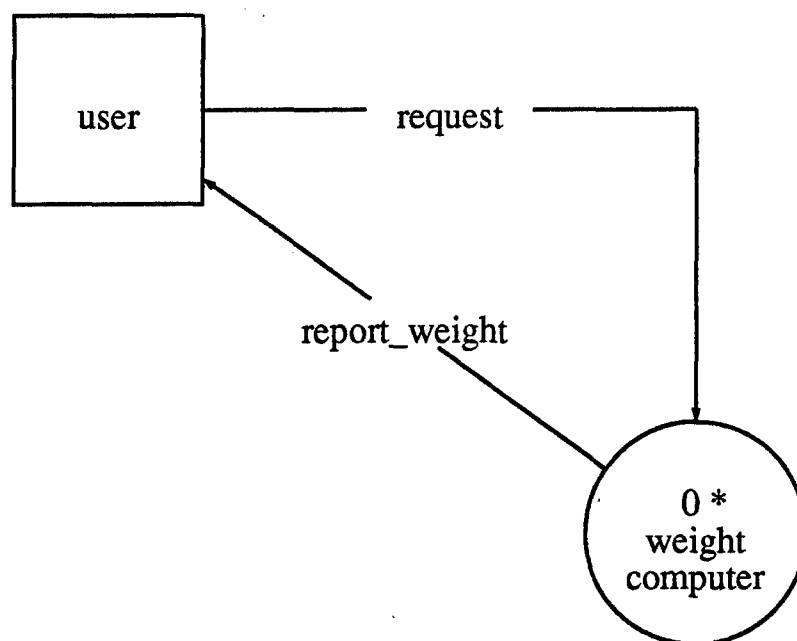


Figure D1-1. Example of a simple context diagram

Project: /staff/imet/banowetz/STP/
System: init_system
Diagram: 0

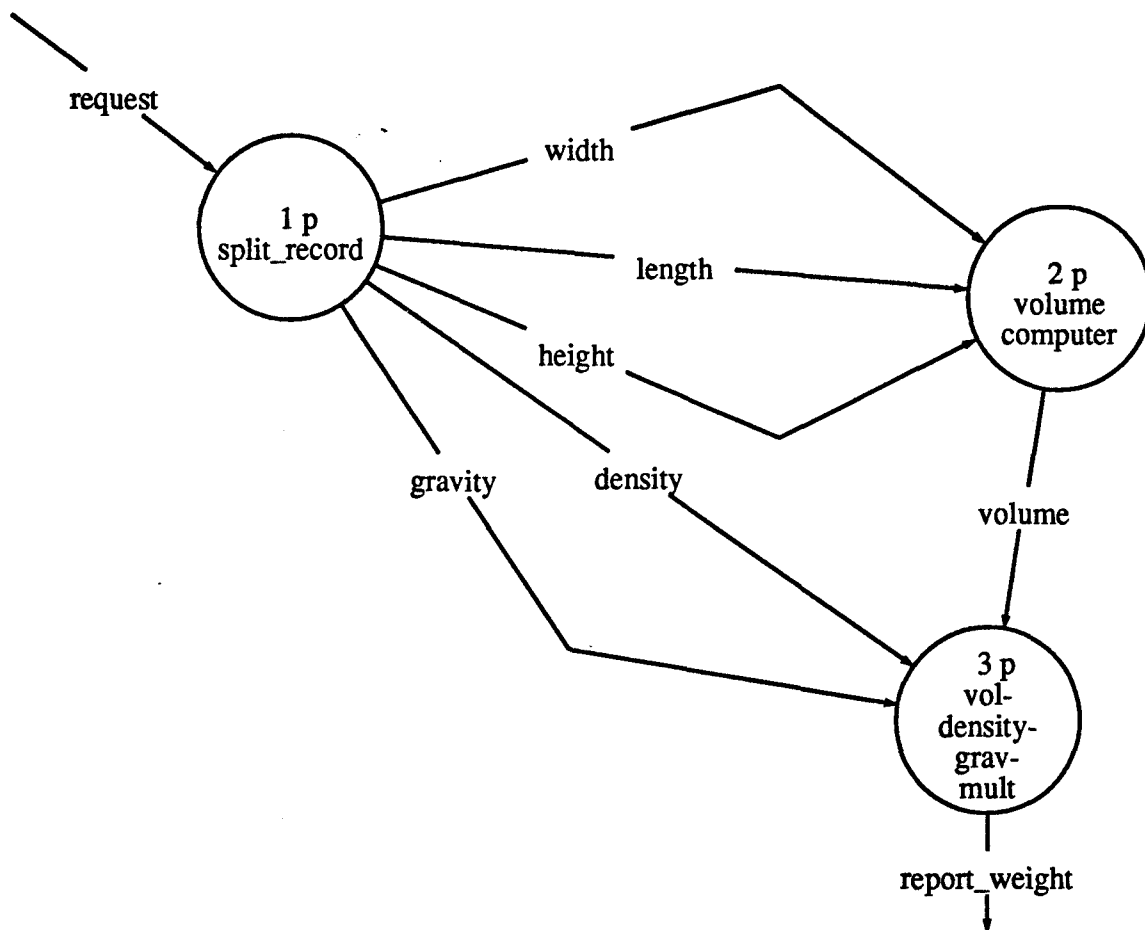


Figure D1-2. Verbose 0 level Data Flow Diagram (DFD) decomposition

Project: /staff/imet/banowetz/STP/
System: init_system
Diagram: 0

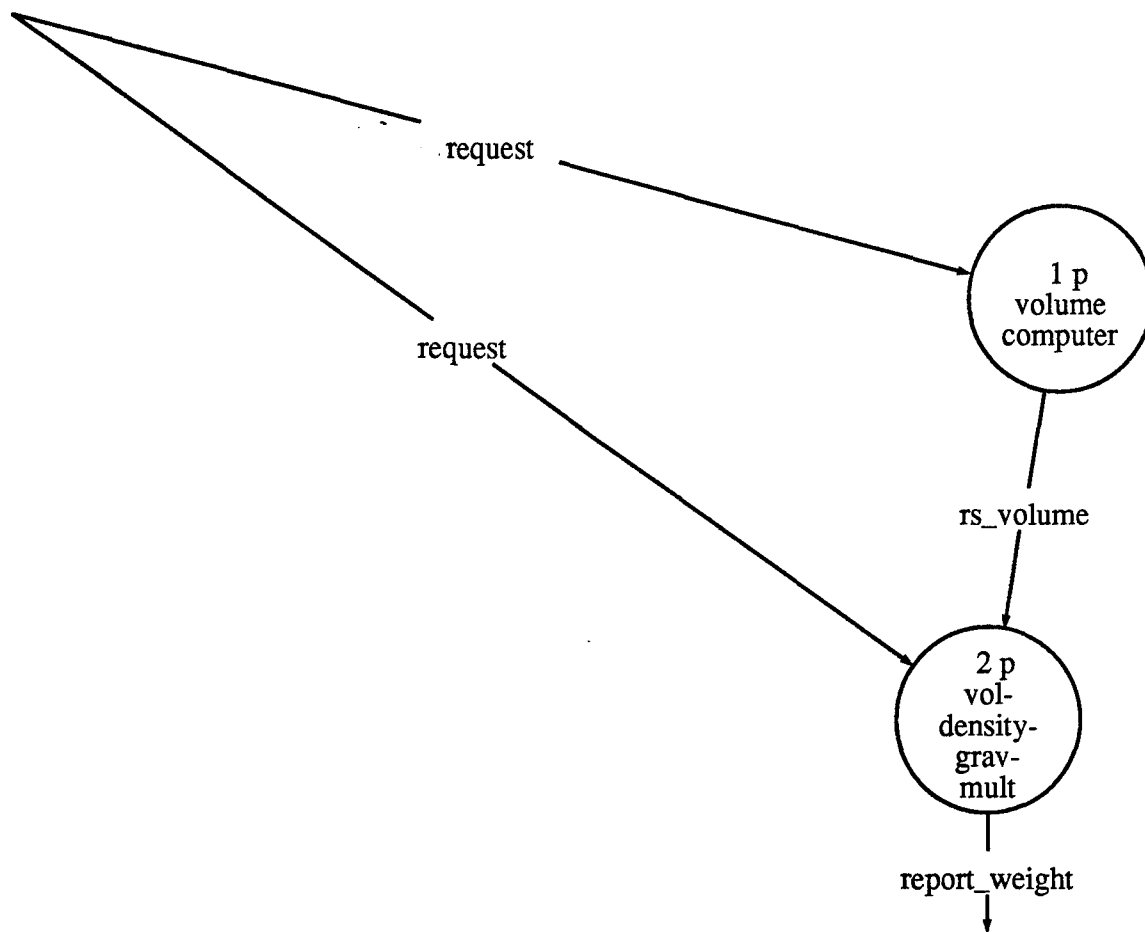


Figure D1-3. Terse 0 level DFD decomposition

Project: /staff/inet/banowetz/STP/
System: init_system
Diagram: 0

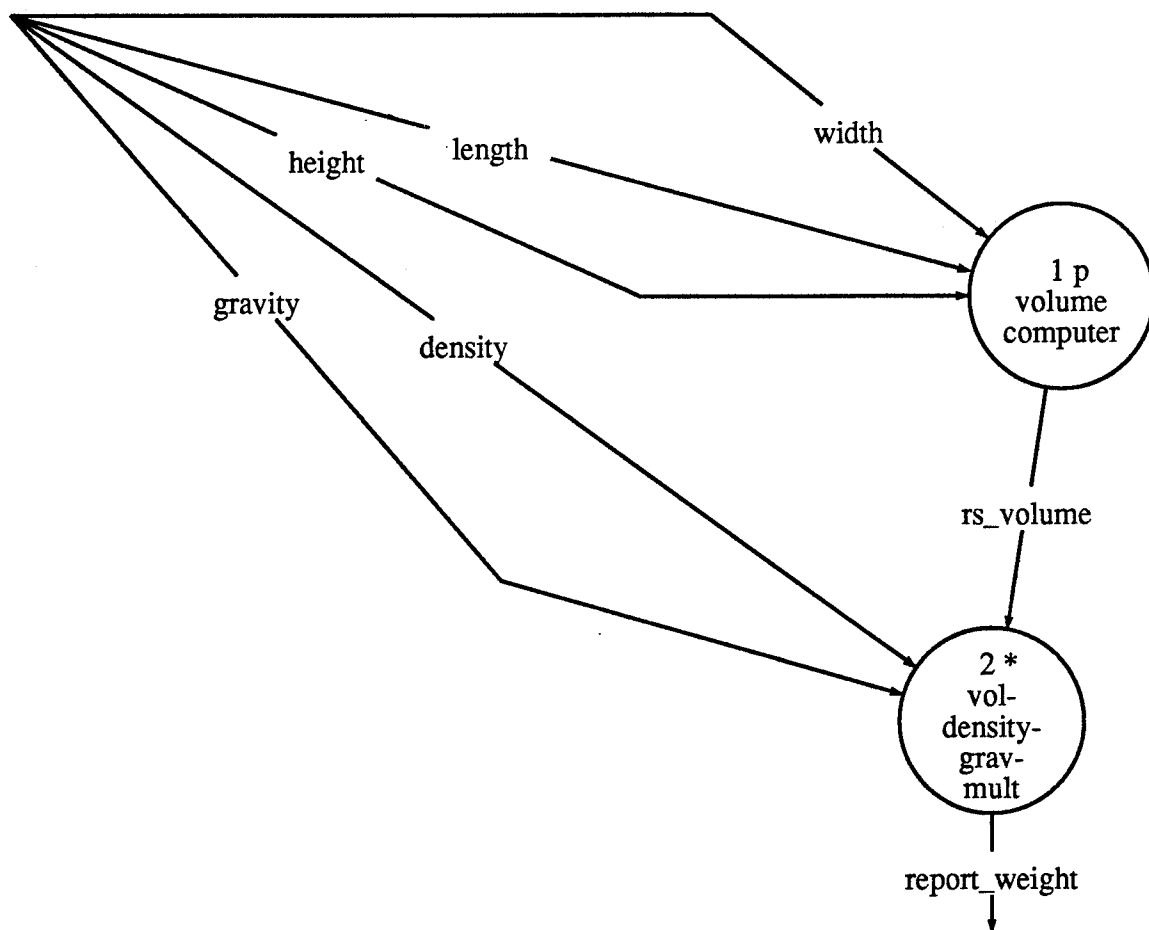


Figure D1-4. Preferred 0 level DFD decomposition

Diagram 0:

Note diag
Top level for a system to compute weight and report to user.

Note Requirement compute weight
RequirementName: compute weight

Process 1, volume_computer:

Note ProcessSpec
Process 1: volume_computer

Pspec generated
21 September 1994 at 14:34:16 by banowetz@crusher
This process has 4 data flows:
rs_volume, length, width, height

input data flows
length, width, height

output data flows
rs_volume

description
multiply length, width, and height
to get volume
end pspec

Process 2, vol-density-grav-mult:

Note process
Name: vol-density-grav-mult

Note Testability
ActsOnlyIf: rs_volume >= 0.0
IsTestedExhaustivelyOn: density
This process is testable only for non negative volume.

Data Flow density:

From: Offpage.0
To: vol-density-grav-mult

Note DataFlow density
Name: density
density and gravity only to 2p

Note InterfaceDesign request
InterfaceName: request
InterfacePriority: N/A
InterfaceType: CSCI or HWCI

Data Flow gravity:

From: Offpage.0
To: vol-density-grav-mult

Data Flow height:

From: Offpage.0
To: volume_computer

Data Flow length:

From: Offpage.0
To: volume_computer

Note DataFlow length
Name: length
length, width, and height go to lp

Note InterfaceDesign request
InterfaceName: request
InterfacePriority: N/A
InterfaceType: CSCI or HWCI

Data Flow report_weight:

From: vol-density-grav-mult
To: Offpage.0

Data Flow rs_volume:

From: volume_computer
To: vol-density-grav-mult

Note DataFlow rs_volume
Name: rs_volume

Note InterfaceDesign volume
InterfaceName: volume
InterfacePriority: N/A
InterfaceType: CSCI or HWCI

Note Scenario 3
ScenarioNumber: 3

Data Flow width:

Project: /staff/imet/banowitz/STP/
System: init_system
Diagram: 2

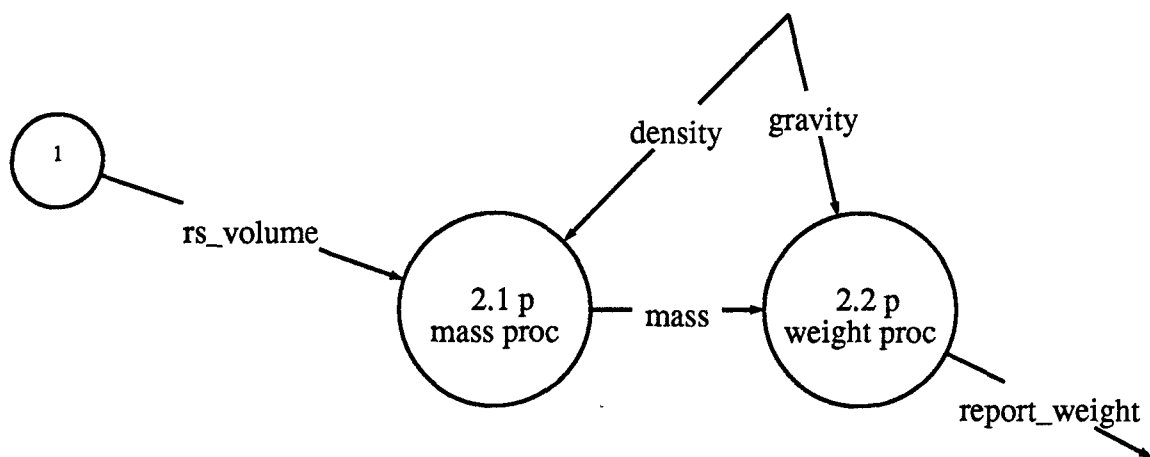


Figure D1-5. Level 2 DFD

Diagram 2:

Note diag
Parent: vol-density-grav-mult

Note Requirement compute weight
RequirementName: compute weight

Process 2.1, mass_proc:

Note ProcessSpec
Process 2.1: mass_proc

Pspec generated
21 September 1994 at 14:34:45 by banowetz@crusher
This process has 3 data flows:
rs_volume, density, mass

input data flows
rs_volume, density

output data flows
mass

description
mass = rs_volume * density
end pspec

Process 2.2, weight_proc:

Note ProcessSpec
Process 2.2: weight_proc

Pspec generated
21 September 1994 at 14:34:45 by banowetz@crusher
This process has 3 data flows:
report_weight, gravity, mass

input data flows
gravity, mass

output data flows
report_weight

description
report_weight = mass * gravity
end pspec

Data Flow density:

From: Offpage.2
To: mass_proc

Data Flow gravity:

From: Offpage.2
To: weight_proc

Data Flow mass:

From: mass_proc
To: weight_proc

Data Flow report_weight:

From: weight_proc
To: Offpage.2

Data Flow rs_volume:

From: Offpage.2
To: mass_proc

D2. Data Structure Diagrams (DSDs)

All data for this model is presented in two compact diagrams. Figure D2-1 shows the data available to the context diagram and Figure D2-2 shows local data. Figure D2-1 shows input "dataitems" arranged in a record (request), the output dataitem (weight), and the "datatypes" needed for the primitive (undecomposed) dataitems. The datatype "dim_type" is used for all three of the dimensional dataitems. The listings which follow each figure show the annotations for all items. Annotations indicate the type of a dataitem and the allowable values or domain for a datatype. The domain is the minimum and maximum value allowed for a type and the increment desired for test variations. StP treats the dataitems and datatypes as similar entities and arranges them alphabetically in these listings.

The DSDs support both the SA diagrams in Section D1 above and the SD diagrams in Section D3 below.

Project: /staff/imet/banowetz/STP/
System: init_system
Diagram: request

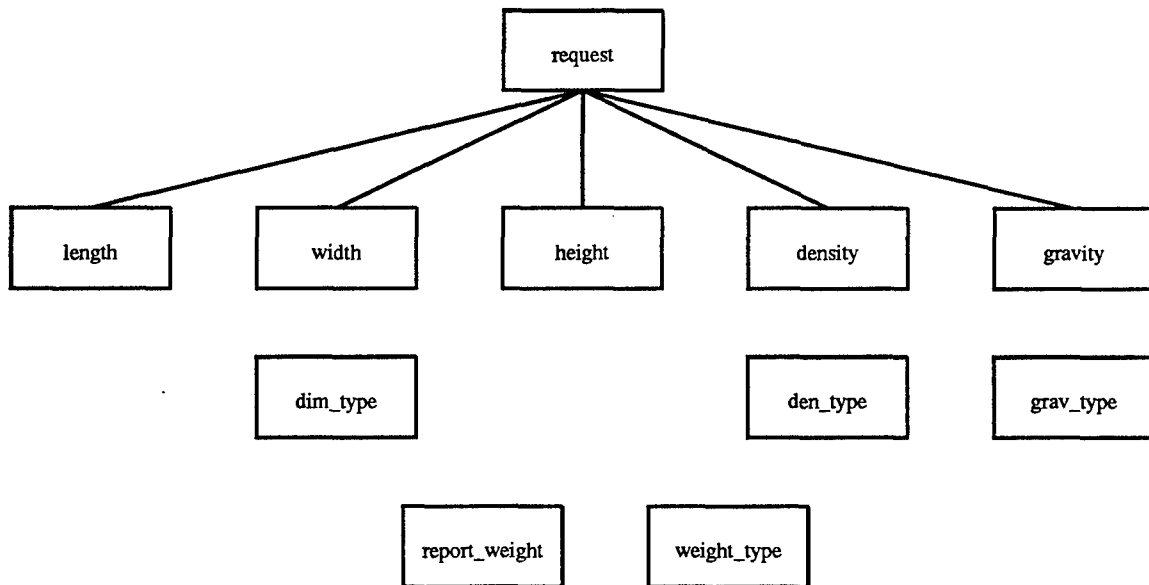


Figure D2-1. Context Diagram Data DSD

Diagram request:

Note diag

Name: request

This diagram has all external data elements. These are the data elements which are I/O to/from an external entity.

Data Structure request:

Note DataDefinition request

Name: request

Element den_type:

Note DataDefinition den_type

Name: den_type

Type: real

Units: kg/meter

Note Domain

Increment: 10.0

Minimum: 0.0

Maximum: 1.0E3

Element density:

Note DataDefinition density

Name: density

Type: den_type

Units: kg/cubic meter

Element dim_type:

Note DataDefinition dim_type

Name: dim_type

Type: real

Units: meters

dimensions: length/width/height

Note Domain

Increment: 1.0

Minimum: 0.0

Maximum: 1.0E2

Element grav_type:

Note DataDefinition grav_type

Name: grav_type

Type: real

Units: newtons/kg

Value: 9.8

Note Domain

Increment: 0.2

Minimum: 0.0

Maximum: 20.0

Element gravity:

Note DataDefinition gravity
Name: gravity
Type: grav_type
Value: 9.8 newtons/kg

Element height:

Note DataDefinition height
Name: height
Size: N/A
Type: dim_type
Units: meters

Element length:

Note DataDefinition length
Name: length
Type: dim_type

Element report_weight:

Note DataDefinition report_weight
Name: report_weight
Type: weight_type
Units: newtons

Element weight_type:

Note DataDefinition weight_type
Name: weight_type
Type: real
Units: newtons

Note Domain
Increment: 1.0
Minimum: 0.0
Maximum: 9.0e99

Element width:

Note DataDefinition width
Name: width
Type: dim_type

Project: /staff/imet/banowetz/STP/
System: init_system
Diagram: rs_volume

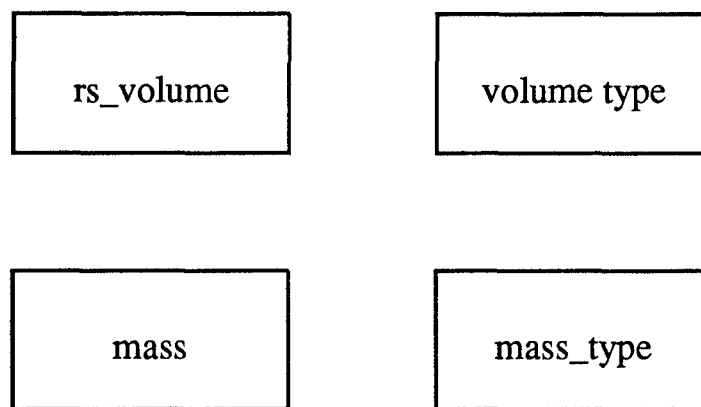


Figure D2-2. Local Data DSD

Diagram rs_volume:

Element mass:

Note DataDefinition mass
Name: mass
Type: mass_type

Element mass_type:

Note DataDefinition mass_type
Name: mass_type
Type: real
Units: kg

Note Domain
Increment: 1.0E8
Minimum: 0.0
Maximum: 1.0E11

Element rs_volume:

Note DataDefinition rs_volume
Name: rs_volume
Type: volume_type
Units: cubic meters
volume is the product of 3 dimensions

Element volume_type:

Note DataDefinition volume_type
Name: volume_type
Type: real

Note Domain
Increment: 1.0E4
Minimum: 0.0
Maximum: 1.0E7

D3. Structure Chart Diagrams (SCDs) for Structured Design (SD)

The SD diagrams implementing the requirements of the SA diagrams in D1 are shown in Figures D3-1 and D3-2. For this illustration, there is precise correspondence between requirements and design which is facilitated by the use of the same DSDs shown in D2 above. Modules for the design match similar processes for the requirements which means that requirements based testing will produce the same test cases as design based testing. The main difference is that the flow of data items for the SD is shown only from the perspective of parent/child relationships so that coding can be segmented and structured.

Project: /staff/imet/banowetz/STP/
System: init_system
Diagram: 0

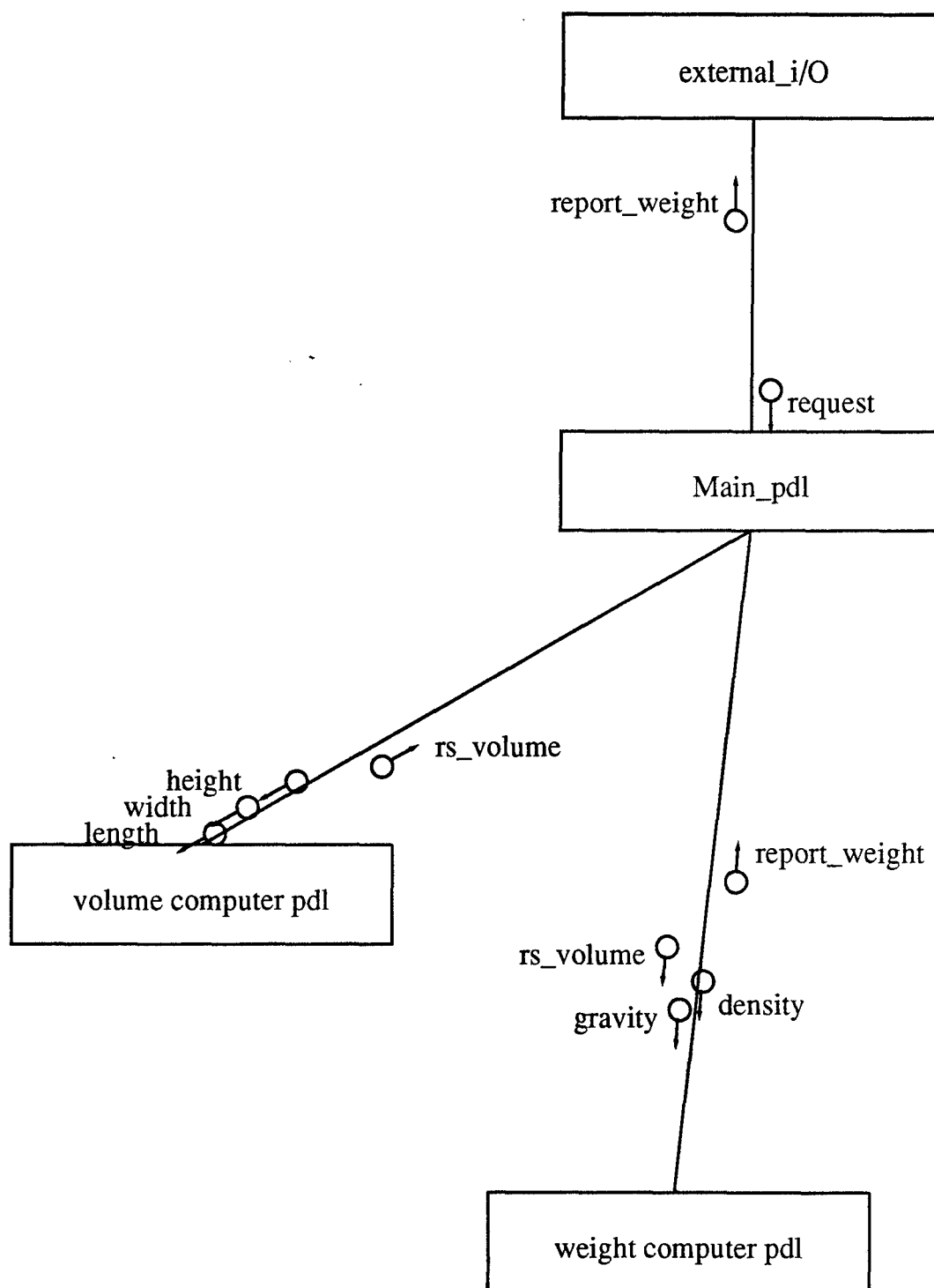


Figure D3-1. Top level System Design Structure Chart Diagram (SCD)

Diagram 0:

Module Main_pdl:

```
Note module
  Name: Main_pdl
  Input 5 values
  Call volume computer pdl
  Call weight computer pdl
```

Module external_i/O:

Module volume_computer_pdl:

```
Note ModulePDL
module volume_computer_pdl

Pdl generated
  6 September 1994 at 16:37:40 by banowetz@crusher
This module has 4 parameters:
  rs_volume, height, width, length

input data params
  height, width, length
input flags

output data params
  rs_volume
output flags

calls

called by
  Main_pdl
description
  rs_volume=height*width*length
end module
```

Project: /staff/imet/banowetz/STP/
System: init_system
Diagram: weight_computer_pdl

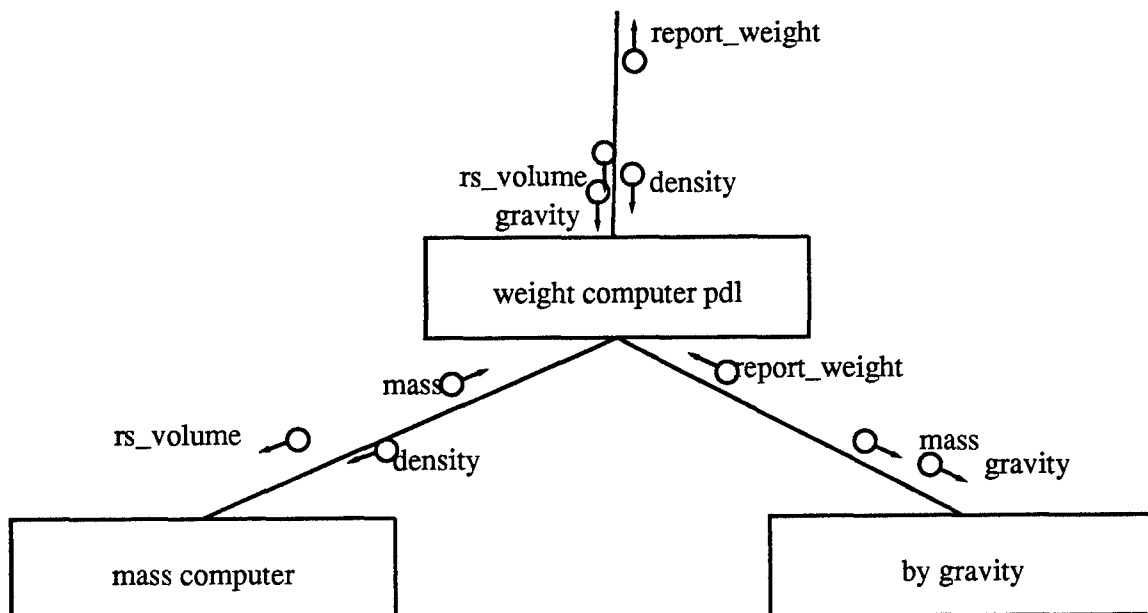


Figure D3-2. Second level System Design SCD

Diagram weight_computer_pdl:

Module by_gravity:

```
Note ModulePDL
module by_gravity

Pdl generated
  17 August 1994 at 12:36:14 by banowetz@crusher
This module has 3 parameters:
  report_weight, mass, gravity

input data params
  mass, gravity
input flags

output data params
  report_weight
output flags

calls

called by
  weight_computer_pdl
description
  report_weight = mass * gravity
end module
```

Module mass_computer:

```
Note ModulePDL
module mass_computer

Pdl generated
  17 August 1994 at 12:36:14 by banowetz@crusher
This module has 3 parameters:
  mass, rs_volume, density

input data params
  rs_volume, density
input flags

output data params
  mass
output flags

calls

called by
  weight_computer_pdl
description
  mass = rs_volume * density
end module
```

Module weight_computer_pdl:

D4. Software Description File (SDF) Listings

SDF files for the three primitive SA processes are shown in Figures D4-1 through D4-3. SDF files for non primitive processes are not provided since version 4.2D of StP/SE does not extract for non primitive processes. SDF files for all SD modules are shown in Figures D4-4 through D4-8. The correspondence between SDF files for the SA and SD are shown in Table D4-1 below.

Unit	SA Figure	SD Figure
length * width * height -> volume	4-1	4-5
volume * density -> mass	4-2	4-7
mass * gravity -> weight	4-3	4-8
Main program	N/A	4-4
volume * density * gravity -> weight	N/A	4-6

Table D4-1 Figures for listings of SDF files for SA and SD

```

s_packet                                Proc_volume_computer.

#include <stp.std>
Action                                volume_computer
    is actiontype internal;
    uses dataitem
        height,
        width,
        length;
    produces dataitem
        rs_volume;.

Dataitem                                height
    is an instance of datatype dim_type.

Dataitem                                width
    is an instance of datatype dim_type.

Dataitem                                length
    is an instance of datatype dim_type.

Dataitem                                rs_volume
    is an instance of datatype volume_type.

Datatype                                dim_type
    is datatypeclass real;
    has value range minimum 0.0;
    has value range maximum 1.0E2;
    has value range resolution 1.0;
    has valid subdomain "as_specified";
    has invalid subdomain "abnormal".

Datatype                                volume_type
    is datatypeclass real;
    has value range minimum 0.0;
    has value range maximum 1.0E7;
    has value range resolution 1.0E4;
    has valid subdomain "as_specified";
    has invalid subdomain "abnormal".

```

Figure D4-1 System Analysis SDF for volume computer

```

s_packet                                Proc_mass_proc.

#include <stp.std>
Action                                mass_proc
    is actiontype internal;
    uses dataitem

        density,
        rs_volume;
    produces dataitem
        mass;.

Dataitem                                size_cat
    is an instance of datatype size_cat_t.

Dataitem                                density
    is an instance of datatype den_type.

Dataitem                                rs_volume
    is an instance of datatype volume_type.

Dataitem                                mass
    is an instance of datatype mass_type.

Datatype                                size_cat_t
    is datatypeclass string;
    has values
        "big",
        "small";
    has valid subdomain "as_specified";
    has invalid subdomain "not_in_list".

Datatype                                den_type
    is datatypeclass real;
    has value range minimum 0.0;
    has value range maximum 1.0E3;
    has value range resolution 10.0;
    has valid subdomain "as_specified";
    has invalid subdomain "abnormal".

Datatype                                volume_type
    is datatypeclass real;
    has value range minimum 0.0;
    has value range maximum 1.0E7;
    has value range resolution 1.0E4;
    has valid subdomain "as_specified";
    has invalid subdomain "abnormal".

Datatype                                mass_type
    is datatypeclass real;
    has value range minimum 0.0;
    has value range maximum 1.0E11;
    has value range resolution 1.0E8;
    has valid subdomain "as_specified";
    has invalid subdomain "abnormal".

```

Figure D4-2. System Analysis SDF for mass computer

```

s_packet                                Proc_weight_proc.

#include <stp.std>
Action                                weight_proc
    is actiontype internal;
    uses dataitem
        mass,
        gravity;
    produces dataitem
        report_weight;.

Dataitem                                mass
    is an instance of datatype mass_type.

Dataitem                                gravity
    is an instance of datatype grav_type.

Dataitem                                report_weight
    is an instance of datatype weight_type.

Datatype                                mass_type
    is datatypeclass real;
    has value range minimum 0.0;
    has value range maximum 1.0E11;
    has value range resolution 1.0E8;
    has valid subdomain "as_specified";
    has invalid subdomain "abnormal".

Datatype                                grav_type
    is datatypeclass real;
    has value range minimum 0.0;
    has value range maximum 20.0;
    has value range resolution 0.2;
    has valid subdomain "as_specified";
    has invalid subdomain "abnormal".

Datatype                                weight_type
    is datatypeclass real;
    has value range minimum 0.0;
    has value range maximum 9.0e99;
    has value range resolution 1.0;
    has valid subdomain "as_specified";
    has invalid subdomain "abnormal".

```

Figure D4-3. System Analysis SDF for weight computer

```

s_packet                               Mod_Main_pdl.

#include <stp.std>
Action                                Main_pdl
    is actiontype internal;
    uses dataitem
        width,
        height,
        length,
        gravity,
        density;
    produces dataitem
        report_weight;.

Dataitem                                width
    is an instance of datatype dim_type.

Dataitem                                height
    is an instance of datatype dim_type.

Dataitem                                length
    is an instance of datatype dim_type.

Dataitem                                gravity
    is an instance of datatype grav_type.

Dataitem                                density
    is an instance of datatype den_type.

Dataitem                                report_weight
    is an instance of datatype weight_type.

Datatype                                dim_type
    is datatypeclass real;
    has value range minimum 0.0;
    has value range maximum 1.0E2;
    has value range resolution 1.0;
    has valid subdomain "as_specified";
    has invalid subdomain "abnormal".

Datatype                                grav_type
    is datatypeclass real;
    has value range minimum 0.0;
    has value range maximum 20.0;
    has value range resolution 0.2;
    has valid subdomain "as_specified";
    has invalid subdomain "abnormal".

Datatype                                den_type
    is datatypeclass real;
    has value range minimum 0.0;
    has value range maximum 1.0E3;
    has value range resolution 10.0;
    has valid subdomain "as_specified";
    has invalid subdomain "abnormal".

Datatype                                weight_type
    is datatypeclass real;
    has value range minimum 0.0;
    has value range maximum 9.0e99;
    has value range resolution 1.0;
    has valid subdomain "as_specified";
    has invalid subdomain "abnormal".

```

Figure D4-4. System Design SDF for top level weight computer (Main)

```

s_packet                                Mod_volume_computer_pdl.

#include <stp.std>
Action                                volume_computer_pdl
    is actiontype internal;
    uses dataitem
        height,
        length,
        width;
    produces dataitem
        rs_volume;.

Dataitem                                height
    is an instance of datatype dim_type.

Dataitem                                length
    is an instance of datatype dim_type.

Dataitem                                width
    is an instance of datatype dim_type.

Dataitem                                rs_volume
    is an instance of datatype volume_type.

Datatype                                dim_type
    is datatypeclass real;
    has value range minimum 0.0;
    has value range maximum 1.0E2;
    has value range resolution 1.0;
    has valid subdomain "as_specified";
    has invalid subdomain "abnormal".

Datatype                                volume_type
    is datatypeclass real;
    has value range minimum 0.0;
    has value range maximum 1.0E7;
    has value range resolution 1.0E4;
    has valid subdomain "as_specified";
    has invalid subdomain "abnormal".

```

Figure D4-5. System Design SDF for volume computer

s_packet Mod_weight_computer_pdl.

#include <stp.std>

Action weight_computer_pdl
is actiontype internal;
uses dataitem
density,
gravity,
rs_volume;
produces dataitem
report_weight;.

Dataitem density
is an instance of datatype den_type.

Dataitem gravity
is an instance of datatype grav_type.

Dataitem rs_volume
is an instance of datatype volume_type.

Dataitem report_weight
is an instance of datatype weight_type.

Datatype den_type
is datatypeclass real;
has value range minimum 0.0;
has value range maximum 1.0E3;
has value range resolution 10.0;
has valid subdomain "as_specified";
has invalid subdomain "abnormal".

Datatype grav_type
is datatypeclass real;
has value range minimum 0.0;
has value range maximum 20.0;
has value range resolution 0.2;
has valid subdomain "as_specified";
has invalid subdomain "abnormal".

Datatype volume_type
is datatypeclass real;
has value range minimum 0.0;
has value range maximum 1.0E7;
has value range resolution 1.0E4;
has valid subdomain "as_specified";
has invalid subdomain "abnormal".

Datatype weight_type
is datatypeclass real;
has value range minimum 0.0;
has value range maximum 9.0e99;
has value range resolution 1.0;
has valid subdomain "as_specified";
has invalid subdomain "abnormal".

Figure D4-6. System Design SDF for mid level weight computer

```

s_packet                                     Mod_mass_computer.

#include <stp.std>
Action                                     mass_computer
    is actiontype internal;
    uses dataitem
        density,
        rs_volume;
    produces dataitem
        mass;.

Dataitem                                     density
    is an instance of datatype den_type.

Dataitem                                     rs_volume
    is an instance of datatype volume_type.

Dataitem                                     mass
    is an instance of datatype mass_type.

Datatype                                     den_type
    is datatypeclass real;
    has value range minimum 0.0;
    has value range maximum 1.0E3;
    has value range resolution 10.0;
    has valid subdomain "as_specified";
    has invalid subdomain "abnormal".

Datatype                                     volume_type
    is datatypeclass real;
    has value range minimum 0.0;
    has value range maximum 1.0E7;
    has value range resolution 1.0E4;
    has valid subdomain "as_specified";
    has invalid subdomain "abnormal".

Datatype                                     mass_type
    is datatypeclass real;
    has value range minimum 0.0;
    has value range maximum 1.0E11;
    has value range resolution 1.0E8;
    has valid subdomain "as_specified";
    has invalid subdomain "abnormal".

```

Figure D4-7. System Design SDF for mass computer

```

s_packet                                Mod_by_gravity.

#include <stp.std>
Action                                by_gravity
    is actiontype internal;
    uses dataitem
        gravity,
        mass;
    produces dataitem
        report_weight;.

Dataitem                                gravity
    is an instance of datatype grav_type.

Dataitem                                mass
    is an instance of datatype mass_type.

Dataitem                                report_weight
    is an instance of datatype weight_type.

Datatype                                grav_type
    is datatypeclass real;
    has value range minimum 0.0;
    has value range maximum 20.0;
    has value range resolution 0.2;
    has valid subdomain "as_specified";
    has invalid subdomain "abnormal".

Datatype                                mass_type
    is datatypeclass real;
    has value range minimum 0.0;
    has value range maximum 1.0E11;
    has value range resolution 1.0E8;
    has valid subdomain "as_specified";
    has invalid subdomain "abnormal".

Datatype                                weight_type
    is datatypeclass real;
    has value range minimum 0.0;
    has value range maximum 9.0e99;
    has value range resolution 1.0;
    has valid subdomain "as_specified";
    has invalid subdomain "abnormal".

```

Figure D4-8. System Design SDF for low level weight computer (by gravity)

D5. Tverify/Tdesign Reports for Main Process

The T output files are itemized in Table D5-1 for the SD extraction for the main module with file names, sizes, the numbers of pages shown, the T subunit generating the files and identifying names for the outputs. The numbers of pages shown are selected based on their instructive value. All of the output files are produced based on the input SDF file and its included files. A brief description of these output files follows:

drf.p	This file is the design rule file produced by Tverify for use by Tdesign based on the SDF. It lists the parameter names for each action, i.e., process or module and the value assignment rules to use for each datatype.
cref.rpt	This file is a cross reference report which list the SDF, all its include files, and various cross reference reports. (not shown)
verify.rpt	This file is a verification report which lists the SDF and any errors messages if applicable. The listing of the include files depends on software switches within the include files.(not shown)
sdmetric.rpt	This file has a list of metrics pertaining to the SDF processed by Tverify. Metrics reported include the numbers of actions, state transitions, conditions, dataitems, datatypes, and states.
sdf.p	This file has the SDF and all its include files,(not shown)
summary.rpt	The Test case Summary Report file shows the test cases with all input completely identified. (Only 2 of its 60 pages are shown)
catalog.rpt	The Test case Catalog Report is a preview of the itemized test cases. (Only 1 of its 10 pages are shown)
samples.rpt	The Samples Report shows the selected values for each variable going into the unit being tested.
design.rpt	The Test case design rule file is produced by Tdesign for use by Tprepare .

filename	file size	pages	T		
	lines	pages	shown	product	comments
drf.p	60	1	1	Tverify	Test design rule file
cref.rpt	544	11	0	Tverify	Cross Reference Report
verify.rpt	264	3	0	Tverify	Verification Report
sdmetric.rpt	21	1	1	Tverify	SDF Metric Report
sdf.p	410	9	0	Tverify	preprocessed SDF
summary.rpt	3260	60	2	Tdesign	Test case Summary Report
catalog.rpt	441	10	1	Tdesign	Test case Catalog Report
samples.rpt	101	2	2	Tdesign	Samples Report
design.rpt	84	1	1	Tdesign	Test case design rules

Table D5-1. T output files for weight computer

Table D5-1. T output files for weight computer

All files listed are labeled with their name at the beginning and end of their listings on the pages which follow.

```

## drf.p  Wed Oct 26 09:38:46 EDT 1994
/* T Design Rule Generation  Version 3.0
** Copyright (C) 1987-1992 Programming Environments, Inc.
*/

```

```

T_Packet          tpacket
s_packet          Mod_Main_pdl
.

```

```

CombinationRule   CR0001
action            Main_pdl;
singular          width,
                  height,
                  length,
                  gravity,
                  density;
.

```

```

SelectionRule     SR0001
datatype          dim_type;
reference         TBD;
valid             as_specified
                  with function, boundary, debug;
.

```

```

SelectionRule     SR0002
datatype          grav_type;
reference         TBD;
valid             as_specified
                  with function, boundary, debug;
.

```

```

SelectionRule     SR0003
datatype          den_type;
reference         TBD;
valid             as_specified
                  with function, boundary, debug;
.

```

```

pe_mark
## /home/eagle/banowetz/mail drf.p

```

```
## sdmetric.rpt Wed Oct 26 09:39:21 EDT 1994
T Software Description Metrics Version 3.0
Copyright (C) 1987-1992 Programming Environments, Inc.
```

```
s_packet: Mod_Main_pdl
unitdate: Wed Oct 26 09:21:03 1994
```

Total	Extra	Verified		Unverified		Dynamic
1	0	1		0		action(s)
0	0	0		0		statetransition(s)

Total	Extra	Verified		Unverified		Static
		In	Out	In	Out	
0	0	0	0	0	0	condition(s)
6	0	5	1	0	0	dataitem(s)
30	26	3	1	0	0	datatype(s)
0	0	0	0	0	0	state(s)

```
Deficiencies:          0
Inconsistencies:      0
## /home/eagle/banowetz/mail sdmetric.rpt
```

s_packet: Mod_Main_pdl
unitdate: Mon Sep 26 12:17:38 1994
t_packet: tpacket
casedate: Mon Sep 26 12:17:52 1994

#####

CASENAME 01000001
EXERCISES Main_pdl
IN STATE <unspecified>
PURPOSE all inputs at reference values
INPUT DATA
Name --- Value

width
--- 50.0
height
--- 50.0
length
--- 50.0
gravity
--- 10.0
density
--- 500.0

START BY <unspecified>
END BY <unspecified>
OUTPUT DATA
Name --- Value

report_weight
--- "<unspecified>"

TRANSITION <none>

#####

CASENAME 01000002
EXERCISES Main_pdl
IN STATE <unspecified>
PURPOSE all inputs at low boundary
INPUT DATA
Name --- Value

width
--- 0.0
height
--- 0.0
length
--- 0.0
gravity
--- 0.0
density
--- 0.0

START BY <unspecified>
END BY <unspecified>
OUTPUT DATA
Name --- Value

report_weight
--- "<unspecified>"

TRANSITION <none>

#####

CASENAME 01000003
EXERCISES Main_pdl
IN STATE <unspecified>
PURPOSE all inputs at high boundary
INPUT DATA
Name --- Value

width
--- 1.0E2
height
--- 1.0E2
length
--- 1.0E2
gravity
--- 20.0
density
--- 1.0E3

START BY <unspecified>
END BY <unspecified>
OUTPUT DATA
Name --- Value

report_weight
--- "<unspecified>"

TRANSITION <none>

#####

CASENAME 01000004
EXERCISES Main_pdl
IN STATE <unspecified>
PURPOSE to probe width at valid as_specified low_bound
INPUT DATA
Name --- Value

width
--- 0.0
height
--- 50.0
length
--- 50.0
gravity
--- 10.0
density
--- 500.0

START BY <unspecified>
END BY <unspecified>
OUTPUT DATA
Name --- Value

report_weight
summary.rpt

s_packet: Mod_Main_pdl
unitdate: Mon Sep 26 12:17:38 1994
t_packet: tpacket
casedate: Mon Sep 26 12:17:52 1994

Casename Purpose (+ exercises action, - fails to exercise action)

01000001	+	action Main_pdl state <unspecified> dataitem all at reference
01000002	+	action Main_pdl state <unspecified> dataitem all at low boundary
01000003	+	action Main_pdl state <unspecified> dataitem all at high boundary
01000004	+	action Main_pdl state <unspecified> dataitem width (valid as_specified low_bound)
01000005	+	action Main_pdl state <unspecified> dataitem width (valid as_specified high_bound)
01000006	+	action Main_pdl state <unspecified> dataitem width (valid as_specified low_debug)
01000007	+	action Main_pdl state <unspecified> dataitem width (valid as_specified high_debug)
01000008	+	action Main_pdl state <unspecified> dataitem width (invalid above_bounds above_bounds)
01000009	+	action Main_pdl state <unspecified> dataitem width (invalid below_bounds below_bounds)
01000010	+	action Main_pdl state <unspecified> dataitem width (invalid out_of_type out_of_type_1)
01000011	+	action Main_pdl state <unspecified> dataitem width (invalid out_of_type out_of_type_2)
01000012	+	action Main_pdl state <unspecified> dataitem width (invalid out_of_type out_of_type_3)
01000013	+	action Main_pdl state <unspecified> dataitem height (valid as_specified low_bound)
01000014	+	action Main_pdl state <unspecified> dataitem height (valid as_specified high_bound)

s_packet: Mod_Main_pdl
unitdate: Mon Sep 26 12:17:38 1994
t_packet: tpacket
casedate: Mon Sep 26 12:17:52 1994

density

Index	SubDomain	Equiv.Class	Label	Value
[1]	valid	as_specified	reference	500.0
[2]	valid	as_specified	low_bound	0.0
[3]	valid	as_specified	high_bound	1.0E3
[4]	valid	as_specified	low_debug	10.0
[5]	valid	as_specified	high_debug	990.0
[6]	invalid	above_bounds	above_bounds	1010.0
[7]	invalid	below_bounds	below_bounds	-10.0
[8]	invalid	out_of_type	out_of_type_1	9
[9]	invalid	out_of_type	out_of_type_2	'a'
[10]	invalid	out_of_type	out_of_type_3	"<not_in_list>"

gravity

Index	SubDomain	Equiv.Class	Label	Value
[1]	valid	as_specified	reference	10.0
[2]	valid	as_specified	low_bound	0.0
[3]	valid	as_specified	high_bound	20.0
[4]	valid	as_specified	low_debug	0.20
[5]	valid	as_specified	high_debug	19.80
[6]	invalid	above_bounds	above_bounds	20.20
[7]	invalid	below_bounds	below_bounds	-0.20
[8]	invalid	out_of_type	out_of_type_1	9
[9]	invalid	out_of_type	out_of_type_2	'a'
[10]	invalid	out_of_type	out_of_type_3	"<not_in_list>"

height

Index	SubDomain	Equiv.Class	Label	Value
[1]	valid	as_specified	reference	50.0
[2]	valid	as_specified	low_bound	0.0
[3]	valid	as_specified	high_bound	1.0E2
[4]	valid	as_specified	low_debug	1.0
[5]	valid	as_specified	high_debug	99.0
[6]	invalid	above_bounds	above_bounds	101.0
[7]	invalid	below_bounds	below_bounds	-1.0
[8]	invalid	out_of_type	out_of_type_1	9
[9]	invalid	out_of_type	out_of_type_2	'a'
[10]	invalid	out_of_type	out_of_type_3	"<not_in_list>"

length

Index	SubDomain	Equiv.Class	Label	Value
[1]	valid	as_specified	reference	50.0
[2]	valid	as_specified	low_bound	0.0
[3]	valid	as_specified	high_bound	1.0E2
[4]	valid	as_specified	low_debug	1.0
[5]	valid	as_specified	high_debug	99.0
[6]	invalid	above_bounds	above_bounds	101.0
[7]	invalid	below_bounds	below_bounds	-1.0
[8]	invalid	out_of_type	out_of_type_1	9

[9]	invalid	out_of_type	out_of_type_2	'a'
[10]	invalid	out_of_type	out_of_type_3	"<not_in_list>"

width

Index	SubDomain	Equiv.Class	Label	Value
[1]	valid	as_specified	reference	50.0
[2]	valid	as_specified	low_bound	0.0
[3]	valid	as_specified	high_bound	1.0E2
[4]	valid	as_specified	low_debug	1.0
[5]	valid	as_specified	high_debug	99.0
[6]	invalid	above_bounds	above_bounds	101.0
[7]	invalid	below_bounds	below_bounds	-1.0
[8]	invalid	out_of_type	out_of_type_1	9
[9]	invalid	out_of_type	out_of_type_2	'a'
[10]	invalid	out_of_type	out_of_type_3	"<not_in_list>"

- saving samples in test design data base

samples.rpt

=====
Translation
=====

```

1  /* T Design Rule Generation Version 3.0
2  ** Copyright (C) 1987-1992 Programming Environments, Inc.
3  */
4
5  T_Packet                tpacket
6    s_packet              Mod_Main_pdl
7    .
8
9  CombinationRule         CR0001
10    action                Main_pdl;
11    singular              width,
12                          height,
13                          length,
14                          gravity,
15                          density;
16    .
17
18  SelectionRule           SR0001
19    datatype              dim_type;
20    reference              TBD;
21    valid                  as_specified
22                          with function, boundary, debug;
23    .
24
25  SelectionRule           SR0002
26    datatype              grav_type;
27    reference              TBD;
28    valid                  as_specified
29                          with function, boundary, debug;
30    .
31
32  SelectionRule           SR0003
33    datatype              den_type;
34    reference              TBD;
35    valid                  as_specified
36                          with function, boundary, debug;
37    .
38
39                          pe_mark

```

- finished translation with 5 recognizable TDRL sentences out of 5

=====
Interpretation
=====

s_packet: Mod_Main_pdl
unitdate: Mon Sep 26 12:17:38 1994
t_packet: tpacket
casedate: Mon Sep 26 12:17:52 1994

- saving rules in test design data base

APPENDIX E. ASQ-212 Samples

E1. StP Output

E1.1 Data Flow Diagrams (DFDs)

E1.2 Data Structure Diagrams (DSDs)

E1.3 Software Description Files (SDFs)

E2. T Output For Process 1.1

E1. StP Output

This appendix illustrates a sample of StP diagrams and Test case results for the Navy ASQ-212 project, the Tactical Mission Software (TMS) segment, the Navigation and Steering Computer Software Configuration Item (CSCI), and the Determine_Steering_Mode process.

E1.1 Data Flow Diagrams (DFDs)

The DFDs needed to support Determine_Steering_Mode (process 1.1) for the Navigation and Steering CSCI are presented in the following figures:

Figure E1-1. Context Diagram for Navigation Steering (top level/ supporting text omitted) Navigation Steering is a Computer Software Configuration Item (CSCI).

Figure E1-2. Level 0 DFD (supporting text omitted)

Figure E1-3. Level 1 DFD (supporting text omitted)

Project: /staff/imet/banowetz/STP/
 System: vb212
 Diagram: top

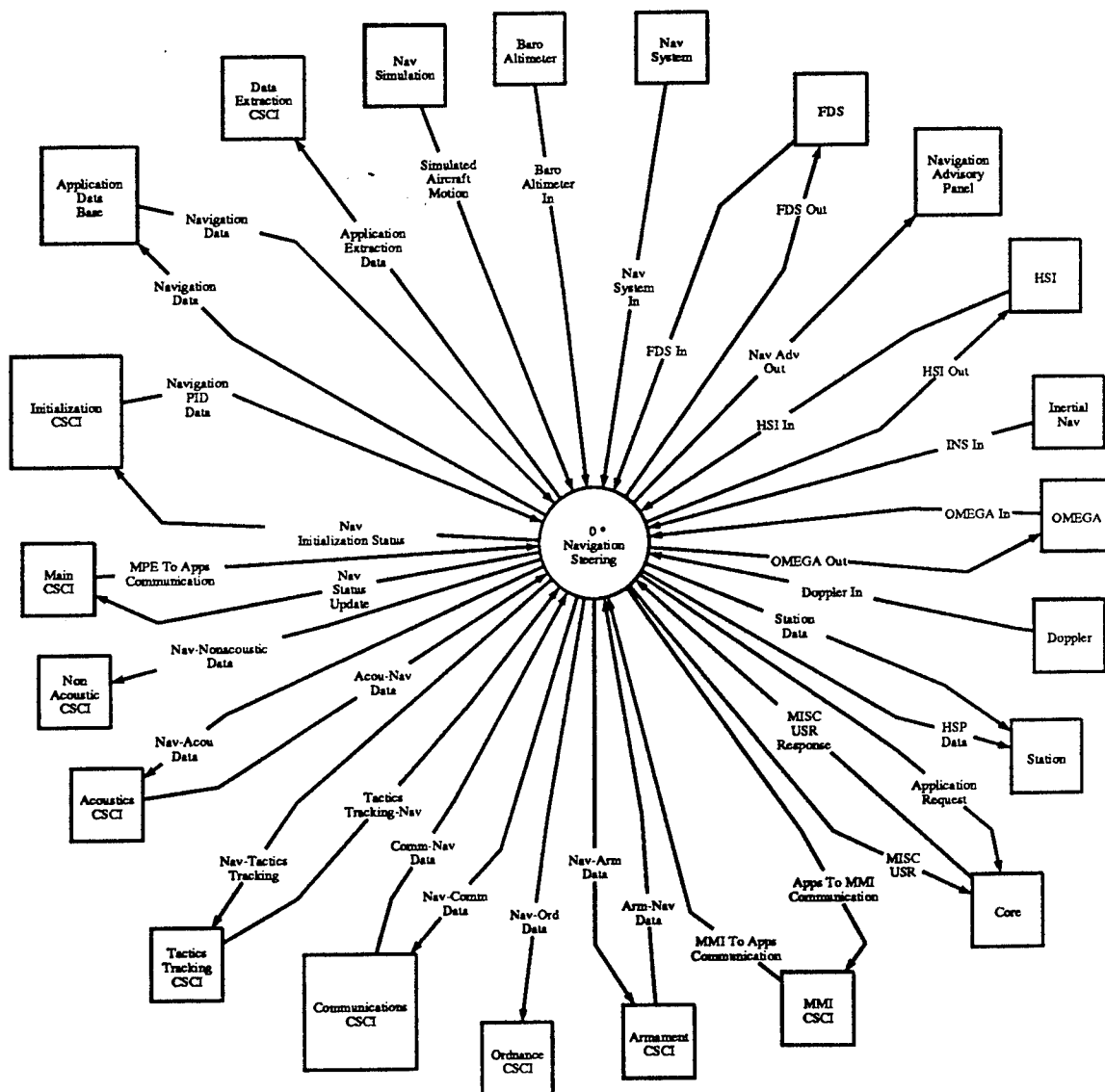


Figure E1-1. Context Diagram for Navigation Steering
 (top level/ supporting text omitted)

```
Project: /staff/imet/banowetz/STP/
System: vb212
Diagram: 0
```

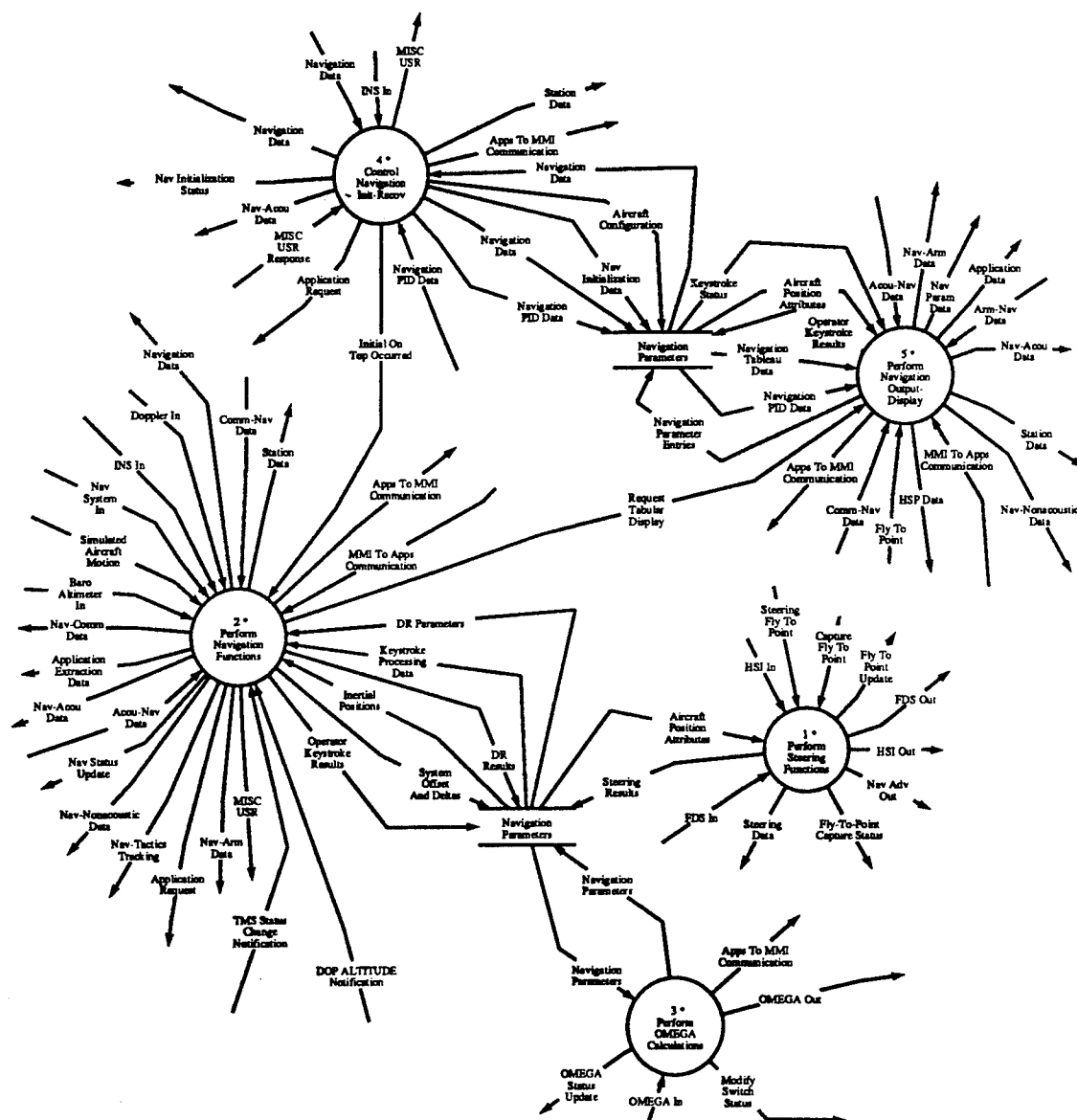


Figure E1-2. Level 0 DFD

```
Project:  /staff/imet/banowetz/STP/
System:   vb212
Diagram:  1
```

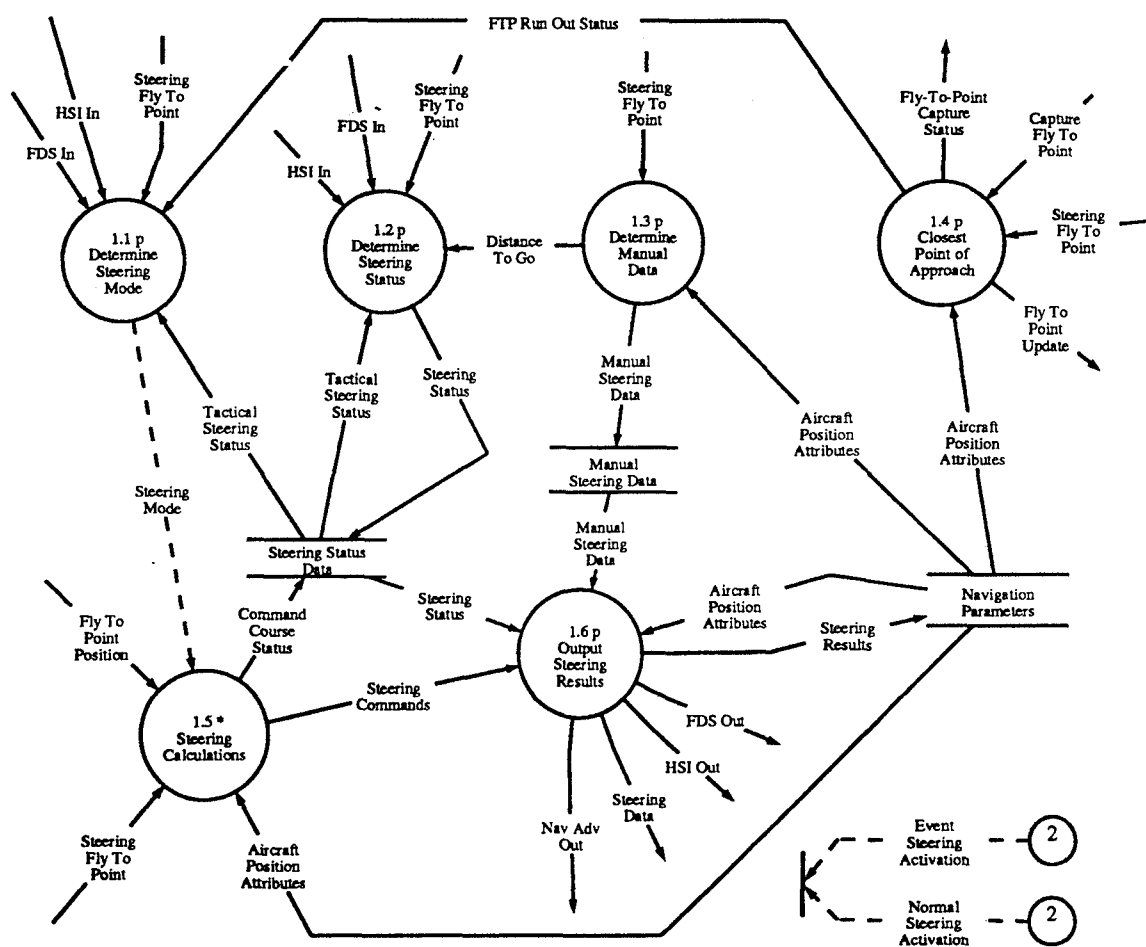


Figure E1-3. Level 1 DFD

E1.2 Data Structure Diagrams (DSDs)

Figure E1-4 has the DSDs for the components of process 1.1 (changes hand marked). The annotations provided with the DSDs provide additional descriptive information on the data items including type and, where applicable, values, or domain. The types for data items used in process 1.1 were changed to be compatible with T. In some cases, this was merely a change in case (Boolean to boolean, String to string, etc.). More typically, the type numeric was used and had to be changed to an added defined type such as Lat_Type, Lon_Type, Radius_Type, or Angle_Type for the Fly_To_Point data structure. In the case of "FTP_Runout_Status", an enumeration object had to be built in the DSD itself to replace an object represented as a list of possible values. The changes to the diagrams are hand annotated.

Project: /staff/imet/banowetz/STP/
System: vb212
Diagram: FDS_In

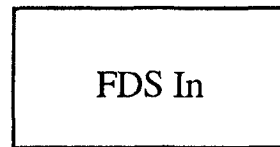


Figure E1-4. Data Structure Diagrams for the components of process 1.1

Diagram FDS_In:

Element FDS_In:

Note DataDefinition FDS_In

Accuracy: N/A
Frequency: N/A
Legality: N/A
Name: FDS_In
Priority: N/A
Precision: N/A
Representation: N/A
Type: boolean
Units: N/A

This interface provides the status of the Computer Track
(COMP TRK CONT) switch on the Flight Direction System.

Note Document IRS
Document: IRS

Note Document SRS
Document: SRS

Project: /staff/imet/banowitz/STP/
System: vb212
Diagram: HSI_In



HSI In

Diagram HSI_In:

Element HSI_In:

Note DataDefinition HSI_In

Accuracy: N/A
Frequency: N/A
Legality: N/A
Name: HSI_In
Priority: N/A
Precision: N/A
Representation: N/A
Type: boolean
Units: N/A

Pilot input from the Horizontal Situation Indicator.
Providing the status of the Pilot Course Selection
switch.

Note Document IRS
Document: IRS

Note Document SRS
Document: SRS

Diagram HSI_In:

Element HSI_In:

Note DataDefinition HSI_In

Accuracy: N/A

Frequency: N/A

Legality: N/A

Name: HSI_In

Priority: N/A

Precision: N/A

Representation: N/A

Type: boolean

Units: N/A

Pilot input from the Horizontal Situation Indicator.
Providing the status of the Pilot Course Selection
switch.

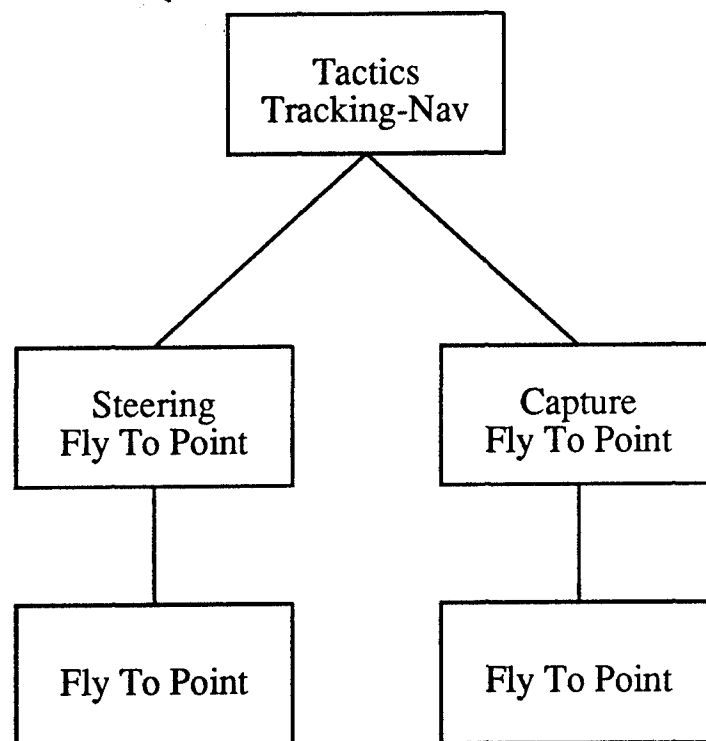
Note Document IRS

Document: IRS

Note Document SRS

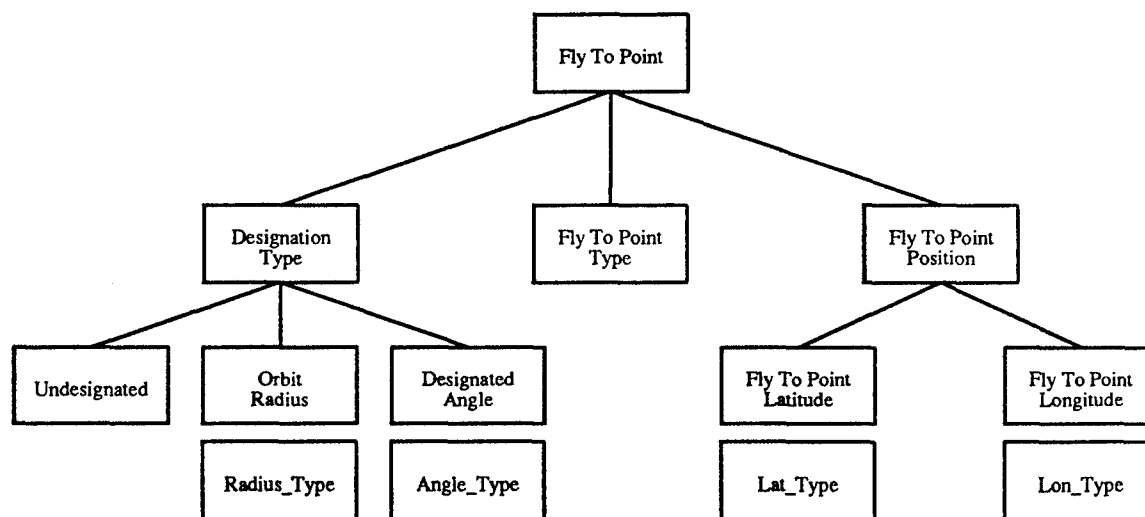
Document: SRS

Project: /staff/imet/banowetz/STP/
System: vb212
Diagram: Tactics_Tracking-Nav



Data Structure Diagram Tactics_Tracking-Nav

Project: /staff/imet/banowetz/STP/
System: vb212
Diagram: Fly_To_Point



Data Structure Diagram Fly_To_Point

Diagram Fly_To_Point:

Data Structure Designation_Type:

Note DataDefinition Designation_Type

Accuracy: N/A
Frequency: N/A
Legality: N/A
Name: Designation_Type
Priority: N/A
Precision: N/A
Representation: N/A
Units: N/A

Indicates if the FTP is designated or undesignated.

Note Document SRS
Document: SRS

Data Structure Fly_To_Point:

Note DataDefinition Fly_To_Point

Accuracy: N/A
Frequency: N/A
Legality: N/A
Name: Fly_To_Point
Priority: N/A
Precision: N/A
Representation: N/A
Units: N/A

This interface provides the FTP type and position for the highest priority FTP.

Note Document IRS
Document: IRS

Note Document SRS
Document: SRS

Data Structure Fly_To_Point_Position:

Note DataDefinition Fly_To_Point_Position

Accuracy: N/A
Frequency: N/A
Legality: N/A
Name: Fly_To_Point_Position
Priority: N/A
Precision: N/A
Representation: N/A
Units: N/A

This interface provides the latitude and longitude for the FTP.

Note Document IRS
Document: IRS

Element Angle_Type:

Note DataDefinition Angle_Type

Name: Angle_Type
Type: real
Units: degrees

Note Domain
Increment: 1.0
Minimum: 0.0
Maximum: 360.0

Element Designated_Angle:

Note DataDefinition Designated_Angle
Accuracy: N/A
Constraint: 0..360
Frequency: N/A
Legality: N/A
Name: Designated_Angle
Priority: N/A
Precision: N/A
Representation: N/A
Type: Angle_Type
Units: Degrees

The designated track angle at the FTP.

Note Document IRS
Document: IRS

Element Fly_To_Point_Latitude:

Note DataDefinition Fly_To_Point_Latitude
Accuracy: N/A
Constraint: [-90.0,90.0]
Frequency: N/A
Legality: N/A
Name: Fly_To_Point_Latitude
Priority: N/A
Precision: N/A
Representation: N/A
Type: Lat_Type
Units: Degrees

The FTP latitude.

Note Document IRS
Document: IRS

Element Fly_To_Point_Longitude:

Note DataDefinition Fly_To_Point_Longitude
Accuracy: N/A
Constraint: (-180,180]
Frequency: N/A
Legality: N/A
Name: Fly_To_Point_Longitude
Priority: N/A
Precision: N/A
Representation: N/A
Type: Lon_Type
Units: Degrees

The FTP longitude.

Note Document IRS
Document: IRS

Element Lat_Type:

Note DataDefinition Lat_Type
Name: Lat_Type
Type: real

Units: degrees

Note Domain

Increment: 1.0
Minimum: -90.0
Maximum: 90.0

Element Lon_Type:

Note DataDefinition Lon_Type

Name: Lon_Type
Type: real
Units: degrees

Note Domain

Increment: 1.0
Minimum: -180.0
Maximum: 180.0

Element Orbit_Radius:

Note DataDefinition Orbit_Radius

Accuracy: N/A
Constraint: 2000..200475
Frequency: N/A
Legality: N/A
Name: Orbit_Radius
Priority: N/A
Precision: N/A
Representation: N/A
Type: Radius_Type
Units: Yards

Radius of the orbit circle.

Note Document SRS

Document: SRS

Element Radius_Type:

Note DataDefinition Radius_Type

Name: Radius_Type
Type: real
Units: yards

Note Domain

Increment: 1.0
Minimum: 2000.0
Maximum: 200475.0

Element Undesignated:

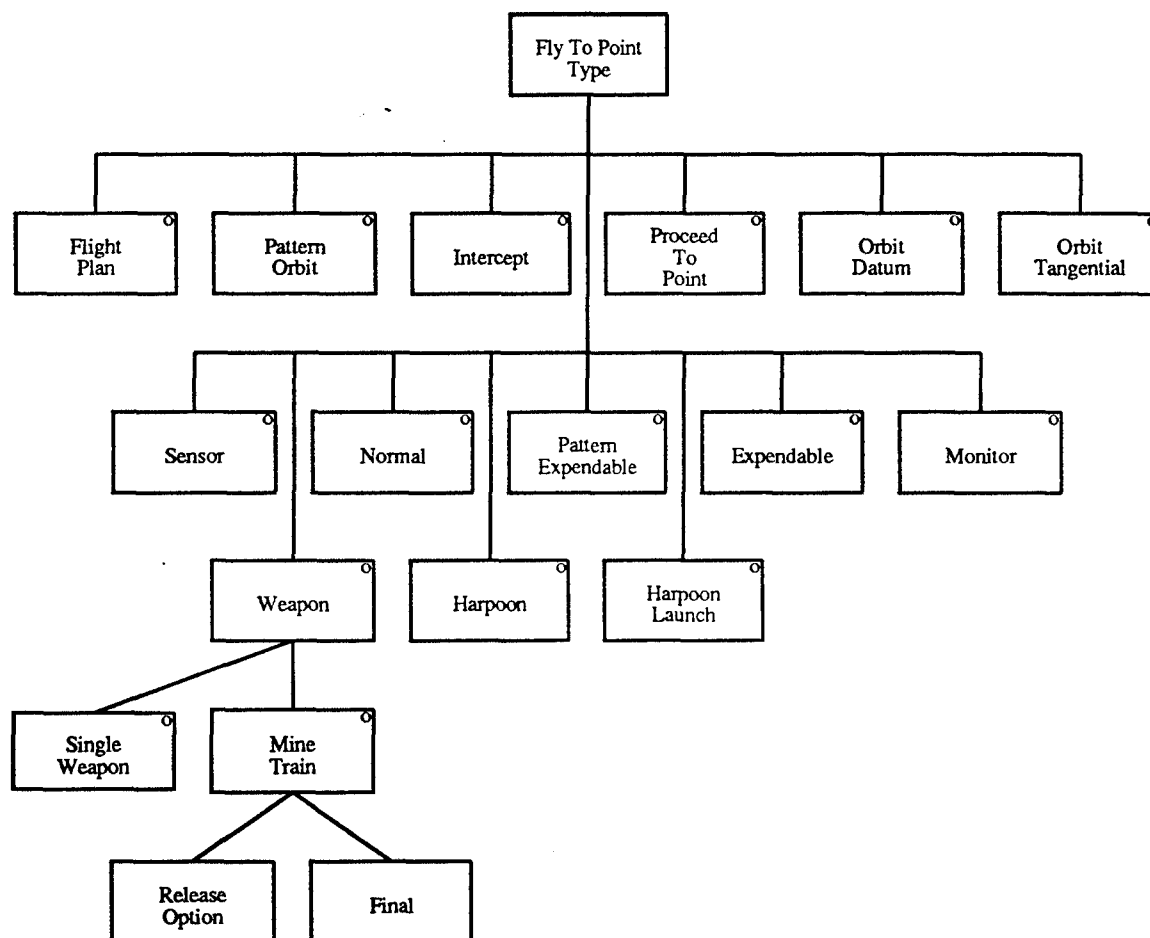
Note DataDefinition Undesignated

Accuracy: N/A
Frequency: N/A
Legality: N/A
Name: Undesignated
Priority: N/A
Precision: N/A
Representation: N/A
Type: boolean
Units: N/A

Provides the indication as to whether the FTP can be captured using any heading.

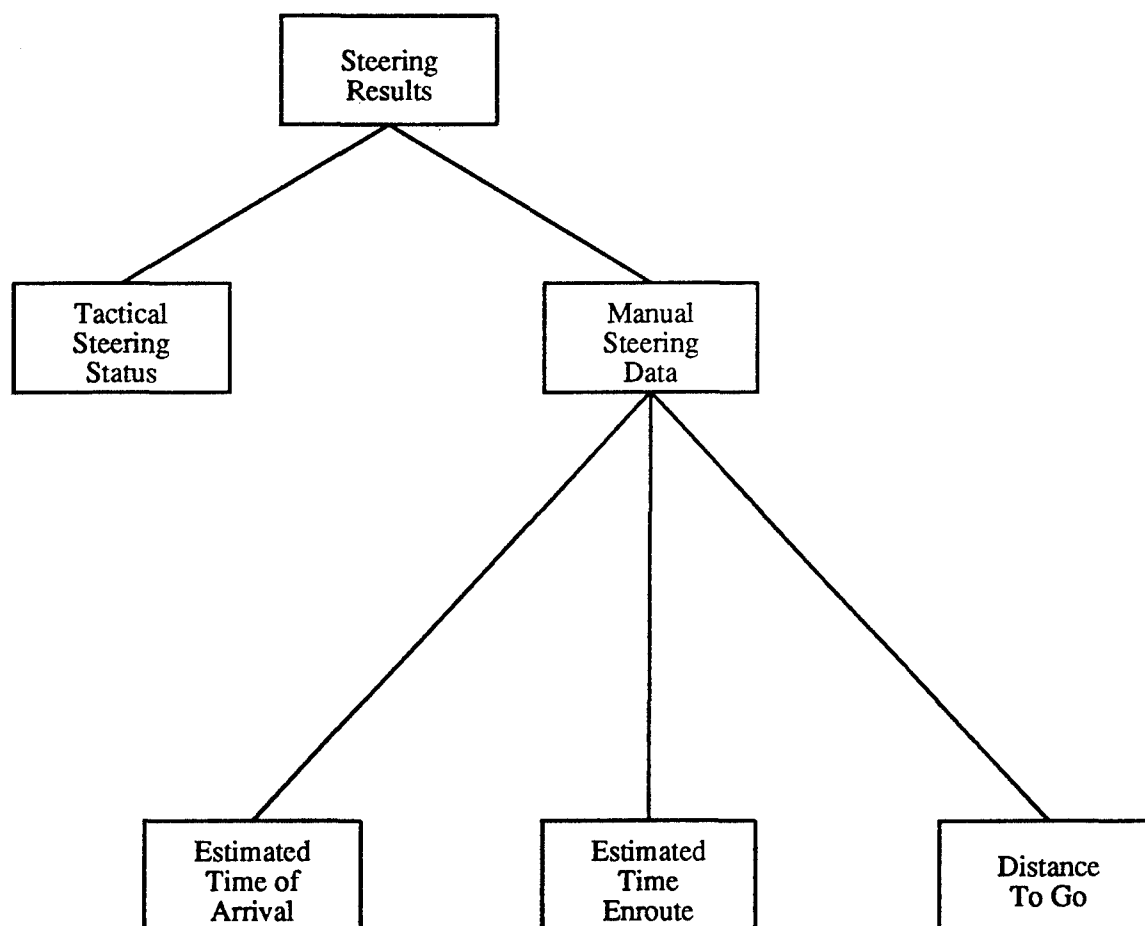
Note Document IRS

Project: /staff/imet/banowetz/STP/
System: vb212
Diagram: Fly_To_Point_Type



Data Structure Diagram Fly_To_Point_Type

Project: /staff/imet/banowetz/STP/
System: vb212
Diagram: Steering_Results



Data Structure Diagram Steering_Results

Diagram Steering_Results:

Note diag
Name: Steering_Results

Note Document SRS
Document: SRS

Data Structure Manual_Steering_Data:

Note DataDefinition Manual_Steering_Data
Accuracy: N/A
Frequency: N/A
Legality: N/A
Name: Manual_Steering_Data
Priority: N/A
Precision: N/A
Representation: N/A
Units: N/A
The Steering Data data that is independent of
which steering mode is currently assigned.

Note Document SRS
Document: SRS

Data Structure Steering_Results:

Note DataDefinition Steering_Results
Accuracy: N/A
Frequency: N/A
Legality: N/A
Name: Steering_Results
Priority: N/A
Precision: N/A
Representation: N/A
Units: N/A
The results of the steering function.

Note Document SRS
Document: SRS

Element Tactical_Steering_Status:

Note DataDefinition Tactical_Steering_Status
Accuracy: N/A
Frequency: N/A
Legality: N/A
Name: Tactical_Steering_Status
Priority: N/A
Precision: N/A
Representation: N/A
Type: boolean
Units: N/A
The tactical steering status as determined by
the steering process.

Note Document SRS
Document: SRS

Project: /staff/imet/banowetz/STP/
System: vb212
Diagram: FTP_Run_Out_Status

FTP Run Out
Status

Diagram FTP_Run_Out_Status:

Element FTP_Run_Out_Status:

Note DataDefinition FTP_Run_Out_Status

Accuracy: N/A

Frequency: N/A

Legality: N/A

Name: FTP_Run_Out_Status

Priority: N/A

Precision: N/A

Representation: N/A

Type: string

Units: N/A

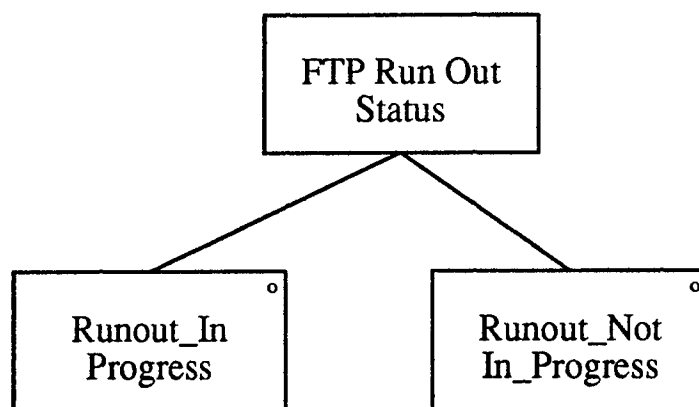
Value: "Runout_In_Progress"

Value: "Runout_Not_In_Progress"

This data indicates whether or not a FTP is in runout.

old copy

Project: /staff/imet/banowetz/STP/
System: vb212
Diagram: FTP_Run_Out_Status



added

Data Structure Diagram FTP_Run_Out_Status

Diagram FTP_Run_Out_Status:

Data Structure FTP_Run_Out_Status:

Note DataDefinition FTP_Run_Out_Status

Accuracy: N/A

Frequency: N/A

Legality: N/A

Name: FTP_Run_Out_Status

Priority: N/A

Precision: N/A

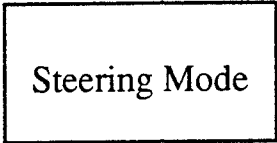
Representation: N/A

Units: N/A

This data indicates whether or not a FTP is in runout.

New way

Project: /staff/imet/banowetz/STP/
System: vb212
Diagram: Steering_Mode



Steering Mode

Data Structure Diagram Steering_Mode

Diagram Steering_Mode:

Element Steering_Mode:

Note DataDefinition Steering_Mode

Accuracy: N/A

Frequency: N/A

Legality: N/A

Name: Steering_Mode

Priority: N/A

Precision: N/A

Representation: N/A

Type: String

Units: N/A

Value: "Designated_Steering"

Value: "Great_Circle"

Value: "Manual"

Value: "Undesignated_Steering"

Note Document SRS

Document: SRS

E1.3 Software Description File (SDF)

Figure E1-5 has the SDF file for process 1.1 (Determine Steering Mode). The SDF action `Determine_Steering_Mode` is produced from the extraction of process 1.1. The dataitems are listed next in the order used by the action. The datatypes follow in the order used. Types not defined here are defined in the file `<stp.std>` included at the top of the file.

```

s_packet                                Proc_Determine_Steering_Mode.

#include <stp.std>
Action                                Determine_Steering_Mode
    is actiontype internal;
    uses dataitem
        Fly_To_Point_Latitude,
        Fly_To_Point_Longitude,
        Fly_To_Point_Type,
        Undesignated,
        Orbit_Radius,
        Designated_Angle,
        Tactical_Steering_Status,
        FTP_Run_Out_Status,
        HSI_In,
        FDS_In;
    produces dataitem
        Steering_Mode;.

Dataitem                                Fly_To_Point_Latitude
    is an instance of datatype Lat_Type.

Dataitem                                Fly_To_Point_Longitude
    is an instance of datatype Lon_Type.

Dataitem                                Fly_To_Point_Type
    is an instance of datatype Fly_To_Point_Type_t.

Dataitem                                Undesignated
    is an instance of datatype boolean_string.

Dataitem                                Orbit_Radius
    is an instance of datatype Radius_Type.

Dataitem                                Designated_Angle
    is an instance of datatype Angle_Type.

Dataitem                                Tactical_Steering_Status
    is an instance of datatype boolean_string.

Dataitem                                FTP_Run_Out_Status
    is an instance of datatype FTP_Run_Out_Status_t.

Dataitem                                HSI_In
    is an instance of datatype boolean_string.

Dataitem                                FDS_In
    is an instance of datatype boolean_string.

Dataitem                                Steering_Mode
    is an instance of datatype DefaultString.

Datatype                                Lat_Type
    is datatypeclass real;
    has value range minimum -90.0;
    has value range maximum 90.0;
    has value range resolution 1.0;
    has valid subdomain "as_specified";
    has invalid subdomain "abnormal".

```

Figure E1-5. SDF file for process 1.1 (Determine Steering Mode)

Datatype Lon_Type
is datatypeclass real;
has value range minimum -180.0;
has value range maximum 180.0;
has value range resolution 1.0;
has valid subdomain "as_specified";
has invalid subdomain "abnormal".

Datatype Fly_To_Point_Type_t
is datatypeclass string;
has values
 "Orbit_Datum",
 "Monitor",
 "Orbit_Tangential",
 "Flight_Plan",
 "Sensor",
 "Pattern_Orbit",
 "Weapon",
 "Normal",
 "Intercept",
 "Harpoon",
 "Pattern_Expendable",
 "Proceed_To_Point",
 "Harpoon_Launch",
 "Expendable";
has valid subdomain "as_specified";
has invalid subdomain "not_in_list".

Datatype Radius_Type
is datatypeclass real;
has value range minimum 2000.0;
has value range maximum 200475.0;
has value range resolution 1.0;
has valid subdomain "as_specified";
has invalid subdomain "abnormal".

Datatype Angle_Type
is datatypeclass real;
has value range minimum 0.0;
has value range maximum 360.0;
has value range resolution 1.0;
has valid subdomain "as_specified";
has invalid subdomain "abnormal".

Datatype FTP_Run_Out_Status_t
is datatypeclass string;
has values
 "Runout_In_Progress",
 "Runout_Not_In_Progress";
has valid subdomain "as_specified";
has invalid subdomain "not_in_list".

E2. T Output for Process 1.1

The T output files are itemized in Table E2-1 for the SA extraction for the 1.1 process with file names, sizes, the numbers of pages shown, the T subunit generating the files and identifying names for the outputs. The numbers of pages shown are selected based on their instructive value. All of the output files are produced based on the input SDF file and its included files. See Appendix D for a brief description of these output files.

filename	file size	pages	T		
	lines	pages	shown	product	comments
drf.p	72	2	2	Tverify	Test design rule file
cref.rpt	559	11	0	Tverify	Cross Reference Report
verify.rpt	190	4	0	Tverify	Verification Report
sdmetric.rpt	21	1	1	Tverify	SDF Metric Report
sdf.p	482	9	0	Tverify	preprocessed SDF
summary.rpt	2310	40	2	Tdesign	Test case Summary Report
catalog.rpt	249	5	1	Tdesign	Test case Catalog Report
samples.rpt	128	2	2	Tdesign	Samples Report
design.rpt	93	2	2	Tdesign	Test case design rules

Table E2-1. T output files for a Navigation-Steering Process

The most interesting of the above files are samples.rpt and summary.rpt. The samples.rpt file shows the selected values for each variable going into the unit being tested. The summary.rpt files shows the test cases with all input completely identified. (Only 2 of its 40 pages are shown). The files verify.rpt, sdf.p and cref.rpt are not shown since they are little more than a repeat or an expansion of the SDF file. All files are labeled with their name at the beginning and end of their listings on the pages which follow.

```

drf.p  Mon Sep 26 11:15:37 EDT 1994
/* T Design Rule Generation Version 3.0
** Copyright (C) 1987-1992 Programming Environments, Inc.
*/

```

```

T_Packet      tpacket
  s_packet    Proc_Determine_Steering_Mode
.

```

```

CombinationRule  CR0001
  action         Determine_Steering_Mode_1;
  singular       Fly_To_Point_Latitude,
                 Orbit_Radius,
                 Designated_Angle,
                 Tactical_Steering_Status,
                 FTP_Run_Out_Status,
                 HSI_In,
                 FDS_In;
.

```

```

CombinationRule  CR0002
  action         Determine_Steering_Mode_2;
  singular       Fly_To_Point_Longitude,
                 Fly_To_Point_Type,
                 Undesignated,
                 Orbit_Radius,
                 Designated_Angle,
                 Tactical_Steering_Status,
                 FTP_Run_Out_Status,
                 HSI_In,
                 FDS_In;
.

```

```

SelectionRule    SR0001
  datatype       boolean_string;
  reference      TBD;
  valid          as_specified
                with function, boundary, debug;
.

```

```

SelectionRule    SR0002
  datatype       Lat_Type;
  reference      TBD;
  valid          as_specified
                with function, boundary, debug;
.

```

```

SelectionRule    SR0003
  datatype       Lon_Type;
  reference      TBD;
  valid          as_specified
                with function, boundary, debug;
.

```

```

SelectionRule    SR0004
  datatype       Fly_To_Point_Type_t;
  reference      TBD;
  valid          as_specified
                with function, boundary, debug;
.

```

```

SelectionRule    SR0005
  datatype       Radius_Type;
  reference      TBD;
  valid          as_specified
                with function, boundary, debug;

```

.
SelectionRule SR0006
 datatype Angle_Type;
 reference TBD;
 valid as_specified
 with function, boundary, debug;

.
SelectionRule SR0007
 datatype FTP_Run_Out_Status_t;
 reference TBD;
 valid as_specified
 with function, boundary, debug;

.
drf.p pe_mark

sdmetric.rpt Fri Sep 23 15:22:05 EDT 1994
 T Software Description Metrics Version 3.0
 Copyright (C) 1987-1992 Programming Environments, Inc.

s_packet: Proc Determine Steering Mode
 unitdate: Fri Sep 23 11:51:31 1994

Total	Extra	Verified	Unverified	Dynamic
2	0	2	0	action(s)
0	0	0	0	statetransition(s)

Total	Extra	Verified In Out	Unverified In Out	Static
0	0	0 0	0 0	condition(s)
11	0	10 1	0 0	dataitem(s)
32	24	7 1	0 0	datatype(s)
0	0	0 0	0 0	state(s)

Deficiencies: 0
 Inconsistencies: 0
 sdmetric.rpt

summary.rpt Fri Sep 23 15:23:23 EDT 1994
T Test Case Definitions Version 3.0
Copyright (C) 1987-1992 Programming Environments, Inc.

s_packet: Proc_Determine_Steering_Mode
unitdate: Fri Sep 23 11:51:31 1994
t_packet: tpacket
casedate: Fri Sep 23 11:51:35 1994

#####

CASENAME 01000001
EXERCISES Determine_Steering_Mode_1
IN STATE <unspecified>
PURPOSE all inputs at reference values
INPUT DATA
Name --- Value

Fly_To_Point_Latitude
--- 0.0
Orbit_Radius
--- 101237.0
Designated_Angle
--- 180.0
Tactical_Steering_Status
--- "FALSE"
FTP_Run_Out_Status
--- "Runout_Not_In_Progress"
HSI_In
--- "FALSE"
FDS_In
--- "FALSE"

START BY <unspecified>
END BY <unspecified>
OUTPUT DATA
Name --- Value

Steering_Mode
--- "<unspecified>"

TRANSITION <none>

#####

CASENAME 01000002
EXERCISES Determine_Steering_Mode_1
IN STATE <unspecified>
PURPOSE all inputs at low boundary
INPUT DATA
Name --- Value

Fly_To_Point_Latitude
--- -90.0
Orbit_Radius
--- 2000.0
Designated_Angle
--- 0.0
Tactical_Steering_Status
--- "TRUE"
FTP_Run_Out_Status
--- "Runout_In_Progress"
HSI_In
--- "TRUE"
FDS_In
--- "TRUE"

START BY <unspecified>

END BY <unspecified>

OUTPUT DATA

Name --- Value

Steering_Mode

--- "<unspecified>"

TRANSITION

<none>

#####

CASENAME 01000003

EXERCISES Determine_Steering_Mode_1

IN STATE <unspecified>

PURPOSE all inputs at high boundary

INPUT DATA

Name --- Value

Fly_To_Point_Latitude

--- 90.0

Orbit_Radius

--- 200475.0

Designated_Angle

--- 360.0

Tactical_Steering_Status

--- "FALSE"

FTP_Run_Out_Status

--- "Runout_Not_In_Progress"

HSI_In

--- "FALSE"

FDS_In

--- "FALSE"

START BY <unspecified>

END BY <unspecified>

OUTPUT DATA

Name --- Value

Steering_Mode

--- "<unspecified>"

TRANSITION

<none>

#####

CASENAME 01000004

EXERCISES Determine_Steering_Mode_1

IN STATE <unspecified>

PURPOSE to probe Fly_To_Point_Latitude at valid as_specified low_bound

INPUT DATA

Name --- Value

Fly_To_Point_Latitude

--- -90.0

Orbit_Radius

--- 101237.0

Designated_Angle

--- 180.0

Tactical_Steering_Status

--- "FALSE"

FTP_Run_Out_Status

--- "Runout_Not_In_Progress"

HSI_In

--- "FALSE"

FDS_In

--- "FALSE"

START BY <unspecified>

catalog.rpt Fri Sep 23 15:24:28 EDT 1994
T Test Catalog Version 3.0
Copyright (C) 1987-1992 Programming Environments, Inc.

s_packet: Proc Determine Steering Mode
unitdate: Fri Sep 23 11:51:31 1994
t_packet: tpacket
casedate: Fri Sep 23 11:51:35 1994

Casename	Purpose (+ exercises action, - fails to exercise action)
01000001	+ action Determine_Steering_Mode_1 state <unspecified> dataitem all at reference
01000002	+ action Determine_Steering_Mode_1 state <unspecified> dataitem all at low boundary
01000003	+ action Determine_Steering_Mode_1 state <unspecified> dataitem all at high boundary
01000004	+ action Determine_Steering_Mode_1 state <unspecified> dataitem Fly_To_Point_Latitude (valid as_specified low_bound)
01000005	+ action Determine_Steering_Mode_1 state <unspecified> dataitem Fly_To_Point_Latitude (valid as_specified high_bound)
01000006	+ action Determine_Steering_Mode_1 state <unspecified> dataitem Fly_To_Point_Latitude (valid as_specified low_debug)
01000007	+ action Determine_Steering_Mode_1 state <unspecified> dataitem Fly_To_Point_Latitude (valid as_specified high_debug)
01000008	+ action Determine_Steering_Mode_1 state <unspecified> dataitem Fly_To_Point_Latitude (invalid above_bounds above_bounds)
01000009	+ action Determine_Steering_Mode_1 state <unspecified> dataitem Fly_To_Point_Latitude (invalid below_bounds below_bounds)
01000010	+ action Determine_Steering_Mode_1 state <unspecified> dataitem Fly_To_Point_Latitude (invalid out_of_type out_of_type_1)
01000011	+ action Determine_Steering_Mode_1 state <unspecified> dataitem Fly_To_Point_Latitude (invalid out_of_type out_of_type_2)
01000012	+ action Determine_Steering_Mode_1 state <unspecified> dataitem Fly_To_Point_Latitude (invalid out_of_type out_of_type_3)
01000013	+ action Determine_Steering_Mode_1 state <unspecified>

catalog.rpt

s_packet: Proc_Determine_Steering_Mode
unitdate: Fri Sep 23 11:51:31 1994
t_packet: tpacket
casedate: Fri Sep 23 11:51:35 1994

Designated_Angle

Index	SubDomain	Equiv.Class	Label	Value
[1]	valid	as_specified	reference	180.0
[2]	valid	as_specified	low_bound	0.0
[3]	valid	as_specified	high_bound	360.0
[4]	valid	as_specified	low_debug	1.0
[5]	valid	as_specified	high_debug	359.0
[6]	invalid	above_bounds	above_bounds	361.0
[7]	invalid	below_bounds	below_bounds	-1.0
[8]	invalid	out_of_type	out_of_type_1	9
[9]	invalid	out_of_type	out_of_type_2	'a'
[10]	invalid	out_of_type	out_of_type_3	"<not_in_list>"

FDS_In

Index	SubDomain	Equiv.Class	Label	Value
[1]	valid	as_specified	reference	"FALSE"
[2]	valid	as_specified	low_bound	"TRUE"

FTP_Run_Out_Status

Index	SubDomain	Equiv.Class	Label	Value
[1]	valid	as_specified	reference	"Runout_Not_In_Progress"
[2]	valid	as_specified	low_bound	"Runout_In_Progress"
[4]	invalid	not_in_list	not_in_list	"<not_in_list>"

Fly_To_Point_Latitude

Index	SubDomain	Equiv.Class	Label	Value
[1]	valid	as_specified	reference	0.0
[2]	valid	as_specified	low_bound	-90.0
[3]	valid	as_specified	high_bound	90.0
[4]	valid	as_specified	low_debug	-89.0
[5]	valid	as_specified	high_debug	89.0
[6]	invalid	above_bounds	above_bounds	91.0
[7]	invalid	below_bounds	below_bounds	-91.0
[8]	invalid	out_of_type	out_of_type_1	9
[9]	invalid	out_of_type	out_of_type_2	'a'
[10]	invalid	out_of_type	out_of_type_3	"<not_in_list>"

Fly_To_Point_Longitude

Index	SubDomain	Equiv.Class	Label	Value
[1]	valid	as_specified	reference	0.0
[2]	valid	as_specified	low_bound	-180.0
[3]	valid	as_specified	high_bound	180.0
[4]	valid	as_specified	low_debug	-179.0
[5]	valid	as_specified	high_debug	179.0
[6]	invalid	above_bounds	above_bounds	181.0
[7]	invalid	below_bounds	below_bounds	-181.0
[8]	invalid	out_of_type	out_of_type_1	9

[9]	invalid	out_of_type	out_of_type_2	'a'
[10]	invalid	out_of_type	out_of_type_3	"<not_in_list>"

Fly_To_Point_Type

Index	SubDomain	Equiv.Class	Label	Value
[1]	valid	as_specified	reference	"Normal"
[2]	valid	as_specified	low_bound	"Orbit_Datum"
[3]	valid	as_specified	high_bound	"Expendable"
[4]	valid	as_specified	element_2	"Monitor"
[5]	valid	as_specified	element_3	"Orbit_Tangential"
[6]	valid	as_specified	element_4	"Flight_Plan"
[7]	valid	as_specified	element_5	"Sensor"
[8]	valid	as_specified	element_6	"Pattern_Orbit"
[9]	valid	as_specified	element_7	"Weapon"
[10]	valid	as_specified	element_9	"Intercept"
[11]	valid	as_specified	element_10	"Harpoon"
[12]	valid	as_specified	element_11	"Pattern_Expendable"
[13]	valid	as_specified	element_12	"Proceed_To_Point"
[14]	valid	as_specified	element_13	"Harpoon_Launch"
[15]	invalid	not_in_list	not_in_list	"<not_in_list>"

HSI_In

Index	SubDomain	Equiv.Class	Label	Value
[1]	valid	as_specified	reference	"FALSE"
[2]	valid	as_specified	low_bound	"TRUE"

Orbit_Radius

Index	SubDomain	Equiv.Class	Label	Value
[1]	valid	as_specified	reference	101237.0
[2]	valid	as_specified	low_bound	2000.0
[3]	valid	as_specified	high_bound	200475.0
[4]	valid	as_specified	low_debug	2001.0
[5]	valid	as_specified	high_debug	200474.0
[6]	invalid	above_bounds	above_bounds	200476.0
[7]	invalid	below_bounds	below_bounds	1999.0
[8]	invalid	out_of_type	out_of_type_1	9
[9]	invalid	out_of_type	out_of_type_2	'a'
[10]	invalid	out_of_type	out_of_type_3	"<not_in_list>"

Tactical_Steering_Status

Index	SubDomain	Equiv.Class	Label	Value
[1]	valid	as_specified	reference	"FALSE"
[2]	valid	as_specified	low_bound	"TRUE"

Undesignated

Index	SubDomain	Equiv.Class	Label	Value
[1]	valid	as_specified	reference	"FALSE"
[2]	valid	as_specified	low_bound	"TRUE"

- saving samples in test design data base

Translation

```

1  /* T Design Rule Generation Version 3.0
2  ** Copyright (C) 1987-1992 Programming Environments, Inc.
3  */
4
5  T_Packet                                tpacket
6      s_packet                          Proc_Determine_Steering_Mode
7      .
8
9  CombinationRule                        CR0001
10     action                            Determine_Steering_Mode_1;
11     singular                          Fly_To_Point_Latitude,
12                                     Orbit_Radius,
13                                     Designated_Angle,
14                                     Tactical_Steering_Status,
15                                     FTP_Run_Out_Status,
16                                     HSI_In,
17                                     FDS_In;
18     .
19
20 CombinationRule                        CR0002
21     action                            Determine_Steering_Mode_2;
22     singular                          Fly_To_Point_Longitude,
23                                     Fly_To_Point_Type,
24                                     Undesignated,
25                                     Orbit_Radius,
26                                     Designated_Angle,
27                                     Tactical_Steering_Status,
28                                     FTP_Run_Out_Status,
29                                     HSI_In,
30                                     FDS_In;
31     .
32
33 SelectionRule                          SR0001
34     datatype                          boolean_string;
35     reference                          TBD;
36     valid                             as_specified
37                                     with function, boundary, debug;
38     .
39
40 SelectionRule                          SR0002
41     datatype                          Lat_Type;
42     reference                          TBD;
43     valid                             as_specified
44                                     with function, boundary, debug;
45     .
46
47 SelectionRule                          SR0003
48     datatype                          Lon_Type;
49     reference                          TBD;
50     valid                             as_specified
51                                     with function, boundary, debug;
52     .
53
54 SelectionRule                          SR0004
55     datatype                          Fly_To_Point_Type_t;
56     reference                          TBD;
57     valid                             as_specified
58                                     with function, boundary, debug;

```

```

59      .
60
61      SelectionRule          SR0005
62      datatype              Radius_Type;
63      reference              TBD;
64      valid                  as_specified
65                               with function, boundary, debug;
66      .
67
68      SelectionRule          SR0006
69      datatype              Angle_Type;
70      reference              TBD;
71      valid                  as_specified
72                               with function, boundary, debug;
73      .
74
75      SelectionRule          SR0007
76      datatype              FTP_Run_Out_Status_t;
77      reference              TBD;
78      valid                  as_specified
79                               with function, boundary, debug;
80      .
81
82                               pe_mark

```

- finished translation with 10 recognizable TDRL sentences out of 10

```

=====
                        Interpretation
=====

```

```

s_packet: Proc_Determine_Steering_Mode
unitdate: Fri Sep 23 11:51:31 1994
t_packet: tpacket
casedate: Fri Sep 23 11:51:35 1994

```

- saving rules in test design data base

design.rpt

APPENDIX F. Documentation Errors

The following documentation errors were reported to IDE:

T Users Guide

TSDL Manual

- Section 2.7: references in Figure 9 are N/A. Ex: to Section 3.3.19
- Section 4, Figure 17: The trace.rpt and tdmtric.rpt reports are not produced by tdesign.
- Section 4.2: "echo on" should be "echo"

Reports Manual

- Section 2: There is no discussion of cref.rpt or sdf.p as listed on page 18 of the Operations Manual
- Section 2.2: Figure 7 does not contain the "Greatest Lookup Level" information as stated.
- Section 3: (Reports-9) There is no discussion of design.rpt as listed on page 18 of the Operations Manual
- Section 3.2: The tdmtric.rpt report is not produced by tdesign.
- Section 3.4: The trace.rpt report is not produced by tdesign.

Application Note: Software through Pictures (StP) and T Integration

- Page 11: The Satisfy Preconditions flag is apparently the "Preconditioned" variable in tconfig.ini