<table>
<thead>
<tr>
<th>TOPIC</th>
<th>PRESENTER</th>
<th>TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>Dr. Jay Zwally/David Hancock</td>
<td>9:00-9:05</td>
</tr>
<tr>
<td>Board</td>
<td>Stan Scott</td>
<td>9:05-9:10</td>
</tr>
<tr>
<td>Science Team Updates</td>
<td>Dr. Bob Schutz</td>
<td>9:10-9:25</td>
</tr>
<tr>
<td>Project Scientist</td>
<td>Dr. Jay Zwally</td>
<td>9:25-9:35</td>
</tr>
<tr>
<td>Instrument Update</td>
<td>Dr. Eleanor Ketchum</td>
<td>9:35-9:45</td>
</tr>
<tr>
<td>I-SIPS Software Requirements</td>
<td>David Hancock</td>
<td>9:45-10:10</td>
</tr>
<tr>
<td></td>
<td><strong>BREAK</strong> (10 minutes)</td>
<td>10:10-10:20</td>
</tr>
<tr>
<td></td>
<td>Software Overview &amp; Architectural Design</td>
<td>10:20-10:45</td>
</tr>
<tr>
<td></td>
<td>Anita Brenner</td>
<td></td>
</tr>
<tr>
<td></td>
<td>I-SIPS Scheduling and Data Management System (SDMS) (Part 1)</td>
<td>10:45-12:00</td>
</tr>
<tr>
<td></td>
<td>Mark Sherman</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>LUNCH</strong> (60 minutes)</td>
<td>12:00-1:00</td>
</tr>
<tr>
<td></td>
<td>I-SIPS Scheduling and Data Management System (SDMS) (Part 2)</td>
<td>1:00-1:45</td>
</tr>
<tr>
<td></td>
<td>Mark Sherman</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Job Descriptions/I-SIPS Operation</td>
<td>1:45-2:15</td>
</tr>
<tr>
<td></td>
<td>Mark Sherman</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GLAS Software Development Process</td>
<td>2:15-2:35</td>
</tr>
<tr>
<td></td>
<td>Gladstone Marcus</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GLAS Software Detailed Design</td>
<td>2:35-2:55</td>
</tr>
<tr>
<td></td>
<td>Anita Brenner</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>BREAK</strong> (15 minutes)</td>
<td>2:55-3:10</td>
</tr>
<tr>
<td></td>
<td>GLAS Executive</td>
<td>3:10-3:40</td>
</tr>
<tr>
<td></td>
<td>Jeff Lee</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Level 1A Processing</td>
<td>3:40-4:10</td>
</tr>
<tr>
<td></td>
<td>Peggy Jester</td>
<td></td>
</tr>
</tbody>
</table>
# AGENDA - DAY 2

<table>
<thead>
<tr>
<th>TOPIC</th>
<th>PRESENTER</th>
<th>TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elevation ATBD</td>
<td>A. Brenner/LeeAnne Roberts/Suneel Bhardwaj</td>
<td>9:00-10:30</td>
</tr>
<tr>
<td>Summary /Products</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Altimeter Waveform Analysis</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Altimeter Elevation Computation (Part 1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>BREAK</strong> (10 minutes)</td>
<td></td>
<td>10:30-10:40</td>
</tr>
<tr>
<td>Elevation ATBD</td>
<td>A. Brenner/L. Roberts/S. Bhardwaj</td>
<td>10:40-11:10</td>
</tr>
<tr>
<td>Summary /Products</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Altimeter Waveform Analysis</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Altimeter Elevation Computation (Part 2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Atmosphere ATBD</td>
<td>Kristine Barbieri/Steve Palm</td>
<td>11:10-12:10</td>
</tr>
<tr>
<td>Summary/Products</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Backscatter Calculations</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cloud Layers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Aerosol Layers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Optical Properties</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>LUNCH</strong> (60 minutes)</td>
<td></td>
<td>12:10-1:10</td>
</tr>
<tr>
<td>Utilities</td>
<td>Anita Brenner</td>
<td>1:10-1:30</td>
</tr>
<tr>
<td>I-SIPS Computing Facility</td>
<td>John DiMarzio</td>
<td>1:30-2:00</td>
</tr>
<tr>
<td>Hardware Configuration</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lessons Learned Version 0</td>
<td>Gladstone Marcus</td>
<td>2:00-2:15</td>
</tr>
<tr>
<td>Wrap-up</td>
<td>David Hancock</td>
<td>2:15-2:25</td>
</tr>
<tr>
<td><strong>BREAK</strong> (30 minutes)</td>
<td></td>
<td>2:25-2:55</td>
</tr>
<tr>
<td>Comments</td>
<td>Review Board</td>
<td>2:55</td>
</tr>
</tbody>
</table>
ICESat SIPS SOFTWARE
CRITICAL DESIGN REVIEW

Day 1

November 30 - December 1, 1999
9:00 a.m.
Goddard Space Flight Center, Building 6 Room W137
INTRODUCTION

Dr. Jay Zwally
ICESAT Project Scientist
NASA Goddard Space Flight Center
Bldg. 33, Room A217
Greenbelt, MD 20771
email: jay.zwally@gsfc.nasa.gov
phone: (301) 614-5643
fax: (301) 614-5644

David W. Hancock III
NASA/GSFC Wallops Flight Facility
Bldg. N159, Room E218
Wallops Island, VA 23337
email: hancock@osb.wff.nasa.gov
phone: (757) 824-1238
fax: (757) 824-1036
ICESAT/GLAS
I-SIPS SOFTWARE DEVELOPMENT

KEY PERSONNEL

Dr. Jay Zwally, NASA/GSFC, ICESat Project Scientist, GLAS Team Member

Mr. David Hancock, NASA/GSFC, Science Software Development Manager

Ms. Anita Brenner, Raytheon ITSS, Deputy Science Software Development Manager

Dr. Bob Schutz, UTCSR, GLAS Science Team Leader
ICESAT/GLAS
I-SIPS SOFTWARE DEVELOPMENT
Team Members

Dr. Jay Zwally
ICESat Project Scientist
NASA Goddard Space Flight Center
Greenbelt, MD 20770
e-mail: jay.zwally@gsfc.nasa.gov
(301) 614-5643

John DiMarzio
Raytheon ITSS
Greenbelt, MD 20770
e-mail: john@icesat2.gsfc.nasa.gov
(301) 614-5893

LeeAnne Roberts
Raytheon ITSS
Greenbelt, MD 20770
e-mail: leeanne@icesat2.gsfc.nasa.gov
(301) 614-5917

Peggy L. Jester
Raytheon ITSS
Wallops Island, VA 23337
e-mail: jester@osb.wff.nasa.gov
(757) 824-2093

David W. Hancock III
NASA/GSFC Wallops Flight Facility
Wallops Island, VA 23337
e-mail: hancock@osb.wff.nasa.gov
(757) 824-1238

Louis Kouvaris
Raytheon ITSS
Greenbelt, MD 20770

e-mail: louis@icesat2.gsfc.nasa.gov
(301) 614-5934

Mark Sherman
Raytheon ITSS
4500 Forbes Blvd
Lanham MD, 20706

e-mail: msherman@pop3.stx.com
(301) 794-3046

Jeffrey E. Lee
Raytheon ITSS
Wallops Island, VA 23337
e-mail: jlee@osb.wff.nasa.gov
(757) 824-1853

Anita Brenner
Raytheon ITSS
Greenbelt, MD 20770

e-mail: anita@icesat2.gsfc.nasa.gov
(301) 614-5914

Gladstone Marcus
Raytheon ITSS
Greenbelt, MD 20770

e-mail: gladstone.marcus@gsfc.nasa.gov
(301) 614-5728

Ingrid Wu
Raytheon ITSS
Greenbelt, MD 20770

e-mail: ingrid@icesat2.gsfc.nasa.gov
(301) 286-3973

Dennis W. Lockwood
Raytheon ITSS
Wallops Island, VA 23337

e-mail: lockwood@osb.wff.nasa.gov
(757) 824-2252

Suneel Bhardwaj
Raytheon ITSS
Greenbelt, MD 20770

e-mail: suneel@icesat2.gsfc.nasa.gov
(301) 614-5729

Steve Palm
Science Systems and Applications Inc.
NASA/Goddard Space Flight Ctr.
Greenbelt, Maryland 20771

e-mail: ssp@virl.gsfc.nasa.gov
301-614-6276

Lisa C. Brittingham
Raytheon ITSS
Wallops Island, VA 23337

e-mail: lcbritt@osb.wff.nasa.gov
(757) 824-2292

Carol T. Purdy
Raytheon ITSS
Wallops Island, VA 23337

e-mail: purdy@osb.wff.nasa.gov
(757) 824-1596

Kristine Barbieri
Raytheon ITSS
Greenbelt, MD 20770

e-mail: kristine@icesat2.gsfc.nasa.gov
(301) 614-5932

Hwa-Ja Rhee
Raytheon ITSS
Greenbelt, MD 20770

e-mail: rhee@icesat2.gsfc.nasa.gov
(301) 286-3913

Ronald L. Brooks
Raytheon ITSS
Wallops Island, VA 23337

e-mail: brooks@osb.wff.nasa.gov
(757) 824-1041

Dr. George S. Hayne
NASA/GSFC Wallops Flight Facility
Wallops Island, VA 23337

e-mail: hayne@osb.wff.nasa.gov
(757) 824-1294
BOARD

Stan Scott
ESDIS Project, Code 423
NASA Goddard Space Flight Center
Greenbelt, MD 20771
email: stanley.scott@gsfc.nasa.gov
phone: (301) 614-5334
BOARD MEMBERS

Chairperson
Stan Scott
ESDIS Project, Code 423
NASA/Goddard Space Flight Center
Greenbelt, MD 20771
dstmary.scott@gsc.nasa.gov
phone: (301) 614-5334

Members
Edward Chang
ICESAT Project
NASA/Goddard Space Flight Center
Greenbelt, MD 20771
dstar.chang@gsfc.nasa.gov
phone: (301) 286-6964

Erich Stocker
TRMM Project
NASA/Goddard Space Flight Center
Greenbelt, MD 20771
dstarocker@tsdis.gsfc.nasa.gov
phone: (301) 614-5178

Dr. Gene C. Feldman
SEAWIFS Project
NASA/Goddard Space Flight Center
Greenbelt, MD 20771
dstar@seawifs.gsfc.nasa.gov
phone: (301) 286-9676

Ron Weaver
University of Colorado
Boulder, CO 80309
weaverr@kyros.colorado.edu

Ed Masuoka
MODIS Project
NASA/Goddard Space Flight Center
Greenbelt, MD 20771
dmasuoka@pop900.gsfc.nasa.gov
phone: (301) 614-5515
SCIENCE TEAM UPDATES

Dr. Robert E. Schutz
GLAS Science Team Leader
Center for Space Research

Suite 200
3925 W. Braker Lane
Austin, TX 78759-5321
email: schutz@utcsr.ae.utexas.edu
phone: (512) 471-4267
fax: (512) 471-3570
INSTRUMENT DESCRIPTION UPDATES

Ronald B. Follas
GLAS Instrument Manager
Science Systems Applications, Inc.

Mail Stop 924.0
NASA/Goddard Space Flight Center
Greenbelt, MD 20771
email: follas@pop900.gsfc.nasa.gov
phone: (301) 286-6609
fax: (301) 286-0213

Dr. Eleanor Ketchum
GLAS System Engineer

Mail Stop 730.0
NASA/Goddard Space Flight Center
Greenbelt, MD 20771
email: ketchum@pop900.gsfc.nasa.gov
phone: (301) 286-1170
fax: (301) 286-0213
I-SIPS SOFTWARE DESCRIPTION/REQUIREMENTS

David W. Hancock III
NASA/GSFC Wallops Flight Facility
Bldg. N159, Room E218
Wallops Island, VA 23337
email: hancock@osb.wff.nasa.gov
phone: (757) 824-1238
fax: (757) 824-1036
I-SIPS SOFTWARE DEFINITIONS

- The GLAS Standard Data Software (SDS) is the system which provides data processing and mission support for the GLAS Investigation.
- The SDS is composed of the ICESat Science Investigator-led Processing System (I-SIPS) and the Instrument Support Facility (ISF) Software.
- The I-SIPS includes the software and operations which produce the GLAS standard data products and their metadata.
- The Instrument Support Facility (ISF) Software includes the GLAS instrument health assessment, instrument commanding, and other functions required for mission support.
I-SIPS SOFTWARE SUBSYSTEMS

- Schedule and Data Management Subsystem (SDMS)- Processing environment to control job flow, data distribution, and archiving
- GLAS Science Algorithm Software – Creates GLAS standard products (controls which products are created and implements ATBD )
- Utility Subsystems- stand-alone software for creating ancillary data, job planning, product QA, reformatting to HDF-EOS
CDR PURPOSE

- To present the I-SIPS design and implementation to a level that demonstrates that the requirements are being met to produce the standard data products in routine production
- To allow suggested improvements in implementation
- To identify any omissions
- Not intended to address the scientific aspects of the Algorithm Theoretical Basis Documents
- Not intended to address the ISF
CDR CONTENTS

- Present system requirements and description
- Present implementation design
- Present detailed work in progress
I-SIPS SOFTWARE DEVELOPMENT DESCRIPTION

- Precision orbit and precision attitude software development at University of Texas at Austin (Ancillary process)
- GLAS science processing software being coded at GSFC under Science Team direction
  - Based on ATBDs
  - Being implemented by one set of developers
  - Under configuration management (using ClearCase)
  - Software designed to handle processing, partial processing, and reprocessing
  - Final Level 2 products in HDF-EOS, others in TBD
- Scheduling and Data Management subsystem (SDMS) being coded at GSFC under I-SIPS proposal funding
DOCUMENTATION

• Current documents provided on: http://glas.wff.nasa.gov
• NASA/Technical Memorandum Series 208641
I-SIPS PROCESSING BASIC REQUIREMENTS

- Process 24 hours of GLAS instrument data into standard data products within 4 hours of receipt of all required inputs
- Ability to distribute to the Science Team Level 1 and Level 2 data products within 24 hours of receipt of Level 0 data (uses predict ancillary data)
- Distribute fully processed Level 1 and Level 2 data products to NSIDC within 14 days of receipt of Level 0 data (after becoming operational and assuming proper funding)
- Support reprocessing requirements without delaying regular processing assuming proper funding
I-SIPS DATA ARCHIVING REQUIREMENTS

- Archives internal data products for internal I-SIPS use, to End-of-Mission
- I-SIPS archives a log of products delivered
- I-SIPS does not perform permanent archive
OPERATIONS

- Autonomous Operation 7 days/week, 24 hours
- Normal Manned Operation is 5 days/week, 12 hours
- Available on-call
- Initial calibration period TBD (as many as required)
I-SIPS BASIC DESCRIPTION

Performs:

• The I-SIPS processing performs the following functions:
  – ingest input data
  – execute algorithms to create the GLAS level 1 and level 2 data products
  – assess data product for quality and content, and produce metadata
  – create processing reports/log
  – deliver data products to the NSIDC DAAC

• Perform reprocessing as required
• Create quick-look data for science team

Does Not Perform:

• All Level 3 and 4 GLAS data products are produced by the science team as special products
# GLAS DATA PRODUCTS

<table>
<thead>
<tr>
<th>Product ID (Identification)</th>
<th>Product Name</th>
<th>Product Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>GLA00</td>
<td>GLAS Instrument Packet</td>
<td>0</td>
</tr>
<tr>
<td>GLA01</td>
<td>Altimetry Data</td>
<td>1A</td>
</tr>
<tr>
<td>GLA02</td>
<td>Atmosphere Data</td>
<td>1A</td>
</tr>
<tr>
<td>GLA03</td>
<td>Engineering Data</td>
<td>1A</td>
</tr>
<tr>
<td>GLA04</td>
<td>SRS and GPS Data</td>
<td>1A</td>
</tr>
<tr>
<td>GLA05</td>
<td>Waveform-based Elevation Corrections</td>
<td>1B</td>
</tr>
<tr>
<td>GLA06</td>
<td>Elevation</td>
<td>1B</td>
</tr>
<tr>
<td>GLA07</td>
<td>Backscatter</td>
<td>1B</td>
</tr>
<tr>
<td>GLA08</td>
<td>Boundary Layer Height</td>
<td>2</td>
</tr>
<tr>
<td>GLA09</td>
<td>Cloud Height for Multiple Layers</td>
<td>2</td>
</tr>
<tr>
<td>GLA10</td>
<td>Aerosol Vertical Structure</td>
<td>2</td>
</tr>
<tr>
<td>GLA11</td>
<td>Thin Cloud/Aerosol Optical Depth</td>
<td>2</td>
</tr>
<tr>
<td>GLA12</td>
<td>Ice Sheet Elevation</td>
<td>2</td>
</tr>
<tr>
<td>GLA13</td>
<td>Sea Ice Roughness</td>
<td>2</td>
</tr>
<tr>
<td>GLA14</td>
<td>Land/Canopy Elevation</td>
<td>2</td>
</tr>
<tr>
<td>GLA15</td>
<td>Ocean Elevation</td>
<td>2</td>
</tr>
</tbody>
</table>
## DATA PRODUCT VOLUMES

<table>
<thead>
<tr>
<th>Prod ID</th>
<th>Record Size (bytes)</th>
<th>Mbytes / Day (approx)</th>
<th>Rate</th>
<th>Num Recs</th>
<th>At 14 revs/day</th>
<th>Granules revolute</th>
<th>Granules/day</th>
<th>Granule size</th>
<th>Mbytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GLA01 - land</td>
<td>4020</td>
<td>1129</td>
<td>1/1 sec</td>
<td>7</td>
<td>4</td>
<td>1 sec</td>
<td>56</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>GLA01 - ocean</td>
<td>4020</td>
<td>552</td>
<td>1/1 sec</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GLA02</td>
<td>26652</td>
<td>2475.5</td>
<td>1/1 sec</td>
<td>1</td>
<td>0.5</td>
<td>1 sec</td>
<td>7</td>
<td>354</td>
<td></td>
</tr>
<tr>
<td>GLA03</td>
<td>1000</td>
<td>864</td>
<td>1/1 sec</td>
<td>1</td>
<td>0.07</td>
<td>1 sec</td>
<td>1</td>
<td>86</td>
<td></td>
</tr>
<tr>
<td>GLA04</td>
<td>28000</td>
<td>2419.2</td>
<td>1/1 sec</td>
<td>1</td>
<td>0.5</td>
<td>1 sec</td>
<td>7</td>
<td>346</td>
<td></td>
</tr>
<tr>
<td>GLA05</td>
<td>9636</td>
<td>832.6</td>
<td>1/1 sec</td>
<td>1</td>
<td>4</td>
<td>1 sec</td>
<td>56</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>GLA06</td>
<td>5236</td>
<td>452.4</td>
<td>1/1 sec</td>
<td>1</td>
<td>4.00</td>
<td>1 sec</td>
<td>56</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>GLA07</td>
<td>63836</td>
<td>5515.4</td>
<td>1/1 sec</td>
<td>1</td>
<td>0.5</td>
<td>1 sec</td>
<td>7</td>
<td>788</td>
<td></td>
</tr>
<tr>
<td>GLA08</td>
<td>780</td>
<td>3.4</td>
<td>1/4 sec</td>
<td>1</td>
<td>0.07</td>
<td>1 sec</td>
<td>1</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>GLA09</td>
<td>2948</td>
<td>63.7</td>
<td>1/4 sec</td>
<td>1</td>
<td>0.07</td>
<td>1 sec</td>
<td>1</td>
<td>64</td>
<td></td>
</tr>
<tr>
<td>GLA10</td>
<td>13812</td>
<td>2983</td>
<td>1/4 sec</td>
<td>1</td>
<td>0.07</td>
<td>1 sec</td>
<td>1</td>
<td>298</td>
<td></td>
</tr>
<tr>
<td>GLA11</td>
<td>568</td>
<td>123</td>
<td>1/4 sec</td>
<td>1</td>
<td>0.07</td>
<td>1 sec</td>
<td>1</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>GLA12</td>
<td>3004</td>
<td>29.1</td>
<td>1/1 sec</td>
<td>1</td>
<td>0.07</td>
<td>1 sec</td>
<td>1</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td>GLA13</td>
<td>4076</td>
<td>176</td>
<td>1/1 sec</td>
<td>1</td>
<td>0.07</td>
<td>1 sec</td>
<td>1</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>GLA14</td>
<td>5416</td>
<td>165.7</td>
<td>1/1 sec</td>
<td>1</td>
<td>0.07</td>
<td>1 sec</td>
<td>1</td>
<td>166</td>
<td></td>
</tr>
<tr>
<td>GLA15</td>
<td>2888</td>
<td>1332</td>
<td>1/1 sec</td>
<td>1</td>
<td>0.07</td>
<td>1 sec</td>
<td>1</td>
<td>133</td>
<td></td>
</tr>
</tbody>
</table>

Total Output per day: 141858
## I-SIPS INPUT DATA

<table>
<thead>
<tr>
<th>Products</th>
<th>Source - Mbytes Transferred per Day</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>EDOS</td>
</tr>
<tr>
<td>GLA00</td>
<td>GLAS PDS (Telemetry Data)</td>
</tr>
<tr>
<td>GLA ANC 01</td>
<td>Meteorological Data File</td>
</tr>
<tr>
<td>GLA ANC 08</td>
<td>Precision Orbit Data File</td>
</tr>
<tr>
<td>GLA ANC 09</td>
<td>Precision Attitude Data File</td>
</tr>
<tr>
<td>GLA SUP 11</td>
<td>Instrument Performance Trend Files</td>
</tr>
<tr>
<td>n/a</td>
<td>Special Products QA Data</td>
</tr>
<tr>
<td>Totals (Mbytes)</td>
<td></td>
</tr>
<tr>
<td><strong>Total Gbytes per Day</strong></td>
<td></td>
</tr>
</tbody>
</table>

n/a = not applicable

* Transfer of GLA00 from NSIDC is a backup to EDOS

Note: Official transfer bytes are maintained in latest Data Product Specifications or the Data Management Plan.
# I-SIPS OUTPUT DATA

## (1st Year)

<table>
<thead>
<tr>
<th>File ID</th>
<th>File Name</th>
<th>SCF</th>
<th>Uteexas</th>
<th>WFF</th>
<th>NSIDC *</th>
</tr>
</thead>
<tbody>
<tr>
<td>GLA00</td>
<td>GLAS PDS (Telemetry Data)</td>
<td>1237.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GLA01</td>
<td>Altimetry Data File</td>
<td>1680.0</td>
<td>840.0</td>
<td>1680.0</td>
<td></td>
</tr>
<tr>
<td>GLA02</td>
<td>Atmosphere Data File</td>
<td>1239.0</td>
<td>2478.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GLA03</td>
<td>Engineering Data File</td>
<td>82.1</td>
<td>82.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GLA04</td>
<td>SRS and GPS Data File</td>
<td>2420.0</td>
<td>2420.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GLA05</td>
<td>Waveform-based Range Corrections File</td>
<td>840.0</td>
<td>840.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GLA06</td>
<td>Elevation File</td>
<td>446.8</td>
<td>446.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GLA07</td>
<td>Backscatter File</td>
<td>5516.0</td>
<td>5516.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GLA08</td>
<td>Boundary Layer and Elevated Aerosol Layer Heights File</td>
<td>7.7</td>
<td>7.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GLA09</td>
<td>Cloud Height for Multiple Layers File</td>
<td>76.7</td>
<td>76.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GLA10</td>
<td>Aerosol Vertical Structure File</td>
<td>303.8</td>
<td>303.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GLA11</td>
<td>Thin Cloud/Aerosol Optical Depth File</td>
<td>17.8</td>
<td>17.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GLA12</td>
<td>Ice Sheet Products File</td>
<td>256.0</td>
<td>256.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GLA13</td>
<td>Sea Ice Products File</td>
<td>361.4</td>
<td>361.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GLA14</td>
<td>Land Products File</td>
<td>455.4</td>
<td>455.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GLA15</td>
<td>Ocean Products File</td>
<td>245.6</td>
<td>245.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GLA ANC 08</td>
<td>Precision Orbit Data File</td>
<td>4.0</td>
<td>4.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GLA ANC 09</td>
<td>Precision Attitude Data File</td>
<td>4.0</td>
<td>4.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n/a</td>
<td>Special Products/Browse/Metadata</td>
<td>7240.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Totals (Mbytes)</td>
<td></td>
<td>10207.3</td>
<td>2420.0</td>
<td>2161.1</td>
<td>23672.4</td>
</tr>
<tr>
<td><strong>Total Gbytes Per Day</strong></td>
<td></td>
<td><strong>38.5</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

n/a - not applicable

* Transfer of GLA00 to NSIDC is a backup to EODS

Note: Official transfer bytes are maintained in latest Data Product Specifications or the Data Management Plan.
## PROCESSING LOAD ESTIMATES TO PRODUCE LEVEL 1 AND LEVEL 2 PRODUCTS

<table>
<thead>
<tr>
<th>Processing Step</th>
<th>CPU-I/O (min) [Note 2]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Level 0 ingest/archive</strong></td>
<td>70.08</td>
</tr>
<tr>
<td><strong>level 1A</strong></td>
<td></td>
</tr>
<tr>
<td>GLA01</td>
<td>45.4</td>
</tr>
<tr>
<td>GLA02</td>
<td>159.03</td>
</tr>
<tr>
<td>GLA03</td>
<td>10.72</td>
</tr>
<tr>
<td>GLA04</td>
<td>90.81</td>
</tr>
<tr>
<td><strong>level 1B</strong></td>
<td></td>
</tr>
<tr>
<td>GLA05</td>
<td>199.36</td>
</tr>
<tr>
<td>GLA06</td>
<td>118.84</td>
</tr>
<tr>
<td>GLA07</td>
<td>157.4</td>
</tr>
<tr>
<td><strong>level 2A</strong></td>
<td></td>
</tr>
<tr>
<td>GLA08</td>
<td>80.07</td>
</tr>
<tr>
<td>GLA09</td>
<td>146.57</td>
</tr>
<tr>
<td>GLA10</td>
<td>214.02</td>
</tr>
<tr>
<td><strong>level 2B</strong></td>
<td></td>
</tr>
<tr>
<td>GLA11</td>
<td>115.49</td>
</tr>
<tr>
<td>GLA12</td>
<td>3.21</td>
</tr>
<tr>
<td>GLA13</td>
<td>2.9</td>
</tr>
<tr>
<td>GLA14</td>
<td>3.05</td>
</tr>
<tr>
<td>GLA15</td>
<td>4.72</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>1421.67</td>
</tr>
</tbody>
</table>

Note 1- K460/8000(2 processors) or RT1000 120 SPECint_rate95, 180 SPECfp_rate95
## NETWORK LOADING (YEAR 1)

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>total Data transfer Mbytes/day</th>
<th>Number of transfers periods/day</th>
<th>Mbytes per transfer period</th>
<th>hours per transfer period</th>
<th>total hours/day transferring</th>
<th>MBITS/Sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>I-SIPS at GSFC B22 Rm. TBD</td>
<td>CSR at UT B?? R?? Austin, TX</td>
<td>3,630</td>
<td>4</td>
<td>908</td>
<td>0.25</td>
<td>1</td>
<td>8.07</td>
</tr>
<tr>
<td>CU LASP</td>
<td>CSR at UT B?? R?? Austin, TX</td>
<td>16</td>
<td>1</td>
<td>16</td>
<td>0.25</td>
<td>0.25</td>
<td>0.14</td>
</tr>
<tr>
<td>I-SIPS at GSFC B22 Rm. TBD</td>
<td>Wallops Flight Facility N159 Rm E2xx</td>
<td>2,161</td>
<td>4</td>
<td>540</td>
<td>0.5</td>
<td>2</td>
<td>2.40</td>
</tr>
<tr>
<td>Wallops Flight Facility B N159 Rm E2xx</td>
<td>I-SIPS at GSFC B22 Rm. TBD</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.02</td>
</tr>
<tr>
<td>EDOS</td>
<td>I-SIPS at GSFC B22 Rm. TBD</td>
<td>8,679</td>
<td>4</td>
<td>2170</td>
<td>0.5</td>
<td>2</td>
<td>9.64</td>
</tr>
<tr>
<td>NSIDC</td>
<td>I-SIPS at GSFC B22 Rm. TBD</td>
<td>22,428</td>
<td>1</td>
<td>22428</td>
<td>2</td>
<td>2</td>
<td>24.92</td>
</tr>
<tr>
<td>NSIDC</td>
<td>I-SIPS at GSFC B22 Rm. TBD</td>
<td>1,360</td>
<td>1</td>
<td>1360</td>
<td>0.5</td>
<td>0.5</td>
<td>6.04</td>
</tr>
<tr>
<td>I-SIPS at GSFC B22 Rm. TBD</td>
<td>GLAS SCF at GSFC B33 Rm B209A</td>
<td>20,415</td>
<td>2</td>
<td>10207</td>
<td>1</td>
<td>2</td>
<td>22.68</td>
</tr>
<tr>
<td>GLAS SCF at GSFC B33 Rm B209A</td>
<td>I-SIPS at GSFC B22 Rm. TBD</td>
<td>7,240</td>
<td>1</td>
<td>7240</td>
<td>1</td>
<td>1</td>
<td>16.09</td>
</tr>
<tr>
<td>GLAS SCF at GSFC B33 Rm B209A</td>
<td>Each alt science team SCF</td>
<td>3,016</td>
<td>1</td>
<td>3016</td>
<td>0.5</td>
<td>0.5</td>
<td>13.41</td>
</tr>
<tr>
<td>GLAS SCF at GSFC B33 Rm B209A</td>
<td>Each met science team SCF</td>
<td>1,318</td>
<td>1</td>
<td>1318</td>
<td>1</td>
<td>1</td>
<td>2.93</td>
</tr>
<tr>
<td>GLAS SCF at GSFC B33 Rm B209A</td>
<td>Each atm science team SCF</td>
<td>8,382</td>
<td>1</td>
<td>8382</td>
<td>2</td>
<td>2</td>
<td>9.31</td>
</tr>
<tr>
<td>Each science team SCF</td>
<td>GLAS SCF at GSFC B33 Rm B209A</td>
<td>478</td>
<td>1.00</td>
<td>478</td>
<td>0.5</td>
<td>0.5</td>
<td>2.12</td>
</tr>
<tr>
<td>ISF at GSFC B22 Rm ???</td>
<td>CU LASP</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.02</td>
</tr>
<tr>
<td>CU LASP</td>
<td>ISF at GSFC B22 Rm ???</td>
<td>16</td>
<td>4</td>
<td>4</td>
<td>0.1</td>
<td>0.4</td>
<td>0.09</td>
</tr>
<tr>
<td>EDOS</td>
<td>NSIDC</td>
<td>8,679</td>
<td>4</td>
<td>2170</td>
<td>1</td>
<td>4</td>
<td>4.82</td>
</tr>
</tbody>
</table>
SOFTWARE PHASE DELIVERY

- Version 0 - working structure to produce products in SCF format without ATBD implemented. No operation systems.
- Version 1 - limited operational version to produce products in SCF format based on 7/99 ATBDs and includes working SDMS.
- Version 2 - fully operational version to produce products. Includes updated ATBDs and SDMS.
STATUS

• I-SIPS S/W version 0 delivered 8/99

• Under Change Control
  – Requirements
  – Architectural design
  – Detailed design for some units

• I-SIPS and SCF H/W selected

• I-SIPS S/W version 1 in progress
SCHEDULE

- I-SIPS S/W CDR - Nov 1999
- I-SIPS Version 1 S/W - July 2000
- I-SIPS Facility Space available - July 2000
- I-SIPS H/W Installation - August 2000
- I-SIPS ready to support S/C Testing - Dec 2000
- I-SIPS ready for launch - Mar 2001
- I-SIPS Operation Readiness Review - May 2001
PDR CONCERNS

- I-SIPS Functionality
- Processing Concept
- DAAC Interface Concept
- I-SIPS Science Team Interface
I-SIPS FUNCTIONALITY PDR CONCERN

The PDR did not address the hardware, network design, operations or data management functions of I-SIPS. No data volumes or processing loads were given, neither average nor peak. The board could not assess the adequacy of the system to meet capacity or functional requirements. The CDR needs to address hardware design, operational scenarios (nominal, error, and end-to-end) and testing. Design requirements need to be justified and matched to budget before development can proceed.
I-SIPS FUNCTIONALITY CDR RESPONSE

- Hardware requirements have been defined and will be presented in CDR.
- Network Loads have been estimated, submitted to ESDIS, and appear reasonable.
- SDMS will address operation and data management.
  - Team reviewed several operational environments and selected the VCL (Vegetation Canopy Lidar) Data Center (VDC) software as model. Major factor was the team experience with the product and common sharing of software enhancements with several other programs provided confidence in development.
- Data volumes and processing loads have been considered.
- Development effort is being actively tracked in Project and appears to be within time and resource budget.
EXECUTIVE OVERHEAD TEST

• Test condition
  – Executive and interface coded, algorithms as stubs
  – Input and output of required data products
  – test input data span of 100 seconds

• Results
  – All products - 8 seconds
  – Altimeter products- 1.4 seconds
  – Elevation products - 1.3 seconds
PROCESSING CONCEPT PDR CONCERN

- The board is concerned about the architecture of the production software. The board recommends that the I-SIPS consider implementation of independently linked and scheduled production steps coupled with smaller granule size to reduce program complexity and interdependence while introducing natural "check points".

- CDR needs to address format and content of EOS Standard Products.
PROCESSING CONCEPT CDR RESPONSE

- Team did performance study of the executive and did not find problems. Continued with design; design will allow processing for individual products.

- Initial processing will be in several jobs. Job breakdown will be presented.

- EOS Standard Products are being defined. ECS and NSIDC are actively involved. Several Earth Science Data Types are submitted. HDF-EOS formatting was more effort than realized. Some question on what to do for Level 1 products.
DAAC INTERFACE CONCEPT

PDR CONCERN:

- The I-SIPS team needs to be actively involved with the DAAC in discussion of scenarios and in validation of assumptions leading to an operations agreement.
- Choice to do distribution by media should be examined.

CDR RESPONSE:

- NSIDC is actively involved in product definition, distribution, and search design.
- Products will be distributed electronically by standard SIPS interface. NSIDC pulls data from I-SIPS.
- I-SIPS/ESDIS ICD in progress, operations agreements will be established later.
I-SIPS SCIENCE TEAM INTERFACE

- **PDR Concern**
  - Need to develop QA scenarios and plans.

- **CDR Response**
  - I-SIPS to SCF interface same methodology as NSIDC interface
  - Utilities defined.
  - QA types defined, but specific implementation will be version 2.
OVERVIEW AND ARCHITECTURAL DESIGN

Anita Brenner
Raytheon ITSS
Bldg. 33, Room B209D
Greenbelt, MD 20771
email: anita@icesat2.gsfc.nasa.gov
phone: (301) 614-5914
ISIPS
CONTEXT DIAGRAM

EDOS

ISF

UTCSR

NSIDC DAAC

L0 Data

L0, L1 Products

Analysis, Trending, Special Products

L1A, L1B, L2 Product Data (HDF-EOS)

Met Data

L0 data

Precision orbits
Precision attitude
GPS-UTC time table
Predicted orbits
Q/A on L1A

SCF

Requested Products

Analysis Products

Special Products Requests And Q/A

L0, L1A, SRS/GPS
ISIPS
SOFTWARE ARCHITECTURE

Legend
SDMS
GLAS Science
External
Data
DBMS
Control
Job Start

External System(s)

Interface Listener(s)

Scheduling

Ingest Job
Preprocess

Processing Job
GLAS Exec

Planning Job
Prod. Rules

Distribution Job
Postprocess

Production Status

Data Management
I-SIPS KEY REQUIREMENTS

• Receive L0 Data from EDOS
• Receive NCEP Meteorological Data from NSIDC
• Process L0 to L1A, L1B, L2 (SCF Format)
• Send L1A SRS/GPS to UTCSR and receive POD/PAD back
• Send L0 and L1 to ISF, receive instrument and spacecraft analysis data back
• Convert L1A, L1B, L2 SCF format to HDF format and send to NSIDC DAAC
• Automatically send a configurable subset of the products created to SCF
CREATION OF STATIC FILES REQUIRED FOR GLAS STANDARD PRODUCTS

Regional DEMs
- Ingest and Reformat DEM

Reference Orbit
- Ingest Reference Orbits and create Ground Track file

Geoid
- Ingest and Reformat Geoid file

Regional masks
- Ingest regional Masks and create Mask file

Global DEM
- Load tide grids

Global model
- Create load tide grid

Global ocean Tide model
- Regional ocean Tide models

GLAS_Exec
- Geoid file

GLAS_Exec
- Mask file

Ingest Ocean tide global model and Create load tide grid

Ingest Ocean tide regional model and Create load tide grid

Land surface Type grid
- Ingest and reformat Surface type file

Surface type Grid for land surfaces

For level 2 Elev. products

IceSat SIPS Software CDR - Day 1

Page 50
I-SIPS SCHEDULING AND DATA MANAGEMENT SYSTEM (SDMS)

Mark Sherman
Raytheon ITSS
4500 Forbes Blvd
Lanham MD, 20706
email: msherman@pop3.stx.com
(301) 794-3046
ICESat SIPS SDMS
DESIGN REVIEW OUTLINE

- I-SIPS SDMS Requirements Overview
- SDMS Development Approach
- SDMS Software Architecture
- SDMS Scheduling Design
- SDMS Data Management Design
- SDMS Production Status Design
- SDMS External Interface Design
- Development Schedule
I-SIPS SDMS REQUIREMENTS OVERVIEW

- Build a highly automated, reliable, and operable system to generate and save products from ICESat GLAS instrument data
- This portion of the Design Review discusses the SDMS, which:
  - allows the plug-in of the GLAS science algorithm software
  - schedules the algorithms according to production rules
  - tracks the progress of the science processing
  - maintains a data repository of the products created
  - transfers data to and from external systems
I-SIPS REQUIREMENTS CONTEXT DIAGRAM

- **SCF**
  - Analysis Products
  - Special Products
  - Requests And Q/A

- **EDOS**
  - L0 Data
  - L0, L1 Products

- **ISF**
  - Analysis, Trending, Special Products

- **ISIPS**
  - L0 data
  - L1A, L1B, L2 Product Data (HDF-EOS)
  - Met Data
  - L0, L1 Products
  - Analysis, Trending, Special Products
  - L1A, SRS/GPS

- **UTCSR**
  - L1A, L1B, L2 Product Data (HDF-EOS)
  - L0 data

- **NSIDC DAAC**
  - L1A, L1B, L2 Product Data (HDF-EOS)
  - Met Data
  - L0 data

- **L1A, SRS/GPS**
  - Precision orbits
  - Precision attitude
  - GPS-UTC time table
  - Predicted orbits
  - Q/A on L1A

- **Requested Products**

- **UTCSR**
  - L0 data

- **ISIPS**
  - Analysis Products
  - Special Products
  - Requests And Q/A

- **SCF**
  - Analysis Products
  - Special Products
  - Requests And Q/A
I-SIPS SDMS KEY REQUIREMENTS

• Provide an environment to run I-SIPS processing:
  – Process L0 to L1A, L1B, L2 (SCF Format)
  – Conversion of L1A, L1B, L2 SCF format to HDF format
  – QA processing (in addition to that on SCF)
• Provide support for reprocessing data when needed
• Provide Interface services for the following:
  – Receive L0 Data from EDOS
  – Receive NCEP Meteorological Data from NSIDC
  – Send L1A SRS/GPS to UTCSR and receive POD/PAD back (turn-around time TBD)
  – Send L0 and L1 to ISF, receive instrument and spacecraft analysis data back
  – Send to HDF and Browse products to NSIDC DAAC
  – Accept special product requests (for L1A) from SCF and send requested products back to SCF
  – Automatically send a configurable subset of the products created to SCF
I-SIPS SDMS KEY REQUIREMENTS (continued)

- Automated processing, everything directly or indirectly triggered by receipt of L0 and ancillary data
- Retain at least two most recent versions of all product levels in computer accessible storage (online or nearline)
- Software is to operate on HP computers
- SDMS Development complete by December 2000
I-SIPS SDMS
ADDITIONAL REQUIREMENTS/GOALS

• Minimize interfaces between SDMS and science software being developed to simplify science software integration:
  – Science software does not need to use a toolkit to work in SDMS
  – Run any stand-alone Unix executable (no language restrictions)
  – No direct access by science software to SDMS database
  – No direct calls to SDMS for resource management
• Operations:
  – 24x7 with no off-shift support and small staff
  – High visibility of current system status
  – Positive control over system workload
  – Good tools for error monitoring and recovery
I-SIPS SDMS
ADDITIONAL REQUIREMENTS/GOALS (continued)

• Reliability:
  – Normal operations with two computer systems, primary and backup
  – Changeover (manually is OK) to single computer system operations within minutes
  – Archive two copies of all product data, one on each computer
  – Can run full capacity (6x) on one computer system
  – Can temporarily run with only one archive, catch-up later
SDMS DEVELOPMENT APPROACH BUILDS

- Raytheon ITSS is currently supporting several projects like the ICESat SIPS:
  - Vegetation Canopy LIDAR Data Center (VDC) at U Md
  - Two Ground Systems for People’s Republic of China
  - Microwave Limb Sounder (MLS) SIPS at JPL
  - Tropospheric Emissivity Spectrometer (TES) SIPS at JPL
- These systems (and several others) are based on a set of core software from the GSFC V0 DAAC and the VDC (we call it “DAACWare”)
- Approach for SDMS is based on this reuse, but requires updates
SDMS DEVELOPMENT APPROACH BUILDS
(continued)

- Two build development approach, synchronized with Science software releases:
  - Version 1 preliminary release
    - coordinated with V1 Science release
    - used for early integration testing and operations experience
    - fully capable system except robotic archive system
  - Version 2 operational release
    - archive system fully incorporated
    - scheduling enhancements
    - additional operational tools (e.g., dbms maintenance)
SDMS DEVELOPMENT APPROACH
SOURCES OF CODE

- Reuse/Upgrade portions of scheduling and data management software from the DAACWare:

<table>
<thead>
<tr>
<th>Source of Reuse</th>
<th>Reuse SLOCS</th>
<th>New SLOCS</th>
<th>Current Status on ISIPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>V0 Scheduler</td>
<td>8200</td>
<td>0</td>
<td>Compiled, not working due to bug with known fix</td>
</tr>
<tr>
<td>V0 Executive</td>
<td>3400</td>
<td>500</td>
<td>Compiled, not tested due to scheduler bug</td>
</tr>
<tr>
<td>V0 Resource Manager</td>
<td>8500</td>
<td>0</td>
<td>Compiled, not tested due to scheduler bug</td>
</tr>
<tr>
<td>VDC Planner</td>
<td>4000</td>
<td>2000</td>
<td>Not on ISIPS yet, some mods expected</td>
</tr>
<tr>
<td>ARCHER</td>
<td>35200</td>
<td>500</td>
<td>Evaluation version(^1) working with no robot</td>
</tr>
<tr>
<td>DCM</td>
<td>6500</td>
<td>6500</td>
<td>Compiled, not tested, being rewritten</td>
</tr>
<tr>
<td>Message Passing</td>
<td>1700</td>
<td>0</td>
<td>Working</td>
</tr>
<tr>
<td>Logging</td>
<td>4300</td>
<td>0</td>
<td>Working</td>
</tr>
<tr>
<td>Prompt Server</td>
<td>2300</td>
<td>0</td>
<td>Working</td>
</tr>
<tr>
<td>DAAC Library Code</td>
<td>60000</td>
<td>0</td>
<td>Working</td>
</tr>
<tr>
<td>Data Fetch/Store/Catalog</td>
<td>0</td>
<td>1000</td>
<td>To be developed</td>
</tr>
<tr>
<td>External Interfaces</td>
<td>0</td>
<td>1500</td>
<td>To be developed</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>134100</strong></td>
<td><strong>12000</strong></td>
<td></td>
</tr>
</tbody>
</table>

Note: SLOCS == Source Lines of Code (i.e., includes all source lines, comments, and blank lines)

\(^1\)Evaluation version will not be deployed, will use newer version from V0 DAAC (discussed later)
SDMS SUPPORT SOFTWARE

• The DAACWare provides a large suite of support software that:
  – is the foundation for the high level DAACWare components
  – will be reused without significant change
  – is not really discussed elsewhere in this presentation

• Significant components that will be reused:
  – Message Passing - mailbox system based on shared memory and sockets
  – Prompt Server - prompts operator when system needs attention
  – Log Server - centralized logging system
  – SDMS support library - utilities, string manipulation, database interfaces, file management, containers, etc.

• Status
  – All recompiled on HP
  – Message Passing and Log Server are tested and running
I-SIPS
SOFTWARE ARCHITECTURE

Legend
- SDMS
- GLAS Science
- External

Production Status

Data Management

Ingest Job
Preprocess

Processing Job
GLAS Exec

Planning Job
Prod. Rules

Distribution Job
Postprocess

External System(s)

Interface Listener(s)

Scheduling

Legend
- Data
- DBMS
- Control
- Job Start
## SDMS SOFTWARE ARCHITECTURE ENVISIONED
### JOB TYPES FOR I-SIPS

- Four broadly classified* types of jobs will be developed:

<table>
<thead>
<tr>
<th>Job Type</th>
<th>Started By</th>
<th>Inputs</th>
<th>Processing</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ingest</td>
<td>• By interface listener as result of message from external system</td>
<td>• Files from external systems</td>
<td>• Preprocess ingest data as needed</td>
<td>• Products to DM</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Check validity of files</td>
<td>• Production Status updates</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Products to DM</td>
<td>• Start messages to scheduler</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Production Status updates</td>
<td></td>
</tr>
<tr>
<td>Planning</td>
<td>• By either Ingest or Processing Job request</td>
<td>• Production Status</td>
<td>• Analyze production status</td>
<td>• Control files to DM</td>
</tr>
<tr>
<td></td>
<td>• By job timer</td>
<td></td>
<td>• Build control files according to rules</td>
<td>• Production Status updates</td>
</tr>
<tr>
<td></td>
<td>• By manual request</td>
<td></td>
<td>• Start jobs</td>
<td>• Start messages to scheduler</td>
</tr>
<tr>
<td>Processing</td>
<td>• By a planning Job</td>
<td>• Control Files, files from DM, static files</td>
<td>• Science Processing</td>
<td>• Products to DM</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• QA Processing</td>
<td>• Production Status updates</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Start messages to scheduler</td>
<td></td>
</tr>
<tr>
<td>Distribution</td>
<td>• By a planning Job</td>
<td>• Control Files, files from DM, static files</td>
<td>• Postprocess data (e.g., reformat to HDF)</td>
<td>• Products to external systems</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Production Status updates</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Start messages to scheduler</td>
</tr>
</tbody>
</table>

* a job can really be anything, so combinations of the above are possible
I-SIPS SCHEDULING DESIGN TERMINOLOGY: “JOBS” AND “STEPS”

- A “step” is an individual Unix process
- A “job” is a set of “steps” that accomplish some common goal (e.g., converting L2 to L3)
- Steps can run in series or in parallel as dictated by a job script
- Scheduling system can run multiple jobs
- Jobs can be distributed among machines
A “TYPICAL” SCIENCE PROCESSING JOB

A Concept Fundamental To The SDMS Design:
Minimize the interface between the processing environment and the science processing so the only interfaces are (1) the command line, (2) files in the default directory, (3) the status returned
SDMS SCHEDULING DESIGN
High Level Requirements

- Run collections of Unix processes (i.e., steps) defined by a script as a job
- Run steps in serial or parallel as dictated by script
- Run multiple concurrent jobs
- Run multiple concurrent computer systems (primary/backup)
- Provide mechanism to control use of system resources
- Provide operator displays for monitoring/controlling workload
- Monitor jobs/steps for failures/success and report to operators
- Provide error recovery tools and procedures
- Be capable of unattended operations
- Maintain a history of every job ever run
SDMS SCHEDULING DESIGN ARCHITECTURE

- **Planner**: 
  - Gets jobs from job database
  - Allocates jobs to multiple executives
  - Updates job database with job status

- **Executive**: 
  - Gets job start messages from planner
  - Parses job scripts
  - Forks and execs step processes
  - Handles requests from processes
  - Monitors processes for completion
  - Interfaces with scheduler to control flow

- **Scheduler**: 
  - Controls job/step work flow
  - Provides operator displays
  - Routes resources request to resource manager

- **Resource Manager**: 
  - Manages disk and discrete resources
SDMS SCHEDULING DESIGN PLANNER PROCESS

- Planner process will be adapted from DAACWare planner
- Maintains a list (in an Oracle database table) of past, current, and future jobs
- Processes that want to run a job add an entry to the job list database
- The planner makes decisions about which jobs from the database that need to be run, based on:
  - Job priority
  - Number of jobs already running
- Operator displays will be provided to allow the operations staff to monitor workload progress and to prioritize upcoming work
- The planner sends ex_start messages to the scheduler to start the appropriate job and awaits notification back that the job has completed
- The status of the job is updated in the database as jobs complete or fail
- The planner is also responsible for restarting the workload after a system failure if necessary
SDMS SCHEDULING DESIGN
Job Plan Display
SDMS SCHEDULING DESIGN
Job Modification Display
SDMS SCHEDULING DESIGN
Executive Process

- The executive process will be a modified version of the DAACWare executive
- For each job, the executive:
  - accepts messages to start jobs from the planner process
  - parses and validates job script
  - creates a working directory for the steps of a job
  - requests permission from the scheduler to run the job
  - On a step by step basis, according to the script
    - Requests permission from scheduler to run step
    - Runs the step (fork and exec), monitoring for completion or failures
    - Reports step status to scheduler
  - returns job completion status to scheduler
  - removes the working directory if all steps complete normally
  - sends a completion message back to the planner
- The executive can run multiple job sequences simultaneously
SDMS SCHEDULING DESIGN
Example of a Script

Example Job Script

```plaintext
job example_job disk=12MB
allocate db_connections=1
  step1 step1_unix_command
  step2 step2_unix_command
parallel
  step3.1 step3.1_unix_command
  step3.2 step3.2_unix_command
  step3.3 step3.3_unix_command
serial
  step3.4.1 step3.4.1_unix_command
  step3.4.2 step3.4.2_unix_command
  step3.4.3 step3.4.3_unix_command
end
ignore-fail step3.5 step3.5_unix_command
step3.6 step3.6_unix_command
end
deallocate
step4 step4_unix_command
end
```

Job uses 12MB of disk space (may wait to get)
Everything in this block needs database (may wait)
These two statements run in series
The statements within the parallel block statement run simultaneously as the third step of the job
The statements within the serial block statement run in series, but the whole serial block statement runs in parallel with the other step3.x statements
Step 4 would run even if step 3.5 failed. Note step 3.6 runs in parallel with 3.5 so it would run anyway
Step 4 runs after all of the Step3.x statements have successfully completed, doesn’t use database
SDMS SCHEDULING DESIGN
Scheduler Process

• The scheduler process will be used as is from the DAACWare
• The primary purpose of the scheduling system is to manage the overall processing workload in a computer system
• The scheduler:
  – accepts requests to run jobs and steps from the executive
  – keeps list of current jobs in prioritized list
  – interacts with the resource manager to determine resource availability
  – selectively grants permission back to executive to proceed with a job or step
• The scheduler supports one or more operator displays that:
  – Show a list of all currently active jobs in a system
  – For operator selected job, shows status of all steps in the job
SDMS SCHEDULING DESIGN

Job Display

Example Job Display
(showing V0 DAAC operational jobs)

<table>
<thead>
<tr>
<th>Job ID</th>
<th>Type</th>
<th>Name</th>
<th>Priority</th>
<th>Arrived</th>
<th>Started</th>
<th>Finished</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>939767006.0000003781</td>
<td>archi copy_brows</td>
<td>-2</td>
<td>Oct 12</td>
<td>Oct 12</td>
<td>--</td>
<td>active</td>
<td></td>
</tr>
<tr>
<td>939767210.0000003781</td>
<td>archi copy_czcs2</td>
<td>-3</td>
<td>Oct 12</td>
<td>Oct 12</td>
<td>--</td>
<td>active</td>
<td></td>
</tr>
<tr>
<td>939767917.0000003781</td>
<td>archi copy_uars</td>
<td>-5</td>
<td>Oct 12</td>
<td>Oct 12</td>
<td>--</td>
<td>active</td>
<td></td>
</tr>
<tr>
<td>939893404.0000003781</td>
<td>archi push_seawi</td>
<td>-82</td>
<td>05:30:14</td>
<td>05:30:16</td>
<td>08:26:06</td>
<td>finished</td>
<td></td>
</tr>
<tr>
<td>939897005.0000003781</td>
<td>archi push_seawi</td>
<td>-103</td>
<td>06:30:15</td>
<td>06:40:16</td>
<td>09:02:17</td>
<td>finished</td>
<td></td>
</tr>
<tr>
<td>939900604.0000003781</td>
<td>archi push_seawi</td>
<td>-105</td>
<td>07:30:14</td>
<td>08:26:18</td>
<td>08:47:52</td>
<td>failed</td>
<td></td>
</tr>
<tr>
<td>939900670.0000003782</td>
<td>retri 84900</td>
<td>-107</td>
<td>07:31:14</td>
<td>07:31:16</td>
<td>08:33:07</td>
<td>finished</td>
<td></td>
</tr>
<tr>
<td>939904205.0000003782</td>
<td>evalu 85985</td>
<td>-108</td>
<td>08:30:10</td>
<td>08:30:12</td>
<td>08:37:05</td>
<td>finished</td>
<td></td>
</tr>
<tr>
<td>939904204.0000003781</td>
<td>archi push_uars</td>
<td>-109</td>
<td>08:30:15</td>
<td>08:48:18</td>
<td>08:50:07</td>
<td>finished</td>
<td></td>
</tr>
<tr>
<td>939904216.0000003781</td>
<td>archi push_seawi</td>
<td>-110</td>
<td>08:30:26</td>
<td>08:50:18</td>
<td>08:55:34</td>
<td>finished</td>
<td></td>
</tr>
<tr>
<td>939904620.0000003782</td>
<td>retri 85985</td>
<td>-111</td>
<td>08:37:04</td>
<td>08:37:18</td>
<td>08:39:08</td>
<td>finished</td>
<td></td>
</tr>
</tbody>
</table>
SDMS SCHEDULING DESIGN
Step Display

Example Step Display
(showing V0 DAAC steps)

![Step Display](image-url)

### Steps for Job 939900604.0000003781 (push_seawifs)

<table>
<thead>
<tr>
<th>Step ID</th>
<th>Type</th>
<th>Priority</th>
<th>Arrived</th>
<th>Started</th>
<th>Finished</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>939900605.0000003781</td>
<td>ingest</td>
<td>-105</td>
<td>08:26:18</td>
<td>08:26:21</td>
<td>08:26:32</td>
<td>finished</td>
</tr>
<tr>
<td>939900606.0000003781</td>
<td>validate</td>
<td>-105</td>
<td>08:26:18</td>
<td>08:26:35</td>
<td>08:26:41</td>
<td>finished</td>
</tr>
<tr>
<td>939900607.0000003781</td>
<td>unix_archive</td>
<td>-105</td>
<td>08:26:18</td>
<td>08:26:45</td>
<td>08:27:17</td>
<td>finished</td>
</tr>
<tr>
<td>939900608.0000003781</td>
<td>primary</td>
<td>-105</td>
<td>08:26:18</td>
<td>08:27:20</td>
<td>08:27:51</td>
<td>finished</td>
</tr>
<tr>
<td>939900609.0000003781</td>
<td>secondary</td>
<td>-105</td>
<td>08:26:18</td>
<td>08:27:54</td>
<td>08:47:43</td>
<td>finished</td>
</tr>
<tr>
<td>939900610.0000003781</td>
<td>expose</td>
<td>-105</td>
<td>08:26:18</td>
<td>08:47:46</td>
<td>08:47:52</td>
<td>failed</td>
</tr>
<tr>
<td>939900611.0000003781</td>
<td>order</td>
<td>-105</td>
<td>08:26:18</td>
<td>--</td>
<td>08:47:52</td>
<td>cancelled</td>
</tr>
<tr>
<td>939900612.0000003781</td>
<td>stage</td>
<td>-105</td>
<td>08:26:18</td>
<td>--</td>
<td>08:47:52</td>
<td>cancelled</td>
</tr>
<tr>
<td>939900613.0000003781</td>
<td>request</td>
<td>-105</td>
<td>08:26:18</td>
<td>--</td>
<td>08:47:52</td>
<td>cancelled</td>
</tr>
<tr>
<td>939900614.0000003781</td>
<td>cleanup</td>
<td>-105</td>
<td>08:26:18</td>
<td>--</td>
<td>08:47:52</td>
<td>cancelled</td>
</tr>
</tbody>
</table>
SDMS SCHEDULING DESIGN

Resource Manager

- The Resource Manager will be used as is from the DAACWare
- The Resource Manager manages two types of resources:
  - Disk Resources: A set of disk partitions used to provide working space for the jobs
  - Discrete Resources: A set of names and counts used to limit quantities of resources used by jobs (e.g., database connections)
- The Resource Manager:
  - accepts requests for combinations of resources from the Scheduler Process
  - allocates all resources at once if sufficient resources are available
  - allocates no resources if at least one resource cannot be allocated
  - returns Go or NoGo to Scheduler as appropriate
- The Resource Manager provides a number of displays for managing and viewing resources
### SDMS SCHEDULING DESIGN

#### Disk Resource Displays

<table>
<thead>
<tr>
<th>Volume Name</th>
<th>Total(KB)</th>
<th>Actual(KB)</th>
<th>Free(KB)</th>
<th>Granted(KB)</th>
<th>Min Free(KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>/ftp/.ingest/3</td>
<td>4439340</td>
<td>4439164</td>
<td>4439164</td>
<td>0</td>
<td>5000</td>
</tr>
<tr>
<td>/ftp/.ingest/2</td>
<td>4439340</td>
<td>4439156</td>
<td>4439156</td>
<td>0</td>
<td>5000</td>
</tr>
<tr>
<td>/ftp/.ingest/1</td>
<td>4439340</td>
<td>4332156</td>
<td>4332156</td>
<td>0</td>
<td>5000</td>
</tr>
<tr>
<td>/ftp/.ingest/4</td>
<td>4439340</td>
<td>4439132</td>
<td>4439132</td>
<td>0</td>
<td>5000</td>
</tr>
<tr>
<td>/ftp/.noncache/3</td>
<td>4439340</td>
<td>4439196</td>
<td>4439196</td>
<td>0</td>
<td>5000</td>
</tr>
<tr>
<td>/ftp/.ingest/9</td>
<td>7850488</td>
<td>7850296</td>
<td>7850296</td>
<td>0</td>
<td>5000</td>
</tr>
<tr>
<td>/ftp/.ingest/5</td>
<td>4439340</td>
<td>4439116</td>
<td>4439116</td>
<td>0</td>
<td>5000</td>
</tr>
<tr>
<td>/ftp/.noncache/1</td>
<td>4439340</td>
<td>4439196</td>
<td>4439196</td>
<td>0</td>
<td>5000</td>
</tr>
<tr>
<td>/ftp/.ingest/6</td>
<td>4439340</td>
<td>4439084</td>
<td>4439084</td>
<td>0</td>
<td>5000</td>
</tr>
<tr>
<td>/ftp/.noncache/2</td>
<td>4439340</td>
<td>4439196</td>
<td>4439196</td>
<td>0</td>
<td>5000</td>
</tr>
</tbody>
</table>

*Change*  *Dismiss*  *Refresh Display*
### SDMS SCHEDULING DESIGN

**Discrete Resource Displays**

![Discrete Resource Displays](image)

<table>
<thead>
<tr>
<th>Name</th>
<th>Min Limit</th>
<th>Max Limit</th>
<th>Limit</th>
<th>Total</th>
<th>Free</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB CONNECTIONS</td>
<td>0</td>
<td>25</td>
<td>25</td>
<td>1</td>
<td>24</td>
</tr>
<tr>
<td>UNITREE_CONNECTIONS</td>
<td>0</td>
<td>10</td>
<td>5</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>REQSrvr_CHILDREN</td>
<td>0</td>
<td>25</td>
<td>25</td>
<td>0</td>
<td>25</td>
</tr>
<tr>
<td>TRANSFER</td>
<td>0</td>
<td>6</td>
<td>3</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>INGEST</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>VALIDATE</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>PRIMARY</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>SECONDARY</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>TERTIARY</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>NOTIFY</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>EXPOSE</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>ORDER</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>STAGE</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>REQUEST</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>CLEANUP</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>TRANSFER_TSDIS</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>INGEST_TRMM_1A</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>VALIDATE_TRMM_1A</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>PRIMARY_TRMM_1A</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>SECONDARY_TRMM_1A</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>TERTIARY_TRMM_1A</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>NOTIFY_TRMM_1A</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
SDMS SCHEDULING DESIGN
Changes to DAACWare for I-SIPS

- Resource Manager will be used “as-is”
- Executive will require changes to support:
  - Creation of working directories from resource manager controlled pool of disk space
  - Changing directories to working directory before running job/steps
  - Adding script-level allocation and deallocation of resources
  - Improved interface to planning process, provide feedback on executive status
  - Capturing “Standard Out” and “Standard Error” streams and logging to the log server process (not clear we will do this)
- Scheduler to be used “as-is” (but may require some minor changes for new resource management policies)
- Current displays will be used “as-is”
- Planner process will be an evolution of the DAACWare planner:
  - Requires changes to support multiple executives and load balancing between primary and backup computers
  - May require some minor display changes
SDMS DATA MANAGEMENT DESIGN APPROACH

- Alter the approach to data management used by the DAACWare in order to accomplish the following goals:
  - Isolate the science processing from having to deal with resource management
  - Keep all data management in one location
  - Disk resource management only deals with transient space required for jobs
  - Simplify system design and structure, leads to maintainability
  - Higher performance
- For Data Management, two systems will be provided:
  - A new system, the Data Server (DS)
    - a rewrite of the Distribution Cache Manager (DCM)
    - new capabilities for archiving files as well as distributing
  - An existing system, ARCHER
    - an archive file management system used to read and write from robotic storage
    - used “as-is”, with small changes to work in HP environment
    - may need to develop new robotic control depending on robotics selected
SDMS DATA MANAGEMENT
Architectural Differences from DAACWare

**Notes:**
1. Separately managed area for input transfers
2. Separately managed area for product output and products must be re-ingested
3. Ingest writes to directly ARCHER
4. DCM gets files only by subscription

**Notes:**
1. Transfer and Ingest now combined
2. Ingest writes to directly Data Server
3. ARCHER exchanges files only with DS
4. Process reads and writes to DS
SDMS DATA MANAGEMENT DESIGN
High Level Requirements

- Data management (DM) is responsible for the storage of all product files and other files
- DM maintains a large working set of files on disk so that recently obtained/produced files are available for further processing
- DM is science processing independent; in other words, it only knows about files, not what they contain
- Client processes typically use two interfaces to DM:
  - Store a file in DM
  - Access a file in DM
- Other interfaces are available for keeping files on disk, for finding files, listing file attributes, removing files, flushing cache, etc.
- Files are stored in DM using names much like Unix
- All files are initially stored on disk
- A subset of the files are archived to tape
- Some tapes are maintained in a digital library, others are stored offline
SDMS DATA SERVER
General Concepts

- Data Server is disk cache manager for Data Management
  - All access to Data Management is through DS interface
  - ARCHER completely hidden from client applications
  - When a file is fetched, it is actually “linked” so no file copy occurs
  - When a file is stored it must be copied from the job directory to cache
- Data Server manages files:
  - File names can be anything, but we will follow traditional Unix convention “/a/b/c/…/file”
  - Some of these files are in DS Cache, some are in ARCHER, often in both
  - Location of file is transparent to client processes
  - Files are NOT stored on disk as “/a/b/c/…/file” (file names are only used to locate files in DS)
Rewrite, using DCM as a model

Current DCM is 6500 source lines of code, estimate rewrite to be similar in size

Will develop as Java Application:
  - Provides OOP approach which will improve readability (DCM is already somewhat OO in nature)
  - Large suite of foundation classes, reduces need for writing custom codes
  - Excellent support for multithreading which will boost performance
  - Simple Oracle database interface through JDBC

Some progress have prototyped (needed to prove Java a valid option):
  - Message Passing interfaces
  - Log system interfaces
  - JDBC interfaces (will be doing performance tests soon)
ARCHER OVERVIEW

ARCHER is a file storage management system originally developed for the V0 DAAC, now in operational use on several other systems. The purpose of ARCHER is to manage the storage and retrieval of large numbers and volumes of data files using digital libraries (e.g., tape robots).

It supports two basic operations:
- Get(unixpath,archerpath) - Retrieve a file from ARCHER
- Put(unixpath,archerpath) - Store a file in ARCHER

Tapes may be:
- On-line In a tape drive
- Near-line Robotically accessible, but not loaded
- Off-line On the shelf

Files stored to ARCHER are initially stored on disk, but migrated to tape as soon as possible. Files requested from ARCHER will cause the appropriate tape to be loaded and the file to be read.

The operators will be prompted to move tapes to and from shelf storage as needed.
ARCHER
RATIONALE FOR USE

- ARCHER is currently operational in several systems similar to I-SIPS
- ARCHER is available to the I-SIPS project for free (under agreement with the GSFC DAAC)
- The source code comes with ARCHER so we can quickly fix problems
- ARCHER uses a non-proprietary tape format (Unix tar)
- ARCHER was designed for an archive system environment:
  - Batch fetches and puts from tape for better throughput
  - Tape operations sorted for optimal throughput
  - Good operator interfaces for monitoring and controlling activity
  - Not integrated with the operating system kernel like many archive systems
- Direct access to ARCHER developers for problem resolution
SDMS DATA MANAGEMENT
ARCHER SOFTWARE ARCHITECTURE

These dashed boxes denote possible separate computer systems.

Client Application
Client I/F

Archive Server
Family Table & Dir Table

File Server
Cache
File Tables
Media Table

Media Mgr

Library Mgr

Display

Robot Server
Drive Server
Drive 1
Drive N

Drive 1
Drive 2
Drive N

Robot 1

Robot 2

Robot N

Display

Family Table & Dir Table

& Dir Table

Display

Display

Cache
File Tables
Media Table

SDMS DATA MANAGEMENT
ARCHER SOFTWARE ARCHITECTURE

Nov.30-Dec.1, 1999
ICESat SIPS Software CDR - Day 1
Page 88
SDMS DATA MANAGEMENT
ARCHER Library Manager GUI
SDMS DATA MANAGEMENT
ARCHER Media Manager GUI
SDMS DATA MANAGEMENT
Deployment Approach

• ARCHER is currently operational in several systems, with multiple types of robotics, on both SGI and Sun computers
• We have a version working without robotics on I-SIPS HP computers, however:
  – V0 DAAC has made recent changes that we want
  – New version in V0 DAAC operations, but still shaking out bugs
• DAAC is also planning to “shrink-wrap” the new version
  – Plan is to make it easy to install and provide good documentation
  – Only shrink-wrapped for SGI, but HP changes should be fairly limited
• For Version 1:
  – SDMS can operate without ARCHER, transparently to the GLAS science
• Early in Version 2 development:
  – We will rehost the new version of ARCHER
  – Have already proven we can port to HP environment
  – No significant code changes are expected
  – May have to add new code to control for the robotics (depending on robotics used)
PRODUCTION STATUS SYSTEM OVERVIEW

• Maintains the state of the I-SIPS system
• I-SIPS design is “data-driven”, so changes in Production Status indirectly cause other events to occur
• Actually just a data base, Oracle selected because of existing data management system needs
• Will provide some library routines for accessing database in consistent manner
• May also provide a set of small applications and/or scripts used for system monitoring and maintenance
• Implementation will be unique to the I-SIPS, but substantial reuse of DAAC DB layout is possible
PRODUCTION STATUS SYSTEM MAIN TABLES

- **Product Table**
  - One row per product per version in system
  - A product can be a well-known ICESat product, an internal temporary product, a control file, or even a static file
  - Contains overall characteristics of product, and high-level production management information for product (e.g., whether or not to archive/distribute the product)
  - Unique Product ID for each product

- **Granule Table**
  - One row per granule (one related set of science data, 1-N files)
  - Contains per granule information, including production status information
  - Unique Granule ID for each granule
  - Has *indexed* Product ID for quick lookup of granules by product ID

- **File Table**
  - One row per file
  - Contains per file information such as size
  - Unique File ID for each file
  - Has *indexed* Granule ID for quick lookup of files by granule ID
SDMS EXTERNAL INTERFACES
OVERVIEW

• The are five key external interfaces to the I-SIPS:
  – EDOS (Source of Level 0 data)
  – NSIDC (Source of Met data, destination for products, can special order products to replace lost files)
  – UTCSR (destination for SRS/GPS products, source of Orbit & Attitude)
  – SCF (Destination for subset of products, catalog updates, diagnostic and production info, source for QA info and special products)
  – ISF (Destination for L0/L1 products, source for instrument/spacecraft performance & analysis products)
SDMS EXTERNAL INTERFACES
DEVELOPMENT APPROACH

- EDOS and NSIDC interfaces are mostly well documented and understood
  - Will be implemented on I-SIPS per existing ICD
  - Developed for V1 but not really needed until V2
- SCF, ISF, and UTCSR interfaces will be developed
  - New development for I-SIPS
  - Needed earlier than interfaces to the other EOSDIS systems
  - Will develop cooperatively with SCF, ISF, and UTCSR staff
  - If possible, we will reuse EDOS and/or NSIDC protocols for exchanges
- For I-SIPS most external interfaces are extremely simple:
  - FTP used to either push or pull files to or from external systems
  - Usually, the presence of a special file is used to indicate data set(s) ready
  - In some cases, email is used for notification
  - Not expected to be a big development effort
  - Will attempt to use same protocol for all interfaces
SDMS
DEVELOPMENT SCHEDULE

• Version 1:
  – Design 11/29/1999
  – Coding 02/11/2000
  – Unit Testing 03/09/2000
  – Integration (Standalone) 04/14/2000
  – Integration (with GSAS) 05/19/2000
  • ARCHER integrated 7/15/2000
• Version 2:
  – Requirements Review 06/29/2000
  – Internal Design Review 08/03/2000
  – Coding 10/13/2000
  – Integration 12/15/2000
JOB DESCRIPTIONS/I-SIPS OPERATIONS CONCEPT

Mark Sherman
Raytheon ITSS
4500 Forbes Blvd
Lanham MD, 20706
email: msherman@pop3.stx.com
(301) 794-3046
OPERATIONS CONCEPT
JOB PAIRS

• Jobs usually run in pairs:
  – Planning Job
  – Processing Job
• The planning job determines that a production job can be run and starts it
• The production job does the actual work
• There are three types of job pairs:
  – Ingest
  – Science
  – Distribution
OPERATIONS CONCEPT
INGEST JOB PAIRS

• Ingest Planning Job:
  – Run periodically (e.g., twice an hour)
  – Looks in FTP area for PDR
  – Constructs control file and inserts in DM
  – Updates Production Status
  – Creates Job Table Entry

• Ingest Job
  – Gets Control File from DM
  – Transfers files from FTP to working directory
  – Processes files (if necessary)
  – Stores files into DM
  – Updates Production Status
## OPERATIONS CONCEPT
### SUMMARY OF INGEST JOB PAIRS

<table>
<thead>
<tr>
<th>Jobs</th>
<th>Freq.</th>
<th>Input Data</th>
<th>Job Processing</th>
<th>Output Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ingest Pred Orbit</td>
<td>1/day</td>
<td>Pred. Orbit from UTCSR</td>
<td>Validate Predicted Orbit, Determine Revs and Granules</td>
<td>Predicted Orbit, Rev Table, Granule Table</td>
</tr>
<tr>
<td>Ingest L0 Data</td>
<td>4/day</td>
<td>L0 PDS from EDOS</td>
<td>Validate L0 Data, Generate L0 Index Files</td>
<td>L0 data, L0 Index Files</td>
</tr>
<tr>
<td>Ingest PAD Data</td>
<td>1/day</td>
<td>PAD data from UTCSR</td>
<td>Validate PAD</td>
<td>PAD data</td>
</tr>
<tr>
<td>Ingest POD Data</td>
<td>1/day</td>
<td>POD data from UTCSR</td>
<td>Validate POD</td>
<td>POD data</td>
</tr>
<tr>
<td>Ingest Met Data</td>
<td>4/day</td>
<td>Met data from NSIDC</td>
<td>Validate Met Data, Subset Met Data</td>
<td>Subsetted Met Data</td>
</tr>
<tr>
<td>Ingest QA Data</td>
<td>6/day</td>
<td>QA from SCF</td>
<td></td>
<td>QA in database, and updated metadata</td>
</tr>
</tbody>
</table>
OPERATIONS CONCEPT
SCIENCE PROCESSING JOBS

- **Science Planning Job:**
  - Run periodically
  - Looks in Production Status to see if all necessary data available to start science processing
  - Constructs control file and inserts in DM
  - Updates Production Status
  - Creates Job Table Entry

- **Science Processing Job**
  - Gets Control File from DM
  - Transfers files from DM to working directory
  - Processes files
  - Stores files into DM
  - Updates Production Status
## OPERATIONS CONCEPT
### SUMMARY OF SCIENCE PROCESSING JOBS

<table>
<thead>
<tr>
<th>Jobs</th>
<th>Freq.</th>
<th>Input Data</th>
<th>Job Processing</th>
<th>Output Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create L1A Data</td>
<td>4/day</td>
<td>Level 0 data, ANC07, Predicted POD, Index Files</td>
<td>Create L1A Data (GLAS Exec)</td>
<td>GLA01-04, ANC06</td>
</tr>
<tr>
<td>Create Trend Data</td>
<td>4/day</td>
<td>GLA01-04, ANC06</td>
<td>Create Trend Data</td>
<td>Trend data, QA Flags, Plots, Statistics</td>
</tr>
<tr>
<td>Create ATM Data, (L1B &amp; L2)</td>
<td>1/day</td>
<td>GLA02, ANC07, Std ATM, Met Data, POD</td>
<td>Create L1B &amp; L2 LIDAR data</td>
<td>GLA07-11, ANC06</td>
</tr>
<tr>
<td>Create Elev Data, (L1B &amp; L2)</td>
<td>1/day</td>
<td>GLA01, GLA07 POD, PAD, Ref Orbit, Node File, ANC07, DEM, Surf, Tide, Region Flag</td>
<td>Create L1B &amp; L2 Elevation data</td>
<td>GLA05-06, GLA12-15, ANC06</td>
</tr>
<tr>
<td>Create Browse Data</td>
<td>6/day</td>
<td>GLAxx</td>
<td>Create Browse Data, (One step for each of 15 xx products)</td>
<td>GLAxx Browse</td>
</tr>
<tr>
<td>Perform Product QA</td>
<td>6/day</td>
<td>GLAxx, GLAxx Browse, ANC06</td>
<td>Perform QA, (One step for each of 15 xx products)</td>
<td>GLAxx QA Flags</td>
</tr>
</tbody>
</table>
OPERATIONS CONCEPT
DISTRIBUTION JOB PAIRS

• Distribution Planning Job:
  – Run periodically
  – Looks in Production Status for files to distribute (see note)
  – Constructs control file and inserts in DM
  – Updates Production Status
  – Creates Job Table Entry

• Distribution Job
  – Gets Control File from DM
  – Fetches files from DM
  – Processes files (e.g., HDF conversion)
  – Stores files and PDR to FTP Area
  – Updates Production Status

Note: some distribution planning steps look at FTP area for special requests from SCF or NSIDC for files
## OPERATIONS CONCEPT

### SUMMARY OF DISTRIBUTION JOBS

<table>
<thead>
<tr>
<th>Jobs</th>
<th>Freq.</th>
<th>Input Data</th>
<th>Job Processing</th>
<th>Output Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSIDC Standard Products</td>
<td>1/day</td>
<td>GLAxx (all), GLAxx Browse (all)</td>
<td>Convert GLAxx to HDF, Validate HDF, (one step for each of 15 xx products)</td>
<td>GLAxx HDF to NSIDC, GLAxx Browse to NSIDC</td>
</tr>
<tr>
<td>SCF Product Subset</td>
<td>6/day</td>
<td>(programmable subset of all products)</td>
<td></td>
<td>Product Subset to SCF</td>
</tr>
<tr>
<td>ISF Products</td>
<td>4/day</td>
<td>GLA01-04</td>
<td></td>
<td>GLA01-04 to ISF</td>
</tr>
<tr>
<td>UTCSR Products</td>
<td>4/day</td>
<td>GLA04</td>
<td></td>
<td>GLA04 to UTCSR</td>
</tr>
<tr>
<td>NSIDC Recovery Requests</td>
<td>Varies</td>
<td>Product Request Info, GLAxx, GLAxx Browse</td>
<td>Parse Request, validate all files exist, Convert to HDF and validate if necessary</td>
<td>Requested products to NSIDC</td>
</tr>
<tr>
<td>SCF Special Requests</td>
<td>Varies</td>
<td>Any products on ISIPS</td>
<td>Parse Request, validate all files exist</td>
<td>Requested products to SCF</td>
</tr>
</tbody>
</table>
I-SIPS TYPICAL DAY
Main Events

- Ingest 4 sets of PDS
- Produce Level 1A Products for current data
- Distribute L1A SRS/GPS to UTCSR
- Distribute L0 and L1A to ISF
- Ingest POD/PAD for past day N (up to two weeks earlier)
- Produce Level 1B & L2 products for day N
- Produce Browse products for all standard products
- Perform QA on all products
- Distribute programmable subset of products to SCF
- Create HDF products, validate, and distribute to NSIDC
- Distribute Browse to NSIDC
- Reprocess data based on updates (POD/PAD or Algorithm)
Example of what the processing timeline might look like (note “↓” = precedes):

- Ingest Predicted Orbit
- Ingest Met
- Ingest Level 0
- Create L1A
- Create Trend
- L1A to ISF
- GLA04 to UTCSR

- Ingest POD
- Ingest PAD
- Create ATM
- Create Elev

- Create Browse
- Perform QA
- Product Subset to SCF
- Std Products to NSIDC
- Requests to NSIDC
- Requests to SCF

Everything in here will be significantly later than creation of L1A because of delta time between sending GLA04 to UTCSR and getting POD/PAD back.
OPERATIONS CONCEPT
OPERATOR HANDLING OF JOBS

• Operators see a line on their displays for each job in system
• For jobs that run with no error
  – Operators need to do nothing
  – Disk space and resource automatically deallocated when done
• For jobs that fail
  – Operators make quick assessment as to cause by:
    • Examining status return value on operator displays
    • Examining log files in the working directory for the job
  – If cause is one that Operations Manual says can be corrected, operators:
    • correct error
    • remove old job (frees disk space)
    • restart job
  – Otherwise, SCF staff is notified to examine job, when completed ops staff:
    • remove failed job from the system
    • restart job if directed to do so by SCF staff
GLAS SOFTWARE DEVELOPMENT PROCESS

Gladstone Marcus
Bldg. 33, Room B209B
Raytheon ITSS
Greenbelt, MD 20771
email: marcus@icesat2.gsfc.nasa.gov
phone: (301) 286-3923
SOFTWARE DEVELOPMENT PROCESS GOALS

• Implement a Software Process to assure software quality.

• Establish Procedures that maintain the integrity of the software.

• Deliver the software on time.
SOFTWARE DEVELOPMENT PROCESS LIFE CYCLE

- Requirements Phase
  - Concept and Initiation
  - Requirements Development
- Design Phase
  - Prototyping
  - Architectural Design
- Implementation and Testing Phase
  - Implementation/Coordination
  - Integration and Test
- Acceptance and Delivery Phase
  - Acceptance
  - Delivery
- Sustaining Engineering and Operations Phase
  - Operations
  - Maintenance
## I-SIPS DOCUMENTATION TREE

<table>
<thead>
<tr>
<th>GLAS Standard Data Software</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Management Plan Volume</strong></td>
</tr>
<tr>
<td>GLAS Science Software Management Plan</td>
</tr>
<tr>
<td>GLAS Science Data Management Plan</td>
</tr>
<tr>
<td><strong>Product Specification Volume</strong></td>
</tr>
<tr>
<td>GLAS Science Software Requirements</td>
</tr>
<tr>
<td>GLAS Level 0 Instrument Data Product Specification</td>
</tr>
<tr>
<td>GLAS Standard Data Products Specification – Level 1</td>
</tr>
<tr>
<td>GLAS Standard Data Products Specification – Level 2</td>
</tr>
<tr>
<td>GLAS Science Software Architectural Design</td>
</tr>
<tr>
<td>GLAS Science Software Detailed Design</td>
</tr>
<tr>
<td>GLAS Science Software Version Description</td>
</tr>
<tr>
<td><strong>Assurance and Test Procedures Volume</strong></td>
</tr>
<tr>
<td>GLAS Science Software Assurance and Test Procedures</td>
</tr>
<tr>
<td><strong>Management, Engineering, and Assurance Reports</strong></td>
</tr>
<tr>
<td>GLAS Science Software Performance/Status Report</td>
</tr>
<tr>
<td>GLAS Science Software Discrepancy Reports</td>
</tr>
<tr>
<td>GLAS Science Software Engineering Change Proposal</td>
</tr>
<tr>
<td>GLAS Science Software Test Report</td>
</tr>
</tbody>
</table>

SOFTWARE DEVELOPMENT PROCESS METHODOLOGY

- Used the Yourdon and DeMarco Method for Designing the Software
  - Design Models of the system
    - Purpose:
      - To validate the system by presenting different, but consistent views.
      - Enables comprehensive review.
    - Process Model - Data Flow Diagrams
      - Shows the flow of data and its transformation.
SOFTWARE DEVELOPMENT PROCESS METHODOLOGY

• State Model - State Transition Diagrams
  – Depicts the important scenarios, and identifies the common threads in the system.
  – Other scenarios supported by Decision Tables.
• Structure Model - Structure Charts
  – Identifies the modules that will comprise the system.
  – Common modules identified. Duplication avoided.
  – Review the Models
  – Refine the Design
  – Write Pseudocode for each Module or construct Flow charts
SOFTWARE DEVELOPMENT PROCESS TOOLS

- Case Tool - WIN/Mac A&D
  - Implements the methodology (Data Flow Diagrams, State Diagrams, Structure Charts)
  - Applies constraints and allows validation
  - Creates a Data Dictionary
  - Generates Reports
- Drawing Tool - WinFlow
  - Allows Flowcharting and architecture diagrams
- Configuration Management Tool - Clearcase
  - Software Version Control
  - Parallel Development
  - Multisite access
  - Report Generation
SOFTWARE DEVELOPMENT PROCESS
PROCEDURES

• Design Review
• PDL Review
• Code Review
• Unit Test Reviews
• Testing
  – Unit Testing
  – Integration Testing
  – Acceptance Testing
SOFTWARE DEVELOPMENT PROCESS
PROCEDURES

• Planning the Builds
  – Build as major Units and Functionalities become available.

• Requirements Trace
  – Unit test and Integration test against the Requirements.
  – Requirements identified for each Subsystem.

• Load Testing to provide statistics for improving efficiency.

• Scheduling.
SOFTWARE DEVELOPMENT PROCESS
CONFIGURATION MANAGEMENT

- GLAS I-SIPS Software Configuration Management Plan, May 1999
- Each process category is tracked using a Central Automated Tracking Utility (CATU).
- Configuration Manager (CM) presents Change Requests (CRs) to the I-SIPS Change Control Board (CCB) for their consideration, and monitors approved CRs for the Board.
- CM is responsible for ensuring the integrity of all software.
SOFTWARE DETAILED DESIGN

Anita Brenner
Raytheon ITSS
Bldg. 33, Room B209D
Greenbelt, MD 20771
email: anita@icesat2.gsfc.nasa.gov
phone: (301) 614-5914
SOFTWARE DESCRIPTION COMPONENTS

• A Way to View the System
  – SDMS - The Data Management and Scheduling environment
  – Science Algorithms Software that runs under SDMS - Glas_Exec + 4 subsystems:
    • L1A Subsystem
    • Atmosphere Subsystem
    • Waveform Analysis Subsystem
    • Elevation Subsystem
SOFTWARE DESCRIPTION COMPONENTS
(continued)

– Utilities
  • A set of standalone, supporting programs that are required for production of GLAS Science Data Products (SDP).
  • Perform pre-processing or post-processing functions.
SOFTWARE DESCRIPTION FEATURES

- Includes all Science Algorithms based on ATBDs as submitted to ESDIS on 7/30/99.
- GLAS_Exec can create all products at once, or in steps. For example:
  - Level 1
  - Level 1B and 2 Atmospheric
  - Level 1B and 2 Elevation
- Modularity - orchestrated by GLAS_Exec
  - Selective calls can be made to subsystems or Processes within subsystems.
  - Allows selective processing and reprocessing.
SOFTWARE DESCRIPTION FEATURES
(continued)

- Each Subsystem is implemented as a Shared Library.
- Each Subsystem computes statistics for Quality Assurance.
- Error Handling is hierarchical. Subsystems will not stop the system directly.
SOFTWARE DESCRIPTION COMPONENTS
I-SIPS
I-SIPS SOFTWARE TOP LEVEL DECOMPOSITION

Level 1B and 2 Elevation
- Ingest
- Stage
- SDMS
- Sched.
- Archive

Utilities
- Reformat
- Browse
- QA

GLAS_Exec

Assessment
- Standard Range Corr
- POD Merge

Level 1B Range Distributions / Waveform
- Instr Corr
- Det Geoloc / Surface Type
- Calc WF Characteristics

POD Merge

PAD Merge

Level 1B and 2 Atmosphere (Cloud and Aerosol)
- Backscatter
- Boundary Layers
- Cross Sections
- Optical Depth

Level 1B and 2 Elevation
- Trop
- Tides
- Rough, Slope
- Std. spot Loc & Elev

POD Merge

PAD Merge

Met Merge

Geoid

Std. spot Loc & Elev

I-SIPS SOFTWARE TOP LEVEL DECOMPOSITION
SOFTWARE DESCRIPTION
GLAS Science Processing
SOFTWARE DESCRIPTION SUBSYSTEMS
Run Science Algorithms
SOFTWARE DESCRIPTION
Generic SubManager

Data from GLAS_Exec

PRE subsystem-specific computations → Science Algorithms → POST subsystem-specific computations

Wrap-up (granule and end of processing) → Write Products
SOFTWARE DESCRIPTION
Production Scenario

Science Algorithm Processing

Utilities
GLAS EXECUTIVE

Jeffrey E. Lee
Bldg. N159, Room E228
Raytheon ITSS
Wallops Island, VA 23337
email: jlee@osb.wff.nasa.gov
phone: (757) 824-1853
OVERVIEW

- GLAS_Exec is the Fortran90 wrapper surrounding the Science Processing Algorithms.
- Provides standardized initialization and control.
- Provides standardized error handling.
- Handles (nearly) all I/O – processes data one second at a time.
- Allows for partial processing and reprocessing scenarios.
REQUIREMENTS

• All standard data products will produced in an internal format (HDF conversion is a utility).

• Input and output products will be delimited by start and stop times.

• Full processing history will be available via metadata.

• Standardized messaging and error-handling using local ancillary files will be available to all subprocesses.

• Changeable parameters will be defined in local ancillary files.
REQUIREMENTS (cont.)

- Capability to fully and partially process and reprocess data with several different Scenarios, including:
  - One processing string to create GLA01 to GLA15.
  - One processing string that starts with a GLA05 input to produce GLA06,12,13,14,15.
  - One processing string that starts with GLA02 input and produces GLA07,08,09,10,11.
  - One processing string that starts with a GLA05 input to produce GLA06.
DESIGN AND CODING PHILOSOPHY OF GLAS_Exec

• GLAS_Exec implemented with extreme modularity and parallelism with many reusable subroutines. GLAS_Exec is just a state machine providing standardized services to the science algorithms.

• The code is very generic. Very, very few architecture-specific assumptions. (Have cross-compiled in Sun 64 bit environment). Exceptions handled with conditional compilations.
DESIGN AND CODING PHILOSOPHY OF GLAS_Exec (cont.)

- Design allows us to code one piece of software now, but allows the option of quickly and easily splitting in case of performance problems.

- Implemented mostly as a set of shared libraries, with emphasis on generic, reusable library routines. Currently 75% of code is implemented in shared libraries; 25% specific to GLAS_Exec.

- Interface between GLAS_Exec and SDMS is a simple Control file. This control file can also be hand-generated.
Rationale for Design Choice

- Science algorithms are independent of input/output data formats and mechanics.
- GLAS_Exec is independent of SDMS.
- Potential for significant reduction in I/O during both processing and re-processing.
- Forces a great deal of consistency among the subsystems.
- Design allows for great deal of code reuse and rapid development when we write companion utility, analysis, and assessment software.
- Life Cycle management is easier with a single executable.
GLAS_Exec Layers

<table>
<thead>
<tr>
<th>GLAS_Exec</th>
<th>L1A_Mgr</th>
<th>WF_Mgr</th>
<th>Atm_Mgr</th>
<th>Elev_Mgr</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>l1a_lib</td>
<td>wf_lib</td>
<td>atm_lib</td>
<td>elev_lib</td>
</tr>
<tr>
<td></td>
<td>anc_lib</td>
<td>cntrl_lib</td>
<td>err_lib</td>
<td>file_lib</td>
</tr>
</tbody>
</table>

GLAS_Exec is written using a layered approach:

**Foundation**
- `anc_lib`: Ancillary file readers and converters (6 modules)
- `cntrl_lib`: Generic control definitions and routines (14 modules)
- `err_lib`: Error handling routines (8 modules)
- `file_lib`: File opening/closing routines (3 modules)
- `platform_lib`: Basic variable type definitions, constants, & platform-specific routines (6 modules)
- `prod_lib`: Product file readers and converters (73 modules)
- `time_lib`: Time conversion routines (8 modules)

**Algorithm Layer**
- L1A_Mgr: Implemented science algorithms (will be detailed by respective developer leads)
- `l1a_lib`: L1A Processing
- `wf_lib`: Waveform Processing
- `atm_lib`: Atmosphere Processing
- `elev_lib`: Elevation Processing

**Manager Layer**
- Wrappers around the science algorithms (same breakdown as Algorithm Layer, 4 modules)

**GLAS_Exec**
- Wrapper around the Managers (20 modules)
STRUCTURE CHARTS

Main Executive Program
STRUCTURE CHARTS (cont.)
STRUCTURE CHARTS (cont.)

MainWrap

CloseFiles
1.7.1
STRUCTURE CHARTS (cont.)
STRUCTURE CHARTS (cont.)

Atmosphere Subsystem: Interpolation Section
STRUCTURE CHARTS (cont.)

Level 1B Waveforms Structure Chart
Elevation Processing Manager Structure Chart
STRUCTURE CHARTS (cont.)
STRUCTURE CHARTS (cont.)

```
WriteATM
  *1.1.5.1.1.6.6.1

  result_code
  fCtrl.out
  gla07_out

write_gla07
  *1.1.5.1.1.6.6.2

  result_code
  fCtrl.out
  gla08_out

write_gla08
  *1.1.5.1.1.6.6.3

  result_code
  fCtrl.out
  gla09_out

write_gla09
  *1.1.5.1.1.6.6.4

  result_code
  fCtrl.out
  gla10_out

write_gla10
  *1.1.5.1.1.6.6.5

  result_code
  fCtrl.out
  gla11_out

write_gla11
  *1.1.5.1.1.6.6.6
```
GENERIC PRODUCT GENERATION DATAFLOW

GLAxx

ReadData

GLAxx

ReadGLAxx

GLAxx_P2A

Pass_GLAxx

xxx_Mgr

GLAxx_A2P

WriteGLAxx

GLAxx
CONTROL FILE REQUIREMENTS/CAPABILITIES

A control file is used to specify what states GLAS_Exec should execute. Controls are provided to:

- Read one or more selected products, with boundaries delimited by start/stop times.
- Write any of the products with boundaries delimited by start/stop times. Multiple input granules may be required to complete an output granule.
- Process using only data over specific regions (land, ocean, ice sheet, sea ice).
- Execute combinations of individual science algorithms (or entire subsystems).
CONTROL FILE ADVANTAGES

- Control file can be completely generated by hand.
- All production rules are enforced via the control file generator.
- Changing production rules or running special-case scenarios requires no recompilation of the GLAS_Exec software.
SAMPLE CONTROL FILE – LEVEL1 PROCESSING

- TEMPLATE=end-to-end.template
- EXEC_KEY=2
- DATE_GENERATED=10 November 1999
- OPERATOR=jlee
- CYCLE=001 2423432 3423423
- REV=001 2423432 34242432
- REV=002 2423432 34234500
- REV=003 2423432 34234600
- INPUT_FILE=GLA00_inname.dat 2423432 3423423
- INPUT_FILE=ANC08_predict_orbit.dat 2423432 3423423
- INPUT_FILE=ANC07_const_err.dat 2423432 3423423
- INPUT_FILE=ANC07_const_glob.dat 2423432 3423423
- INPUT_FILE=ANC07_const_lo.dat 2423432 3423423
- INPUT_FILE=ANC07_const_wf.dat 2423432 3423423
- OUTPUT_FILE=GLA01_gran_name.dat 0 2423432 3423423
- OUTPUT_FILE=GLA02_gran_name.dat 0 3242343 4234322
- OUTPUT_FILE=GLA03_gran_name.dat 0 2342342 2342343
- OUTPUT_FILE=GLA04_gran_name.dat 0 2342342 2342343
- OUTPUT_FILE=GLA05_gran_name.dat 0 2342342 2342343
- OUTPUT_FILE=ANC06_inname.dat 2423432 3423423
- SURFACE_TYPE=ALL
- L1A_PROCESS=ALL
- WAVEFORM_PROCESS=ALL
- END_OF_CONTROL_FILE
PROCESSING/REPROCESSING SCENARIOS
L1A-Only

Shaded modules are not executed
PROCESSING/REPROCESSING SCENARIOS
Atmosphere-Only

Shaded modules are not executed
PROCESSING/REPROCESSING SCENARIOS
Waveform & Elevation

Shaded modules are not executed
PROCESSING/REPROCESSING SCENARIOS
Partial Elevation

Shaded modules are not executed
ERROR HANDLING

• The GLAS_Exec design attempts to make very few assumptions. Coding is purposely very generic. GLAS_Exec provides a standard error facility.

• GLAS_Exec handles ALL error termination conditions. Sub-systems are not allowed to directly terminate processing, but can indicate the occurrence of a “fatal” error.

• The error handler is designed such that the “fatality” of specific errors can be changed in a constants file without recompiling software.

• The error handler is designed such that the output frequency of repetitive, non-fatal error and status messages can be changed in a constants file without recompiling code.

• All error and/or status information is written both to stdout and the ANC06 file. Keyword=value format of ANC06 makes it easy to “grep out” specific information.
ERROR CONDITIONS

• Bad (or no) Control file – Contents are checked for errors. Most control file errors are fatal.

• Bad (or no) Constants files – ANC constants files are checked for errors. Errors are fatal.

• Missing data – all processing has been designed to handle missing data. Processing does not assume data are continuous.

• I/O errors – Read errors are considered EOF on the input file. Write errors are considered fatal. Operator must rely on metadata to determine if there is really a problem.

• GLAS_Exec returns a result code to SDMS. A non-zero result code will signal the SDMS to leave all temporary files in place and alert an operator that a problem has occurred.
STATUS

- GLAS_Exec runs.

- Code consists of:

  22,438 LOC in libraries
  4,592 LOC in GLAS_Exec

- Libraries have been proven to be sufficiently generic to write very simple product readers using only 138 lines of new code (per product).

- Coding on schedule. Currently coding Version 1 product definitions. Future milestones include:
  - integrating the science algorithms
  - reading remaining ANC products
  - performing time conversion
  - performing time synchronization with multiple input products
  - creating header and metadata info
CONCERNS

• To date, team has not had experience with target hardware, operating system, or compiler.
  STATUS – new hardware/software is on order and is expected before the end of CY99.

• To date, have not seen real data. Experience says that somehow it will be different than planned. (timing, format, …)
  STATUS – we will get data from ground testing. This should be in time that we are comfortable with what to expect in-flight.
Supplemental Information
PASS-THRU LOGIC

- !
- ! Read Synchronous Data
- !
- do until all records of all files read

- if (need GLA00 record) then
  - read GLA00
  - Convert from P(roduct) to A(lgorithm)
- endif
- !
- ! ALWAYS COPY PASSTHRUS
- !
- Copy like (passthru) P and A variables from GLA00 to GLAxx (where xx > 00)

- if (need GLAxx record) then
  - read GLAxx
  - Convert from P to A
- endif
- Copy like P and A variables from GLAxx to GLAyy (where yy > xx)
- !
- ! Execute Managers
- !
- do each Manager routine (based on control)
- for variables modified by the routine executed...
  - Convert from A to P
  - Copy like P and A variables from GLAxx to GLAyy (where yy > xx)
- enddo
- !
- ! Write Data
- !
  - Based on control, write GLAxx
PRODUCT/ALGORITHM DEFINITIONS AND FLOW

definitions

- types_mod
- GLAxx_prod_mod
- GLAxx_alg_mod
- A2P_GLA_mod
- P2A_GLAxx_mod
- xxx_Mgr_mod

kinds_mod

GLAxx_mod
- ReadGLAxx
- WriteGLAxx

public data structures

arguments

- subATBD Stub
- subATBD Stub
- subATBD Stub
- subATBD Stub
LEVEL 1A Processing ATBD

Peggy L. Jester
Bldg. N159, Room E214
Raytheon ITSS
Wallops Island, VA 23337
phone: (757) 824-2093
email: jester@osb.wff.nasa.gov
L1A PROCESSING REQUIREMENTS

- Perform Level 1A Processing as defined in the Level 1A Processing ATBD, Draft 1, 11/5/99
- Obtain location from Predicted Orbit
- Process engineering data first since the altimeter and LIDAR data may require temperature, current, voltage, or status values for processing
- Automatically handle flight software and instrument hardware configuration and mode changes during Level 1A processing
- Check telemetry packets for suspect data
- Collect data for trend analysis
- Collect instrument and spacecraft position and attitude data
- Additional requirements for the L1A LIDAR Product are found in the GLAS Atmosphere Data Products ATBD
L1A PROCESSING INPUTS

- Raw instrument telemetry data, time ordered into 1 second records, delineated by APID
- Predicted Orbit
- Conversion coefficients and constants
L1A PROCESSING OUTPUTS

- Level 1A Data
  GLA01 - Altimetry Data Product
  GLA02 - Atmosphere Data Product
  GLA03 - Engineering Data Product
  GLA04 - GPS/SRS Data Product

- Metadata (QA and other descriptive information about the data products)

- Processing Status
L1A PROCESSING DFD

2.1* L1A Altimeter Processing

2.2* L1A LIDAR Processing

2.3* Engineering Data Processing

2.4* Collect Instrument and S/C Position and Attitude

2.5* Create L1A QA Statistics and Trend Data

2.6 Get Predicted Location

L1A Alt In

L1A LID_In

L0_Pos_Att_In

L1A cntl_alt

cntl_PredOrb

predOrbit

predLocQA

pred_loc

Time

L1A Alt_out

L1A LID_Out

Eng_In

Eng_Out

L1A cntl_lid

PredOrbit

L1A Alt_out

L1A_altQA

L1A_LID_Out

L1A_LID_In

engQA

L1A_altQA

L1Acntl_lid

L1A-txt

Time

predLocQA

L1A_Pos_Att_Out

L1Acntl_lid

L1A_LID_In

Eng_In

Eng_Out

L1A(cntl_lid)

L1Acntl_lid

L1A(txt)

L1Acntl_lid

L1A(txt)
# L1A PROCESSING DECISION TABLE

<table>
<thead>
<tr>
<th>Var</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1A ALT input or code updates</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>L1A LIDAR input or code updates</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>Engineering input or code updates</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>Instrument and S/C Location and Attitude data collection input or code updates</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>Actions</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Process L1A Altimeter Data</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Process LIDAR Data</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Process Engineering Data</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Collect Location/Attitude Data</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Get Predicted Location</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Perform QA and Trending</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>
L1A DECISION CRITERIA

- An entire L1A process will be executed for any update related to that process.
- QA and trending will always be executed if a process is rerun.
- Predicted location will always be obtained if a process is rerun.
- Altimeter, LIDAR, and the location and attitude data processing is dependent on engineering processing.
- The Level 1A data will always be re-created from the Level 0 data.
- The altimeter, LIDAR, and the location and attitude data processing are not dependent on each other and can be executed in any order or concurrently.
L1A PROCESSING STATE DIAGRAM

Initial L1A State

L1AInput

Level1AComputations
Process Level 0 Data into Level 1A Data

L_Eng_proc
Process Engineering Data

L_Alt_proc
Process L1A Altimeter Data

L_LID_proc
Process L1A LIDAR Data

L_Loc_Att_proc
Collect Instrument and S/C Location and Attitude Data

L_GetPredLoc
Get Predicted Location

cntl_PredOrb

L1A_cntl_alt

L1A_cntl_lid

L_GetPredLoc

cntl_PredOrb

Final L1A State
L1A ERROR PROCESSING

- File I/O (Predicted Orbit File) - Critical
- Math Errors (mitigate with data checks) - Warning
- Missing data will be flagged but not cause processing to halt.
L1A LIDAR PROCESSING STRUCTURE CHART
POSITION AND ATTITUDE DATA COLLECTION
STRUCTURE CHART

L_Pos_Att_Coll

L1A_Pos_Att
pos_att_eng_chk
pos_att_eng_con
L0_Pos_Att
pos_att_err
L_chkposattdat

L_configposatt

L_posattqatrnd

L1APosAtt_out
PosAtt_QATrend
L1A QA AND TREND HISTORY STRUCTURE CHART
L1A PROCESSING DEVELOPMENT SCHEDULE

- Implement Engineering Processing - 42 Man Days - in Progress
- Implement the L1A Altimeter Processing - 42 Man Days
- Implement the L1A LIDAR Processing - 45 Man Days
- Coding and Unit Testing to be Completed by 4/14/00
L1A PROCESSING UNIT TESTING

- Unit Testing is included in the schedule
- Realistic simulated data or instrument test data will be used as input
L1A PROCESSING DEVELOPMENT
STATUS/CONCERNS

• 1st draft of ATBD in Review - designing to 1st draft
• Program to Generate Test Data in Progress
• QA and Trend Analysis not defined - defer to V2
  * Will gather detailed information during I&T of Flight Instrument
• Instrument and Spacecraft Position and Attitude Data Collection not completely defined - defer to V2
  * Need to finalize output format requirements and input specification
• Will update engineering unit conversion code after calibration of the flight instrument
Supplemental Information
L1A ALTIMETER PROCESSING DFD

2.1.1 Check ALT Packets

2.1.2 ALT Engineering Unit Conversion

2.1.3 Perform L1A ALT Data Monitoring

2.1.4 Collect L1A ALT Data for QA and Trend History
L1A LIDAR PROCESSING DFD

2.2.1 Check LIDAR Packets

2.2.2 LIDAR Engineering Unit Conversion

2.2.3 Perform L1A LIDAR Data Monitoring

2.2.4 Collect L1A LIDAR Data for QA and Trend History

L1A_cntl_lid

L1A_LID_In

LID_EUC_In

L1A_LID

L1A_LID_Out

L1A_lid_mon_data

L0_lid_pkt_err

L1A_lid_trend

L1AlidQA_data
ENGINEERING DATA PROCESSING DFD

2.3.1 Check ENG Packets

ENG_EUC_In

Eng_In

2.3.2 ENG Engineering Unit Conversion

L1A_eng

2.3.3 Perform ENG Data Monitoring

Eng_Out

eng_mon_data

2.3.4 Collect ENG Data for QA and Trend History

eng_pkt_err

engQA_data

eng_trend
INSTRUMENT AND S/C POSITION AND ATTITUDE DATA PROCESSING DFD

2.4.1 Check Instrument and S/C Position and Attitude Data

L1A_cntl_PosAtt
L0_Pos_Att
L0_Pos_Att_In
posatt_data_err

2.4.2 Configure Instrument and S/C Pos and Att Data for Output

L1A_Pos_Att_Out
PosAtt_QA

2.4.3 Collect Instrument and S/C Pos and Att Data for QA and Trend History

L1A_PosAtt_trend
L1A_PosAttQA_data
L1A_PosAttOut
L1A QA AND TREND HISTORY DFD

2.5.1 Compute L1A QA and Processing Stats
- predLocQA
- L1AaltQA_data
- L1AlidQA_data
- engQA_data
- L1A_PosAttQA_data

L1AQA

2.5.2 Compute Trend History Data
- L1A_alt_trend
- L1A_lid_trend
- eng_trend
- L1A_PosAtt_trend

L1Atrend_data