

**Software Architecture Document (SAD)
for the
Next ESA SAR Toolbox (NEST)**

Contract Number: 20809/07/I-LG

Prepared by:

Array Systems Computing Inc.
1120 Finch Ave. West, 7th Floor
Toronto, Ontario, Canada M3J 3H7

Prepared for:

The European Space Agency (ESA)

Proprietary/Copyright Information

This material contains proprietary and/or copyright information of Array Systems Computing Inc. (ARRAY) and may not be copied, used or disclosed without permission of Array Systems Computing Inc.

© Copyright Array Systems Computing Inc. (2010)

COMMERCIAL IN CONFIDENCE

**Software Architecture Document (SAD)
for the
Next ESA SAR Toolbox (NEST)**

Contract Number: 20809/07/I-LG

Author:

Luis Veci

Date

Project Manager:

Rajesh Jha

Date

Quality Assurance:

Iris Buchan

Date

COMMERCIAL IN CONFIDENCE

Software Architecture Document (SAD) for the Next ESA SAR Toolbox (NEST)

Contract Number: 20809/07/I-LG

Revision History

Version	Date	Revised By	Description
1.0	February 22, 2008	Luis Veci	Initial release.
2.0	January 27, 2010	Luis Veci	Updated for 3C Release Table 1 – Opeartors list updated Tools used updated Supported formats updated
3.0	December 10, 2012	Luis Veci	Update for version 5.0 of the software

Array Systems Computing Inc.

Commercial in Confidence

Printed on December 10, 2012

Page ii

Use, duplication, or disclosure of this document or any information contained herein is subject to the restriction on the title page of this document.

Table of Contents

1	SCOPE	1
1.1	Identification	1
1.2	System Overview	1
1.3	Document Overview.....	1
2	REFERENCED DOCUMENTS.....	3
2.1	Customer Documents	3
2.2	Array Documents	3
2.3	Other Documents.....	3
3	DESIGN METHODOLOGY.....	4
3.1	Object-Oriented Design.....	4
3.2	Test-Driven Development	4
4	DEVELOPMENT ENVIRONMENT	6
4.1	Java.....	6
4.2	Maven Build Tool	7
4.3	Integrated Development Environment.....	7
4.4	Version Control	7
4.5	Continuous Integration	7
4.6	Coding Standards	8
4.7	Detailed Design Documentation.....	8
4.8	Bug tracking	8
5	SYSTEM ARCHITECTURE DESIGN.....	9
5.1	Portable Source Code and Hardware Independence	9
5.2	Dependencies.....	9
5.3	Delivered Software	Error! Bookmark not defined.
6	SOFTWARE ARCHITECTURE DESIGN	10
6.1	BEAM Architecture.....	10
6.2	NEST Architecture	11
6.3	Display and Analysis Tool	14
6.3.1	Views.....	15
6.3.1.1	Image View	15
6.3.1.2	Co-Registered View	Error! Bookmark not defined.
6.3.1.3	Quicklook View.....	Error! Bookmark not defined.
6.3.1.4	3D View	15
6.3.1.5	Products ToolView	16
6.3.1.6	PixelInfo ToolView	16
6.3.1.7	Metadata View.....	16
6.3.1.8	Bitmask Overlay ToolView.....	16
6.3.1.9	Navigation & Zoom ToolView.....	16
6.3.1.10	Statistics ToolView	17
6.3.1.11	WorldMap ToolView	17
6.3.2	Regions of Interest.....	17

Array Systems Computing Inc.

Commercial in Confidence

Printed on December 10, 2012

Page iii

Use, duplication, or disclosure of this document or any information contained herein is subject to the restriction on the title page of this document.

6.3.3	Projects and Workspaces	18
6.3.4	Vector Data Overlays	18
6.4	Data Model and IO	18
6.4.1	Data Model	18
6.4.1.1	Product.....	19
6.4.1.2	Bands	19
6.4.1.3	Tie-Point Grids	20
6.4.1.4	Geo-coding	20
6.4.1.5	Metadata	20
6.4.1.6	Flag-coding.....	20
6.4.1.7	Virtual Bands and Band Arithmetic	20
6.4.1.8	Product Data	20
6.4.2	Readers	20
6.4.2.1	Data Product Readers	21
6.4.2.2	Common EO File Formats.....	23
6.4.2.3	Digital Elevation Maps	23
6.4.3	Writers	23
6.4.3.1	Common EO File Formats.....	23
6.4.3.2	COTS Image File Formats.....	23
6.4.3.3	Generic Binary (BIL, BSQ, BIP).....	23
6.5	Graph Processing Tool	24
6.5.1	Graph Builder User Interface	25
6.5.2	Tiling	25
6.5.3	Operators	26
7	ACRONYMS AND ABBREVIATIONS	28

List of Figures

Figure 1: Test-Driven Development Flow Chart.....	5
Figure 2: The Java 1.6 Platform	6
Figure 3: NEST-BEAM File Structure	Error! Bookmark not defined.
Figure 4: Continuous Integration Development	Error! Bookmark not defined.
Figure 5: BEAM-NEST Module Reuse	Error! Bookmark not defined.
Figure 6: BEAM-NEST Core Framework.....	11
Figure 7: NEST Flow of Data	12
Figure 8: NEST's Primary Applications	13
Figure 9: Sample Module Manifest.....	14
Figure 10: Views and ToolViews of the DAT	Error! Bookmark not defined.
Figure 11: Product Data Model	19
Figure 12: ENVISAT Format Readers	21
Figure 13: CEOS Readers	22
Figure 14: TerraSAR-X and Radarsat-2 Readers	22
Figure 15: HDF 5 CEOS Readers	22
Figure 16: A Directed Acyclic Graph	24
Figure 17: The Graph Builder	25

List of Tables

Table 1: NEST GPF Operators.....	26
Table 2: Acronyms and Abbreviations.....	28

1 SCOPE

1.1 Identification

- [1] Array Systems Computing Inc. (Array) submits this Software Architecture Document (SAD), identified as ARR-NEST-RS07-016, to the European Space Agency (ESA). The SAD is used to describe the high-level design of the Next ESA SAR Toolbox (NEST), the identification of its Configuration Items (CIs), and its operational and support environments.
- [2] The architecture design is derived from the requirements as specified in the Technical Specification (TS) document.
- [3] The SAD Version 1.0 was presented at the Preliminary Design Review (PDR) meeting.
- [4] The SAD Version 2.0 is updated to correspond with the NEST 3C Release. The high level architecture has remained the same as described in Version 1.0 and the changes in this version are minor.
- [5] The SAD Version 3.0 is update to correspond with the NEST 5.0 Release.

1.2 System Overview

- [1] The European Space Agency (ESA) has been delivering Synthetic Aperture Radar (SAR) data to users since 1991, and a large archive of SAR images has been created. Users of ESA SAR data need a set of tools for reading the custom data formats in which these data are provided. Although ESA SAR Toolbox development started in the late 1990's, the software is currently approaching the end of its development cycle. The current closed source software cannot activate the user-base to develop new features and tools for the toolbox. ESA requires a fully open source toolbox that has the potential to grow in the future through modules implemented by the user community.
- [2] In October 2007, ESA awarded Array a contract to develop the Next ESA SAR Toolbox (NEST). NEST is an open source ([GNU GPL](#)) toolbox for reading, post-processing, analysing and visualising the large archive of data (from Level 1) from ESA SAR missions including ERS-1 & 2, ENVISAT and in the future Sentinel-1. In addition, NEST supports handling of products from third party missions including JERS-1, ALOS PALSAR, TerraSAR-X, Radarsat-2 and Cosmo-Skymed. NEST has been built using the [BEAM](#) Earth Observation Toolbox and Development Platform.
- [3] NEST includes the functionality of the ESA SAR Toolbox, BEST (Basic Envisat SAR Toolbox), with additional, newly developed functionality. NEST strives to be as compatible as possible with the architecture of the BEAM toolbox (Basic ERS & Envisat (A)ATSR and MERIS Toolbox) and therefore has a similar, easily portable, modular design. Adopting an architecture closely resembling the BEAM architecture benefits NEST development since many libraries and functions can be directly borrowed from BEAM and costly redevelopment work can be avoided.
- [4] The NEST toolbox will be able to help in preparing SAR data products for use in popular COTS software, but will not be intended to replace existing COTS software nor function as a SAR processor.

1.3 Document Overview

- [1] This document is structured into 6 sections:
 - Section 1 provides an overview of the document.
 - Section 2 lists all documents referenced within this document.

Array Systems Computing Inc.

Commercial in Confidence

Printed on December 10, 2012

Page 1

Use, duplication, or disclosure of this document or any information contained herein is subject to the restriction on the title page of this document.

-
- Section 3 describes the NEST Design Mythology
 - Section 4 describes the Development Environment.
 - Section 5 describes the NEST System Architecture
 - Section 6 describes the NEST Software Architecture.
 - Section 7 defines acronyms and abbreviations used in this document.

2 REFERENCED DOCUMENTS

2.1 Customer Documents

- [1] **Title:** NEST Statement of Work
Reference: ENVI-DTEX-EOPG-SW-06-0005
Date: January 2007

2.2 Array Documents

- [1] **Title:** NEST Technical Specification Worksheet
DCN: ARR-NEST-RS07-017
Date: Latest issue
- [2] **Title:** Standards for Design
DCN: ASC00511
Date: Latest issue

2.3 Other Documents

- [1] **Title:** Design Patterns
Author: Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides
Publisher: Addison Wesley
ISBN: 0-201-63361-2
- [2] **Title:** Code Conventions for the Java Programming Language
Date: April 20, 1999
<http://java.sun.com/docs/codeconv/>
- [3] **Title:** BEAM Architecture Document
Author: Norman Fomferra
Date: May 2006

3 DESIGN METHODOLOGY

- [1] Software development is a complicated process that requires careful planning and execution to meet the goals.
- [2] Design for this project will be based on Array's "Standards for Design". This document provides guidance for methodology and tool selection, system architecture design, software design, design reuse, design documentation, design verification and design quality.
- [3] Array maintains System Development Folders (SDF) throughout the life of the project. The SDFs contain the history of design decisions, the rationale behind those decisions and other related information. SDFs are made up of a collection of logs, emails and problem reports for changes made to the software.

3.1 Object-Oriented Design

- [1] The software architecture for NEST will use an object oriented (OO) approach to the design. Object Oriented Programming is a paradigm to programming that is critical for any large scale software. It allows for greater flexibility, maintainability, modularity and reuse.
- [2] The first stage in an OO approach is to analyse the problem domain and identify the things of importance. These things in turn will be converted to software Classes. An Object becomes a particular instance of a Class. This indicates that there is a close relationship between real world things and software objects.
- [3] Through hierarchical decomposition, high level structures can be defined that are then refined into more detailed structures. This dramatically reduces the complexity of the project into workable smaller parts while at the same time having a systems view of the overall project.
- [4] The OO approach also makes use of other fundamental concepts such as Inheritance where subclasses can inherit attributes and behaviours of its parent class, Abstraction for simplifying complexity, Encapsulation for information hiding, and Polymorphism for handling derived classes in similar ways.
- [5] As with any well-structured object oriented design, our design will be rooted in the widely accepted solutions to common problems in software design known as Design Patterns and popularized by Gamma et al.

3.2 Test-Driven Development

- [1] Test-Driven Development is a core part of the Agile methodology with the purpose of designing the code based on the tests.
- [2] It is a style of development where test cases for code units are first written, and then only the code necessary to pass the test is implemented. The code is then refactored and tested again. Continuous feedback is given on stable working code, and changes can be made without inadvertently breaking dependent units. Furthermore, only code required by the test will be written. This will ensure that the test coverage is complete and that the code only does what is needed.
- [3] The following steps comprise test-driven development:
 - 1. Prepare Basic Plan and Model
 - 2. Write test first

Array Systems Computing Inc.

Commercial in Confidence

Printed on December 10, 2012

Page 4

Use, duplication, or disclosure of this document or any information contained herein is subject to the restriction on the title page of this document.

3. Develop design with test
4. Write the code
5. Update the test and go back to step 2 until complete.

[4] Figure 1 illustrates the flow of test-driven development.

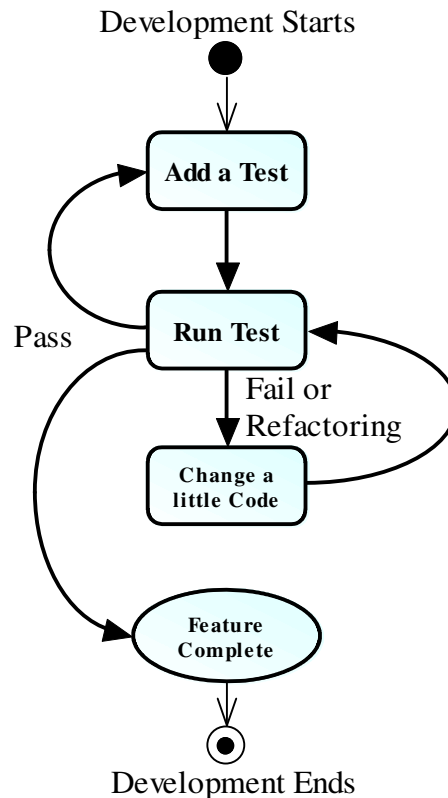


Figure 1: Test-Driven Development Flow Chart

- [5] Since the code is based on the tests, testability is guaranteed. It helps ensure exhaustive test coverage, and it keeps the code and tests in sync.
- [6] NEST uses the standard JUnit Framework for unit testing. JUnit provides a simple and easy to use framework for automating the project's tests.

4 DEVELOPMENT ENVIRONMENT

- [1] Development of NEST takes place on Linux and Windows 7 operating systems running on COTS hardware. All development tools are freely available such that a similar development environment could be reproduced by anyone following links from the NEST website.

4.1 Java

- [1] NEST is written exclusively in Java. Java is an OO programming language capable of allowing the same program to be executed on multiple operating systems.
- [2] Oracle Java 1.7 JDK Platform shown in Figure 2 has many Application Programming Interfaces (APIs) and tools ready for use which will minimize development cost and help to produce robust, cross-platform code. With over 3000 classes, the platform is quite large, but will provide NEST with many useful libraries.

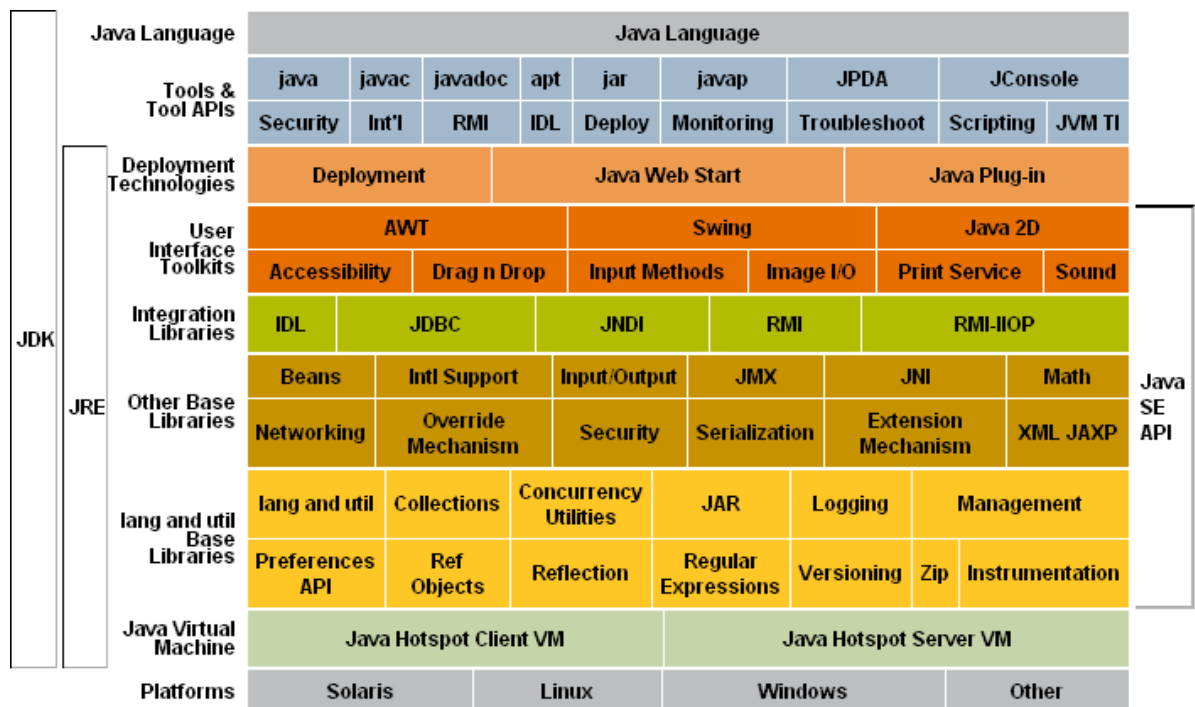


Figure 2: The Java 1.7 Platform

- [3] In addition to the tools provided by the Java JDK, NEST relies heavily on two extensions to the JDK namely the Java Advanced Imaging (JAI) and JAI Imaging I/O extensions.
- [4] JAI is a powerful imaging package with the capability of rendering very large images and applying an extensive collection of common image operators.

- [5] The Imaging IO package provides a foundation for creating image codec to load and save data formats in a standard way.

4.2 Maven Build Tool

- [1] NEST use the open source Apache Maven build tool version 3.0 to set up the project structure and build environment.
- [2] Maven is a software project management and comprehension tool. Based on the concept of a project object model, Maven can manage a project's build, reporting and documentation from a central project object model file. Maven uses a set of plug-ins that are shared by all projects using Maven, providing a uniform build system.
- [3] Maven also provides patterns for the build infrastructure to compile, test, document, and deploy.
- [4] <http://maven.apache.org/download.html>

4.3 Integrated Development Environment

- [1] The community edition of IntelliJ IDEA is used as an Integrated Development Environments (IDE) for coding and debugging. The IDEs are designed to maximise developer productivity by providing the necessary tools and graphical environment for all areas of development including coding, debugging and testing.
- [2] <http://www.jetbrains.com/idea/>

4.4 Performance Profiling

- [1] For profiling performance bottlenecks, finding memory leaks and resolve threading issues, the NEST project uses JProfiler from EJ-Technologies.
- [2] <http://www.ej-technologies.com/products/jprofiler/overview.html>

4.5 Version Control

- [1] For concurrent version control of the code repository git is used. Git is a distributed revision control and source code management (SCM) system with an emphasis on speed. Git was initially designed and developed by Linus Torvalds for Linux kernel development; it has since been adopted by many other projects. Every Git working directory is a full-fledged repository with complete history and full revision tracking capabilities, not dependent on network access or a central server.
- [2] The latest NEST source code is available on github at <https://github.com/lveci/nest>

4.6 Continuous Integration

- [1] Continuous Integration helps to provide a functioning product at all times.
- [2] The code repository is continuously monitored for code changes. When triggered, the latest code from the repository will be checked out automatically by scripts and built on each of the target platforms.
- [3] Any build problems or failed tests will immediately trigger an email to notify the developer responsible for that code.

-
- [4] Upon a successful build, unit tests, integration tests, coding standards checks and metrics will be executed automatically. In this way, problems are identified quickly and developers can have confidence in the code as they continue to refactor and add new features.

4.7 Coding Standards

- [1] NEST follows the Coding Conventions of the Java Programming Language from Sun Microsystems. The guidelines are recommendations to the programmers, and should be followed except when rigorous adherence to them undermines the achievement of readability, maintainability, or portability, and can be justified.
- [2] Java coding standards and conventions are routinely monitored using IntelliJ IDEA, a highly configurable development tool to help programmers write Java code that adheres to a coding standard.

4.8 Detailed Design Documentation

- [1] The JavaDoc tool is used for creating documentation directly from the source code. From formatted comments within the code, JavaDoc can automatically generate associated documentation in HTML and PDF formats.
- [2] All source code must be properly documented. There will be coding standard checks to verify that no class has been left without proper JavaDoc headers. In places of the code where a JavaDoc header has not been completed properly or is missing parameter(s), IntelliJ highlights the comments.

4.9 Bug tracking

- [1] Bug tracking is managed internally using the Trac bug tracking and reporting system and will be accessible to ESA at the website in order to keep ESA informed about the state of each release and to allow ESA users to submit their own bugs or change requests.

5 SYSTEM ARCHITECTURE DESIGN

- [1] The NEST is a software-only solution intended to be as hardware independent as possible.

5.1 Portable Source Code and Hardware Independence

- [1] Hardware independence is achieved by the use of Java. Java code is implicitly portable. Java source code is compiled into a universal format called byte-code. This compiled Java *byte-code* can be executed by a Java runtime interpreter on any platform. The runtime system is a Virtual Machine that performs all the normal activities of a real processor, including executing the instruction set and managing memory.
- [2] The Java interpretation is a strictly defined open specification that can be implemented by anyone who wants to produce a Java-compliant virtual machine for a particular platform. In this way, the same Java application byte-code can run on any platform that provides a Java runtime environment.
- [3] The portability of Java software will allow NEST to be written once, compiled once, and then be able to run on any hardware and operating system supporting the Java virtual machine. As such, NEST will be able to run on Microsoft Windows, Mac OS, Linux, Solaris and other flavours of UNIX.

5.2 Dependencies

- [1] The NEST software will be dependent on some 3rd party libraries, including BEAM, Ceres, JAI, JOGL, HDF, and NetCDF. All 3rd party libraries are included with a NEST distribution including it's own Java Runtime Environment for each platform.
- [1] on a distribution disk will also be available to download from the NEST website.

6

SOFTWARE ARCHITECTURE DESIGN

6.1

BEAM Architecture

- [1] The NEST architecture is built upon the proven and extensible architecture of BEAM. BEAM features an Application Programming Interface (API) that has been designed and developed from the beginning to be used by 3rd parties to extend BEAM or re-use it in their own applications.
- [2] BEAM is an application suite which facilitates the utilization, viewing and processing of MERIS, AATSR and ASAR data products of the ESA ENVISAT environmental satellite. It provides a multitude of tools and functions to visualize, analyze, manipulate and modify earth observation data.
- [3] Originally intended to support scientists working with ENVISAT AATSR and MERIS as well as ERS ATSR data, BEAM has evolved to be used on a number of ESA projects for Earth Observation (EO) data analysis and is supporting additional instruments including MODIS, AVHRR, ALOS PRISM and AVNIR2, as well as generic data formats such as netCDF.
- [4] Beside general functions such as image display or map projections, sensor-specific tools such as retrieval of water constituents from MERIS data or sea surface temperature calculation from AATSR data, are part of the toolbox. BEAM also supports some ENVISAT ASAR products with the basic image visualization and image manipulation functions.
- [5] Although BEAM has been intended specifically for optical data, the BEAM architecture has been designed and developed to facilitate the extension and re-use of its code base such that it is feasible to use BEAM's core framework as the building blocks to develop a Synthetic Aperture Radar (SAR) toolbox.
- [6] NEST and BEAM share a common core that enables the exchange of modules between the two toolboxes. This common core is maintained cooperatively by both Array Systems Computing Inc and Brockmann Consult, the developers of BEAM.
- [7] The majority of the NEST functionality is encapsulated in BEAM plug-in modules. As such, some modules will be interchangeable between the two systems.
- [8] The common BEAM-NEST Core architecture consists of various plug-in modules which are all independently developed, modified and versioned. The core framework shown in Figure 3 is made up of the *beam-core*, *beam-gpf*, *beam-ui* and *beam-visat* modules.

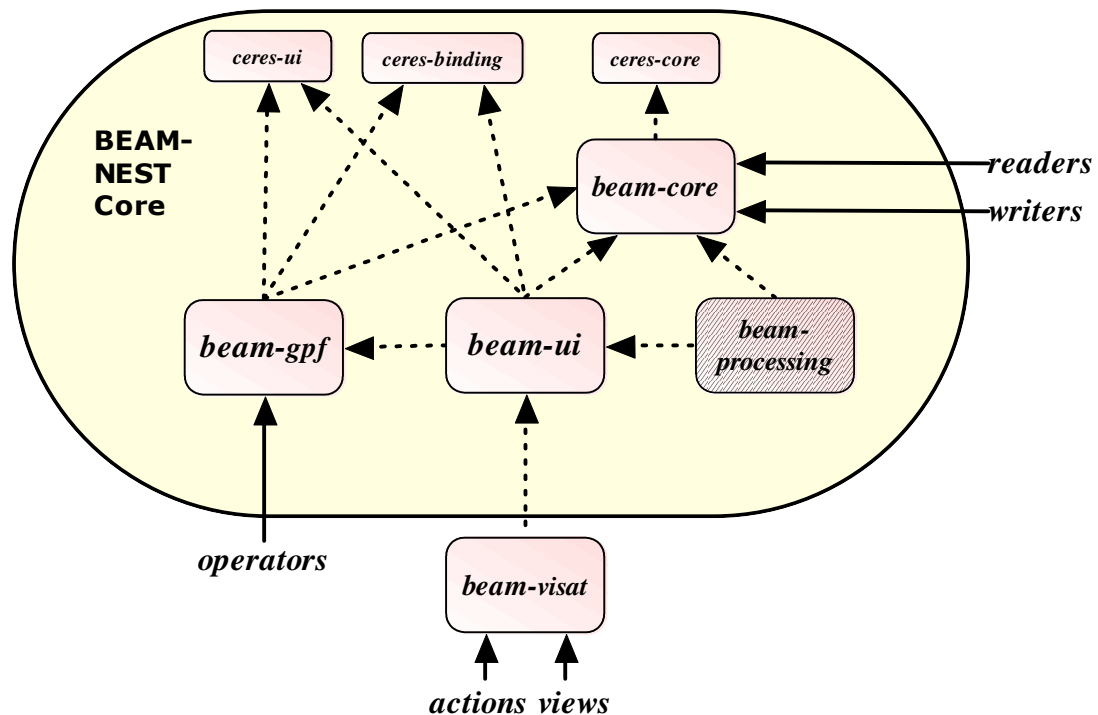


Figure 3: BEAM-NEST Core Framework

- [9] The core modules make use of *ceres-core*, *ceres-ui* and *ceres-binding*. These modules package utility classes for module registration, versioning, application building and swing user interface helper functions.
- [10] The *beam-core* contains most of the managers and data model including the data IO for the product readers and format writers. The *beam-gpf* is the Graph Processing Framework (GPF), which implements the new processing framework introduced in version 4 of the software. Processing tasks are implemented as Operators. The GPF provides a way to execute a chain of sequential processing steps on an image. The *beam-processing* is the old version 3 processing framework, which is not used by NEST directly but may still have dependencies for some BEAM operators. The *beam-ui* provides the user interface framework for creating applications, windows and dialogs. *Beam-visat* is the primary application and user interface to the tools in BEAM. VISAT supports extensions for views for displaying data and actions to add menu items and toolbar buttons to trigger user initiated events.

6.2 NEST Architecture

- [1] NEST consists of a collection of processing modules and data product readers and writers.
- [2] All modules are centered on the Generic Product Model (GPM) of the BEAM-NEST Core. The GPM is a common, unified data model designed such that all data readers convert data into this data model and all analysis and processing tools use this data model exclusively.

Array Systems Computing Inc.

Commercial in Confidence

Printed on December 10, 2012

Page 11

Use, duplication, or disclosure of this document or any information contained herein is subject to the restriction on the title page of this document.

- [3] NEST's primary goals are to read in ESA and third part SAR data products, provide tools for calibration, orthorectification, co-registration, interferometry, and data manipulation, and then convert the data into common file formats used by other 3rd party SAR software.
- [4] Figure 4 shows the flow of data through the components of NEST.

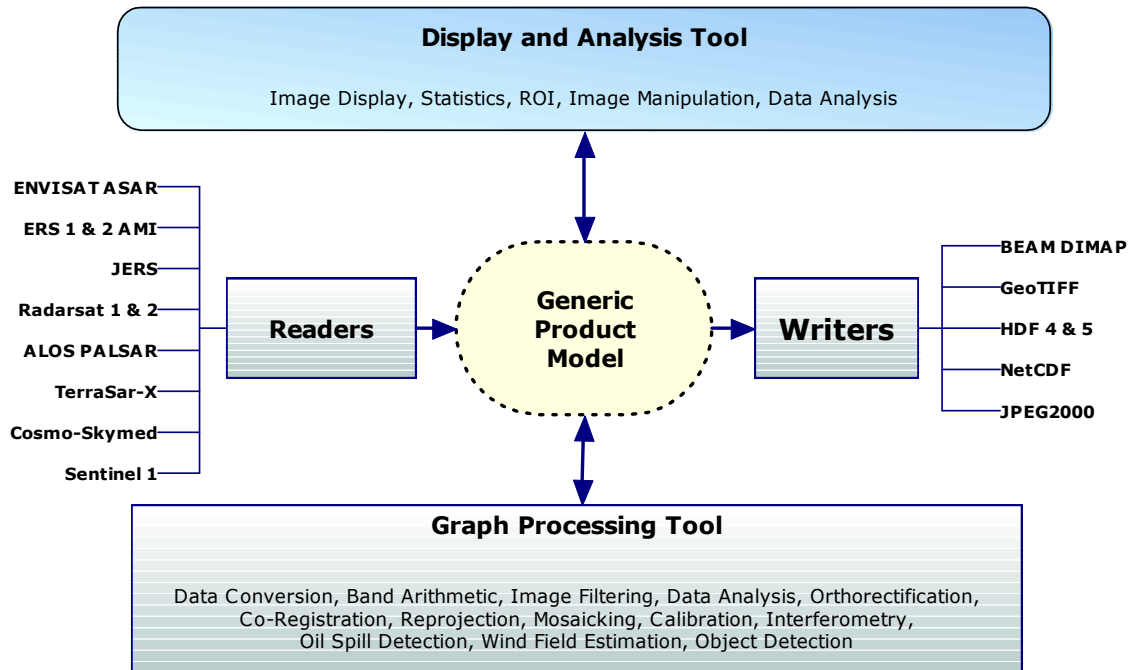


Figure 4: NEST Flow of Data

- [5] This collection of modules are driven by two primary applications: the Display and Analysis Tool (DAT) and the Graph Processing Tool (GPT) as shown in Figure 5. The DAT integrates the various tools within a rich graphical user interface, while the GPT allowa the tools to be executed independently from the command line, scripts or callable from other COTS software.

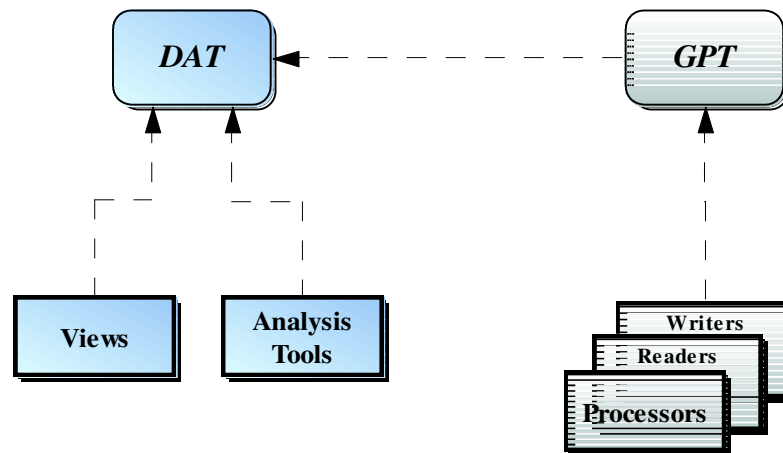


Figure 5: NEST's Primary Applications

- [6] A major goal of the architecture is to keep NEST's functionality as modular as possible. All plug-in modules are based on the Ceres reusable module framework and rely on it for module management, versioning, and updating.
- [7] Modules are designed for strong cohesion within a module, and weak coupling between modules. Through modularity, great flexibility and expandability can be achieved. New requirements needed by the user community could potentially be implemented easily by adding or replacing modules in isolation without affecting other modules or requiring changes to the core framework code. With a plug-in architecture, there is no need to re-write the software to add new capabilities. This allows the end user to create his/her own modules by following the interface descriptions detailed in the user manual.
- [8] Following on the BEAM framework, in NEST there are plug-in modules for most functions including data IO, data layer rendering, map projections, processing and processor user interfaces. All modules are able to append their own menu items and add their own dialog boxes to the system to provide an integrated user interface for the module.
- [1] NEST follows the plug-in system used by the beam-core. Plug-in modules are inserted into the system via module manifest files which hold Meta information about the module.
- [2] Module manifests help the core system to register the module and store information such as version, description and license as well as where it should fit in the system and how it should be called.
- [3] A manifest that defines an extension for an action, for example, may be realized as a menu item within the DAT that instantiates the module when clicked on by the user.
- [4] Figure 6 shows a sample module manifest for a NEST module.

```
<?xml version="1.0"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <parent>
    <artifactId>nest</artifactId>
    <groupId>org.esa.nest</groupId>
    <version>5.0</version>
  </parent>

  <groupId>org.esa.nest</groupId>
  <artifactId>nest-reader-dem</artifactId>
  <name>NEST DEM Reader</name>
  <version>5.0</version>

  <packaging>jar</packaging>

  <dependencies>
    <dependency>
      <groupId>org.esa.beam</groupId>
      <artifactId>beam-core</artifactId>
    </dependency>
    <dependency>
      <groupId>org.esa.nest</groupId>
      <artifactId>nest-core</artifactId>
    </dependency>
    <dependency>
      <groupId>org.esa.beam</groupId>
      <artifactId>beam-geotiff</artifactId>
    </dependency>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <scope>compile</scope>
    </dependency>
  </dependencies>
</project>
```

Figure 6: Sample Module Manifest

6.3

Display and Analysis Tool

- [1] The Display and Analysis Tool (DAT) is the primary application which NEST users use to visualize and analyze the data and access the data manipulation tools available in the toolbox. The user is able to see multiple images within child windows and interact with the data analysis tools with the mouse.
- [2] The DAT is based on the VISAT application from BEAM and inherits most functionality from BEAM VISAT including views and actions. If at some time during the project VISAT is incompatibly changed, then the DAT will use the last compatible version of VISAT as a base.

Array Systems Computing Inc.

Commercial in Confidence

Printed on December 10, 2012

Page 14

Use, duplication, or disclosure of this document or any information contained herein is subject to the restriction on the title page of this document.

- [3] The DAT window contains menus and toolbars to rapidly allow the user to access commonly-used operations using both the mouse and the keyboard. Toolbars will have descriptively drawn icons to reflect their operation and have tool tips describing their functionality to the user. The DAT will include user-friendly GUI components such as progress bars and online help available directly from dialogs and menus.
- [4] Input controls protect the user from entering invalid values. Default values for input parameters as well as upper and lower boundaries will be driven by user-adjustable XML files. This validate-on-entry mechanism prevents invalid data from getting to the underlying system. One of the GUI designer's primary roles is to select the most appropriate control for the data type/range that it represents. The appropriate design of these GUI components can make the application intuitive for the user.
- [5] The DAT is able to create multiple windows using the same dataset. This will allow a zoom window and an overview window, in addition to the main window, to all work with the same dataset. The overview window will show a highlighted area representing the current view in the main window. By dragging this highlight, the user will be able to navigate within the main window. Likewise, the zoom window will show the selected portion of the dataset at full resolution or greater magnification. The user will be able to see the area represented in the zoom window as a highlight in the main window and will be able to drag the highlight to reposition the data shown in the zoom window. Displays will be able to be linked together such that they all show the same geographical area as the user pans in one of them.
- [6] Datasets are viewed using a user-selected colour table. Users will be able to define their own colour tables and save and load them. Colour composites can be made up from up to three different co-registered datasets, and colour tables may be defined for all components. The user will be able to select which band of a dataset is to be used for the red, green and blue components. For complex images, the DAT will represent the intensity of the data as colour intensity and the phase by the particular colour from a colour table.

6.3.1 Views

- [1] The DAT will be able to open the data in various windows offering a different representation of the data as Views and ToolViews. Views are for analysis of a particular band of the data. ToolViews are shared between datasets, and they highlight the data from the currently selected dataset.

6.3.1.1 Image View

- [1] The image view displays the sample values of raster datasets such as bands and tie-point grids as an image.
- [2] Datasets will be viewed using a user-selected colour table. Users will be able to define their own colour tables and save and load them.

6.3.1.2 3D View

- [1] With the 3D View, the user will be able to view datasets as layers on a 3D Earth. If a Digital Elevation Map (DEM) has been co-registered, the data images will be draped over the DEM in 3D. The user will be able to adjust the exaggeration factor of the DEMs height scale.
- [2] This 3D display will be implemented using JOGL the Java binding for the OpenGL API. JOGL provides hardware supported 3D visualization and is both portable and open source.

Array Systems Computing Inc.

Commercial in Confidence

Printed on December 10, 2012

Page 15

Use, duplication, or disclosure of this document or any information contained herein is subject to the restriction on the title page of this document.

- [3] OpenGL is a powerful API that provides high quality, high performance 3D graphics which can be hardware accelerated by most graphic cards. The OpenGL API acts as the interface between the software and the graphics card installed on the target system.
- [4] The 3D view will use a spherical coordinate system. The General Perspective Projection is used positioning a camera at a near space distance above the Earth. Panning and zooming is handled by OpenGL and the graphics card, making the system very efficient.
- [5] The system will be able to efficiently and intuitively pan and zoom to user selected positions. The left mouse button is used for selecting and dragging objects and grid points. The middle mouse button and wheel are used for navigation. Clicking and dragging the middle mouse button pans, double-clicking re-centers at the location of the mouse pointer and the mouse wheel zooms in and out. The right mouse button calls up a context sensitive pop-up menu for operations specific to the objects or overlays currently under the mouse pointer. All functions can be accessed via a menu item or a shortcut key.

6.3.1.3 Products ToolView

- [1] The Products ToolView maintains all opened products in a tree where the user may access bands, flags and metadata.

6.3.1.4 PixelInfo ToolView

- [1] The PixelInfo ToolView shows readout information over the current mouse position on the image view.
- [2] As the user moves the mouse over the image the following information will be shown:
 - Geo-location: image position, the geographic-location and also the map co-ordinates.
 - Tie Point Grids: the values of the tie-point grids.
 - Time Info: the time information associated with the current line.
 - Bands: the value of the pixel beneath the mouse pointer.
 - Flags: the state of the flags at the current pixel.

6.3.1.5 Metadata View

- [1] When opening a data product, the DAT will be able to inspect and display information about the data without actually reading in the full dataset. Information such as metadata and geographical extent will be displayed and optionally quick-look views and overviews may be requested and viewed.
- [2] Complete header analysis will be performed for all products and the results may be optionally saved to a text file.

6.3.1.6 Bitmask Overlay ToolView

- [1] This view allows the user to activate overlay flags and combine of them on a loaded band Image View.

6.3.1.7 Navigation & Zoom ToolView

- [1] The Navigation View allows the user to see an overview of the whole dataset and a highlight of the currently viewed area in the active Image View.
- [2] It can also be used to control the zooming and panning of the Image View and to link multiple views together such that they show the same geographic area.

Array Systems Computing Inc.

Commercial in Confidence

Printed on December 10, 2012

Page 16

Use, duplication, or disclosure of this document or any information contained herein is subject to the restriction on the title page of this document.

6.3.1.8

Statistics ToolView

- [1] The DAT will calculate the following statistical variables for a real dataset:
 - mean
 - median
 - max
 - min
 - number of pixels
 - standard deviation
 - coefficient of variation
 - histogram with normalized percentage frequency
 - equivalent number of looks.
- [2] Statistics will be displayed in the DAT in a tabular format and can be saved to a text file optionally with relevant meta-data from the data product. The user will be able to adjust the parameters used to generate the statistics and be able to specify the valid range within which the statistics will be computed.
- [3] In addition, the current mouse cursor position will produce a readout shown in a common status area. The readout will have the following values:
 - Pixel coordinates in line/sample and latitude/longitude
 - Pixel value
 - Scaled pixel value if applied
 - Values from the ADSs.
- [4] The readouts will be computed by interpolation including incidence angle, Doppler value, satellite position and refined incidence angle using a slope from a DEM. The current cursor position in Latitude and Longitude and DEM elevation in meters will be shown in the status bar as the user moves the mouse. All windows will show the current zoom factor and information about pixel dimensions and ranges.

6.3.1.9

WorldMap ToolView

- [1] A world map toolview is available to show the extents of the current dataset. NEST integrate NASA's WorldWind for the world map view.

6.3.2

Regions of Interest

- [1] The user will be able to graphically define a Region of Interest (ROI) by drawing a rectangle or a polygon. Likewise, an ROI could be specified in an XML-processing request file to a tool by including parameters such as coordinate system, top left corner, bottom right corner, etc., or a ROI polygon by specifying the number of vertices and the list of vertex coordinates. Coordinates could be entered as longitude, latitude or more precisely as row and column.
- [2] The ROI will be used by the user to identify a particular feature in the image such as a built-up area, an oil spill or a water body in order to extract useful information about it. Within the ROI, the user will analyze tone or relative brightness of objects, the measured size of features, the shape and structure of objects and the textures.
- [3] A ROI will be convertible into a binary bit mask of the same dimensions as the dataset to which it relates. If a ROI exceeds the limits of the dataset, then the particular tool being used will optionally set excess pixels in the output to zero or clip excess pixels from the output.

Array Systems Computing Inc.

Commercial in Confidence

Printed on December 10, 2012

Page 17

Use, duplication, or disclosure of this document or any information contained herein is subject to the restriction on the title page of this document.

6.3.3 Projects and Workspaces

- [1] NEST will allow the user to store and restore the currently opened set of data products as project files. Project files will be stored as XML. The current workspace will also be saved in the project files which include all the relative positions of all currently opened views that have been created to inspect and analyse the data.

6.3.4 Vector Data Overlays

- [1] In addition to the raster data products, the DAT is able to load and display vector data as layers. This is helpful for overlaying shoreline, political boundaries, navigational charts, etc., and using this data to mask land or water areas.
- [2] NEST uses a shape file reader for reading vector data. Shapefiles are a common geospatial vector data format that can represent one class of data; for example, there may be one file that represents rivers and another that represents lakes.
- [3] Vector data is broken down into simple features such as points, lines, and polygons. The user will be able to define a presentation model by selecting colours, styles, labels, etc., for each class of features.

6.4 Data Model and IO

- [1] Data product reader modules will know how to ingest all the data of a particular product and transform it into the GPM data structure. The GPM is abstract enough to handle all types of data products without losing any information.
- [2] The NEST tools will then have only one interface to the GPM in order to work with the data. Data product writers would be able to take the data from the GPM and produce an external file format. With a GPM, conversions from one file format to another could be achieved with the appropriate reader and writer modules.
- [3] Furthermore, the DAT, tools and future plug-ins will be independent of which data products are supported or any specific complexities of the file formats.

6.4.1 Data Model

- [1] Figure 7 illustrates the Product Data Model.

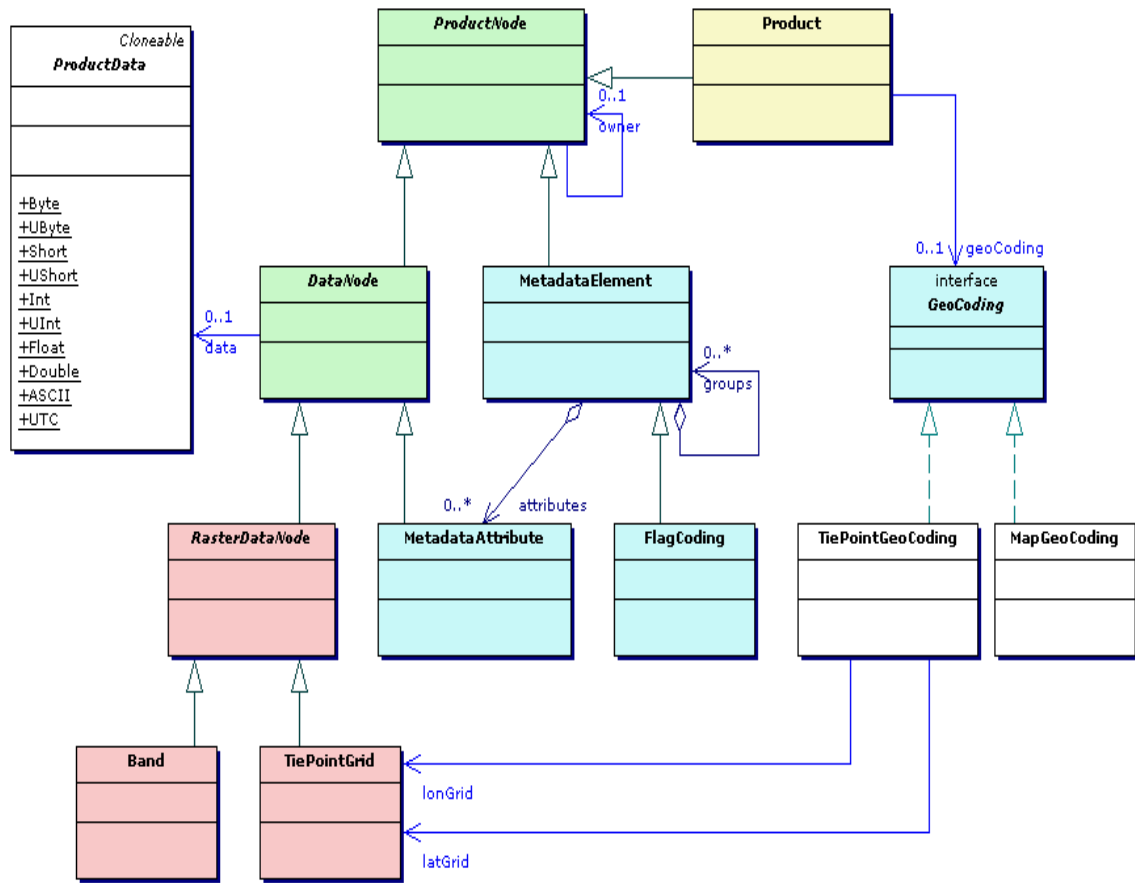


Figure 7: Product Data Model

6.4.1.1

Product

- [1] The main component of the product model is the Product class. This represents a single EO data product which can exist as either a physical file or as an in-memory object in the computer's RAM. A product represents one or more earth observation scenes and is therefore a composite of one or more raster data bands. Optional components of a product are the geo-coding, ancillary data stored as tie-point grids, flag-codings and the metadata.

6.4.1.2

Bands

- [1] Objects of the Product class can have one or more bands for storing raster data. A band provides sample values, geophysical parameter values or quality flags as a function of the pixel position.
- [2] Raster data nodes have a scaling factor and a scaling offset which converts raw counts into geophysical units, and a no-data value which indicates missing pixel values.

6.4.1.3**Tie-Point Grids**

- [1] A Product can also have any number of TiePointGrids, which internally store ancillary data on a regular sub-grid. All of the imaging, processing and analysis functions are applicable to tie-point grids as they are to bands.
- [2] When opened, data is interpolated to be the same dimensions as the scene raster data.

6.4.1.4**Geo-coding**

- [1] The geo-coding is responsible for the transformation of pixel coordinates into geographic coordinates and vice-versa. GeoCoding implementations include the TiePointGeoCoding which is based on latitude/longitude tie-point grids, the MapGeoCoding which uses a map-projection in order to perform the coordinate transformations, and the PixelGeoCoding which operates on the latitude and longitude values given at the band resolution.

6.4.1.5**Metadata**

- [1] A product's metadata, such as the main and specific product headers and annotation datasets, are stored as nested groups of MetadataElements. A product can have any number of metadata elements. Most metadata is composed of single name-value pairs. These are stored as generic MetadataAttributes as part of a MetadataElement.

6.4.1.6**Flag-coding**

- [1] Flag-codings are a special type of metadata. The FlagCoding class describes the classification/quality flags stored as per-pixel bit fields in "flag bands". A product can have multiple flag bands and therefore also multiple flag-codings.

6.4.1.7**Virtual Bands and Band Arithmetic**

- [1] A special type of band is represented by the VirtualBand class. The raster data of this band is computed from a mathematic expression instead of being read directly from a product data file. The mathematical expression is given as a character string and can be composed of the names of any band, tie-point grid or single flag value contained in the product. The strength of the virtual band concept is its great flexibility for working with temporary bands without needing to save them to files.

6.4.1.8**Product Data**

- [1] DataNodes such as Bands, TiePointGrids and MetadataAttribute store the actual product data in objects of the ProductData class. This class is an abstraction of arrays of primitive data types, namely 8-, 16-, 32-bit integer and 32-, 64-bit floating point numbers.

6.4.2**Readers**

- [1] When opening a data product, the DAT will be able to inspect and display information about the data without actually reading in the full dataset. Information such as metadata and geographical extent will be displayed and optionally quick-look views and overviews may be requested and viewed. Complete header analysis will be performed for all products and the results may be optionally saved to a text file.
- [2] The user will be able to select which measurement datasets are read in entirely or specify a user-defined region of interest to be read in.

Array Systems Computing Inc.

Commercial in Confidence

Printed on December 10, 2012

Page 20

Use, duplication, or disclosure of this document or any information contained herein is subject to the restriction on the title page of this document.

6.4.2.1

Data Product Readers

- [1] NEST is able to ingest a variety of data products into a common internal representation. Each tool calls the product manager to open a data product which would then in turn call each product reader to attempt to read the data. The type of data product to be read will automatically be determined by the software. Internally, the data will be abstracted in the GPM. If there isn't a product reader available for reading the data, the user will be informed.
- [2] All or parts of metadata from the Main Product Header (MPH), Specific Product Header (SPH) and all Annotation Data Sets (ADSs) and Measurement Data Sets (MDSs) will be read from the product and optionally presented to the user. The user will have the option to only read in parts of the data product such as selected measurement datasets or user-defined ROI.
- [3] NEST is able to read in data formats from SAR missions including:
 - Envisat ASAR
 - Envisat orbit files
 - ERS-1 & 2 AMI
 - JERS SAR
 - ALOS PALSAR
 - TerraSAR-X
 - RADARSAT-1 & 2
 - Cosmo-Skymed
- [4] The ENVISAT Reader uses the ENVISAT Data Dictionary database used by BEAM to interpret the ENVISAT format for ASAR data products.
- [5] ERS 1 & 2 PGS data in ENVISAT format is handled by a subclass of the ENVISAT reader as shown in Figure 8.

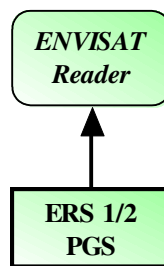


Figure 8: ENVISAT Format Readers

- [6] The CEOS format is a common specification for earth observation data. As such, a common base class will provide the necessary helper functions for parsing and interpreting the format.
- [7] Subclasses for ERS 1 & 2 VMP/PGS, RADARSAT-1, JERS SAR and ALOS PALSAR in CEOS format will implement the specific details required to read each data product.

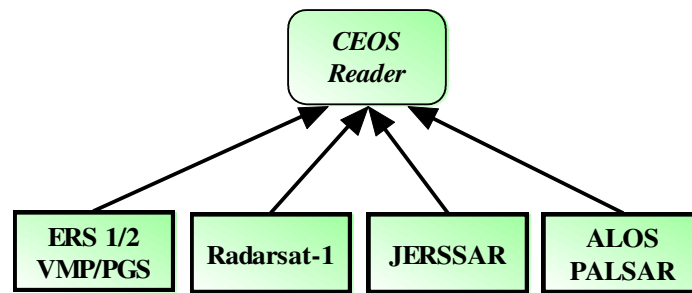


Figure 9: CEOS Readers

- [8] Data products such as TerraSar-X and Radarsat-2 use GeoTIFF for image data and XML for metadata. Classes for these readers will wrap the GeoTIFF reader and an XML reader as shown in Figure 10.

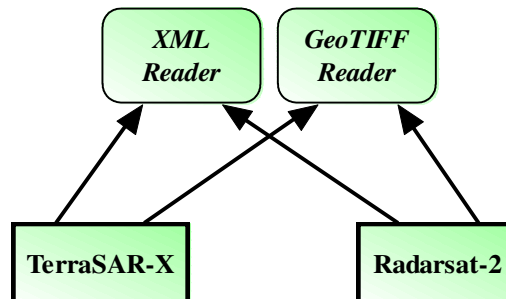


Figure 10: TerraSAR-X and Radarsat-2 Readers

- [9] Cosmo-Skymed data products will be read in by wrapping a HDF 5 CEOS reader as shown in Figure 11.

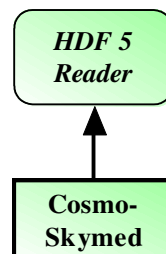


Figure 11: HDF 5 CEOS Readers

6.4.2.2 Common EO File Formats

- [1] NEST ingests common file formats used by third party earth observation (EO) software such as GeoTIFF, HDF 4, HDF 5 EOS, NetCDF, and BEAM DIMAP.
- [2] Only NEST supported map projections are handled.

6.4.2.3 Digital Elevation Maps

- [1] Digital Elevation Maps (DEMs) are imported from various formats including SRTM, ACE DEM, and GETASSE30 DEM and abstracted into a common internal format for use by the NEST tools.
- [2] Elevations are interpolated based on the image data product to which the DEM is being applied. Holes in the DEM datasets are corrected using the Fill Hole operator.
- [3] SRTM is the Shuttle Radar Topography Mission offering the most complete high resolution elevation models available. It covers the area from 56 °S to 60 °N. The dataset contains no-data areas affected by mountain and dessert areas. SRTM is available in 30 and 3 arc second resolutions for most of the globe and in 1 arc second resolution of the United States.
- [4] ACE is the Altimetry Corrected Elevations at 30 arc second resolution covering the entire globe.
- [5] DTED is the Digital Terrain Elevation standardized in U.S. Military Specification *MIL-PRF-89020B*. It provides Level 0 at 30, Level 1 at 3 and Level 2 at 1 arc seconds.
- [6] GETASSE30 is the Global Earth Topography and Sea Surface Elevation at 30 arc second resolution. It is a composite digital elevation model composed from the SRTM30, ACE, MSS and EGM96 ellipsoid datasets courtesy of Marc Bouvet of ESA/ESRIN.

6.4.3 Writers

6.4.3.1 Common EO File Formats

- [1] NEST writes common file formats used by third party earth observation (EO) software such as GeoTIFF, HDF 4, HDF 5 EOS, NetCDF, JPEG200 and BEAM DIMAP.
- [2] Only NEST-supported map projections are handled.

6.4.3.2 COTS Image File Formats

- [1] The DAT is able to save images from datasets as they are currently viewed into the following common COTS image file formats:
 - JPEG
 - PNG
 - TIFF
 - BMP.
- [2] The user is prompted with the appropriate parameters for the user-selected output format via a Save As dialog. It will be possible to also produce a KML file such that the image could be imported and overlaid into Google Earth.

6.4.3.3 Generic Binary (BIL, BSQ, BIP)

- [1] NEST has a generic binary import and export capability for Band Interleaved by Line (BIL), Band Sequential (BSQ) and Band Interleaved by Pixel (BIP) formats. With BIL, the first line for all bands is written in sequential order followed by the second line for all bands etc. With BSQ, each band is

Array Systems Computing Inc.

Commercial in Confidence

Printed on December 10, 2012

Page 23

Use, duplication, or disclosure of this document or any information contained herein is subject to the restriction on the title page of this document.

written fully one after the other. With BIP, the first pixel for all bands is written in sequential order followed by the second line for all bands, etc.

6.5 Graph Processing Tool

- [1] NEST relies on BEAM's Graph Processing Framework (GPF) to create processing graphs. The GPF is based on the Java Advanced Imaging (JAI) rendering chain.
- [2] A graph is a set of nodes connected by edges. In this case, the nodes will be the processing steps. The edges will show the direction in which the data is being passed between nodes; therefore it will be a directed graph. The graph will have no loops or cycles, so it will be a Directed Acyclic Graph (DAG) as shown in Figure 12.
- [3] The sources of the graph will be the data product readers, and the sinks can be either a product writer or an image displayed on the DAT.

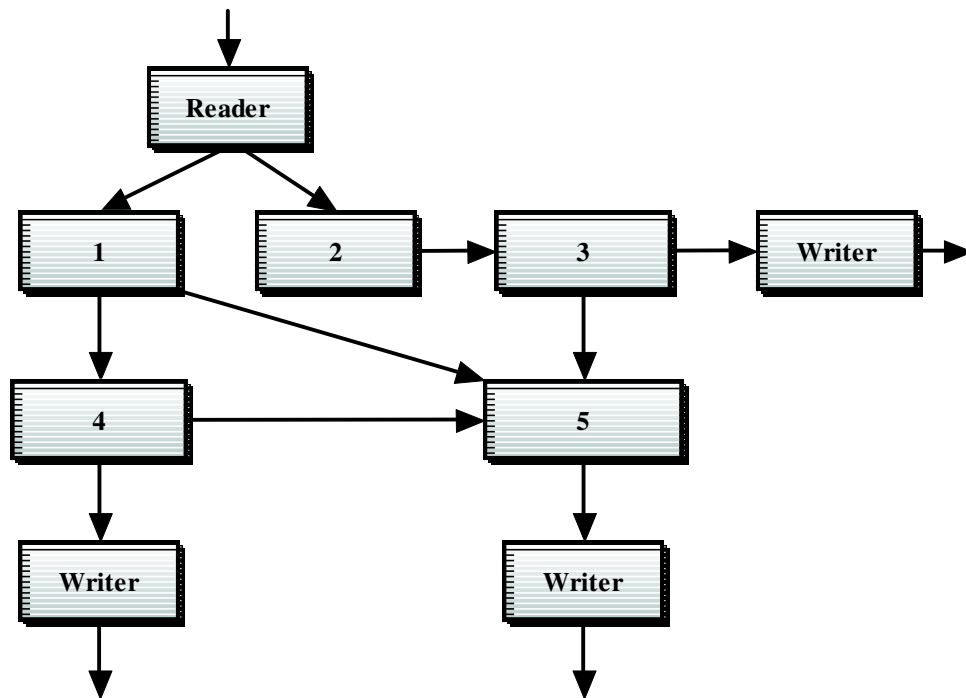


Figure 12: A Directed Acyclic Graph

- [4] The GPF uses a Pull Model, where a request is made from the sink backwards to the source to process the graph. This request could be to create a new product file or to update a displayed image. Once the request reaches a source, the image is pulled through the nodes to the sink. Each time an image passes through an operator, the operator transforms the image, and it is passed down to the next node until it reaches the sink.
- [5] The graph processor will not introduce any intermediate files unless a writer is optionally added anywhere in the sequence.

6.5.1 Graph Builder User Interface

- [1] The user is able to create graphs visually with the Graph Builder, process them directly from the DAT and save them as XML files. These saved graphs can then also be used as the input for the command line Graph Processing Tool (GPT) with a different set of input data products.
- [2] The Graph Builder shown in Figure 13 allows the user to assemble graphs from a list of available operators and connect operator nodes to their sources.

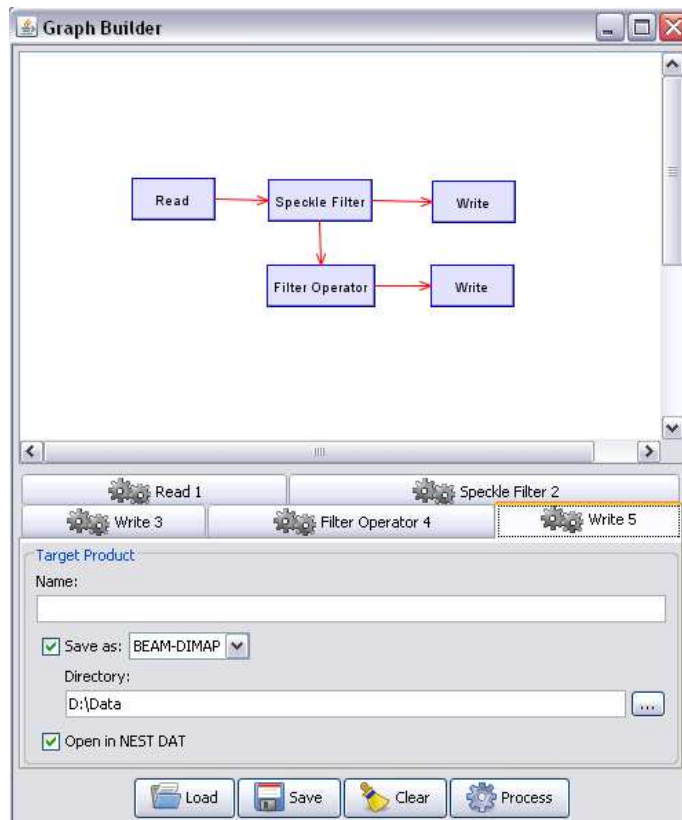


Figure 13: The Graph Builder

- [3] As operators are added, their corresponding OperatorUI will be created and added as tabs to a property sheet. The OperatorUIs will accept user input for the Operator's parameters.
- [4] Before saving or processing a graph, the Graph Builder will call each operator to validate its parameters. Validation will also occur at the graph level to ensure there are no cycles and that each operator has the appropriate connections.

6.5.2 Tiling

- [1] The memory management of NEST allows for very large data products, of which all the data can not be stored in available memory, to be handled by the processing tools and visualization. To do so, a tiled approach is used.

Array Systems Computing Inc.

Commercial in Confidence

Printed on December 10, 2012

Page 25

Use, duplication, or disclosure of this document or any information contained herein is subject to the restriction on the title page of this document.

- [2] The dataset are divided into workable areas called tiles consisting of a subset of the data read from disk in one piece. Only the data for tiles being visualized are read in, and in some cases the data could be down-sampled to view the desired area at the expense of resolution.
- [3] Depending on the tool, data is ingested for a tile or a set of tiles, and processing is applied only to the current set of tiles. The data is then written to a file and released from memory. The process will then be repeated on a new set of tiled data from the large data product.
- [4] From the DAT, in order to allow zooming out and viewing of the entire image, a pyramid of tiled images at different resolutions is used.
- [5] Tiling is generally transparent to the user.

6.5.3 Operators

- [1] In order to provide the greatest flexibility to the end user, processing algorithms such as orthorectification and co-registration will be broken down into unit processing steps as Operators. These operators may be reused to create new processing graphs for other purposes.
- [2] However, in some cases it may prove beneficial to combine processing steps within an operator to either improve performance or simplicity.
- [3] Some functionality such as mosaicking will require prior processing to have been done before images can be stitched together such as reprojection and orthorectification. For this reason, it may be required to put a number of graphs together as one.
- [4] Table 1 describe some of the Operators to be implemented for NEST.

Table 1: Subset of NEST GPF Operators

Operator	Description
Reader	Opens the data with the appropriate product reader for the user specified input file
Writer	Writes the data with the appropriate file format writer for the user specified format
Subset	Creates a spatial subset of a source data product. All operators having subset as a source will be using the area defined by the subset operator as its input image.
Band Arithmetic	Uses an arithmetic expression to create new image sample values derived from existing bands
Image Filter	Provides general filters for Averaging, Edge Detection, Enhancement, and Noise Reduction
Speckle Filter	Provides specialized filters for speckle reduction including Mean, Lee, Refined Lee, Gamma-Map, and Frost
Reprojection	Uses the geolocation information taken from the product header to transform the image to the current map projection. Common coordinate systems such as UTM, UPS and Plate Carree will be handled.
GCP Selection	Generates the Ground Control Points
Warp Evaluation	Computes a warp function from the GCP to fit samples from a slave image to a master image
EllipsoidCorrectionRD	Range Doppler Ellipsoid Correction using average height

Array Systems Computing Inc.

Commercial in Confidence

Printed on December 10, 2012

Page 26

Use, duplication, or disclosure of this document or any information contained herein is subject to the restriction on the title page of this document.

Operator	Description
EllipsoidCorrectionGG	Geolocation Grid Ellipsoid Correction
RangeDopplerTC	Range Doppler Terrain Correction
SARSimulation	Create a simulated SAR image from a DEM
SARSimTC	Terrain Correction using a simulated SAR image
Mosaic	Create a mosaiced product from several input products
ApplyOrbitFile	Correct orbit information from DORIS, PRARE and DELFT orbit files
FillHole	Interpolate DEM holes
ConvertDataType	Converts a product from one data type to another with appropriate scaling
DeburstWSS	Consolidates the bursts in a WSS product
Multilook	Averages the power across a number of lines in both the azimuth and range directions
CreateLandMask	Creates a land mask
AdaptiveThresholding	Ship Detector
WindFieldEstimation	Estimates the speed and direction of the wind from a SAR image
ObjectDiscrimination	Eliminates objects detected outside of a specified min/max range
OilSpillDetection	Oil Spill Detector

7

ACRONYMS AND ABBREVIATIONS

[1] Table 2 shows the acronyms and abbreviations used in this document.

Table 2: Acronyms and Abbreviations

Acronyms and Abbreviations	Explanation
3D	Three-Dimensional
AATSR	Advance Along Track Scanning Radiometer
ACE	Altimetry Corrected Elevations <i>format</i>
ADD	Algorithm Description Document
ADS	Annotation Data Sets
ALOS PALSAR	Advanced Land Observing Satellite Phased-Array-type L-band Synthetic Aperture Radar
ALOS PRISM	Advanced Land Observing Satellite Panchromatic Remote-sensing Instrument for Stereo Mapping
AMI	Active Microwave Instrument
API	Application Programming Interface
Array	Array Systems Computing Inc.
ASAR	Advanced Synthetic Aperture Radar
ASCII	American Standard Code for Information Interchange
ATSR	Along Track Scanning Radiometer
AVHRR	Advanced Very High Resolution Radiometer
AVNIR2	Advanced Visible and Near Infrared Radiometer 2
BC	Brockmann Consult
BEAM	Basic ERS & Envisat (A)ATSR and MERIS
BEST	Basic Envisat SAR Toolbox
BIL	Band Interleaved by Line <i>format</i>
BIP	Band Interleaved by Pixel <i>format</i>
BMP	BitMaP <i>format</i>
BSQ	Band SeQuential <i>format</i>
CEOS	Committee on Earth Observation Satellites
COTS	Commercial Off-The-Shelf
CVS	Concurrent Versioning System
DAG	Directed Acyclic Graph
DAT	Display and Analysis Tool
DCN	Document Control Number

Array Systems Computing Inc.

Commercial in Confidence

Printed on December 10, 2012

Page 28

Use, duplication, or disclosure of this document or any information contained herein is subject to the restriction on the title page of this document.

Acronyms and Abbreviations	Explanation
DEM	Digital Elevation Model
DIMAP	Digital Image MAP
DTED	Digital Terrain Elevation Data <i>format</i>
DVD	Digital Versatile Disk
EGM	Earth Gravitation Model
ENVISAT	ENVironmental SATellite
EO	Earth Observation
ERS	Earth Remote Sensing Satellite
ESA	European Space Agency
ESRIN	European Space Research Institute
GCP	Ground Control Point
GeoTIFF	TIFF-based interchange format for GEOreference <i>for raster imagery</i>
GETASSE30	Global Earth Topography and Sea Surface Elevation at 30 arc second resolution
GNATS	<i>Represents the GNU problem report management system</i>
GNU	GNU's Not Unix (<i>recursive acronym</i>)
GPF	Graph Processing Framework
GPM	Generic Product Model
GPT	Graph Processing Tool
GUI	Graphical User Interface
HDF	Hierarchical Data Format
HTML	Hyper Text Mark-up Language
IO	Input / Output
IDE	Integrated Development Environment
JAI	Java Advanced Imaging
JDK	Java Development Kit
JERS	Japanese Earth Resources Satellite
JOGL	Java OpenGL
JPEG200	Joint Photographic Experts Group <i>image format</i> 200
KML	Keyhole Mark-up Language <i>for managing 3D geospatial data</i>
MDS	Measurement Data Sets
MERIS	Medium Resolution Imaging Spectrometer
MODIS	Moderate-Resolution Imagine Spectrometer
MSS	Mean Sea Surface

Array Systems Computing Inc.

Commercial in Confidence

Printed on December 10, 2012

Page 29

Use, duplication, or disclosure of this document or any information contained herein is subject to the restriction on the title page of this document.

Acronyms and Abbreviations	Explanation
MPH	Main Product Header
NEST	Next ESA SAR Toolbox
NetCDF	Network Common Data Form
OO	Object-Oriented
OpenGL	Open Graphics Language
OS	Operating System
PDF	Portable Document Format
PDR	Preliminary Design Review
PNG	Portable Network Graphics <i>format</i>
RAM	Random Access Memory
ROI	Region Of Interest
SAD	Software Architecture Documents
SAR	Synthetic Aperture Radar
SDF	System Development Folder
SPH	Specific Product Header
SRTM	Shuttle Radar Topography Mission <i>format</i>
SVN	Subversion
TIFF	Tagged Image File Format
TS	Technical Specification
UNIX	<i>originally UNICS, UNiplexed Information and Computing Service, the acronym evolved to UNIX</i>
UPS	Universal Polar Stereographic <i>projection</i>
UTC	Universal Time Co-ordinated
UTM	Universal Transverse Mercator <i>projection</i>
VISAT	VISualization and Analysis Tool
XML	Extensible Markup Language

Array Systems Computing Inc.

Commercial in Confidence

Printed on December 10, 2012

Page 30

Use, duplication, or disclosure of this document or any information contained herein is subject to the restriction on the title page of this document.