ASTER Science Data Processing Development: A Look Back

G. N. Geller, M. Pniel, and N. Dehghani
Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena, CA 91109-8099

***Abstract*-ASTER (Advanced Spaceborne Thermal Emission and Reflection Radiometer) is a 14 band, imaging instrument built by Japan and flying on the Terra spacecraft. The US ASTER Science Team is responsible for developing the algorithms and software for generating eight Level 1 standard data products. This paper discusses the approach used for that development and evaluates its effectiveness. The approach worked very well, resulting in timely software deliveries, few software problems, and products that met quality expectations.**

One of the key elements of this approach was to shield the algorithm developers from the complex and bureaucratic system environment that the software runs in. Another was a conservative schedule that was driven by the software rather than the algorithm developers, and vigorously enforced by management. And a third element was an independent test team that developed a variety of test tools, allowing rigorous, automated testing.

I. Introduction

ASTER (Advanced Spaceborne Thermal Emission and Reflection Radiometer) is an imaging instrument built by Japan and flying on the Terra spacecraft. It has three telescopes and a total of 14 bands. The visible and near-infrared telescope has three bands plus a backward-looking band for same-orbit stereo observations. The resolution is 15 meters. The shortwave infrared telescope has six bands with 30 m resolution, and the thermal infrared telescope has five bands and 90 m resolution. ASTER acquires about 600 60 km by 60 km scenes per day. Because of the high data rate the duty cycle is limited to 8% and scenes are acquired based on specific requests by the science team or other individuals.

Japan is responsible for the development and generation of the Level 1 products, while the U.S. ASTER Science Team (AST) and ASTER Science Project at JPL develops the various Level 2 products (the Level 3 DEM was developed elsewhere). The Level 1 products are shipped from Japan and archived at and distributed from the EROS Data Center (EDC) DAAC in Sioux Falls, SD. The DAAC also generates, archives, and distributes the Level 2 and 3 products, either routinely or by request, depending on the product.

This paper summarizes the methods used in development of the algorithms and code for the Level 2 products, and evaluates the success of those methods. It begins by discussing the basic problems that had to be addressed and describing the basic approach and development process that was chosen. Next, it discusses the effectiveness of the approach and process used, describes those components that worked well, and discusses the most difficult areas.

TABLE I
ASTER STANDARD PRODUCTS

Level	Product
1A	Reconstructed, unprocessed instrument data
1B	Registered radiance at sensor
2	Decorrelation stretch--VNIR
2	Decorrelation stretch--SWIR
2	Decorrelation stretch--TIR
2	Brightness temperature
2	Surface reflectance
2	Surface radiance--VNIR, SWIR
2	Surface radiance--TIR
2	Surface emissivity
2	Surface kinetic temperature
2	Polar surface and cloud classification
3	Digital elevation model (DEM)

II. Players

Many individuals and organizations were involved in the development of the U.S. ASTER data products. These included:

1) Algorithm developers (two at JPL, three at other locations)
2) Product Generation System developers and lead (JPL)
3) DAAC (EROS Data Center)
4) Team Leader, Project Manager, and System Engineer (JPL)
5) AST members (~12, in ~8 locations)
6) EOS Core System (ECS; dictates software environment)

This large number of geographically and culturally dispersed players required that care be taken to ensure good communication and working relationships.

III. Basic Problems To Address

In addition to a diversity of players there were a variety of other basic problems that needed to be addressed, including:

1) How to create high-quality algorithms
2) How to create high-quality code
3) How to meet schedule milestones
4) How to incorporate complex constraints and requirements on software
5) How to effectively divide responsibility between players and use their strengths

IV. Approach And Process

The basic approach selected was simple: let the Algorithm Developers (ADs) focus on algorithm development, by having the software developers (called the Product Generation System; PGS) focus on everything else. Thus the PGS developers handled all the constraints and requirements on the software. This included those imposed by the operational environment (toolkits, operating systems, interface files), those imposed on the products (metadata content and format, file format--which was HDF), and performance considerations (reliability, resource usage). Thus the ADs were shielded from the complex and somewhat bureaucratic morass of system constraints and requirements.

The rationale for this was twofold. First, the difficult task of learning such a complex system would be handled by a single group in a single location; this would concentrate and minimize the training required, as well as the confusion. Second, it freed the algorithm developers to do what they do best and have the best temperament for--that is, develop algorithms.

Additionally, the schedule was driven by the PGS. While the algorithm developers "bought in" to the schedule, they had a relatively passive role in meeting it. The PGS, with its ties to the project (which had commitments to the sponsor), was the active player on schedule matters and it structured the development process with the schedule in mind.

Fig. 1 provides an overview of the development process that was used. The first step was for the AD to create a software prototype and/or to describe the algorithm in a document; the specific method selected was chosen by the AD. The PGS then developed the software so that it conveyed the algorithm and was consistent with the system constraints and requirements. PGS then passed the code back to the AD, who then evaluated the performance of the algorithm, and updated it as appropriate. These updates were then passed back to the PGS, who updated the code.

This iterative process continued until the algorithm and code were nearly ready for delivery to the DAAC. The PGS then ran some pre-delivery tests in the development environment at JPL, and then repeated these tests in the DAAC environment. After passing these tests the code was officially delivered to the DAAC. The DAAC would install the code, which had to pass the same tests before it was made operational. Such repeated testing in different environments was made possible by the highly automated test scripts that had been developed by the test group.

V. Evaluation

A. How Well Did It Work?

This approach was very effective. Software deliveries met their functional and schedule milestones, and the quality of the software was high, as indicated by the very few code problems at the DAAC. Also, data products met the quality

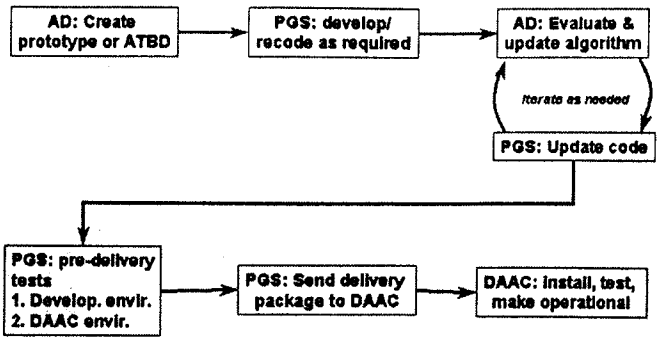

Fig. 1. Overview of ASTER science software development process.

expectations of the ADs, and all products have been released as "Beta" or "Provisional". The only caveat is that external factors -- delays in acquisition of the spacecraft orbit and in obtaining specific scenes needed for validation -- caused some delays in public release.

B. Valuable Elements Of The Process

The following were key elements in making the process work:

1) Focus: People could focus on what they do best. The ADs focused on developing the algorithm, and the PGS focused on the software, the schedule, and the many constraints and requirements imposed by the system.

2) Responsibilities: A clear demarcation of responsibilities meant there were few gaps or overlap among the players. If problems did arise they were addressed by the System Engineer. Consequently, things rarely "fell through the cracks", and there was no wasted effort or contention due to overlap.

3) Good relationships: The relationships between the ADs and PGS developers (and the DAAC) were excellent. This was due to a variety of factors, including support and encouragement from the team leader, the clear demarcation of responsibilities, and the personalities of the people involved.

4) Good communication: Communication between the ADs and the PGS was free from barriers -- significant because it was important that the ADs understand what the PGS developers were doing. This was largely due to the good relationships, but also because all ADs were software-oriented, and because the computer languages used were familiar to both the ADs and the PGS.

5) Schedule: Although the ADs bought into the schedule, the PGS and the project were the real drivers. Also, the project manager and the PGS lead relentlessly emphasized

the importance of schedule. Finally, the schedule itself was realistic for the available resources and the tasks at hand. Schedule was probably the single most difficult hurdle in the process, and the PGS developers and management were wise to be very conservative and to not attempt to do too much.

6) Tailored approach: Each AD could select how to communicate the algorithm and updates to the PGS.

7) Dedicated (single task) PGS test team: Thorough testing was critical to meeting schedule milestones. Having some members of the PGS team focus only on testing ensured that testing was always rigorous, and allowed the testers to take time to develop automated scripts and evaluation tools that enhanced the quality and speed of testing. Because the testers were part of the PGS team they had good relationships with the coders, which helped resolve problems quickly.

8) Definition of science requirement:. Adequate definition of the science requirements allowed the PGS developers to correctly capture the algorithm as code. Even better definition of the science requirements probably would have been helpful.

C. Difficult Areas

1) How many iterations? How many iterations should there be between the AD and PGS? The tendency is to iterate forever, because some additional improvements can always be made. Several strategies were used to overcome this. First, the schedule was very clear and strongly enforced, and each delivery had a well-defined end point. Second, most late requests for changes were handled by putting them into the next delivery.

2) PGS delivery schedule: Delivering software on time is always difficult, and ASTER data product development was no exception. Despite this, the code was mostly delivered on time, though such timeliness was never easy and required both will and effort.

3) Hesitancy to release: The ADs, for understandable reasons, wanted the initially released products to be of high-quality. Consequently, they sometimes hesitated to release the products to the public even as a "Beta" product. Better-defined criteria for Release Designations (e.g. "Beta", "Provisional", and "Validated") would have helped.

4) Beyond developer control: Some things were beyond the control of the developers and resulted in some delays in the public release of products. In particular, there were delays in getting proper scenes for validation.

VI. SUMMARY

Developing data product software is a difficult task involving many people and organizations. While the process will probably always be difficult, it can nonetheless still be effective and lead to the timely delivery of high-quality code that produces high-quality products. The elements that were most effective for ASTER were:

1) Letting people and organizations focus on what they do best
2) Creating a conservative, realistic schedule and completely committing to it
3) Having a dedicated (single task) test team