AWIPS II Technical Interchange Meeting

Primary audience: GOES-R Proving Ground
Time: 2 PM to 4 PM EDT
Date: September 17, 2010

An audio recording of the call is available.

Notes prepared by Jordan Gerth

Attendance: Jason Burks, Kevin Fuell, Kevin McGrath, Eric Lau, Deb Molenar, Gene Petrescu, Matt Smith, Gary Jedlovec, David Plummer, Phil Ardanuy, Jim Heil, Bill Ward, Bill Sjoberg, Bill Campbell, Ed Mandel, Jordan Gerth, Kaba Bah, Doug and Frank Griffith (Raytheon)

Introduction: Ed Mandel, informs all about the intent of the GOES-R Proving Ground and what the call participants have accomplished so far looking at the software. General questions have come up, and this introductory TIM serves to answer them. We will be walking through the topics that were provided with the work assignment. Subsequent TIMs will be scheduled to address the questions and topics which result from this meeting.

> *Current Code Overview*
> A summary and question session on the current code delivered with the AWIPS Development Environment (ADE), particularly pertaining to end-to-end satellite imagery ingest, processing, storage, and display, is essential for understanding the architecture of the new software.

We need to know and understand an end-to-end analysis of the current code, looking at how the operational products are entered into the system, how they stored in the database and metadatabase, and displayed on the screen. We are going to be expanding the number of satellite imagery and products for NPP and GOES-R.

Basically in AWIPS II, all the different data types, information about that data type, everything that happens to it is encapsulated in a plugin. For satellite, there is a plugin currently tailored toward the GINI format. Other plugins would have to be used for other satellite data types. A McIDAS plugin has been developed at Raytheon. Basically, the GINI format is a binary product format for satellite that comes over the Satellite Broadcast Network. It has a header and a body. It is a compressed format. The decoder can handle both compressed and uncompressed data. The raw data is geolocated and projected onto a standard projection. The header data describes that projection. In that plugin, there is a Java class that decodes it and extracts the metadata out as well as the basic data record. It does create a coverage object which is then put into a field tools style geometry object for doing spatial queries. That is a general thing. The metadata that is in the header is converted into the metadatabase. The binary data itself, depending on the resolution and the size, is tiled by JAI (Java Advanced Image) library into an automatic number of levels. The one-kilometer GOES visible imagery has 5 levels of quad tiling to it. Some of the smaller products are not tiled. So if the data has a higher resolution, the JAI would automatically deal with that. The data is written to a HDF5 file organized by hour. The metadata goes into a

postgres database table along with the spatial information. There is a data URI that references that particular area file with the HDF5 data.

Standard projections are basic map projections. The data is not reprojected. Most of the data is in a Lambert Conformal projection.

There is a lot happening on the display side. The data can be stitched together to create a single area of coverage. You could look at the satellite plugin and see that the metadata is all defined in one record. They use annotation. It is the satellite record Java class that defines the metadata. The annotations define the pieces that go into the metadata record in the database. Annotations define what the data URI becomes and how that particular data is referenced by the display side.

There are many data types which AWIPS II handles. Each of those types is handled by its own plugin. However different kinds of point data are basically combined into one plugin.

The LDM (Local Data Manager) has been modified by Raytheon somewhat so that it sends messages to the ingest side whenever a file comes in, but to facilitate testing and development, there is a manual dropbox, which data can be placed without routing it through the LDM.

One complication is that the routing is controlled by the WMO header and there is an xml file with each plugin which has the WMO filters appropriate and specific to the product it can decode. There is a satellite ingest xml. That may need to be modified if you are making up your own name.

There was a McIDAS plug-in that has been developed by Raytheon for testing and other purposes, but not for delivery with AWIPS. It would not take too much to turn that into something that could be included. The only satellite plugin currently handles GINI format (as discussed previously). Dave Plummer notes that NCEP is working on a McIDAS plugin that will be delivered operationally. The capability would be needed by OCONUS sites. They have a box which converts GINI into AREA currently, and a McIDAS plugin would eliminate the need for external conversion.

Action: Work to collect, and then distribute information on the McIDAS AREA plugins currently developed by Raytheon, NCEP, and SPoRT so that a unified decoder can be delivered operationally to support the experimental product ingest.

In the AWIPS menu system, for satellite, there is a menu reference for the various products that are available. If you are interested in infrared imagery, there would be a reference. That menu item would have the latest time from the metadatabase. It uses the data URI to go to the HDF5 file and reference the data. There are several activities that go on simultaneously. What happens is that the base map that you want to display it on is used to convert the tiled data into that projection. It creates the corners of that tile and how it would go onto that projection. The data itself it treated like an OpenGL texture. Whatever format it is in, it is loaded into the graphics card as a texture. It is rotated around and is forced into the display projection. Reprojection occurs on the graphics card. The other thing that happens is a shader program is loaded when CAVE starts. It maps the color table to the texture/pixel values. The colors for display are

stored on disk as xml structures. It is a standard RGBA format. Along with that menu item, the xml file also describes the color table. It gets loaded as a one-dimensional texture.

The other aspect of the display is that tiles cover the display. If you zoom in and out, you get different resolution tiles. Tiles are loaded as you pan. There is constantly activity going on between the display and the repository via the graphics card.

A lot of the configurability was handled in flat text, pipe-delimited files that were used to set value mappings, set legends, and make menu items in legacy AWIPS. We have been going through and looking at some of the xml to find out, for example, exactly how the legends within the viz are set. They are hard-coded into the viz as a public static class.

Basically, there is a series of classes to handle conversions between raw value and the infrared temperature, for example.

Action: Raytheon needs to further describe how the legends are created and displayed. Information on the shader language and process is also important. The maximum and minimum values are parameterized when the data selection occurs, and that controls what you see on the screen as far as color.

Action: Determine exactly at what point in the ingest and display process is the conversion between bit space and value space performed. Does/could the interpolation done on the graphics card lead to a false return value?

> *Data Storage Architecture*
> Discussion is required to determine if the current file and directory structure is suitable for current and expandable uses of satellite imagery and products. Also, it is important to assess whether the current file naming convention based on the ingest time is useful, and if it is desirable to have the data bins completely decoupled from the metadatabase.

In this section, we discuss whether the current file and directory structure is sufficient. Right now, hourly bins based on ingest time are used.

Just recently, because of a request from the WDTB (Warning Decision Training Branch), Raytheon has changed the HDF5 binning to be by data time. Everything will now be in data time. The files themselves have that hour (data time) in the file name. If they are taken out, you know what you have. As far as the coupling with the metadata, the concept of the data URI is basically a fundamental aspect of the way that works. The main purpose of the metadata is for discovery of the data (for creating catalogs, volume browser, etc.) The record class defines what the metadata item becomes. There is a connection between the two and that is the data URI. It is the annotations in the record class that actually define that. Plugin and time are standard in all of the data URIs. After that, it is basically a progressive refinement of the data. You can see that structure in the CAVE's classic data browser. The first level lists the satellite products that have come in. It breaks it down into Alaska, Hawaii, etc. It goes further into the channel. That structure is basically met with the annotations in the record class.

We have noticed an unbalanced interaction between the repository and metadatabase. The entire cycle is now locked in a transaction. It is a relatively remote instance when information is written to the metadatabase without a coexisting data set using the concept of a transaction. Between the postgres database and the HDF5 bins, the transaction sequence is that data is first written to the HDF5 bin, then it goes ahead and makes the metadata record. If the metadata record fails, it does not really hurt anything aside from an abandon HDF5 bin. It hurts if you have a metadata record without an HDF5 bin, however.

Action: More information on this process. More broadly, we need to identify fail points in the software. Memory management and execution time should be closely watched to not impact the rest of the system. More investigation is needed to determine if the JAI libraries could be problematic with really large datasets.

Data Format Architecture

It is necessary to understand where the dependencies in the data bin array lie between the storage, metadatabase, and the display. The mapping of colors to data bin values for display is central to adequately handling true color bit depths without image combinations.

The first thing to look at is the GRIB2 plugin. The GRIB2 data is stored as a floating point in the HDF5. That is what is loaded into the card as a texture. The card can support multiple data types. There is not inherent limitation with 24-bit. You would have to create a new plugin to support a different type.

Each channel in GOES-R is going to be approximately 12 bits. The graphics card does the shading. The shader language routines are very small. You could come up with a high performance theme where you have the RGB values mapped dynamically. It allows you to have control and manipulate. The other approach is to combine that data in the main processor. The shader object is optimized for vector level processing. Without a whole lot of changes, that could accommodate the additional bits and the compositing.

Action: Work with Raytheon to determine how the shader language works with the graphics card for displaying satellite imagery at greater than 8 bits, as well as compositing multiple images.

The alpha channel controls each layer independently. That is an important advantage. There are people doing intensive image analysis and distinguishing between cloud and ice in Alaska Region, and using a third party tool for performing interactive image analysis. The current capabilities are a flavor of what could be done. The imaging widget is all dynamic and instantly applies to an animated loop. It controls the display and there are algorithms tied into it which could be used for this.

The color tables are in xml. There is no real limitation to the number of values that can be rendered. We can show in excess of 256 (8-bit) individual colors via expanding the xmls to contain an infinite in the number of colors.

Action: Beyond the current capabilities, discussing with Raytheon the exact process in linking colors to value mappings and defining a maximum and minimum for the displayed range. Can more of this be defined in the configurable xml?

Metadatabase Architecture
The use of metadata stored in the Postgres database is not well documented and understood. This knowledge is necessary for expanding on currently implemented components and take advantage of the mapping capabilities (understand loss of supplementary mapping files).

The big need-to-know item is exactly what components of the raw data is stored and used in the display. Incoming data is stored natively in the HDF5 when it arrives. As far as the value in the additional information, the satellite metadata has a lot of items in data record that is really not used. There is a lot more metadata than is necessary to display it or catalog it.

Action: Determine what from the metadatabase is actually used.

The design concept is such that the intention for metadata is for use in discovery purposes. If you have a data URI that maps directly to the HDF5, you do not need to go to the metadata to work with the data. Complexities will require more metadata. Some algorithms may need that. Design level thinking is required. For calculations or other things, it should be in the HDF5 itself. You can set structures in the HDF5. That probably makes more sense in most cases.

They used to put the decoded values in a metadatabase and discovered performance issues. For example, now all that is in the metadatabase is enough information to reference the METAR. The actual data resides in an array structure in HDF5.

Action: Based on our future expectations from NPP and GOES-R, identify what needs to be handled in the metadatabase and what needs to be stored in the HDF5.

For instance, suppose we want to maintain the individual times of the polar passes in creating a composite image, and are really trying to bring more metadata out front on the viz side. Do we store the pass information completely in the repository, including the time, which is already in the metadatabase for discovery? As we get more into evolving AWIPS II, user configurability, image blending and compositing, on the polar and geostationary satellite side, are paramount. There is quite a bit of blending already going on in image subtractions.

Suppose we create a derived product pattern that cuts across data types. You can have basically combinations like that which are calculated in Python and under forecaster control. If you look in the derived parameter scripts, an example is the water vapor divided by infrared product which is calculated. It is how it is envisioned that this kind of thing be done: in real time with the display and using numPy behind the seen.

There are also image combination actions combining GOES-EAST and GOES-WEST and updating the data/display correctly. That is built into the satellite mechanism. That is done in the satellite display resource.

Sample project: Using the display side to combine polar and geostationary data dynamically to create a maximum coverage, maximum resolution product as part of the display. Raytheon says this is probably within the range of what we can do by relaxing the time matching, or using a threshold in the time matching.

Action: Developers should talk about what is done now with image blending. Subsequent TIMs should address capabilities of the graphics card and the PG partners should outline a strategy of how compositing and image subtractions should desirably work.

Action: Submit any specific questions. Raytheon could have a developer help with answering the specifics. All participants need to have the next step in the knowledgebase. There is not much in the way of documentation. It comes down to digging into it and sometimes you need the developer to explain why things are the way they are at the design level.

The issue of looping and loading: the loop moves forward in time without all frames loaded. There has been progress in this area.

> *Image Blending and Use of Alpha Channel*
> A strategy for the implementation of three-way image overlays with equal image blending could be used as a proxy for easily maximizing the function of true color (24-bit) displays.

CIRA has had some interest in compositing up to 32 bits. We have already established that the blending is done through the graphics card cores. If the intent is a highly interactive system, that is the way that it needs to be done.

It is not preferable to do things when the data arrives. Performance outside of the graphics card is not efficient. One of the more challenging algorithms is the cloud height algorithm. That runs very well dynamically. You are loading a set of gridded data at the same time you are running a complex algorithm, displaying the result as you move the mouse around. It is a combination of the graphics card and what goes on in the display software.

The environment data package loads virtual layers. You see sample values. The sampling is intended to drill through all the layers. However, there is radar sampling that includes the topography data and the radar data so that you get the height above ground. The infrared imagery sampling combines different data types. The mechanisms are there and should be flexible enough to do what is necessary for satellite compositing.

> *Generalization of Satellite Decoder*
> Can the Satellite Decoder be generalized to handle McIDAS AREA files, since a bulk of the GINI format is based in AREA? Can we break free of the current 8-bit standard?

The bulk of the GINI format is based on the McIDAS format, though it is confined. Is it possible to develop a one-size fits all plug-in? It is reasonable to rethink the satellite plug-in from first principles to come up with a more general approach which could accommodate a category that

contains only one plug-in. Variety could be accommodated by configuration. It may be possible to come up with a McIDAS plug-in that reads GINI as part of it. The McIDAS format is more expandable and flexible.

In terms of an acceptable transmission format, netCDF4 might be an appropriate format, with the CF conventions. That format can be used for different types of data. At some point in the future of development, we will have to think about transitioning to a more universal format. We have also been discussing the use of the HDF EOS format. More design work is necessary.

The intent of the groups is similar. We are looking to build a set of tools. You can accommodate advanced compression and a very flexible schema structure. With the CF metadata, you can standardize the naming convention. There is not a very good standardization of parameter naming at the current time (for example, between point data and grid data).

Part of the design tradeoff is what from the raw data goes in metadata and what goes in HDF5. If it is used dynamically for display, it is better to have it in HDF5. If you have a structure associated with a record which has some of the metadata it in, that is preferable. The metadata is really designed for discovery and that is it. It is not intended to be accessed during the display.

In AWIPS I, there were complex, specific structures in the netCDF3.

If we generalize our format structure, we may get to the point where we can deliver a data bin without going through a decoder, but not desirable. You need to create notification so that all the descriptions and displays can get updated, as well as making the data discoverable. It is necessary to have a metadata record in the database. It is important in building an enterprise system to have all that. You could deliver data for scientific purposes and then have a tool like what WDTB is making which generates metadata. Let the tool adapt it to the running system.

> *Envisioned Implementation of Plug-ins*
> A consistent, organized approach is desired in formulating and implementing plug-ins for the software, not only to minimize conflict between builds, but between other plug-ins. How can this best and most efficiently be done without interfering with the operational, essential components of the software and recoding with subsequent releases?

If we develop a plug-in, what is the best way to implement it without interfering with operational, essential components of the software while limiting recoding in subsequent releases? NCEP has experience with both display and ingest-level plug-ins, independently of Raytheon. The plugins are decoupled from each other well enough to make that possible. You do not want to lag from the release too much if you are using a lot of the infrastructure.

You can have separate decoder strings put in one plugin. Most of them have one decoder class that is oriented toward the data that they have. If the metadata is common, you could accommodate different ingest types with multiple decoder paths.

Action: Developer involvement is required on how to implement a new or generalized decoder, and identify any interference with other classes which may result.

The ingest pipeline can be controlled. You can setup how many threads and pipelines are desired. Pipelines are in categories, which specify how many threads are allocated. It is in the xml file for the EDEX. There is a pipeline for text data which is separate from gridded data. There is a generic pipeline the majority of data uses. There are currently three or four pipelines. The pipelines have a configured number of threads. It is all configurable. It is possible to add a pipeline for new satellite data and keep the thread count low so that it does not interfere with existing satellite data.

Threads are balanced between memory and execution. It is hard to balance the CPU and the memory. There are three JVMs (Java Virtual Machines). There is a file which configures which plugin goes into which JVM. Each of them has a custom amount of memory specified for them. Running on a cluster, this can be optimized. If you hit the system with a lot of data, the CPUs should not be used entirely ("maxed out"), but use as much as possible, without going into swap. It is constantly adjusted throughout the development.

Operating system wise, backward compatible with AWIPS I is important. Raytheon research and development is working exclusively in 64-bit. A lot of libraries are compiled against 32-bit. Raytheon has locally modified some of the libraries.

Action item: Acquiring, building, and developing in 64-bit at Proving Ground sites.

GRIB2
Expandability and flexibility of the GRIB2 implementation is desired for a larger delivery of data in this format, including satellite imagery and products.

There is some configuration that goes with ingesting GRIB2. The NCEP GRIB2 decoder is what it used. Support for all the products that NCEP puts out. NCEP says that the GRIB2 decoder is supported against the WMO standard. NCEP would need to be more involved with ingest of satellite data with GRIB2. It would automatically be picked up with AWIPS II as long as they are using the updated decoder. There have been some issues with the large gridded data and using JPEG2000 compression. Size limitations exist. GRIB2 has different compression formats.

Action: We need to find out the specifics of ingesting a local model into AWIPS II. It is not too difficult and should be able to be accomplished at the configuration level.

Localization works through a data store on EDEX which contains the localized data for a site. Mainly for the GFE (Graphical Forecast Editor), there is special processing required. EDEX is run for a particular site. Kevin Johnson has been working on a task order. This area has been changing and evolving.

Action: Obtain and distribution the latest documentation on localization.

For maps, menus, and color tables, it is dynamic. For example, you could have one site connect to another and everything would look like the other office without having to move any data around. On the server side, it is a little more difficult, because of the GFE. You usually have to

do a single localization. Most of the configurability is handled in xml. Some of the local library files are preserved from AWIPS II. For hydrology, things work a little differently.

Action: Gather specific topics based on the results of what came out of this TIM.

Action: Submit items which each participant on the call is working on or concerned about.

Action: Submit items which require a design-oriented discussion. Think long term.

Subsequent TIMs will likely be held in Omaha, in person. The next TIM will be in four to six weeks or beyond, depending on planning and group submissions.

There has been no recent action in governance.

Initial draft: Jordan Gerth (CIMSS/SSEC) September 19, 2010