LESSONS LEARNED DURING ASTER DATA PRODUCT DEVELOPMENT

Gary N. Geller
On Leave of Absence from Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109-8099 USA
Email: garygeller888@earthlink.net

ABSTRACT

The Advanced Spaceborne Thermal Emission and Reflection Radiometer (ASTER) is a 14-band, imaging instrument built by Japan and launched on the NASA Terra spacecraft in 1999. The U.S. ASTER Science Team is responsible for developing the algorithms and software that generate eight Level 2 standard data products. This paper discusses the approach used for that development and evaluates its effectiveness.

The approach worked well, providing timely software deliveries that contained very few problems, and products that met quality expectations. One of the key elements of this approach was to allow people and organizations to focus on what they did best. Largely, this meant shielding the algorithm developers from the complex and bureaucratic system environment that the software runs in. Another key element was a conservative schedule 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.

1. INTRODUCTION

ASTER (Advanced Spaceborne Thermal Emission and Reflection Radiometer) is a 14-band, three-telescope imaging instrument built by Japan, and was launched on NASA's Terra spacecraft in December 1999. The visible and near-infrared telescope has three bands plus a backward-looking band for same-orbit stereo observations, with 15 m resolution. The shortwave infrared telescope has six bands and 30 m resolution, and the thermal infrared telescope has five bands and 90 m resolution. About 600 60 km by 60 km scenes are acquired each 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 and ASTER Science Project at JPL develops various Level 2 products (Table 1). The Level 1 products are shipped from Japan, then archived at and distributed from the EROS Data Center DAAC in Sioux Falls, South Dakota, U.S.A. The DAAC also generates, archives, and distributes the Level 2 and 3 products, either routinely or by request, depending on the product.

Table 1. ASTER standard data 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)

This paper first summarizes the context and problems associated with developing the ASTER Level 2 science algorithms. Next it explains the methods that were developed to address those problems. Finally, the success of those methods is evaluated, and the most important elements, as well as difficult areas, are discussed. It is hoped that these lessons learned are of use to future data product developers who may face many of the same problems.

2. PLAYERS

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

1. Algorithm developers (two at JPL, three at other locations)
2. Software developers and lead (JPL)
3. DAAC (EROS Data Center, South Dakota)
4. Team Leader, Project Manager, and System Engineer (JPL)
5. U.S. ASTER Science Team members (~12, in ~8 locations)
6. EOS Core System (dictates software environment; Maryland)

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

3. BASIC PROBLEMS TO ADDRESS

In addition to a diversity of players there were a variety of other 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 and changing constraints and requirements on software
5. How to cleanly divide responsibility between players and make best use of their strengths

4. APPROACH AND PROCESS

One of the basic philosophies of the approach that was devised was simple: let each group focus on what they do best. In practice this meant that the Algorithm Developers (ADs) focused on algorithm development, while the Software Developers (SDs) focused on everything else. The SDs handled all the constraints and requirements on the software, including 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). This shielded the ADs from the complex and bureaucratic set 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 concentrated and minimized 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, it allowed the schedule to be driven by the SDs. Although the ADs "bought in" to the schedule, they had a relatively passive role in meeting it. The SDs, with their ties to the project (which had commitments to the sponsor), were the active player on schedule matters, and structured the development process with the schedule in mind.

Figure 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 method was selected by the AD. The SDs then developed the software, incorporating the algorithm and following the many system constraints and requirements. SDs then passed the code back to the AD, who evaluated the performance of the algorithm, and updated it as appropriate. These updates were then passed back to the SDs, who updated the code.

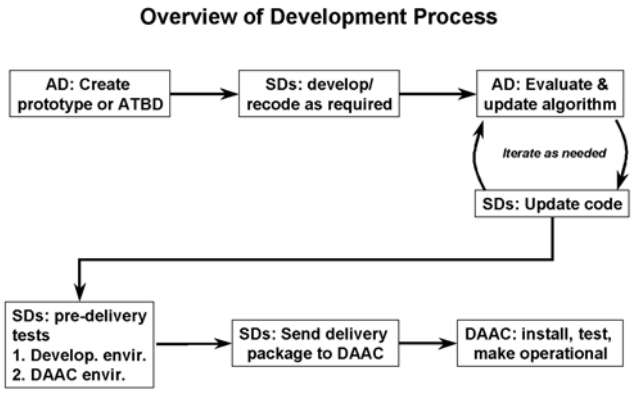

Figure 1. Overview of ASTER science software development process. (ATBD=Algorithm Theoretical Basis Document)

This iterative process continued until the algorithm and code were nearly ready for delivery to the DAAC. The SDs 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 then installed the code and reran the tests before making the code operational. Such repeated testing in different environments was made possible by the highly automated test scripts that had been developed by the test group.

5. EVALUATION

5.1 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 expectations of the ADs. While there were some delays in release of the data to the public, these resulted primarily from delays in acquisition of the spacecraft orbit and in obtaining specific scenes needed for validation.

5.2 Key Elements

The following were the key elements in making the process work:

1. *Focus:* People could focus on what they do best. Basically, this meant that the ADs focused on developing the algorithm, and the SDs 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 little wasted effort or contention due to overlap.

3. *Relationships:* The relationships between the ADs, SDs, 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 personnel who made a special effort to communicate well and maintain good working relationships.

4. *Communication:* Communication between the ADs and the SDs was free from barriers -- significant because it was important that the ADs understand what the SDs 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 SDs.

5. *Schedule:* Although the ADs bought into the schedule, the SDs and the project were the real drivers. Also, the project manager and the Software Development 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 SDs and management were wise to be very conservative and to not commit to more than they could realistically do.

6. *Tailored approach:* Each AD could select how to communicate the algorithm and updates to the SDs, allowing them to use an approach that worked best for them.

7. *Dedicated (single task) SD test team:* Thorough testing was critical to meeting schedule milestones. Having some members of the SD 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 SD 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 SDs to correctly capture the algorithm as code. Even better definition of the science requirements probably would have been helpful.

5.3 Difficult Areas

The overall effectiveness of the approach could not, unfortunately, remove all difficulties. The major difficult areas were:

1. *Software 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.

2. *Number of iterations:* How many iterations should there be between the AD and SDs? 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.

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.

6. 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 result in 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

7. ACKNOWLEDGMENT

The research described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.