

SANDIA REPORT

SAND97-3179 • UC-903

Unlimited Release

Printed December 1997

The Waveform Correlation Event Detection System Global Prototype Software Design

Judy I. Beiriger, Susan G. Moore, Julian R. Trujillo, Chris J. Young

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation,
a Lockheed Martin Company, for the United States Department of
Energy under Contract DE-AC04-94AL85000.

Approved for public release; further dissemination unlimited.



Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Prices available from (615) 576-8401, FTS 626-8401

Available to the public from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Rd
Springfield, VA 22161

NTIS price codes
Printed copy: A03
Microfiche copy: A01

SAND97-3179
Unlimited Release
Printed December 1997

Distribution
Category UC-903

The Waveform Correlation Event Detection System Global Prototype Software Design

Judy I. Beiriger, Susan G. Moore, Julian R. Trujillo
Decision Support Systems Architectures Department

Chris J. Young
Geophysical Technology Department

Sandia National Laboratories
P. O. Box 5800
Albuquerque, NM 87185-1138

Abstract

The WCEDS prototype software system was developed to investigate the usefulness of waveform correlation methods for CTBT monitoring. The WCEDS prototype performs global seismic event detection and has been used in numerous experiments. This report documents the software system design, presenting an overview of the system operation, describing the system functions, tracing the information flow through the system, discussing the software structure, and describing the subsystem services and interactions. The effectiveness of the software design in meeting project objectives is considered, as well as opportunities for code reuse and lessons learned from the development process. The report concludes with recommendations for modifications and additions envisioned for a regional waveform-correlation-based detector.

Acknowledgments

This work was funded by the Automatic Data Processing portion (ST485D) of the Comprehensive Test Ban Treaty Research & Development Program at Sandia National Laboratories under Department of Energy contract DE-AC04-94AL85000.

Contents

Introduction	5
Background	7
IDC/US NDC Operational Environment	7
Seismic Wave Propagation	8
System Overview	9
System Design	14
Object Diagram	14
Architecture	14
Subsystem Descriptions	15
WCEDS Graphical User Interface	15
Specification Manager	19
Transaction Manager	20
WCEDS Capabilities	20
WCEDS Seismic Domain Library	23
Travel Time Model Library	25
Earth Grid Library	25
Base Seismic Data Library	25
Extended Seismic Data Library	26
General Purpose Utility Library	27
Third Party Libraries	27
Discussion	27
Effectiveness of Design	27
State of Implementation	29
Reusability	29
Lessons Learned	30
Viewing Correlated Data As Analogous To Waveforms	30
Database Interface	30
Input Parameter Parsing	31
Commercial Library Use	31
Software Development Process	32
Conclusions	33
References	35

Figures

Figure 1.	WCEDS Information Flow	6
Figure 2.	WCEDS Detailed Design Object Diagram	16
Figure 3.	WCEDS Subsystems Block Diagram	18
Figure 4.	Sample WCEDS GUI Screen	19
Figure 5.	Organization of Correlation Matrix	24

Introduction

To verify compliance with a Comprehensive Test Ban Treaty (CTBT), treaty monitoring systems will be required to process and interpret significantly greater volumes of sensor data. As part of an effort to develop new, more efficient data processing algorithms, the Waveform Correlation Event Detection System (WCEDS) project was initiated in the spring of 1995 to investigate the potential utility of a waveform correlation-based approach to seismic event detection and location. Initially, the project explored the waveform correlation concept, focusing on rapid prototyping and experimentation with an existing code (Shearer 1994). This work yielded valuable insight into how to structure a correlation-based detector. Next, the project developed and functionally tested a global WCEDS prototype. During the design process, the more general seismic monitoring functionality was extracted from the WCEDS-specific requirements and developed into independent C++ libraries. Finally, the project ran experiments to determine the effectiveness of the WCEDS prototype, processing seismic sensor data from the International Monitoring System (IMS) primary network and comparing the results to the event bulletin from the International Data Center (IDC).

This report describes the global WCEDS prototype software design and implementation. It documents design decisions, the final object model, software implementation, reusable components, and lessons learned for the global WCEDS prototype, and will provide a reference for investigating regional waveform correlation methods. The WCEDS system design followed the Object Modeling Technique (OMT) design methodology (Rumbaugh et al. 1991), and generated analysis and system design documents (Beiriger et al. 1996a and b). This report may be viewed as the third and final software design document, taking the place of the object design document. However, because the WCEDS software development task was primarily a research effort and not a production software development project, the implementation details and requirements were constantly changing. Consequently, the documentation is less formal and detailed than might be expected for a production system. The software design choices emphasized flexibility, access to intermediate processing results, adaptability to changing requirements, and generality to seismic monitoring. Production software requirements such as execution speed and robustness, though not ignored, were treated as lower priority and developed less fully than for an operational system. System design information and parameter descriptions can be found in the WCEDS analysis, system design, and user manual documents (Beiriger et al. 1996a, 1996b, and 1997a).

The remainder of this document is organized as follows:

Background. This section provides seismological and treaty monitoring system background information that may be helpful to a software engineer unfamiliar with this application domain.

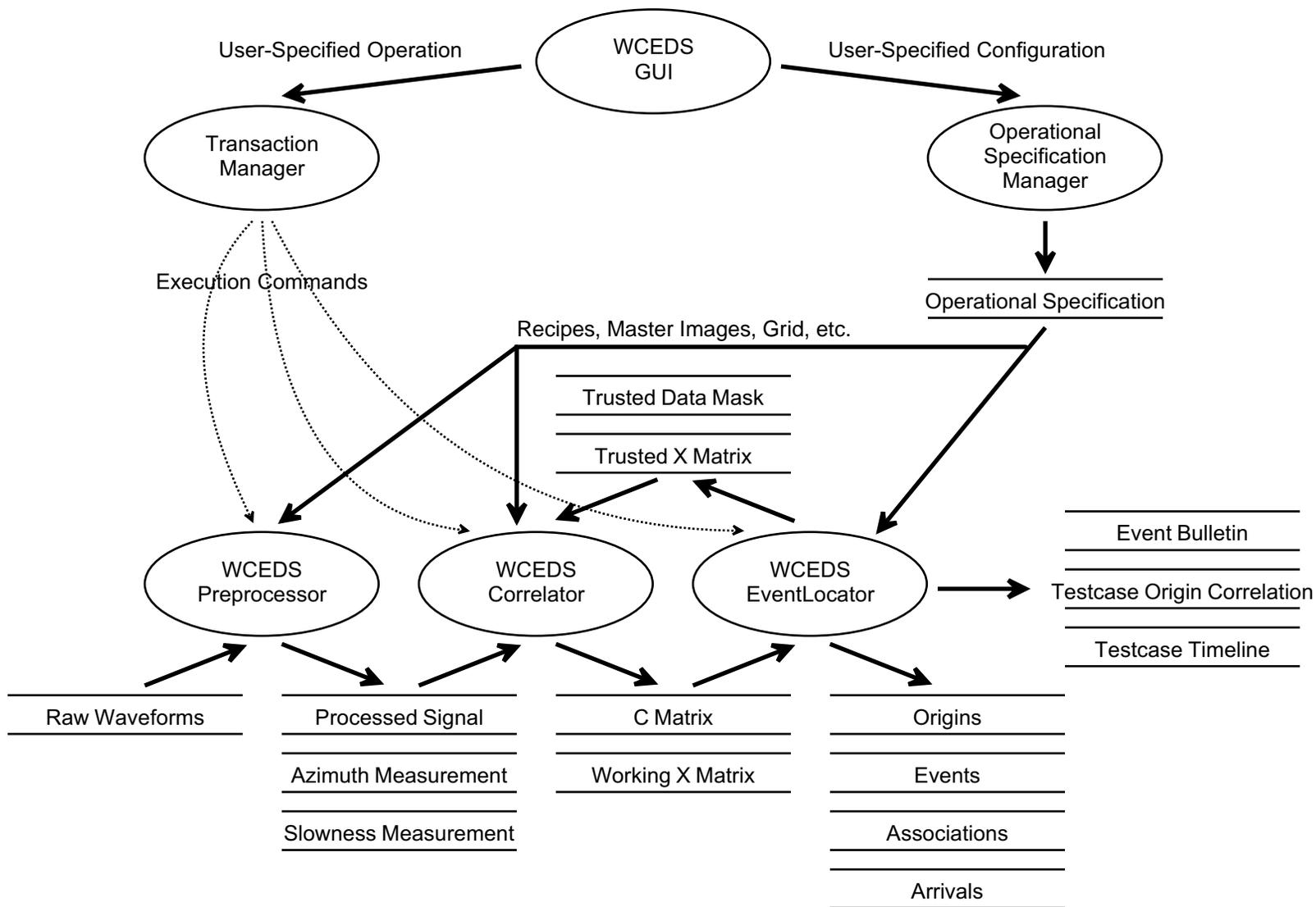


Figure 1. WCEDS Information Flow

System Overview. This section provides a high-level description of the WCEDS prototype, discussing functional requirements, operation, and information flow.

System Design. This section describes how the system is designed to meet the requirements. It describes the object model, system architecture, and subsystem partitioning. Each subsystem is discussed, including a description of the purpose of or service provided by the subsystem, the software components that comprise it, and information flow between it and other subsystems or external entities.

Discussion. This section discusses the effectiveness of the software design, the state of the software implementation, reusability, and lessons learned.

Conclusions. This section contains insights and recommendations that may be helpful for future waveform correlation or other CTBT advanced data processing software development efforts.

Background

IDC/US NDC Operational Environment

The operational environment for the CTBT event detector is expected to be essentially identical to the existing monitoring system environment, known as “the pipeline”, found at the IDC and at the United States National Data Center (US NDC). Raw seismogram data is collected continuously, packaged into blocks of data corresponding to time intervals, and fed into the pipeline. From the other end of the pipeline comes a notification of detected events. The pipeline itself consists of a sequence of independent processes that, in aggregate, perform event detection and location and provide data for analyst review. Each process takes in the results of the previous step, operates on the block of data, and passes its results on to the next step. Analysts review and may refine or reject the automatically-built events, process the remaining signal detections that were not automatically associated with any event, and finalize the event list.

Interprocess communication and process scheduling are provided automatically by a third party software package called ISIS. Intermediate processing results are passed from one process to the next via a database, with each process retrieving from and storing to separate input and output accounts. The Center for Seismic Studies version 3.0 database schema, widely used in the seismic community, is used for this purpose (Anderson et al. 1990).

The IMS primary network stations installed around the world continuously monitor and record seismic activity. A seismic recording station has one or three channels, each corresponding to the orientation of an emplaced sensor such as North (N), East (E), or Vertical (Z). Depending on the type of recording instrument, the stations sample the seismic energy at various rates; e.g., 1, 20, or 40 samples per second. A seismic array is a set of stations installed at certain locations relative to one another such that the recorded data may be combined to reduce noise and enhance signals, as well as to determine the direction to the source of the seismic energy. An automated data acquisition system collects the data from each station at regular intervals and stores it in a central CSS 3.0 schema database.

The WCEDS prototype fits into the pipeline immediately after data acquisition. The detector reads the recorded waveforms for a given time interval from the CSS 3.0 database. It executes one or more independent processes and outputs a list of candidate events, including estimated locations and times and associated phase arrivals at each station. This information is also stored in the database for subsequent processing and analyst review.

Seismic Wave Propagation

Understanding the waveform correlation technique requires a basic understanding of seismic wave propagation, so a brief discussion is given here. More comprehensive explanations can be found in the literature. Kulhanek (1990) is a good introductory text.

A seismic event such as an earthquake or explosion radiates seismic waves in all directions. Recording instruments installed on the Earth's surface detect and measure the resulting ground motion. There are two major types of seismic waves -- body waves, which travel through the Earth's interior, and surface waves, which travel along the Earth's surface or other discontinuities in the interior -- and additional subcategories within these major types. Different types of waves produce different types of ground motion, travel at different speeds, have different frequency content, and exhibit velocity dispersion and amplitude attenuation with distance. Discontinuities in the propagation path cause reflection, refraction, and transformation of the waves. Energy from each of these possible seismic paths, called phases, can be distinctly observed at the recording stations. Knowing these characteristic features of seismic wave propagation, a set of seismograms may be analyzed to determine important event information such as location, depth, origin time, and magnitude.

Body waves propagate with either compressional or shear ground motion. Compressional waves generate ground motion in line with the direction of wave travel. These are the fastest of the seismic waves, arriving first at a recording station, and hence are called primary or P waves. Shear waves generate ground motion perpendicular to the direction of wave travel, and are called secondary or

S waves. The propagation paths of body waves can be well approximated with ray geometry. Since wave velocity is not uniform within a layer of the Earth, but tends to increase with depth, the seismic rays curve upward toward the surface of the earth instead of being straight. When a ray hits a discontinuity between layers, such as the core-mantle boundary or the surface-air boundary, energy may be reflected, refracted, or converted to the other wave mode (i.e., converted from P to S or S to P). The phases are labeled to indicate the propagation path and mode conversions; for example, PS is the phase that begins as a P wave, hits the surface, and is reflected as an S wave, and PcP is the phase that begins as a P wave, hits the outer core, and is reflected back up as a P wave.

Surface waves originate from the interactions of body waves with discontinuities in the Earth, and produce ground motion that propagates along the surfaces of the discontinuities. There are two basic types of commonly observed surface waves, Love waves and Rayleigh waves, denoted LQ and LR. Love waves generate ground motion in a horizontal plane perpendicular to the direction of propagation, with no vertical displacement. Rayleigh waves generate ground motion in a vertical plane in line with the direction of propagation, with displacement following a retrograde elliptical path. Love waves travel faster than Rayleigh waves. Both exhibit velocity dispersion, with long period waves arriving ahead of short period waves. Surface waves can be quite large, and often dominate the seismograms of shallow earthquakes. As the name implies, however, amplitudes diminish rapidly with depth (or distance from the discontinuity), and deep earthquakes don't usually produce surface waves of any consequence.

The different seismic phases propagate at sufficiently different velocities such that distinct phase arrivals are readily observable at most distances. Distance is usually measured in degrees of arc from the epicenter; on the surface of the earth, one degree is approximately one hundred eleven kilometers. At regional distances ($< 20^\circ$), the local earth structure complicates wave propagation and may generate unique regional phases. At teleseismic distances ($> 20^\circ$), the effects of local structure have largely dissipated and wave propagation is more uniform around the globe. Using a homogeneous earth model suitable for teleseismic distances, travel time curves have been developed that plot travel time versus distance traveled for each phase. That is, the curves show the time expected for a particular phase to travel a particular distance that is reasonably accurate at teleseismic distances. Regional earth models and travel time curves are currently a subject of intense interest and research.

System Overview

The primary goal of the global WCEDS prototype was to implement a global detection algorithm based on waveform correlation, and evaluate its usefulness

in an automated CTBT monitoring system. Two important requirements for the software system were operation in an environment compatible with the IDC and the USNDC, and the incorporation of existing IDC and USNDC software and commercial third-party libraries where needed capabilities could be obtained readily. An additional goal of the software design effort was to develop the WCEDS-specific software entities and functions in a separate application layer on top of the more general seismic processing software. The reusable, general seismic software base is available to support other CTBT advanced data processing research projects.

The WCEDS prototype system consists of a set of related executable programs that are accessed and controlled through the WCEDS GUI. This compartmentalized approach allows the major processing phases of the waveform correlation technique (that is, preprocessing, correlation, and detection) to be developed and investigated independently. It is also an arrangement compatible with the IDC/USNDC pipeline scheduling. Operation of the WCEDS prototype is illustrated by tracing a typical data interval through the system from raw waveforms to detected events. The data flow through the system is summarized in Figure 1.

The WCEDS executable programs operate on one data interval and then terminate. They are scheduled and launched with the appropriate input parameters, so that successive data intervals are sequenced through all of the specified processing steps. To initiate a run, the user must first specify the desired configuration and operational parameters to be applied to all of the executable programs during a single WCEDS run. These include:

- the set of phases and the time and distance resolution for the travel time models,
- the time and distance resolution for candidate event locations,
- the set of sites to use for seismic data input, and site-specific information for each site, such as channels, site/channel groups, processing algorithms, filters, recipes, and recipe parameters,
- seismic event and arrival detection thresholds, and optional azimuth and slowness criteria,
- the database interface,
- optional testcase origins,
- the desired start time,
- either continuous operation or a fixed duration,
- which processing steps to perform,
- and preprocessing, correlation, and location interval sizes.

In a typical research scenario, preprocessing, correlation, and event detection are to be performed on a particular block of waveform data. The set of stations includes both arrays and three-component stations, which require different waveform processing algorithms. The block of waveforms is broken into smaller time segments, which are processed consecutively by separate invocations of the appropriate executable program. The executables do not coordinate directly, but are loosely coupled; each starts when its input data becomes available.

First, the waveforms are preprocessed to provide the signal, azimuth, and slowness measurements. The preprocessing interval size specifies the time interval of waveform data for the WCDPreprocessor program, say 15 minutes. The WCDPreprocessor reads the recipes defined for each site and performs the specified processing. Processing differences between sites -- different filter bands, different algorithms, different input parameters for the same algorithm, etc. -- can be achieved by defining different recipes. The typical sequence is bandpass filtering of all the waveform channels, a multichannel signal detection method like spatial coherence, scaling of the processed signal, and f-k analysis to take azimuth and slowness measurements at each output data sample time. For each recipe, three processed data streams are stored in the CSS database as waveforms: the processed signal, stored using the recipe name as the channel name, and the azimuth and slowness measurement streams, stored with an 'A' or an 'S', respectively, appended to the recipe name. The WCDPreprocessor program then terminates. When the next 15-minute interval of raw waveforms is available, another WCDPreprocessor program is started.

Once sufficient preprocessed data becomes available, a WCDCorrelator program is started for the first location interval. The epicentral distance interval from 0 to 180 degrees is discretized to a specified spacing, and the travel time curve for a particular phase is presented as boxcars of finite duration centered at the expected travel time for each discrete distance. The correlation time interval specifies the length of the processed waveforms that are to be correlated against "waveforms" derived from the expected travel time curves, yielding a correlation result for a single postulated origin time. The correlation interval must be long enough to encompass all the phases of interest, say 40 minutes. The location interval size specifies the time interval of postulated origin times for which trusted correlations will be made, say 1 hour. Trusted correlations are those for which processed data were available for the full correlation time interval. The WCDCorrelator program needs processed data for the location interval plus the correlation interval to compute results for the location interval. It actually produces correlations for postulated origin times as far out as it has data, though results beyond the location interval can not be trusted since full correlation was not possible. The need for the trusted/untrusted model for resolving false correlations near the end of the location interval, due to an event in the succeeding interval, became apparent during the initial WCEDS prototype testing (Young et al. 1996).

The desired behavior of the waveform correlation detector is that, for a given postulated origin (grid point and time), the processed signals for each site are correlated with specified phases of the travel time curve at the corresponding site-to-grid point distance, and added together to yield a single correlation value for the postulated origin. For performance reasons, the software implementation splits the distance correlation and the summing into two steps. Since each site must be tested at each of the discrete correlation distances for at least one grid point, and at the same distance from many grid points, its correlation at each discrete distance is computed once and saved. There is also a data mask channel for each processed signal channel that could block segments of the processed waveform from being correlated. Blocked segments could include previously-identified arrivals or data that failed quality control criteria. (Quality control is not implemented in the WCEDS prototype.)

The correlation results for each origin time are saved in a Correlation Matrix (C Matrix). The C Matrix originally contained a column for each site and a row for each distance. Each cell contained a single summed value for all of the phase correlations for all of the processed recipe channels for the corresponding site and distance. The C Matrix has since been generalized to hold separate correlation values for individual phases and channels, as well as azimuth and slowness measurements, and to allow different (possibly overlapping) distance ranges for different channels. The azimuth and slowness measurements corresponding to an individual phase correlation are taken to be the values of the azimuth and slowness channels at the time of maximum correlation between the travel time-based predicted signal and the processed observed signal. In addition, the C Matrix contains a correlation mask (X Matrix). If the phase correlation value does not exceed the specified threshold, the X Matrix is set to prevent it from being included in a sum for some grid point. The user may also optionally specify a slowness check for individual phases. If slowness checking is requested, the WCDCorrelator program compares the slowness measurement to the expected travel time slowness for the specified phase. If the slowness values differ by more than the specified threshold, the X Matrix is set to mask that phase correlation value. After the C Matrix and X Matrix are stored, then the WCDCorrelator program terminates. Another will not start until preprocessed data for the next location interval plus correlation interval are available. Correlation intervals after the first will also need phase identification information from the WCDEventLocator, provided as initial X Matrix and data masks for the interval, before they can run.

When a location interval of trusted correlation data (and its corresponding untrusted correlation data) becomes available, a WCDEventLocator program is started. It proceeds by computing, for each postulated origin time in the trusted and untrusted intervals, the correlation sum for each grid point. A C Matrix map defines the summing path for each grid point. The map is a set of offsets from the start of the C Matrix to the individual phase correlations for each site channel at the appropriate site-to-grid point distance. The X Matrix has the same structure,

so it may be navigated with the same map. At each map offset, if the X Matrix value allows, the C Matrix phase correlation value is added to the grid point sum. The user may optionally specify an azimuth check for individual phases. If azimuth checking is requested, the WCDEventLocator program compares the azimuth measurement to the site-to-grid point azimuth for the specified phase. If the azimuth values differ by more than the specified tolerance, the phase correlation value is not included in the sum.

If the maximum of the correlation sums for all grid points and time points exceeds the specified threshold, an event is detected. Associated phases are then identified. Phase identification is accomplished by duplicating the correlation process for the event time and location, but using the phase identification and screening configuration, rather than the detection configuration, of the phase set and the azimuth and slowness criteria. The screening configuration must at least guarantee that the detected signals will be claimed by this event. Typically, the screening phase set is larger than the detection set, and the azimuth and slowness checks are less stringent. A phase correlation that meets the threshold criteria is considered an arrival at the time corresponding to the peak correlation with the travel time boxcar. The WCDEventLocator program stores the event information in the CSS database origin, arrival, and assoc tables. The author, algorithm, and velocity model fields are set to indicate that the record was created by WCEDS. Possible event types are trusted, untrusted, or testcase. Events with times in the trusted interval are also inserted into the event table.

WCEDS tracks the start and stop times corresponding to the travel time boxcar to identify the data segments claimed by arrivals. The data mask for a claimed data segment must be set properly so this arrival will not add into other computations. Similarly, the X Matrix must be set to mask travel time correlations with the data segment. This is done by computing the origin times when the data segment correlated with any phase in the detection travel time model, and masking that phase correlation value. For trusted events, separately-maintained trusted data and X Matrix masks are updated. These trusted masks are used by the WCDCorrelator and WCDEventLocator programs operating on the next location interval. Using the new X Matrix, the WCDEventLocator program then repeats the process of finding the maximum correlation sum. When the maximum sum falls below a specified threshold, or a specified number of events has been found in the interval, the program stores the trusted mask information and terminates. Another WCDEventLocator program will be started when the next location interval of trusted correlation data becomes available.

The user may optionally specify testcase origins or locations for which additional output is produced for analysis. An “event” testcase specifies a target latitude, longitude, and time. The closest grid point and postulated origin time used by the detector are treated as an event, and both the detection and screening correlations are performed. The user may choose to see all phase correlation results, or only those that pass the threshold criteria. A “timeline” testcase specifies a tar-

get latitude and longitude, but allows time to vary. The correlation sum for the closest grid point is output for each postulated origin time, so the seismic activity at that point can be watched in time. A separate utility program, CorrSnapshot, stores the correlation sum for all grid points for a set of origin times, and outputs the data so it can be viewed by an animation program.

System Design

Object Diagram

The *object model* describes the static structure of the system - the objects (things that have meaning within the context of the problem) and the static relationships between them. An object (class) is an abstraction of a real-world entity, where every occurrence (instance) of the real-world entity has the same characteristics (attributes), relationships (associations), and behaviors (operations). An association is an abstraction of a relationship between instances of objects. Relationships may be traversed from object to object, deriving relationships between objects not explicitly shown on the object model. For example, if the relationships “A is related to B” and “B is related to C” are modeled, then the relationship “A is related to C through B” is implied and is not modeled directly.

The *object diagram* is a graphical representation of the system with objects as nodes and relationships as links between nodes. The WCEDS analysis effort employed the conventions of the Object Modeling Technique (OMT) advanced by Rumbaugh, et al. (1991). The symbolic notation used to draw the links between objects specify important aspects of the relationship, including multiplicity, conditionality, subclass/superclass hierarchy, and aggregation. The OMT process follows an evolutionary approach of moving from analysis to design through elaboration and iteration, so that the analysis effort does not yield a complete object model. The final object diagram used for detailed design and implementation, showing the classes and associations added as a result of design choices, is shown in Figure 2.

Architecture

The WCEDS prototype is organized into several subsystems, each providing a particular service to the system through a well-defined and relatively small interface. A subsystem is a package of interrelated software components, associations, and operations that can be designed and implemented independently of other subsystems.

The WCEDS system is structured as a closed architecture of layers and partitions, as shown in Figure 3. The bottom two layers encompass the hardware sys-

tem and available resources: a distributed network of Sun workstations running the Solaris 2.5.1 operating system, the Network File System (NFS), an Oracle database, the CSS 3.0 database schema, some third party libraries, and general utility routines. The top layer provides the WCEDS application - the global capabilities and services - to the user through a Graphical User Interface (GUI). In between is a layer encompassing the WCEDS prototype functionality, divided into three vertical partitions. The Specification Manager Subsystem manages the user specification of the WCEDS components and analysis parameters, providing them to the user interface for display and editing, and providing them to the detector code for operation. The Transaction Manager Subsystem manages the execution of the WCEDS detector according to the user specification; several detector application and utility programs are available. The WCEDS Capabilities Subsystem provides the high-level WCEDS functions and executables that can be launched by the Transaction Manager. It is built upon the WCEDS Seismic Domain Subsystem, which provides seismological information and processing services. The WCEDS Seismic Domain Subsystem is further subdivided into Travel Time Model, Earth Grid, Base Seismic Data, and Extended Seismic Data Subsystems.

Relationships between the layers are one-way, client-server relationships: a layer knows about the layer immediately below it, but not above it. The WCEDS GUI Subsystem requests services of the Specification Manager Subsystem interactively. It "requests" services of the Transaction Manager Subsystem indirectly, and through it services of the WCEDS Capabilities Subsystem, by specifying parameter files and launching execution. The WCEDS Capabilities Subsystem requests services of the WCEDS Seismic Domain Subsystem by directly invoking object operations.

Relationships between partitions may be one-way or two-way relationships: a partition knows about the other partitions in the same layer. The relationships between the Specification Manager Subsystem and each of the other subsystems in that layer are one-way relationships: the Specification Manager provides the operational specification to the other subsystems in a data store. The relationship between the Transaction Manager Subsystem and the WCEDS Capabilities Subsystem is a two-way, peer-to-peer relationship: these subsystems interact through asynchronous, message-based communications. (Currently, this is a one-way interaction: the Transaction Manager launches a WCEDS executable.) The various executables in the WCEDS Capabilities Subsystem, such as the preprocessor, correlator, and locator executables, do not relate to each other directly: they interact indirectly through data maintained in the Seismic Domain Subsystem.

Subsystem Descriptions

WCEDS Graphical User Interface. The WCEDS user interaction and object specification management are implemented in the WCEDS GUI executable. The

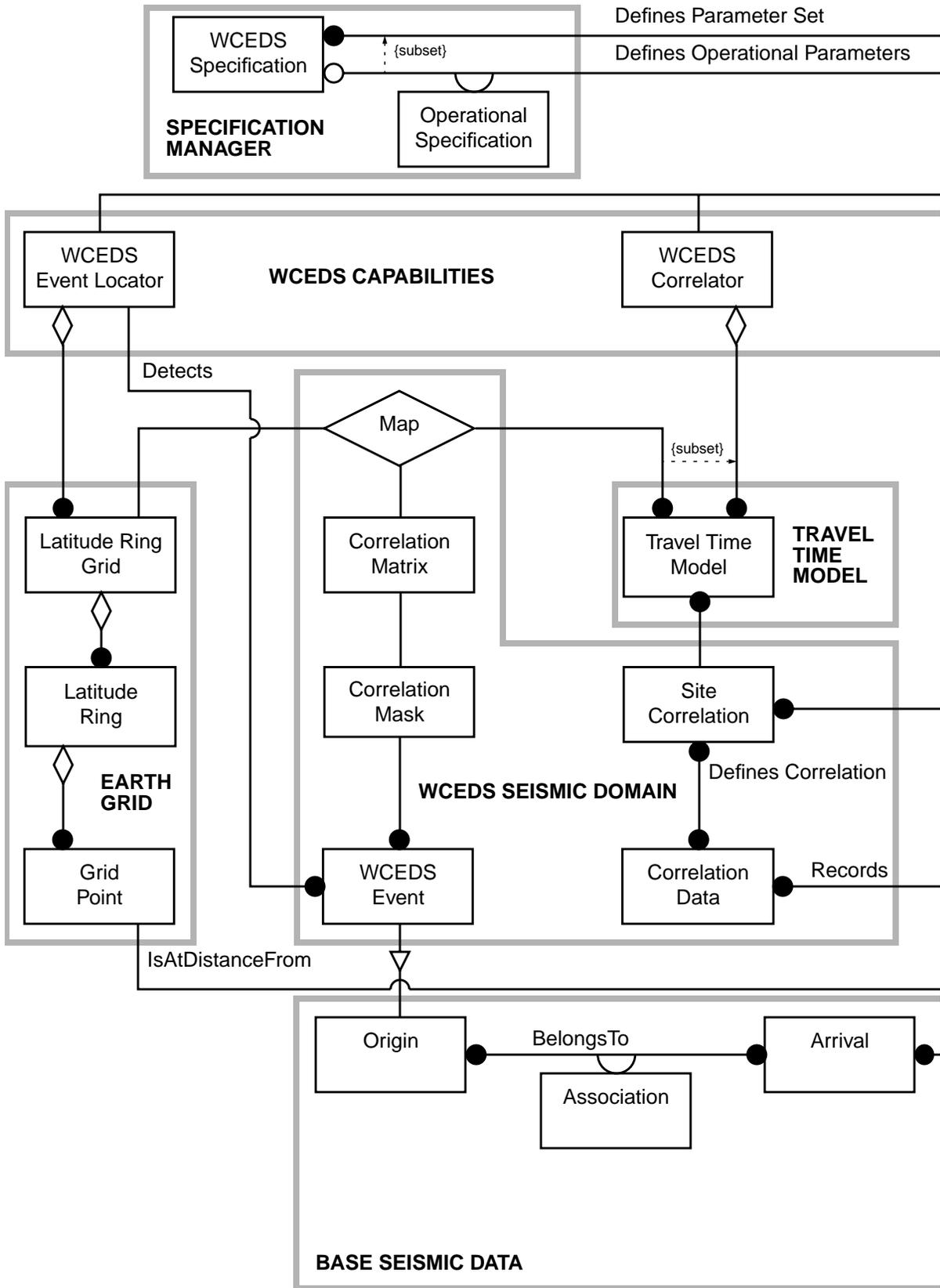


Figure 2. WCEDS Detailed Design Object Diagram

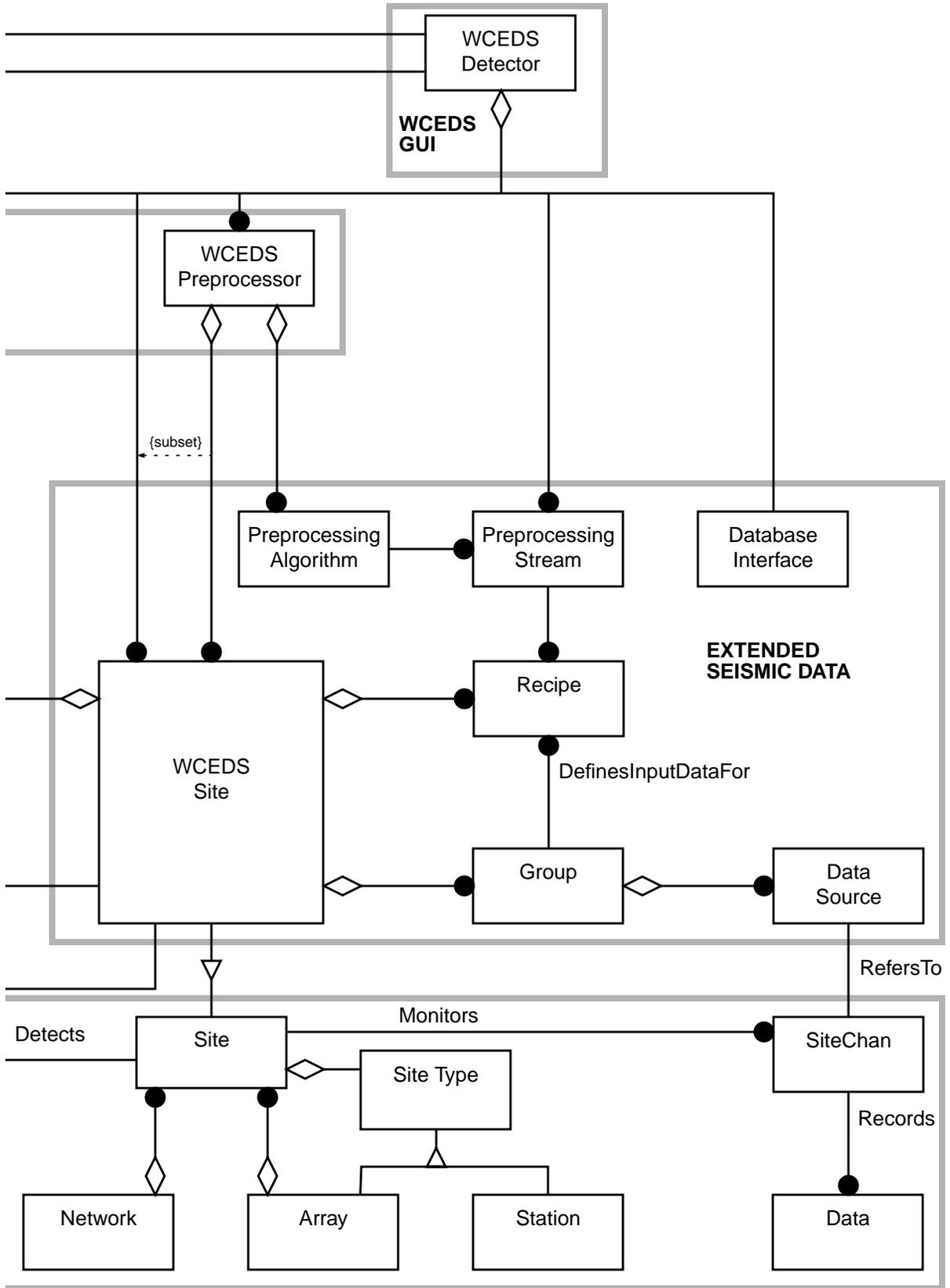


Figure 2. WCEDS Detailed Design Object Diagram (cont'd)

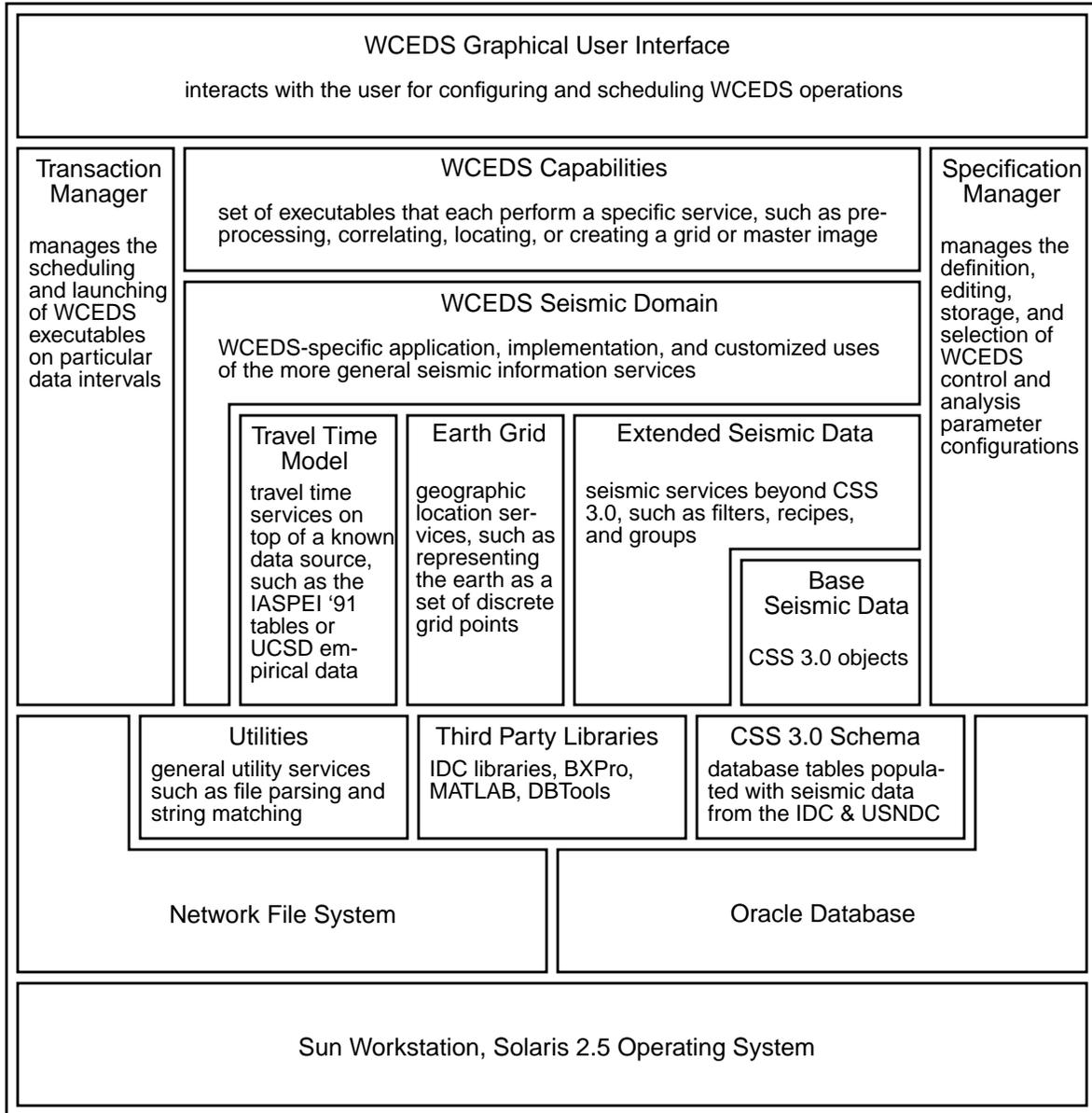


Figure 3. WCEDS Subsystems Block Diagram

GUI allows the user to view and set the operational control parameters for the WCEDS prototype detector: detector start time and duration, which of the pre-processing, correlation, and location steps are to be performed, preprocessing, correlation, and location interval sizes, and whether to start fresh or use event history information. The GUI is being extended to provide management of the system configuration parameters. Information for launching WCEDS executable programs is provided to the transaction manager. In theory, the WCEDS GUI design reflects the operator requirements for the operation of the WCEDS algorithms. This is the goal of the design. The control parameters were implemented for simplified operation of the WCEDS program in the first phase of the project.

The management of the configuration parameters was envisioned for a more complete operator specification in the second phase. Once the GUI is complete it can be used for regression testing of the underlying application, WCEDS, with an automated procedure.

The *WCEDS* program is a C++ executable implemented with the user interface development tool, BXPPro, from Integrated Computer Solutions. Tabs organize the parameters into categories to ease viewing and setting of related parameters. The *Scope*, *Function*, *Output*, and *DACS* tabs were implemented for management of the operational control parameters. The *Testcase*, *Travel Time*, *Database*, *Stream*, *Grid*, and *Site* tabs are designed for management of detector configuration parameters. A sample screen is shown in Figure 4.

Specification Manager. The specification manager is designed to extend the GUI to address the problem of difficult organization of parameter files or configurations. In addition to the control parameters, the GUI will also manage the configuration parameters such as the set of sites, preprocessing recipes, testcase locations, master image configurations, etc. The design allows a user to set configurations, and then store and recall them by name rather than traversing directories searching and editing parameter files manually. The WCEDS GUI provides the operational specification to the detector code through a data store. The operational data store is defined, and implemented as a set of parameter files. At the time of this writing, the specification manager is not operational, and the parameter files are maintained and edited manually.

Transaction Manager. The transaction manager is currently implemented as a shell script. It receives user-specified information from the GUI to launch and sequence the executable programs. It is responsible for ensuring that the requested data intervals receive the requested processing, but is not robust. The WCEDS design plans for the transaction manager to be the Distributed Applications Control System (DACS) developed by SAIC for the USNDC. We performed some simplified prototyping to verify proof-of-concept, but have not implemented the DACS interface.

The *start_wceds* C-shell script takes control parameters passed from the WCEDS GUI (or on the command line) and launches WCEDS executables accordingly. The script determines the time intervals for which processing has been requested. It sequences through the time intervals in order, invoking the WCD-Preprocessor, WCDCorrelator, and WCDEventLocator programs for each interval, depending on the input control parameters.

The *set_wceds_environment* file lists environment parameters that must be set appropriately when compiling or running the WCEDS prototype. The variable WCEDS_HOME must be set to the top of the WCEDS directory tree. The PATH variable should include \$WCEDS_HOME/bin. Variables needed to locate the Ora-

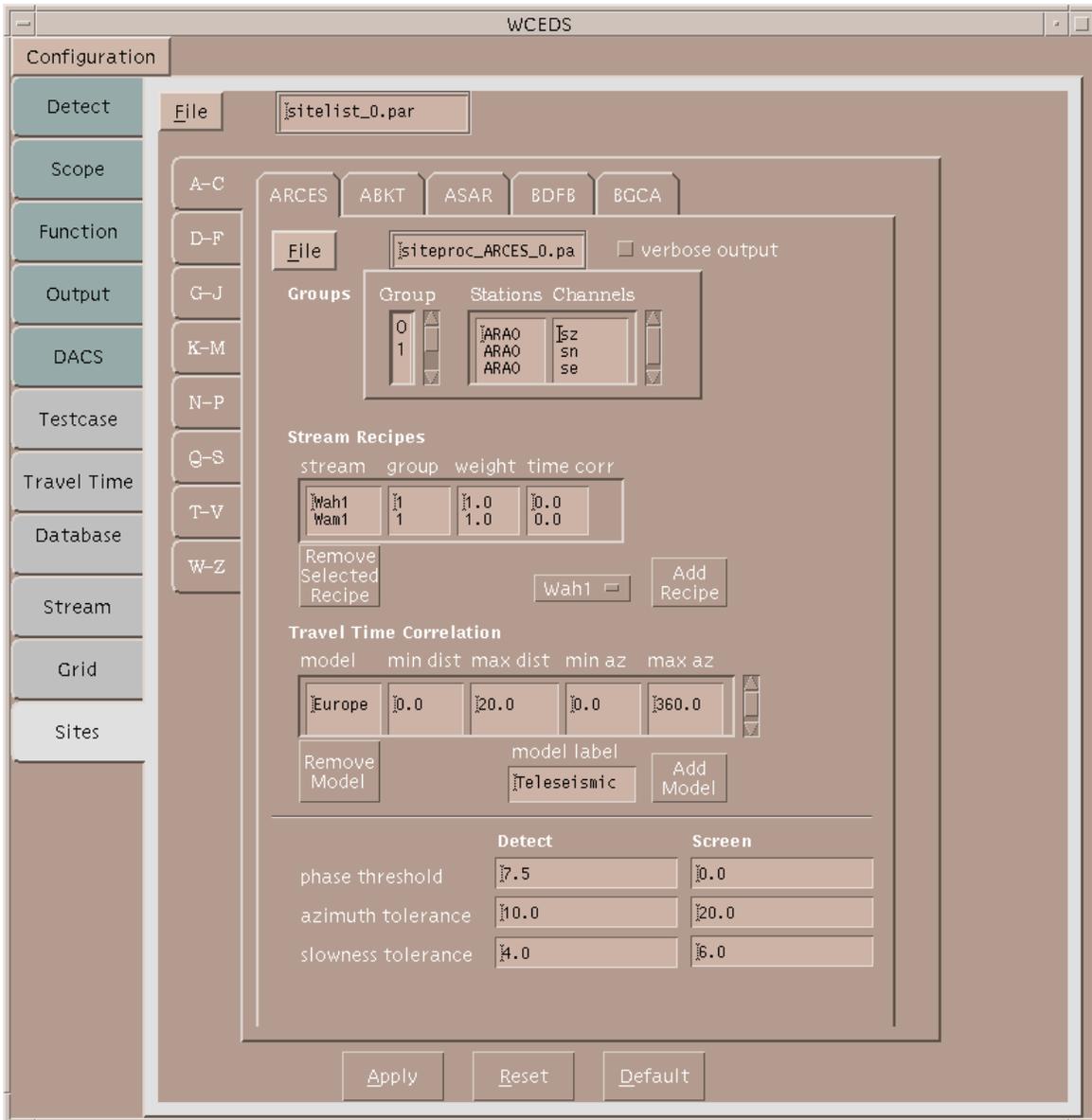


Figure 4. Sample WCEDS GUI Screen

cle database, ORACLE_HOME, ORACLE_SID, and TWO_TASK, must be set. The variable LD_LIBRARY_PATH must be set appropriately to find runtime libraries, including the RogueWave dynamically-linked libraries, Sun math libraries, and X11 libraries.

WCEDS Capabilities. WCEDS capabilities are implemented as separate, but related, executable programs. Coordination and consistency between the programs is provided by the operational specification data store and the scheduling and launch parameters provided by the transaction manager. The major seismic event processing is provided by the programs WCDPreprocessor, WCDCorrelator,

WCDEventLocator. There is another variation, WDCorrLoc, which combines correlation and detection for faster performance if it is not necessary to save the interim results. In addition, several utility programs for managing the WCEDS prototype have been developed, including programs to build grids and master images, and programs to assist with managing the database.

The *WCDPreprocessor* program is a C-language executable that performs site-specific signal processing to enhance signals with respect to noise prior to correlation. The input parameter files define the set of sites and the preprocessing time interval. The program also reads recipes and station/channel group definitions for all sites. For each site group, WCDPreprocessor retrieves the seismic waveforms from the database, and executes the algorithm specified by each recipe that uses that group. A recipe specifies a filter band, a station/channel group, a correlation distance range, a scale factor, a known preprocessing algorithm such as fixed short term average (STA) or spatial coherence, and a character string of input parameters to pass into the specified algorithm. A typical algorithm performs bandpass filtering of each waveform, a multichannel signal detection method that produces a single processed data stream, amplitude scaling, and, if necessary, rate conversion of the output signal to the sample rate of the master image. Some algorithms were developed in C, while others were developed in MATLAB and then converted into a C function using the MATLAB C compiler. The processed data streams are stored in the database using the recipe name as the channel name. The seismic waveforms also undergo f-k analysis in the recipe-specified filter band, with azimuth and slowness measurements taken at the master image sample rate. These streams are stored in the database using the recipe name with 'A' or 'S' appended as the channel name.

The *WDCorrelator* program is a C++ executable that correlates the site data with the detection master image for the set of discrete distances and possible origin times being considered. The input parameter files define the set of sites and recipes, the correlation and location time intervals, and the detection master image. The program reads the processed signal, azimuth, and slowness data streams from the database for each site recipe. WDCorrelator steps through each possible origin time, detection master image phase, site recipe, and recipe-dependent correlation distance range, correlating the master image phase "box-cars" with the processed signal at the appropriate time offsets. The azimuth and slowness measurements are determined as the data samples of the azimuth and slowness channels at the time of maximum correlation between the travel time pulse and the processed signal. The results are stored in the Correlation Matrix. For possible origin times in the location interval (all processed data are available at the master image phase times), the C Matrix results are trusted. For possible origin times later than the location interval (some processed data are unavailable at some master image phase times), the C Matrix results are untrusted. The following pseudocode illustrates the process:

for each possible origin time T in (location interval + correlation interval)
for each phase P in detection master image MI
for each site recipe R
for each distance D in distance range specified by recipe
if (MI phase defined at distance AND data available at MI phase time)
C Matrix value(T, P, R, D) = correlate signal and MI phase
C Matrix peak(T, P, R, D) = max correlation time
if (amplitude criteria fail OR slowness criteria fail)
C Matrix mask(T, P, R, D) = masked

The *WCDEventLocator* program is a C++ executable that detects seismic events in the C Matrix following a grid search algorithm. The input parameter files define the set of sites and recipes, the correlation and location time intervals, the detection and screening master images, and the grid. The program reads in the C Matrix and the processed data streams. Before the program can compute correlation sums, it must build a map for each grid point. The map specifies the offset to the phase correlation value at the correct site-to-grid point distance for each site recipe. *WCDEventLocator* steps through each grid point at each possible origin time, summing the phase correlation values for each site recipe at the appropriate site-to-grid point distance, if the value is unmasked and the azimuth criteria are met. If the maximum correlation sum exceeds a threshold, the corresponding grid point and origin time are considered a detected origin. The processed signals are correlated with the screening master image for the origin location and time. Any phase correlations that meet the amplitude, azimuth, and slowness criteria are considered associated arrivals. The time intervals during which the arrivals correlate with the detection master image are masked in the C Matrix. The grid search is repeated with the new mask until the maximum correlation falls below the event threshold. Origins that are detected at trusted times are considered legitimate seismic events. Arrivals associated with trusted origins are masked when processing the next location interval, so that they may not be built into another event. Origins that are detected at untrusted times are considered suspect. Arrivals associated with untrusted origins are unmasked when processing the next location interval, so that they may be rebuilt into the same, but now trusted, origin, or combined with new arrivals into a different origin.

The *WCDCorrLoc* program is a C++ executable that combines the functions of the *WCDCorrelator* and *WCDEventLocator* programs into a single program. Since the C Matrix is not written to file between the correlation and location steps, this program runs faster.

Two auxiliary programs create variations of components used by the *WCEDS* detector. The *BuildMImage* program generates master images with desired time resolution and duration, distance resolution and range, phase sets, pulse shapes, and scaling. The *BuildGrid* program generates a surface grid for a desired grid point spacing.

Several utility programs support a variety of analysis and visualization functions that have been useful to the project. *ClosestPointToSite* program lists the grid point closest to each site. *FindClosestPoint* lists the grid point closest to an arbitrary latitude and longitude. *ListGridPoints* lists all of the grid points in an ASCII file that can be viewed or input to a mapping tool. The *CorrSnapshot* program produces a set of snapshot files at a specified time spacing that contain the correlation sum for each grid point at a point in time. These global snapshot files are input to an animation program. *CompareWaveforms* compares the waveforms for two different channels for the same site to see if they are within a specified tolerance of each other.

Another category of utility programs assist with managing the CSS 3.0 database. *CreateTables* and *DropTables* are used to manage WCEDS-unique tables, if needed. *InsertSiteChans* and *RemoveSiteChans* are useful for managing the processed signal, azimuth, and slowness channel definitions, so these waveforms are visible to other tools that read the sitechan table. *CleanOrigins* and *CleanWaveforms* help remove the WCEDS output from the database after a research run.

WCEDS Seismic Domain Library. WCEDS-specific application or implementation uses of the more general seismic domain information is implemented in an application library (libwceds). This includes functions like translating the user specification into operations on the general seismic objects, or customizing them in some way. For example, the WCEDS application allows optional variations of the master image, such as distance-dependent phase scaling or modification of overlapping phases, and uses the grid points as candidate seismic event locations. Libwceds also provides the correlation mappings between the grid points, seismic monitoring stations, and master images, performance-enhancing implementation objects like the C Matrix, non-CSS uses of the Oracle database, and mechanisms for coordinating among the separate WCEDS executables.

The *CMatrix* module is at the heart of the WCEDS correlation and event detection process. The *CMatrix* object contains, for each possible origin time, detection phase, site recipe, and possible recipe distance bin, a phase correlation value, peak correlation time offset, working correlation mask, and trusted correlation mask. They are stored as separate, one-dimensional arrays in the order shown in Figure 5. The *CMatrix* provides methods for correlating a master image with site data, mapping a grid and set of sites to site recipe distance bin offsets, finding the maximum correlation sum for the map, masking an event, and managing trusted event mask data between location intervals.

The *WCEDSController* module manages control parameters such as time intervals, operating mode, processing steps to perform, and input and output directories and files.

<i>Time 1</i>	<i>Phase 1</i>	<i>Stream 1</i>	<i>Dist 1...Dist Nd1</i>	
		<i>Stream 2</i>	<i>Dist 1...Dist Nd2</i>	
		...		
			<i>Stream Ns</i>	<i>Dist 1...Dist Nds</i>
	<i>Phase 2</i>	<i>Stream 1</i>	<i>Dist 1...Dist Nd1</i>	
		<i>Stream 2</i>	<i>Dist 1...Dist Nd2</i>	
		...		
			<i>Stream Ns</i>	<i>Dist 1...Dist Nds</i>
	...			
	<i>Phase Np</i>	<i>Stream 1</i>	<i>Dist 1...Dist Nd1</i>	
		<i>Stream 2</i>	<i>Dist 1...Dist Nd2</i>	
		...		
			<i>Stream Ns</i>	<i>Dist 1...Dist Nds</i>
	<i>Time 2 { Phase 1 Stream 1 Dist 1 ... Phase Np Stream Ns Dist Nds }</i>			
	...			
<i>Time Nt { Phase 1 Stream 1 Dist 1 ... Phase Np Stream Ns Dist Nds }</i>				

Figure 5. Organization of Correlation Matrix

The *WCEDSData* module provides WCEDS-specific indexing for the processed signal, azimuth, slowness, and mask data to speed performance. It contains *SiteData*, *SiteStreamIndex*, *SiteDataIndex*, and *SiteDataPointers* data structures.

The *WCEDSEvent* module provides WCEDS-specific event and arrival attributes and methods, such as arrival association and the arrival correlation interval. It contains the *WCEDSEvent* (subclass of *Origin*), *WCEDSArrival* (subclass of *Arrival*), and *WCEDSArrivalList* objects.

The *WCEDS_DBI* module provides WCEDS-specific database access methods, such as storing and retrieving *CData* instances. It also manages the database connection based on specified account and table names. It contains the *WCEDS_DBI* (subclass of *DBInterface*) and *WCEDS_DBIGenerator* objects.

The *WCEDS_MI* module provides WCEDS-specific configuration of the general master image. This includes distance-dependent phase scaling, options for merging overlapping phases, and using subsets of phases for detection and screening. It contains the *MIConfiguration* and *MICGenerator* objects.

The *CData* module manages correlated data in a manner analogous to waveforms. It interfaces to a WCEDS-specific database table, *cddisc*, and provides the database and memory-related objects *CDataD* and *CData*, analogous to *wfdisc*

and *wfmem*, respectively. It also contains the objects *CDataDSet* and *CDataSet* for managing sets of correlated data records.

The *GridSiteLocation* module provides relative location information for a specified grid and set of sites, such as site-to-grid point distance, azimuth, and back azimuth.

The *MIMap* module provides WCEDS-specific mapping information to match a grid point and site to a master image distance phase list (object *MIMap*) or distance bin offset (object *MIBinMap*). This map is used when the master image and processed data are correlated directly, without using a *CMatrix*, as when identifying phases associated with a detected event.

Travel Time Model Library. The travel time model is implemented as a library package called the master image library (*libMImage*). Travel time information from a known source, such as the IASPEI 91 tables or empirical data collected by the University of California at San Diego, is interpolated and reformatted to provide expected travel time data at an arbitrary, specified time and distance spacing. The master image uses a specified shape function (e.g., a boxcar) to spread the expected travel time from a point to a finite-width pulse; for example, the expected time for P to travel 1 degree could be spread from 19.17 seconds to an arbitrarily-shaped pulse with a duration from 14 to 24 seconds. The master image was designed to be a general use library, independent of WCEDS. It has a set of operations that can be called to present the travel time information for requested phases to an application program. The master images used for the WCEDS detector operation to date have been based on the IASPEI 91 travel time curves. At the beginning of the project, we experimented with various time and distance spacings and pulse widths and shapes for the WCEDS application, and eventually settled on a 1-second, 1-degree spacing and a square pulse (boxcar), of variable width chosen so that the phase may span the distance between grid points.

The *libMImage* library is a collection of C-language functions and data structures that provide the master image travel time data to a calling program. These include routines that generate a set of master image files from IASPEI tables or other empirical travel time data, read the master image files and populate data structures for a calling program, create master images for a subset of phases, and provide memory management for the master image data structures.

Earth Grid Library. The earth model is implemented as a library package called the grid library (*libgrid*). It presents geographic location information as a set of discrete grid points (latitude, longitude, and depth) with a specified spacing. It was designed to be a general-use package. It provides geographic functionality such as calculating the distance and azimuth between a station and a grid point and finding the closest grid point to a specified target location, but not WCEDS-specific uses of the grid.

The *libgrid* library consists of a single C++ module, *Grid*. It contains the *GridPoint*, *LatitudeRing*, *LatRingGrid*, and *LatRingGridGenerator* objects.

Base Seismic Data Library. The base seismic library (*libbaseeis*) provides the CSS 3.0 database tables and relations as objects for use by an application program. The complete CSS 3.0 database has not been implemented -- just the capabilities needed by the WCEDS application. This includes selection and insertion for many of the fundamental tables such as *wfdisc*, *origin*, *event*, *arrival*, *assoc*, *site*, and *sitechan*, and a few other functions.

The *libbaseeis* library contains a separate module for each table. These are *Arrival*, *Associate*, *Data* (this is the *wfdisc* implementation), *Event*, *Origin*, and *Site*. A separate data file, *seismic_data_definitions.H*, defines native data types for the CSS 3.0 fields.

The *Data* module provides separate *DataD* and *Data* objects for database and memory-related operations analogous to *wfdisc* and *wfmem*, respectively. It was given the more generic name because it can represent data that are not, strictly speaking, waveforms. A *SourceSet* object groups a set of *Data* instances for the same station and channel, and a *DataSet* groups an arbitrary set of *SourceSet* instances. Originally, the software design envisioned simpler waveform operations, but as research progressed, the *Data* module evolved to resemble the *wfindex* structure found in the USNDC library *libwfm*.

The *Site* module provides operations for the site table. It contains *Site*, *SiteType*, *Station*, *Array*, and *SiteSet* objects. It was decided to implement the site type -- station or array -- as an aggregation (*Site-has-a-SiteType*), rather than an inheritance hierarchy (*Site-is-a-Station* and *Site-is-an-Array*). This allows *Site* to be used as a base class for application domain extended site classes, such as *WCEDSSite*, with station and array features accessible through the base *Site*. If the inheritance alternative had been implemented, application domain extended site classes would need to add another level to the inheritance hierarchy, using the subclasses as base classes; instead of a single *WCEDSSite* object, there would have been two additional objects, *WCEDSStation* and *WCEDSArray*.

The *DBInterface* module isolates the other base seismic objects from the specific details of database access. All of the database connection, table access, and error handling methods are provided by the *DBInterface* module. It contains the *DBInterface*, *DBIError*, and *DBIGenerator* objects.

Extended Seismic Data Library. The extended seismic library (*libwcdseis*) is built on top of the base seismic library, and provides seismic information and functionality beyond the CSS database. In some cases, this is simply adding an attribute or relation to a CSS object. Other common seismic objects like filters, signal processing algorithms, recipes, and station/channel groups are located in

this library. This library is incomplete, as a C++ version of the WCEDS preprocessor has not been fully developed.

The *DPAlgorithm* module provides preprocessing algorithms. A *DPAlgorithm* base class implements attributes and methods common to all algorithms, such as name, help text, default parameter string, station types for which the algorithm is valid, and a virtual execution method. Different algorithms are created as subclasses of *DPAlgorithm*, such as *DPAFixedStaLta3C* and *DPASpatialCoherence*.

The *DPStream* module provides WCEDS processed channel definitions, including a channel name, filter, correlation distance range, and preprocessing algorithm. It contains the *DPStream*, *DPStreamSet*, and *DPSSetGenerator* objects.

The *Filter* module provides a Butterworth filter as the *BWFilter* object. The filter methods call C functions in the USNDC library *libfilter*.

The *WCEDSSite* module provides WCEDS-specific extensions to the *Site* object. The *WCEDSBaseSite* object provides site-specific information not included in the CSS database, such as station/channel groups, recipes, and site detection thresholds. A *DataSource* data structure pairs a station and channel. A *Group* data structure defines the *DataSources* processed by some recipe for a station. A *Recipe* data structure defines a particular output data channel produced for a station, linking a *Group*, a *DPStream*, a scale factor, and a static station correction. The *WCEDSSite* object is a combination of a *WCEDSBaseSite* and a basic *Site*. This module also contains the *WCEDSSiteSet* and *WSSetGenerator* objects.

General Purpose Utility Library. Miscellaneous general purpose C++ utility functions are provided in a separate library (*libgpu*). These include directory path and file name manipulation, keyword matching, and string operations (module *GeneralUtility*); file i/o utility functions such stripping comment lines from input files and a file error class hierarchy (module *FileUtility*); and date/time utilities (module *DTUtility*).

Third Party Libraries. Where appropriate, the WCEDS prototype incorporates some other libraries not developed at Sandia. A number of USNDC libraries are used for seismic functions; examples include waveform filtering, f-k analysis, and input parameter parsing. The USNDC libraries incorporated into WCEDS are *libfk*, *libfilter*, *libgeog*, *libpar*, *libwav*, and *libwfm*. To minimize the resources spent developing software not fundamental to the waveform correlation or general seismic monitoring problems, commercial libraries were used for a few specific, ancillary purposes. BXPro from Integrated Computer Solutions allowed a simple GUI to be implemented quickly. DBTools from RogueWave allowed quick development of database access functions. The MATLAB Library and Compiler from The Math Works allowed signal processing algorithms to be developed quickly in the MAT-

LAB environment, and then compiled into the WCEDS executable. The commercial libraries primarily support rapid prototyping; their use in an operational environment would need to be reviewed.

Discussion

Effectiveness of Design

Overall, the WCEDS software design effectively accomplished its mission:

- The prototype functions as a global detector based on a waveform correlation algorithm, reading in raw waveforms as input and reporting detected seismic events as output. It has been used in numerous experiments investigating the usefulness of waveform correlation in an automated CTBT monitoring system. Given the small magnitude of events that must be detected, and the locations of the IMS primary stations, experience with the WCEDS prototype shows that waveform correlation on a global scale will not provide significant improvement over the current monitoring methods. Applied on a regional basis, however, the power of the technique to sense multiple phases may be beneficial.
- The prototype operation is compatible with the USNDC environment. A CSS 3.0 database is the primary input/output interface. WCEDS result data has been successfully exchanged with other programs that access a CSS 3.0 database, such as ARS and MatSeis. The various executable programs operate independently of each other. A WCEDS executable may be scheduled when a time interval of input data is available.
- Existing USNDC software and commercial third-party libraries have been incorporated, supplying a variety of functions: seismic processing capabilities such as filtering and f-k analysis, GUI feature implementation, database access, and general signal processing capabilities.
- The WCEDS-specific software entities and functions have been developed in a separate application layer on top of the more general seismic processing software. The general seismic libraries are available to support other seismic monitoring applications.
- The software design proved to be fairly robust and flexible, adapting readily to new requirements. Often one experiment would suggest changes or new ideas to try. These were accommodated without changing the object

model, by adding attributes or methods to existing objects. The Correlation Data object was dropped for performance reasons, and the Correlation Matrix was populated via the *Site-Has-A-Site Correlation Set* relationship.

- The GUI shows great promise for helping the user understand, configure, and track WCEDS execution.

The WCEDS software design has certain shortcomings:

- Other than the early formulation of the Correlation Matrix to improve performance over a straight-forward correlation algorithm, little attention has been paid to performance issues. Memory usage and system interfacing have not been designed, but left to default operation.
- The WCEDS prototype requires a knowledgeable user. End-to-end operation, from waveform to detected event, requires knowledge of system parameters at an implementation level, how to sequence the WCEDS executables and how they interact, what intermediate and final results are produced, how one run could affect another, and how to clean up the system. It is anticipated that a fully-functional GUI could assist tremendously with managing WCEDS.

State of Implementation

The WCEDS GUI is not yet fully implemented. Currently, the GUI manages WCEDS operation. The user may set operational parameters such as start time and interval size, and launch the transaction manager to control the scheduling and sequencing of executables. The vastly more complicated task of configuring analysis and processing parameters is under development. Shortcomings of the BXPro GUI-development tool are evident here: The tool allowed the rapid development of a simple user interface, but as the WCEDS GUI has grown larger and more complex, the tool has become more difficult to use. Discussions with developers on other projects indicates that this is a common experience in using most GUI builders.

The object model encompasses a more complex correlation algorithm than has been implemented. The WCEDS prototype employs a single, global, teleseismic, surface master image, and a surface grid. The design intended a set of master images and grids that spanned several depth ranges, but implementation continually took lower precedence to other project needs. The design also intended to use different master images for different regions of the world, so that the master image that a site correlated with for a particular grid point depended on the site-to-grid point distance and azimuth. This has not been implemented.

Reusability

The WCEDS prototype provides two likely candidates for software reuse: the general seismic processing libraries and the GUI. The master image, grid, and base and extended seismic libraries provide a solid framework for building other applications. Many objects, however, have not been fully implemented for the general case, but only those aspects needed by WCEDS. Implementing additional methods to round out a general capability can be done as the opportunity arises. The WCEDS GUI encompasses the fundamental concepts and relationships, and many of the parameters, needed to configure seismic processing routines at the IDC or USNDC. Some of the details may vary, but the GUI is extensible to the operational systems.

Lessons Learned

Viewing Correlated Data As Analogous To Waveforms. It was recognized early on that performance could be greatly improved by splitting the correlation algorithm into two steps. The first step correlates the site data with the master image for the set of discrete distances and possible origin times being considered. The second step sums, for a given grid point and origin time, the correlated data value for each site at the appropriate site-to-grid point distance.

In the first step, there are no dependencies between sites or between time intervals. Correlated data was viewed as a continuous record analogous to a waveform. It could be generated for a given site and time interval independent of other considerations, and stored in a database table and flat file format analogous to the `wfdisc` and `.w` file of the CSS 3.0 schema. Later, the location code could retrieve correlated data for an arbitrary set of sites and an arbitrary time interval, independent of the correlator operation.

Unfortunately, this implementation choice proved disastrous for performance. The second step must sum across all sites at the same point in time. Populating a Correlation Matrix organized time-first, site-second from a set of Correlation Data organized site-first, time-second, was horrendously time consuming. Different organizations of the Correlation Matrix were considered, but all required a proliferation of pointers and mapping complexity for the summing process. The Correlation Data object was dropped. Instead, the first step of the correlation process populates the Correlation Matrix directly, requiring full knowledge of the set of sites and the time interval needed by the locator.

As experiments with the WCEDS prototype yielded new insights into the waveform correlation technique, the Correlation Matrix grew. Originally containing a single correlation value per site for a given distance and time, it expanded to include separate correlation values each phase, as well as azimuth and slowness measurements and mask data for each correlation value. Eventually it grew

so large that it takes longer to write or read a file containing a Correlation Matrix than it does to compute one. A combined correlator/locator executable was developed to avoid the file access time. This also ensured the consistency of input parameters between the correlation and location steps. Unless there was an analytic or diagnostic need to examine the Correlation Matrix, typical WCEDS operation now uses the combined program instead of separate correlator and locator programs.

Database Interface. The general seismic libraries contain objects that correspond to relational database tables or views. The database access methods must convert the results of database queries into instances of objects. Object-oriented databases should provide advantages in faster performance and version control, and may bear consideration in the future. To shield the seismic objects from the database particulars, the access methods that interface to the actual database are isolated in a database interface object. The database interface object implementation is unsatisfactory in that it must know about each of the seismic objects, and will potentially grow large and cumbersome. Implementing it as a template library may be a better solution.

Input Parameter Parsing. The USNDC libpar library provides extremely useful and simple-to-use functions for command line and file-based input parameter parsing. These functions were used extensively in the WCEDS prototype. Libpar seems to lack one feature, though, that seemed to make the most sense for implementing some WCEDS parameters -- the ability to input a set (vector) of character strings.

It would be possible to use libpar for this type of parameter set, by using a count parameter for the number of strings and a sequentially-numbered keyword sequence for the individual string parameters. Numbered parameters are labor-intensive and awkward to maintain as the parameter sets change; since the WCEDS control and analysis parameters changed frequently during development, the numbered parameter approach was rejected. Consequently, the WCEDS objects that need input parameters of this type must parse the input file directly. Some low level file i/o functions were developed in the WCEDS general purpose utility library, libgpu, to aid this process. Little effort was expended in this development, by the expedient of placing these parameters in a separate file from the parameters parsed using libpar routines. The lines of the file could be parsed easily with the C++ iostream white-space-delimited functions.

Maintaining multiple parameter files for configuring the same object is a less-than-ideal solution. One alternative is to change the WCEDS functions to recognize both white space and equal sign delimited keywords. While the parameters could then be mixed in a single file, the file would need to be read twice. A better approach is to use the GUI to hide the messy parameter file details from the user,

and use libpar to parse the file. The best solution, perhaps, is to add this capability to libpar.

Commercial Library Use. The commercial libraries used -- BXPro, DBTools, and MATLAB -- aided rapid development and prototyping, but their future use should be considered carefully. Generality often comes at the cost of performance, with a lot of overhead, and these packages were no exception.

Using BXPro jump-started the GUI development; without it, the GUI development would have proceeded at a significantly slower pace. As the GUI implementation grew in size and complexity, however, the tool became difficult to use. It restricts design and implementation choices and slows with large structures. Discussions with developers on other projects indicate that some projects which need more than a simple GUI eventually abandon any GUI-development tool, and implement directly at the Motif level.

DBTools provided an easy means to program object-oriented database access, and transitioned pretty smoothly with an upgrade of Oracle. Since all data is stored as DBTools-specific data types, it entails the overhead of conversion to native types to prevent the proliferation of DBTools data types throughout the application code. Licensing is cumbersome. It isn't clear that DBTools provided significant advantage over the database access methods currently in use.

MATLAB proved invaluable as a vehicle to experiment with signal processing algorithms. Using the C compiler to convert the MATLAB routine into a C function was both easier to integrate and faster to execute than the previous MATLAB engine method. It is likely, however, that faster performance could be achieved by implementing an algorithm directly.

Software Development Process. The WCEDS prototype design and implementation expended a fair amount of effort on operational aspects of the software appropriate to a production software delivery. As a result, the formal analysis and design produced a general seismic processing base that can be reused for other projects. On the other hand, it took longer to accomplish the research objectives than may have been possible if the formalism were relaxed and the delivery considerations deemphasized. Future research-oriented software development efforts should consider carefully the extent to which operational requirements are allowed to affect the development process.

The WCEDS prototype development closely followed the Object Modeling Technique (OMT) (Rumbaugh et al. 1991, Derr 1995). As a whole, the software development team believes that this object-oriented approach resulted in a superior product than functional decomposition or other ad hoc methods would have produced. The software implementation is easily understandable in terms of the application domain -- seismic monitoring and waveform correlation -- which was

invaluable in meeting the applied research objectives of the project. The object-oriented approach was easier to develop and maintain, affords greater opportunity for reuse, and minimizes the ripple impact of modifications.

Some members of the team believe that OMT, though not a bad choice, was flawed in certain aspects. The object model is developed iteratively, and may be changed substantially at any point during the analysis, design, and implementation phases. The level of detail incorporated into the model is left to the discretion of the developers; it is permissible to ignore attributes and relationships if the developer considers them trivial or obvious. Relationships between objects are implied by the lines and symbols drawn between them, rather than represented explicitly as attributes. Relationships can be ambiguous, with one notation representing multiple kinds of relationships, or multiple notations for the same kind of relationship. It is not necessarily clear how to proceed from design to implementation. The iterative and ambiguous nature of the object model will likely cause important design decisions to be delayed until the implementation phase, contrary to the generally accepted best practice of making decisions as early as possible to minimize impact.

Of course, a design team can agree what to include and how to represent it, but this requires time to reach consensus, and the decisions wouldn't necessarily be consistent with other projects using OMT. Also, the more "rules" a team adopts, the more the process resembles a different object-oriented methodology. The argument for the OMT approach is that the implementation is flexible; the developer isn't constrained and possibly prevented from doing something. The argument that more formal methods do, in fact, prevent a developer from implementing any capability has not been made.

The informality and incompleteness of OMT can lead to problems, some of which were experienced by the WCEDS team. It was extremely difficult to come to closure on the analysis phase because of disagreement over the level of detail needed. The team did adopt some conventions for representing relationship attributes and notation, and strove to develop a high degree of completeness. Future projects should consider another technique that seems to produce a more formal, complete, and unambiguous object model, with a straightforward mapping between the model and implementation, without any loss of capability.

Conclusions

The WCEDS prototype has been used in numerous experiments, and has proven to be a successful vehicle for investigating waveform correlation techniques in the context of CTBT monitoring. Experience shows that, except for large events, the IMS primary seismic network does not see many phases beyond

the first arrival. Since much of the power of waveform correlation comes from aligning multiple phases, a global WCEDS that sees first arrivals only for smaller magnitude events will not likely produce a better-quality event bulletin than the current processing. On a local or regional scale, however, with more phases present, WCEDS shows promise.

To use the WCEDS prototype in a regional application, some additional capabilities would need to be developed. Some of these have already been identified and designed, but not yet been implemented. Consideration of the seismological properties of regional signals may lead to further requirements. Some suggestions are listed below:

- Implement the multiple depth range capability.
- Implement regional travel time models.
- Modify the Grid object to represent regional grids.
- Develop techniques for eliminating signals from outside the region.
- Investigate seismological properties of regional signals like frequency content, visibility of smaller magnitude signals, coda complexity, etc. that may hinder or aid waveform correlation, and determine additional software requirements.
- Investigate the impact of a small or even single station set on the results of the waveform correlation technique, such as false alarm rate and implications for grid and time resolution, and determine additional software requirements.
- Investigate use of reference events to indicate similarity with historical events, possibly by correlating with a reference event master image.

References

- J. Anderson, W. E. Farrell, K. Garcia, J. Given, H. Swanger, 1990, *CSS Version 3.0 Database: Schema Reference Manual*, SAIC Center For Seismic Studies, ARPA Order No. 6266-1, 5, & 7.
- J. I. Beiriger, S. G. Moore, and J. R. Trujillo, 1996a, Waveform Correlation Event Detection System Analysis Document report for the Department of Energy, Sandia National Laboratories, Decision Support Systems Department 9432.
- J. I. Beiriger, S. G. Moore, and J. R. Trujillo, 1996b, Waveform Correlation Event Detection System Design Document report for the Department of Energy, Sandia National Laboratories, Decision Support Systems Department 6532.
- J. I. Beiriger, S. G. Moore, and J. R. Trujillo, 1997a, Waveform Correlation Event Detection System User's Manual report for the Department of Energy, Sandia National Laboratories, Decision Support Systems Department 6532.
- J. I. Beiriger, S. G. Moore, J. R. Trujillo, and C. J. Young, 1997b, Software Design and Operational Model for the WCEDS Prototype, in *Nineteenth Annual Seismic Research Symposium On Monitoring A Comprehensive Test Ban Treaty*, held in Orlando, FL, September 23-25, 1997. M. J. Shore, R. S. Jih, A. Dainty, and J. Erwin, ed., Defense Special Weapons Agency, Alexandria, VA, HQ Air Force Technical Applications Center, Patrick AFB, FL, and Department of Energy, Washington, DC.
- K. W. Derr, 1995, *Applying OMT: A Practical Step-by-Step Guide to Using the Object Modeling Technique*, SIGS Books, New York, NY.
- O. Kulhanek, 1990, *Anatomy of Seismograms*, Elsevier Science Publishing Company B. V., Amsterdam, The Netherlands.
- J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W. Lorensen, 1991, *Object-Oriented Modeling And Design*, Prentice-Hall, Inc., Englewood Cliffs, NJ.
- P. M. Shearer, 1991, Imaging global body wave phases by stacking long-period seismograms, in *J. Geophys. Res.*, vol. 96, pp. 20353-20364.
- P. M. Shearer, 1994, Global seismic event detection using a matched filter on long-period seismograms, in *J. Geophys. Res.*, vol. 99, pp. 13713-13725.
- C. J. Young, J. M. Harris, J. I. Beiriger, S. G. Moore, J. R. Trujillo, M. M. Withers, and R. C. Aster, 1996, *The Waveform Correlation Event Detection System*

Project, Phase I: Issues in Prototype Development and Testing, SAND96-1916, Sandia National Laboratories, Albuquerque, NM.

- C. J. Young, 1997a, Status Report On The Waveform Correlation Event Detection System (WCEDS) Project report for the Department of Energy, Sandia National Laboratories.
- C. J. Young, J. I. Beiriger, J. M. Harris, S. G. Moore, J. R. Trujillo, M. M. Withers, and R. C. Aster, 1997b, Testing the Waveform Correlation Event Detection System: Teleseismic, Regional, and Local Distances, in *Nineteenth Annual Seismic Research Symposium On Monitoring A Comprehensive Test Ban Treaty*, held in Orlando, FL, September 23-25, 1997. M. J. Shore, R. S. Jih, A. Dainty, and J. Erwin, ed., Defense Special Weapons Agency, Alexandria, VA, HQ Air Force Technical Applications Center, Patrick AFB, FL, and Department of Energy, Washington, DC.

Distribution:

- | | | | |
|---|---|----|---|
| 1 | Department of Energy
1000 Independence Ave. SW
Washington D. C. 20585-0420
Leslie Casey DOE/NN-20 | 2 | Lawrence Livermore National
Laboratory
PO Box 808
Livermore, CA 94551-0808
Dave Harris LLNL L205
Jay Zucca LLNL L205 |
| 3 | Airforce Technical Applications
Center
1030 South Highway A1A
Patrick AFB, FL 32925-3002
Carol Finn AFTAC/TTR
Dave Russell AFTAC/TTR
Mark Woods AFTAC/TTR | 1 | Pacific Northwest National
Laboratory
PO Box 999
Richland, WA 99352
Dan Hagedorn PNNL K5-12 |
| 4 | SAIC
10260 Campus Pt. Drive
San Diego, CA 92121
Ronan LeBras
Tom Sereno
Jon Sessions
Henry Swanger | 1 | MS 0655 D. B. Carr, 5736 |
| 1 | Center for Monitoring Research
1300 North 17th Street, Suite 1450
Arlington, VA 22209
Bob North CMR/SAIC | 1 | 0655 E. P. Chael, 5736 |
| 1 | Department of Earth Sciences
New Mexico Institute of Mining
and Technology
Socorro, NM 87801
Rick Aster | 1 | 0655 J. P. Claassen, 5736 |
| 1 | Center for Earthquake Research
and Information
University of Memphis
Memphis, TN 38152
Mitch Withers | 1 | 0655 J. M. Harris, 5736 |
| 2 | Los Alamos National Laboratory
PO Box 1663
Los Alamos, NM 87545
Wendee Brunish LANL F659
George Randall LANL C335 | 1 | 0750 C. J. Young, 6116 |
| | | 1 | 0750 M. C. Walck, 6116 |
| | | 1 | 0979 D. B. Shuster, 5704 |
| | | 1 | 0979 L. S. Walker, 5704 |
| | | 1 | 1138 H. M. Armstrong, 6533 |
| | | 10 | 1138 J. I. Beiriger, 6532 |
| | | 1 | 1138 S. K. Chapa, 6533 |
| | | 1 | 1138 T. L. Edwards, 6532 |
| | | 1 | 1138 L. J. Ellis, 6531 |
| | | 1 | 1138 D. R. Funkhouser, 6532 |
| | | 1 | 1138 J. R. Hipp, 6532 |
| | | 1 | 1138 R. G. Keyser, 6532 |
| | | 1 | 1138 B. N. Malm, 6532 |
| | | 3 | 1138 S. G. Moore, 6532 |
| | | 1 | 1138 E. Shepherd, 6532 |
| | | 1 | 1138 R. W. Simons, 6532 |
| | | 1 | 1138 J. R. Trujillo, 6532 |
| | | 1 | 9018 Central Technical File,
8940-2 |
| | | 5 | 0899 Technical Library, 4916 |
| | | 2 | 0619 Review & Approval
Desk, 12690,
For DOE/OSTI |