**Chronos: The Chamber Recording System**

**Specification and Design Document**

by Eric Lyness <eric.i.lyness@nasa.gov>

This document format is based on the Software Design template originally created by Bradford D. Appleton, Copyright © 1994-1997 by Bradford D. Appleton

**Table of Contents**

[1. Introduction 1](#_Toc166398920)

[1.1. Document Outline 2](#_Toc166398921)

[1.2. Document Description 4](#_Toc166398922)

[1.2.1. Introduction 4](#_Toc166398923)

[1.2.2. System Overview 5](#_Toc166398924)

[2. Design Considerations 5](#_Toc166398925)

[2.1. Assumptions and Dependencies 5](#_Toc166398926)

[2.2. General Constraints 5](#_Toc166398927)

[2.3. Goals and Guidelines 6](#_Toc166398928)

[2.4. Development Methods 6](#_Toc166398929)

[3. Architectural Strategies 6](#_Toc166398930)

[4. System Architecture 7](#_Toc166398931)

[4.1. Subsystem Architecture 8](#_Toc166398932)

[5. Policies and Tactics 8](#_Toc166398933)

[6. Detailed System Design 9](#_Toc166398934)

[6.1. Classification 9](#_Toc166398935)

[6.2. Definition 9](#_Toc166398936)

[6.3. Responsibilities 10](#_Toc166398937)

[6.4. Constraints 10](#_Toc166398938)

[6.5. Composition 10](#_Toc166398939)

[6.6. Uses/Interactions 10](#_Toc166398940)

[6.7. Resources 10](#_Toc166398941)

[6.8. Processing 10](#_Toc166398942)

[6.9. Interface/Exports 11](#_Toc166398943)

[6.10. Detailed Subsystem Design 11](#_Toc166398944)

[7. Glossary 11](#_Toc166398945)

[8. Bibliography 11](#_Toc166398946)

# Introduction

This document describes the system called Chronos. Chronos records temperature sensors, pressure sensors and other instrumentation monitoring a thermal-vacuum chamber.

This document contains two main sections: The Specifications section and the Software Design section. The specification section defines the hardware, hardware interfaces and software functionality. The design section describes the software implementation of the system.

The purpose of this document is twofold: First, it communicates the operation of the system to the larger team of operators, managers, technicians and designers. Second, it records the as-built implementation to guide future engineers in maintaining and updating the system.

This document is not a user manual for chamber operators. Related documents are:

* The Chronos User Manual -- provides step-by-step handbook for operators
* The Chronos Test Plan – Provides steps for verifying operation and performance

## Document Outline

Here is an overview of this document’s outline:

* Introduction
* System Overview
* Design Considerations
  + Assumptions and Dependencies
  + General Constraints
  + Goals and Guidelines
  + Development Methods
* Architectural Strategies
  + strategy-1 name or description
  + strategy-2 name or description
  + ...
* System Architecture
  + component-1 name or description
  + component-2 name or description
  + ...
* Policies and Tactics
  + policy/tactic-1 name or description
  + policy/tactic-2 name or description
  + ...
* Detailed System Design
  + module-1 name or description
  + module-2 name or description
  + ...
* Glossary
* Bibliography

# System Overview

This system is intended to replace the ageing Thermal-Vacuum Data System (TVDS) that records data for the Sample Analysis at Mars vacuum chamber and the Mars Organic Molecule Analyze (MOMA) vacuum chamber. TVDS relies on an Oracle database that is expense and difficult to update. Also, both MOMA and SAM use the same Oracle server. It is preferable for each chamber to be independent of each other.

The system consists of data acquisition hardware that records temperatures, pressures and other instrumentation and the software that records, displays and archives the data. Figure 1, below, shows a schematic of Chronos.

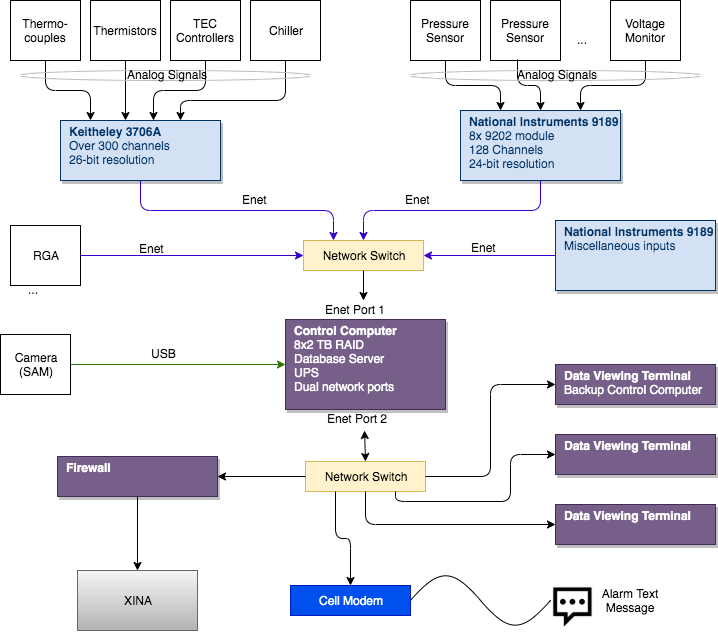


Figure System Layout

## System Overview: Hardware

As shown in Figure 1, the hardware consists of:

* Keithley 3706A acquisition system for up to 300 channels with 26-bit resolution
  + Relatively low speed but high resolution
* National Instruments 9189 with up to 128 channels with 24-bit resolution
  + Relatively high speed, high reliability
* A Residual Gas Analyzer (RGA)
  + Not required, can be enabled/disabled in software
* A secondary National Instruments 9189 for extra inputs as required by special configurations of the instrument and additional GSE.
* Server computer with 2TB RAID array with uninterruptable power supply
* Firewall computer allows access outside the chamber network to the cloud-based database system referred to as XINA
* A cell modem on a simple computer such as an Arduino provides a means to send text messages in the event of an emergency such as a limit exception or power outage.
* Network hardware such as cables and switching systems

## System Overview: Software

TVDS acts as the model for Chronos software. Some key features of TVDS and the Chronos implementation of those features:

|  |  |
| --- | --- |
| TVDS | Chronos |
| Records all channel in database once every two minutes. | Records all data in database once every minute. Some channels can be customized to record at higher rates |
| Screens update once every two minutes. | Screen updates can be defined per channel, up to once per second. |
| Data is recorded on a local server in an Oracle database. | Data is recorded on a local server in a National Instrument database (TDMS) and stored in the cloud in XINA’s MySql database |
| Allows channels to be renamed and reconfigured to support different instrument configurations. | Same |
| Allows data to be transmitted to instrument ground software such that it is recorded with instrument telemetry. | Same |
| Data can be viewed from any number of terminals on the same network with the Oracle Database | Data can be viewed on any terminal on the network or from anywhere on the Internet with XINA access (Internet-based viewer includes some latency and only displays once-per-minute data) |
| Provides visual alarm when values exceed limits. | Provides visual alarms and text-message notifications when limits are exceeded or the system fails. |

# Design Considerations

This system monitors temperatures, pressures and other instrumentation connected to flight hardware. The primary design consideration is reliability. The software needs to be robust and the archive needs to be secure such that past records can be easily and reliably retrieved in the event of a flight article failure, even if the failure occurs years later.

The system should be scalable such that, as a goal, it can record data and support normal operations for a period of 10 years without hardware upgrades.

## Assumptions and Dependencies

We assume that 690 information technology team will supply and maintain the firewall computer and further they will continue to provide access through the firewall to the XINA cloud-based database. The server and other computers will run the most recent Windows operating system and the developers will have administration access to the computers.

We assume the cell modem can be configured to get reception in the labs where the chambers are located.

## General Constraints

The software should be capable of running continuously, without interruption for up to 30 days at a time. There should be no degradation in performance over time.

All data should be recorded and archived on the local server as well as transmitted to XINA.

Outside connections, such as the connection to the XINA cloud-based database, do not have the same requirement. Therefore, the system needs to be able to cache data and upload to XINA when the link is available.

Hardware connection should be via Ethernet whenever possible. USB devices interface should be avoided if possible as they have limit distance, strain relief issues and they do not scale well.

## Goals and Guidelines

Chronos should do everything that the existing TVDS software is capable of plus many additional features.

Previous thermal-vacuum campaigns have often used extraneous ground support equipment. In many cases additional instrumentation and laptop computers were used to store data from this GSE. This data was then difficult to correlate with the chamber data and instrument telemetry. One of the goals of the upgrade is to allow the relatively easy addition of data inputs to the system.

The system should also be capable of dealing with data at different rates. While the bulk of the data is recorded at a one-minute pace, other rates should be allowed, up to 1Hz. Rates faster than 1Hz should be allowed where hardware and computing power supports it and as long as reliability is not sacrificed. High speed data may be recorded in a separate archive, if necessary.

## Development Methods

The software will be developed entirely in LabVIEW. There should be no dependencies on external development environments such as Python or C.

Once developed, the software will run as an executable and not require a LabVIEW development license.

# Specifications

This section describes the detailed operation of the software.

The software can be broken down into these interfaces:

* Acquisition Hardware Interface – Communication with Keithley and National Instrument hardware
* Channel Definitions
* Local storage and archive of data – Definition of the local database
* Message Logs
* Alarms
* Data Viewing User Interface
* Instrument Ground System Communications
* XINA Interface
* External Instrumentation – Specifically the Residual Gas Analyzer

## Acquisition Hardware Interfaces

The primary hardware interfaces connect the server with the Keithley 3706A multiplexing digital multimeter and the two National Instruments 9189 chasses. Both of these devices communicate over Ethernet.

### Keithley 3706A Interface



Figure Keithley 3706A

The Keithley 3706A is a “main frame” that is capable of containing up to 6 slots of expansion cards for as many as 576 two-wire channels. Wires are landed on cards the slide into the slots of this main frame. Both SAM and MOMA chamber already use this instrument.

Communications with the instrument is via Ethernet. All configured channels will be scanned once per minute. Each physical channel’s type can be configured in software according to the capabilities of the Keithley 3706A.

### National Instruments 9189 Chassis



Figure NI 9189 Chassis

The NI 9189 is an Ethernet-based chassis that can contain up to 8 modules for reading different types of electrical signals.

Chronos has two NI 9189 chasses, referred to as the Primary chassis and the All-purpose chassis. The Primary chassis is wired to the most critical values from the chamber such as pressures that need to be continuously monitored in most chamber operational modes. The All-purpose chassis can be re-configured to support extra GSE inputs depending on the test configuration. There should be no permanent connections to the All-purpose chassis such that it can be moved around to convenient locations when a particular configuration requires it.

Each cRIO module of either chassis can be configured for a different data acquisition rate, from once per second to once per minute. The All-purpose chassis can also have modules that are samples at faster rates, depending on the capability of the module. However, data acquired at rates greater than 1 sample per second will be recorded in files rather than the database. See section TBD below.

## Channel Definitions

Each temperature, pressure or other physical signal is referred to as a channel. Channels can also be “virtual” meaning that they are values computed from other channels.

Each channel has many attributes that can be explicitly defined by operators via the Chronos user interface. These channel definitions can be viewed but not edited in XINA.

Table Channel Attributes

|  |  |
| --- | --- |
| **Attribute** | **Purpose** |
| Name | User selected name for the channel to be used in plots and reports |
| Device | Physical device such as “NI-9189-Primary” or “Keithley-3706” |
| Module | The module or slot the signal is wired two. The NI-9189 has up to 8 modules, each with several channels. |
| Channel | The physical channel this signal is wired to on the device. |
| Limits | The yellow and red alarm limits for both high a low. Limits can also be disabled on channel. |
| Alarm Reaction | A list of things to do in the event an alarm condition occurs on this channel. This can include setting off audible alarm, sending a text message, etc. |
| Scaling | Parameters for scaling an input signal into engineering units. Scaling can be linear or polynomial. Default is no scaling. |
| Configuration | Configuration reflects the physical signals and depends on the device. For instance, the Keithley-3706A allows several different types of temperature sensors, voltage measurements, resistance measurements and other configurations. This attribute tells the software how to configure the device. |
| Unit Type | The general class of measurement without units. For instance, “Temperature” or “Resistance” as opposed to “C” or “Ohms” |
| Units | The text that will be displayed on plots and reports of this channel. |
|  |  |
|  |  |

The user interface allows operators to create “Virtual Channels”. These are channels that are not physical but are the result of calculations from other channels. For instance, a virtual channel “Resistance” could be created that is a voltage channel divided by a current channel.

Virtual channels have the same attributes as physical channels but the “Device” has the value “Virtual” and the equation to calculate the value is in the “Configuration” field.

For example, if the Keithley 3706A has a channel to measure voltage named “Device Voltage” and a channel to measure current named “Device Current”, the virtual channel named “Device Resistance” could be created and the configuration field would have the value “Device Voltage / Device Current”.

## Local Storage and Archive

Data will be stored in two places: locally on the server computer and remotely in the XINA database. The local storage will always be considered the master copy.

Local data will be stored on the Chronos server. The server has a large disk sized for 10 years of recording and a RAID system to ensure the safety of the data. Normal rate data (1 Hz and slower) is stored in National Instruments Technical Data Management Streaming (TDMS) files. Faster data is stored in separate files.

The Chronos software will send data to the XINA server in the cloud via the firewall computer at one-minute intervals. The network connection to XINA should not interfere with Chronos operation. If the XINA server cannot be contacted, the data should be recorded in a cache and sent to XINA when the software and connects again.

If the XINA database is corrupted, it can be fully re-populated with the data located on the local server.

High speed data is stored in files. These files are sent to XINA but not into the database. They can be downloaded by XINA users but viewing the contents of the files may not be supported.

## Message Logs

Events will be recorded in a running log of time stamped messages. Events include, but are not limited to, alarms, changes to channel definitions, hardware errors, user operations such as starting to stopping the software, open and close of connections to GSE. Messages can also be manually entered by the operator.

Message logs are displayed on the server’s terminal and are sent to XINA.

## Alarms

Every channel, physical or virtual, has four limits:

Table Alarm Levels

|  |  |
| --- | --- |
| Low-Red | If channel value falls below this, set red alarm state |
| Low-Yellow | If channel value falls below this, set yellow alarm state |
| High-Yellow | If channel value exceeds this, set yellow alarm state |
| High-Red | If channel value exceeds this, set the red alarm state |

Each of the four limits also has the following attributes (defined by the users):

Table Alarm Attributes

|  |  |
| --- | --- |
| Enable | If false, this limit is ignored |
| Deadband | Time, in seconds, during which a value must be out of limits before the alarm is set. |
| Action List | Any number of the following actions: Alarm Light, Text Message, Go Safe |

Each new data point for each channel is evaluated against the limits defined for the channel. If a channel changes to alarm state, that alarm is logged and indicated on the screen. If there are any actions in the channel’s action list, they are executed.

### Alarm Actions

When an alarm it occurs, it will always be indicated on the Chronos local screens. In addition, each alarm can have a list of actions to perform when alarm state occurs. The list can contain 0 or more actions.

Table Alarm Actions

|  |  |
| --- | --- |
| Alarm Light | Illuminate the alarm light in the room |
| Text Message | Send a text message to a given set of people |
| Go Safe | (Future) If the system has control over any devices, set those devices into a safe state |

## Data Viewing User Interface

Operators locally in the control room or on remote terminals on the control room network, can select any channels and view them on a time-based graph. Operators can add any number of channels to a graph. Operators can open any number of graphs.

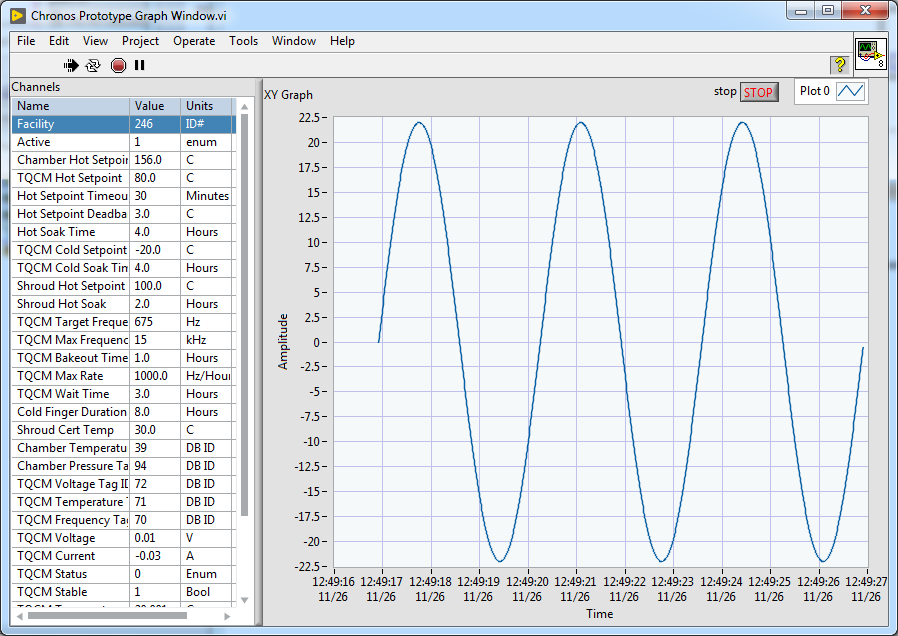
Graphs have a maximum history of approximately 30,000 (TBC) data points per channel. Therefore, the high-speed channels will show less time but the same number of data points.

Table Data Rates and Graph History

|  |  |
| --- | --- |
| **Data Rate** | **Time on graph** |
| 1 Sample/Minute | 20 days |
| 1 Sample/Second | 8 hours |
| 10 Samples/Second | 50 Minutes |
| 1000 Samples/Second | 30 Seconds |

Older data can be viewed using XINA or a data viewer designed for archive data viewing that runs on the server with the archives.

Table Sample User Interface



## Interface to Instrument Ground Software

The MOMA and SAM GSE software can connect to Chronos via a TCP socket connection. The protocol will be defined in detail in the Chronos User Manual.

The GSE software will connect to Chronos and subscribe to a list of channels by channel name. The Chronos software then sends a packet to the GSE each time one of the subscribed channels updates. If the GSE attempts to subscribe to a channel that is not defined, an error message will be sent but any channels that are valid are still sent.

## XINA Interface

XINA is an application running on a cloud server that provides an interface to a database containing time-based telemetry. The Chronos server connects to XINA via the firewall computer (as shown in Figure 1 System Layout). The details of this connection are defined in the Chronos user manual.

Chronos will send data to XINA once per minute. If data is recorded at rates faster than once per minute, all data recorded during the minute will be averaged and a single value sent for each channel. For high speed data (rates faster than 1Hz), only files containing the data will be sent to XINA. The files will be sent when they are closed.

Data is sent to XINA in .JSON format as defined in the Chronos User Manual.

XINA users can log into XINA from any internet connection using a Chrome browser. The web-based user interface allows selection of any time range and any combination of channels. (This is already implemented on XINA for instrument housekeeping data.)

Files from high speed data will be available for download but not necessarily display on the XINA user interface.

## External Instrumentation (RGA)

Chronos provides a general purpose means to add additional instrumentation to the system. The first instrument will be the Residual Gas Analyzer.

The software allows display of the latest spectrum returned by the instrument and stores the spectra at selected rate (on the order of 1/hour) into the XINA database. XINA provides a custom interface for reviewing RGA spectra, including selecting a given set of masses and viewing over time.

# Architectural Strategies

The software should be written in LabVIEW. LabVIEW works seamlessly with National Instruments hardware and with Keithley hardware, allowing for easy expansion of the hardware and software.

As much as possible, the software should be multi-threaded such that an communications error or other blocking event does not stop monitoring and recording of other data.

LabVIEW also provides a built-in database format, TDMS, that is ideal for logging and archiving the relatively slow speed chamber data.

Describe any design decisions and/or strategies that affect the overall organization of the system and its higher-level structures. These strategies should provide insight into the key abstractions and mechanisms used in the system architecture. Describe the reasoning employed for each decision and/or strategy (possibly referring to previously stated design goals and principles) and how any design goals or priorities were balanced or traded-off. Such decisions might concern (but are not limited to) things like the following:

* Use of a particular type of product (programming language, database, library, etc. ...)
* Reuse of existing software components to implement various parts/features of the system
* Future plans for extending or enhancing the software
* User interface paradigms (or system input and output models)
* Hardware and/or software interface paradigms
* Error detection and recovery
* Memory management policies
* External databases and/or data storage management and persistence
* Distributed data or control over a network
* Generalized approaches to control
* Concurrency and synchronization
* Communication mechanisms
* Management of other resources

Each significant strategy employed should probably be discussed in its own subsection, or (if it is complex enough) in a separate design document (with an appropriate reference here of course). Make sure that when describing a design decision that you also discuss any other significant alternatives that were considered, and your reasons for rejecting them (as well as your reasons for accepting the alternative you finally chose). Sometimes it may be most effective to employ the "pattern format" for describing a strategy.

# System Architecture

This section should provide a high-level overview of how the functionality and responsibilities of the system were partitioned and then assigned to subsystems or components. Don't go into too much detail about the individual components themselves (there is a subsequent section for detailed component descriptions). The main purpose here is to gain a general understanding of how and why the system was decomposed, and how the individual parts work together to provide the desired functionality.

At the top-most level, describe the major responsibilities that the software must undertake and the various roles that the system (or portions of the system) must play. Describe how the system was broken down into its components/subsystems (identifying each top-level component/subsystem and the roles/responsibilities assigned to it). Describe how the higher-level components collaborate with each other in order to achieve the required results. Don't forget to provide some sort of rationale for choosing this particular decomposition of the system (perhaps discussing other proposed decompositions and why they were rejected). Feel free to make use of design patterns, either in describing parts of the architecture (in pattern format), or for referring to elements of the architecture that employ them.

If there are any diagrams, models, flowcharts, documented scenarios or use-cases of the system behavior and/or structure, they may be included here (unless you feel they are complex enough to merit being placed in the Detailed System Design section). Diagrams that describe a particular component or subsystem should be included within the particular subsection that describes that component or subsystem.

Note:

This section (and its subsections) really applies only to newly developed (or yet-to-be developed) portions of the system. If there are parts of the system that already existed before this development effort began, then you only need to describe the pre-existing parts that the new parts of the system depend upon, and only in enough detail sufficient to describe the relationships and interactions between the old parts and the new parts. Pre-existing parts that are modified or enhanced need to be described only to the extent that is necessary for the reader to gain a sufficient understanding of the nature of the changes that were made.

## Subsystem Architecture

If a particular component is one which merits a more detailed discussion than what was presented in the System Architecture section, provide that more detailed discussion in a subsection of the System Architecture section (or it may even be more appropriate to describe the component in its own design document). If necessary, describe how the component was further divided into subcomponents, and the relationships and interactions between the subcomponents (similar to what was done for top-level components in the System Architecture section).

If any subcomponents are also deemed to merit further discussion, then describe them in a separate subsection of this section (and in a similar fashion). Proceed to go into as many levels/subsections of discussion as needed in order for the reader to gain a high-level understanding of the entire system or subsystem (but remember to leave the gory details for the Detailed System Design section).

If this component is very large and/or complex, you may want to consider documenting its design in a separate document and simply including a reference to it in this section. If this is the option you choose, the design document for this component should have an organizational format that is very similar (if not identical to) this document.

# Policies and Tactics

Describe any design policies and/or tactics that do not have sweeping architectural implications (meaning they would not significantly affect the overall organization of the system and its high-level structures), but which nonetheless affect the details of the interface and/or implementation of various aspects of the system. Such decisions might concern (but are not limited to) things like the following:

* Choice of which specific product to use (compiler, interpreter, database, library, etc. ...)
* Engineering trade-offs
* Coding guidelines and conventions
* The protocol of one or more subsystems, modules, or subroutines
* The choice of a particular algorithm or programming idiom (or design pattern) to implement portions of the system's functionality
* Plans for ensuring requirements traceability
* Plans for testing the software
* Plans for maintaining the software
* Interfaces for end-users, software, hardware, and communications
* Hierarchical organization of the source code into its physical components (files and directories).
* How to build and/or generate the system's deliverables (how to compile, link, load, etc. ...)

Each particular policy or set of tactics employed should probably be discussed in its own subsection, or (if it is large or complex enough) in a separate design document (with an appropriate reference here of course). Make sure that when describing a design decision that you also discuss any other significant alternatives that were considered, and your reasons for rejecting them (as well as your reasons for accepting the alternative you finally chose). For this reason, it may frequently be convenient to use one of the more popular "pattern formats" to describe a given tactic.

For this particular section, it may become difficult to decide whether a particular policy or set of tactics should be discussed in this section, or in the System Architecture section, or in the Detailed System Design section for the appropriate component. You will have to use your own "best" judgement to decide this. There will usually be some global policies and tactics that should be discussed here, but decisions about interfaces, algorithms, and/or data structures might be more appropriately discussed in the same (sub)section as its corresponding software component in one of these other sections.

# Detailed System Design

Most components described in the System Architecture section will require a more detailed discussion. Other lower-level components and subcomponents may need to be described as well. Each subsection of this section will refer to or contain a detailed description of a system software component. The discussion provided should cover the following software component attributes:

## Classification

The kind of component, such as a subsystem, module, class, package, function, file, etc. ....

## Definition

The specific purpose and semantic meaning of the component. This may need to refer back to the requirements specification.

## Responsibilities

The primary responsibilities and/or behavior of this component. What does this component accomplish? What roles does it play? What kinds of services does it provide to its clients? For some components, this may need to refer back to the requirements specification.

## Constraints

Any relevant assumptions, limitations, or constraints for this component. This should include constraints on timing, storage, or component state, and might include rules for interacting with this component (encompassing preconditions, postconditions, invariants, other constraints on input or output values and local or global values, data formats and data access, synchronization, exceptions, etc.)

## Composition

A description of the use and meaning of the subcomponents that are a part of this component.

## Uses/Interactions

A description of this components collaborations with other components. What other components is this entity used by? What other components does this entity use (this would include any side-effects this entity might have on other parts of the system)? This concerns the method of interaction as well as the interaction itself. Object-oriented designs should include a description of any known or anticipated subclasses, superclasses, and metaclasses.

## Resources

A description of any and all resources that are managed, affected, or needed by this entity. Resources are entities external to the design such as memory, processors, printers, databases, or a software library. This should include a discussion of any possible race conditions and/or deadlock situations, and how they might be resolved.

## Processing

A description of precisely how this components goes about performing the duties necessary to fulfill its responsibilities. This should encompass a description of any algorithms used; changes of state; relevant time or space complexity; concurrency; methods of creation, initialization, and cleanup; and handling of exceptional conditions.

## Interface/Exports

The set of services (resources, data, types, constants, subroutines, and exceptions) that are provided by this component. The precise definition or declaration of each such element should be present, along with comments or annotations describing the meanings of values, parameters, etc. .... For each service element described, include (or provide a reference) in its discussion a description of its important software component attributes (Classification, Definition, Responsibilities, Constraints, Composition, Uses, Resources, Processing, and Interface).

Much of the information that appears in this section is not necessarily expected to be kept separate from the source code. In fact, much of the information can be gleaned from the source itself (especially if it is adequately commented). This section should not copy or reproduce information that can be easily obtained from reading the source code (this would be an unwanted and unnecessary duplication of effort and would be very difficult to keep up-to-date). It is recommended that most of this information be contained in the source (with appropriate comments for each component, subsystem, module, and subroutine). Hence, it is expected that this section will largely consist of references to or excerpts of annotated diagrams and source code. Any referenced diagrams or source code excerpts should be provided at any design reviews.

## Detailed Subsystem Design

Provide a detailed description of this software component (or a reference to such a description). Complex diagrams showing the details of component structure, behavior, or information/control flow may be included in the subsection devoted to that particular component (although, unless they are very large or complex, some of these diagrams might be more appropriately included in the System Architecture section. The description should cover any applicable software component attributes (some of which may be adequately described solely by a source code declaration or excerpt).

# Glossary

An ordered list of defined terms and concepts used throughout the document.

# Bibliography

A list of referenced and/or related publications.

Brad Appleton <brad@bradapp.net>

http://www.bradapp.net