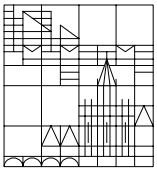

University of Constance
Department of Computer Science

AOCS FRAMEWORK - FRAMEWORK INSTANTIATION

Abstract

This document was written as part of the study "Design and Prototyping of a Software Framework for the AOCS" done under contract Estec/13776/99/NL/MV for ESA-Estec. The purpose of the study is the development of a software framework for the Attitude and Orbit Control Subsystem (AOCS) of a satellite. The framework was tested by using it to generate the software for a prototype AOCS. This document defines the test environment that was used to test the AOCS prototype and the results of the tests.

Written By: A. Pasetti

Date: 30 April 2002

Issue: 1.0

Reference: SWE/00/AOCS/002

TABLE OF CONTENTS

1 REFERENCES

RD1 E. Gamma *et al.* (1995), *Design Patterns – Elements of Reusable Object Oriented Software*, Reading, Massachusetts: Addison-Wesley

RD2 A. Pasetti (2000), *AOCS Framework – Concept Level Description*, AOCS Framework Document ref. SWE/99/AOCS/004

RD3 Deleted

RD4 A. Pasetti (2000), *Methodological Issues*, AOCS Framework Document ref. SWE/99/AOCS/018

RD5 A. Pasetti (2000), *Inter-Component Communication Framelet* AOCS Framework Document ref. SWE/99/AOCS/005

RD6 A. Pasetti (2000), *Object Monitoring Framelet*, AOCS Framework Document ref. SWE/99/AOCS/008

RD7 A. Pasetti (2000), *Data Processing Framelet*, AOCS Framework Document ref. SWE/99/AOCS/006

RD8 A. Pasetti (2000), *AOCS Unit Management Framelet*, AOCS Framework Document ref. SWE/99/AOCS/017

RD9 A. Pasetti (2000), *Reconfiguration Management Framelet* AOCS Framework Document ref. SWE/99/AOCS/015

RD10 A. Pasetti (2000), *Operational Mode Management Framelet* AOCS Framework Document ref. SWE/99/AOCS/009

RD11 T. Brown, A. Pasetti (2000), *Manoeuvre Management Framelet*, AOCS Framework Document ref. SWE/99/AOCS/012

RD12 A. Pasetti (2000), *Failure Detection Management Framelet*, AOCS Framework Document ref. SWE/99/AOCS/010

RD13 A. Pasetti (2000), *System Management Framelet*, AOCS Framework Document ref. SWE/99/AOCS/021

RD14 A. Pasetti (2000), *Failure Recovery Management Framelet*, AOCS Framework Document ref. SWE/99/AOCS/011

RD15 A. Pasetti (2000), *Telemetry Management Framelet*, AOCS Framework Document ref. SWE/99/AOCS/003

RD16 A. Pasetti (2000), *Telecommand Management Framelet*, AOCS Framework Document ref. SWE/99/AOCS/014

RD17 A. Pasetti, T. Brown (2000), *Controller Management Framelet*, AOCS Framework Document ref. SWE/99/AOCS/016

RD18 A. Pasetti (2000), *AOCS Prototype Definition*, AOCS Framework Document ref. SWE/99/AOCS/020

RD19 *MACS Bus Handbook*

RD20 A. Pasetti (2000), *AOCS Prototype Definition*, AOCS Framework Document ref. SWE/99/AOCS/020

RD21 A. Pasetti (2000), *AOCS Test Report*, AOCS Framework Document ref. SWE/00/AOCS/001

2 ACRONYMS

AAD	Attitude Anomaly Detection
AOCS	Attitude and Orbit Control Subsystem
AST	Autonomous Star Tracker
CSS	Coarse Sun Sensor
ES	Earth Sensor
FDIR	Failure Detection, Isolation and Recovery
FPM	Fine Pointing Mode
FSS	Fine Sun Sensor
GYR	Gyroscope
KF	Kalman Filter
IAM	Initial Acquisition Mode
MIMO	Multi-Input-Multi-Output
NM	Normal Mode
NTT	Non-Time-Tagged
OBDH	On-Board Data Handling system (aka as OBDS)
OCM	Orbit Control Mode
OO	Object-Oriented
PD	Proportional-Derivative controller
PI	Proportional-Integral controller
PID	Proportional-Integral-Derivative controller
RRM	Rate Reaction Mode
RTOS	Real-Time Operating System
RW	Reaction Wheel
SAS	Sun Attitude Sensor
SBM	Stand-By Mode
SISO	Single-Input-Sinle-Output
SPS	Sun Presence Sensor
STR	Star Tracker
SLM	Slewing Mode
SM	Safe Mode
TC	Telecommand
THU	Thruster
TM	Telemetry
TT	Time-Tagged

3 INTRODUCTION

This document describes the instantiation of the *AOCS prototype framework* as it was done to generate the *AOCS prototype application*. The AOCS prototype is a simplified AOCS application which is implemented using the constructs offered by the AOCS prototype framework. The AOCS prototype serves as a test bed for the AOCS prototype framework.

The instantiation policy described here was specifically targeted at the instantiation of the AOCS prototype. No claim is made about its generality and optimality.

3.1 Context

The context for the definition of the prototype framework is the architectural design of the full framework. This is presented at system concept definition level in RD2 and at framelet concept and at framelet architectural definition level in RD5 to RD17. The prototype framework is defined in RD18. The AOCS prototype whose instantiation is described here is presented in RD20.

3.2 General Approach

The framework instantiation is based on three items:

- a set of abstract factories (in the sense of RD1) to construct and provide the application components
- an instantiation module to configure the components
- a configuration file to define the configuration parameters.

The next three sections describe each of the three above items in greater detail.

3.3 Tuning the Instantiation Process

In general, the characteristics of the instantiated AOCS applications can be changed in the following manners:

- by subclassing the framework factories to create different kinds of framework components,
- by changing module `InstantiateAocs` to change the configuration of the framework components generated by the factories,
- by changing the `AocsConfiguration.h` to change the parameters in the components generated by the factories.

4 THE ABSTRACT FACTORIES

For each framelet, or group of related framelets, an abstract factory is provided. The table provides a list of the abstract factories together with the components that they can supply to their clients:

Abstract Factory Class	Generated Components
`AocsUnitFactory`	AOCS units, unit triggers, unit trigger mode managers, unit reconfiguration managers
`ControllerFactory`	Controllers, controller manager, controller mode manager
`FailureDetectionFactory`	Failure detection manager, failure detection mode manager, lists of monitoring check objects, lists of consistency checkable objects
`FailureRecoveryFactory`	Failure recovery manager, failure recovery actions, failure recovery strategies
`ManoeuvreFactory`	Manoeuvre manager, manoeuvre mode manager, manoeuvre objects
`ModeFactory`	Mode change actions
`MonitoringFactory`	Change objects
`SystemFactory`	System manager, mission mode manager, AOCS clock, data pools, event repositories
`TelemetryFactory`	Telemetry manager, telemetry mode manager, telemerable lists

The factories are designed to be completely independent of each other and could in principle be used in isolation of each other.

Factories expose two kinds of methods: *make methods* and *get methods*.

The make methods are used to create the components (without necessarily returning a reference to the caller). Wherever possible, each make method takes as parameters all the information that is necessary to fully configure the components that it creates. The goal is to

ensure that the components that are created with a call to a make method should return true to a call to their `isConfigured` method.

The get methods are used to retrieved components already created by make methods. The get methods should only be called after the make methods have been called.

The memory for components that are created in the factories is allocated dynamically from the heap.

As a rule, components, when they are exported by the factories, are cast to the basic framework types. Thus, for instance, the FSS proxy object may be created within the factory as an instance of type `FssPrototype` (or some other implementation of the base abstract class `AocsUnit`) but it is exported – through a getter method – as a component of type `AocsUnit`. In this way, the AOCS application is forced to deal with the framework interfaces which effectively protects it from the framework implementation details.

The factories do some basic checking to verify the correctness of the configuration parameters that they receive from their clients. If inconsistencies are found, they are flagged as configuration errors.

4.1 THE FRAMEWORK INSTANTIATION MODULE

The framework instantiation module is called `InstantiateAocs`. It contains a suite of procedures that call the factory methods – with appropriate parameters – to create the application components. The application main program should call its procedure `instantiateAocs` to start the framework instantiation process.

Module `InstantiateAocs` receives the framework factories through an instance of the following type:

```
struct FrameworkFactories {
     SystemFactory*              systemFactory;
     TelemetryFactory*           telemetryFactory;
     TelecommandFactory*         telecommandFactory;
     AocsUnitFactory*            aocsUnitFactory;
     FailureRecoveryFactory*     failureRecoveryFactory;
     FailureDetectionFactory*    failureDetectionFactory;
     MonitoringFactory*          monitoringFactory;
     ModeFactory*                modeFactory;
     ManoeuvreFactory*           manoeuvreFactory;
     ControllerFactory*          controllerFactory;
};
```

Procedure `instantiateAocs` proceeds in four steps:

- It creates all the framework components
- It loads the following object lists:
  - the telemetry lists (defining the telemetry layout)
  - the failure detection lists (defining the failure detection checks)
  - the unit trigger list (defining the unit triggering patterns)
  - the controller list (defining the closed-loop controllers)
- It calls the isSystemConfigured of the system manager to verify that all application components are correctly configured
- It checks that no configuration errors have occurred during the instantiation process

5 THE AOCS CONFIGURATION FILE

The configuration of the components created by the AOCS factories is defined by the parameters passed by factory clients to factory method calls and by constants internal to the factories. The latter are stored as `#define` constants in the header file `AocsConfiguration.h`.