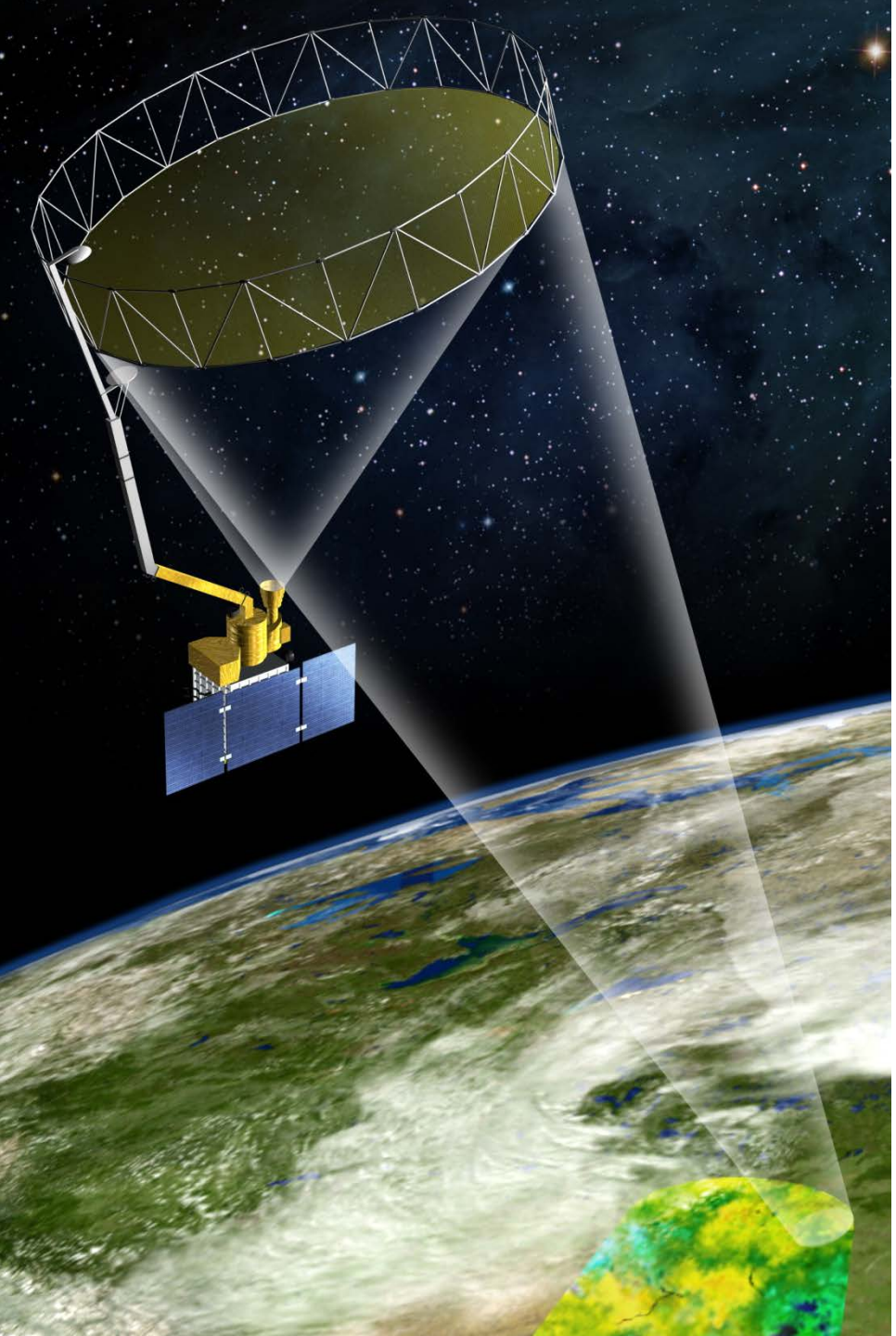


The SMAP Dictionary Management System (DMS)

Kevin Smith

Jet Propulsion Laboratory, California Institute of Technology

DMS Lead Developer

Christopher Swan

Jet Propulsion Laboratory, California Institute of Technology

SMAP Flight System Engineering / DMS System Engineer

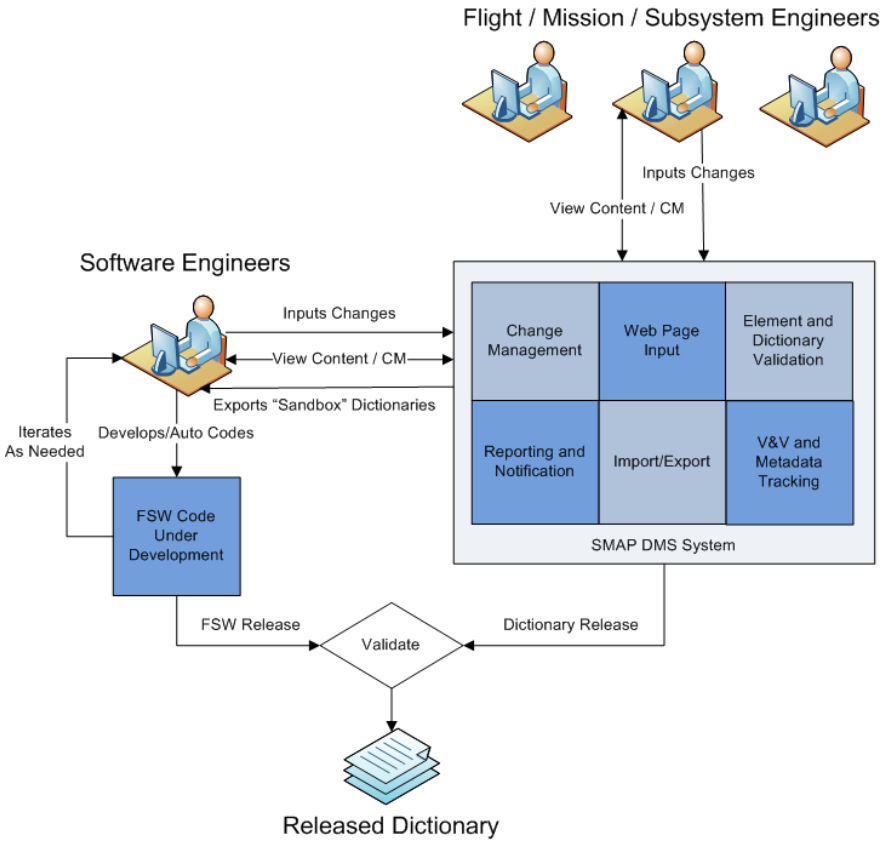
SpaceOps2014

May 6, 2014

http://smap.jpl.nasa.gov

What Is DMS?

- Dictionary Management System (DMS) is a Web Driven Database that manages the SMAP dictionaries.
  - Ruby on Rails application with MongoDB database
- Inputs come via dynamic web forms and XML/TSV files
- Exports information in XML (and limited TSV)
- Aggregates information and validates it based on schema (and validation rules beyond schema)
  - This includes dictionary interdependencies (i.e. all parameters and linked the command to change them and the telemetry that includes their values)
- Manages changes an applies them to targeted releases (based on approval)
- Produces a variety of reports which can automatically run and send the results to specific users via email
- Collects and integrates a variety of ancillary information about dictionary elements
  - Including detailed V&V results and meta data like FSW developer/V&V assignment, etc.
- Allows the building of sandbox dictionaries for development/test
- Adaptable to different schemas – can support schema migration of existing dictionaries
- Changes to the dictionary are tracked per element (and every change is logged)



- Metrics (February 2014)
  - ~32,000 individual change requests processed
  - 11 Official Dictionaries Released
  - Hundreds of Sandbox Dictionaries Current Dictionary (R4.0.0.2) has 10,754 discrete elements (commands, channels, etc)
  - ~9000 V&V results recorded (by dictionary element)
  - One schema migration performed

Software Requirements or Interface Specification?

- One thing that was observed early in development on SMAP was that the purpose of dictionary elements can change over their life cycle.
  - Most dictionary elements start their life cycle as specifications to software to implement.
  - Once they have been implemented they serve as a detailed, machine readable, interface document between flight and ground software.
- In an effort to streamline requirements sources the SMAP project decided to treat all dictionary content as requirements.
  - This centralized the specification of many detailed aspects of the design in a system which provided configuration management and a machine readable output.
- As development progressed the dictionary content is iterated between the stakeholders and migrates to an interface specification on release (of the dictionary and corresponding code)
- While some aspects of the SMAP Dictionary (FP/Parameters) never become part of the flight / ground interface they are requirements on software and are closely coupled with that interface.
  - A hidden benefit of this was the formalization of the formats of these specifications (in xml) which allowed a number of efficiencies
  - For example: The software developer responsible for developing fault protection discovered he could auto code parameter definitions based on the parameter specification. (~1000 parameters)

Expanding Role of the Dictionary

- The "Dictionaries" on SMAP serve to manage a large amount of detailed design specification
- This is all handled in a machine readable format (xml) that both FSW and GDS ingest.
  - These formats are called the dictionary schema.
- Historically (at JPL) this has not included Fault Protection or Parameters, and Alarms were managed separately
  - SMAP chose to integrate this information together because of the interdependencies and to streamline workflow.
- Major Stakeholders of the Dictionaries:
  - Flight Systems (all)
  - Flight Software (all but alarms, ground derived channels, data product viewers)
  - Mission Systems
    - GDS for ground derived channels/Data Product viewers
    - MOS for alarms (as well a general interest overall)
  - Individual subsystems/disciplines involved in the specification of their dictionary content.

SMAP Dictionaries

MSL/MSAP/MER

- APID: (Application ID)
  - Default data priority, streaming product limits
- EH&A (Channelized Telemetry)
  - Also includes Ground Derived Channels
- EVRs (Event Messages)
- Data Products (Files produced by Spacecraft)
  - Also specifies ground software to display them
- Commands
  - Hardware and Software

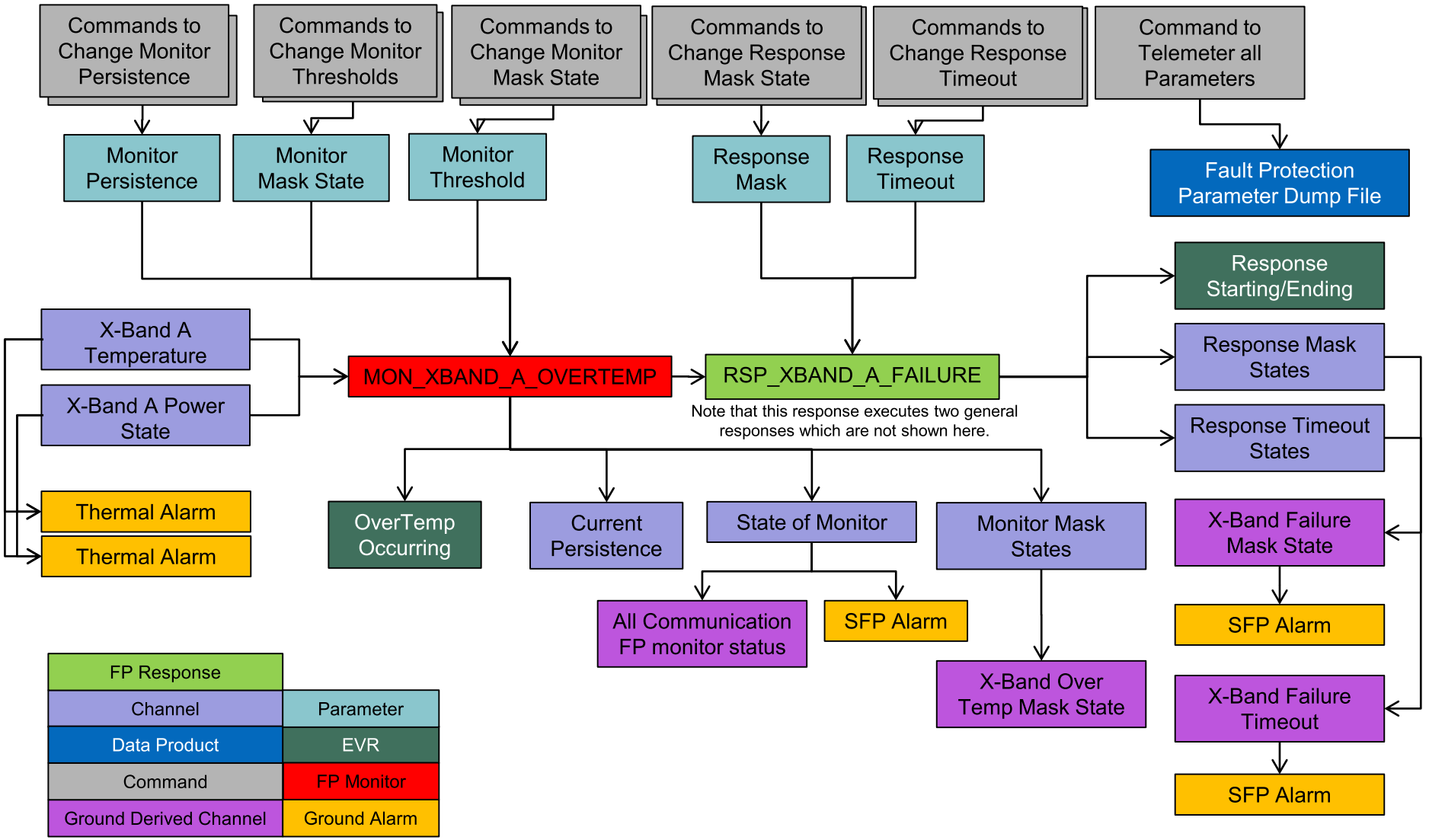
Added By SMAP

- Parameters
  - Includes default value, rationale, mechanism to change/telemeter
- Fault Protection: Monitors
  - Local and System Monitors, error test definition, map to system responses, local responses, associated cmds, eha, evrs, params
- Fault Protection: Responses (System)
  - Actions (by tier), associated cmds, telemetry, evrs, parameters

MSL/MPCS

- Alarms
  - Ground based alarms on individual or multiple combinations of channels

Dictionary Interconnectivity

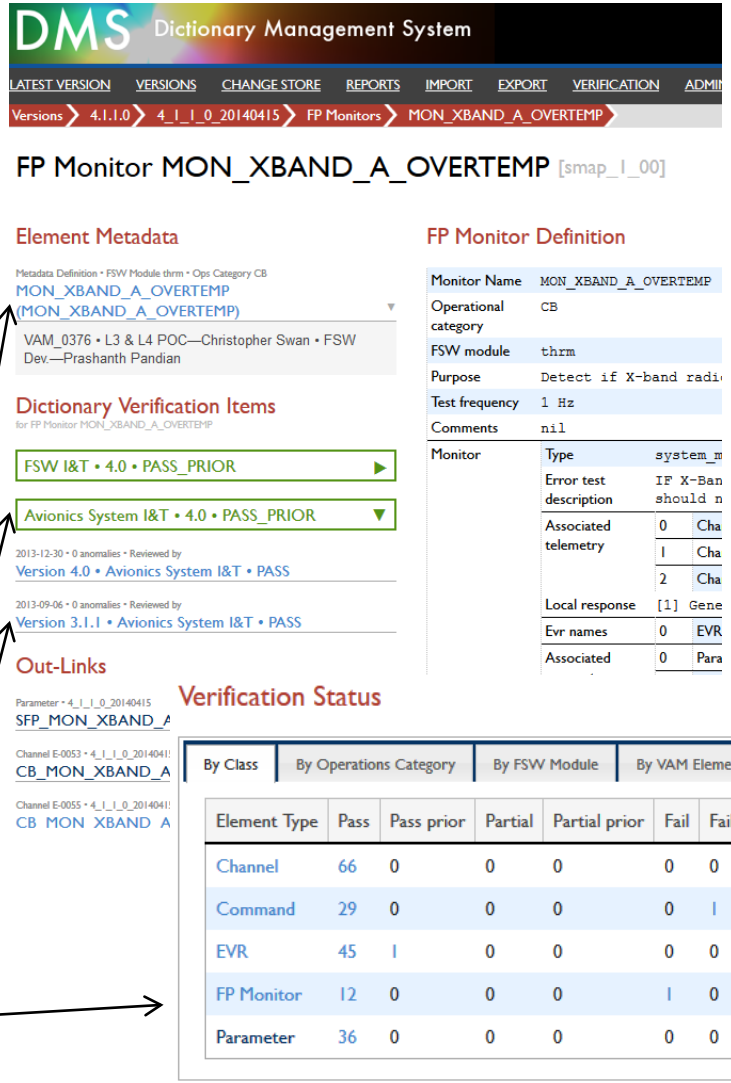


SMAP Dictionary Example Life Cycle: Stellar Reference Unit (SRU)

- FS Engineer worked with SRU Cog'E to develop initial request
  - Incorporated source information from SRU IICD/Specification, GN&C FDD, applicable FMECA/FTA information
- Initial request was reviewed by cross cutting team
  - Including broader GN&C team, FS/FP team, FSW, AVSE, Mission Systems
  - Discussion focused on specific challenges of the SRU and information that could drive behavior
    - The SRU 1553 message content is SRU mode specific which complicates EH&A telemetry collection
      - In some modes this information is an image or memory dump, while in others it could be a quaternion or result in a rejected request
    - The SRU has firmware that needed to be patched in flight
  - Team brainstormed on problem and elected to route most common telemetry to EH&A and create a general data product for all other information. Team also reached consensus that a mechanism needed to exist to update SRU firmware.
  - Team also agreed that key aspects of this (mode dependent telemetry, firmware upload, general data product) constitute change to behavior and would require an change request to software behavior specification..
- FS Engineer closed action items and entered initial request in DMS
  - 55 FSW EH&A channels, 4 FP Monitor, 1 Data Product, 26 Commands, 6 EVRs
- FSW Cog'E assigned SRU to R3 and a developer began work
- Error found in initial request
  - In review of change request the software behavior specification it was discovered that several 1553 messages planned for the data product were needed for EH&A and a requested command was invalid (and should be removed)
  - FSE coordinated this with FSW developer (with oversight from FSW Cog'E and FSE Operability Lead)
  - 68 FSW EH&A channels, 4 FP Monitor, 1 Data Product, 25 Commands, 6 EVRs
- FSW developer identified problems with requested FP Monitors
  - FSE coordinated between FP/GN&C/FSW to resolve the issue – FP monitors consolidated and moved to higher level GN&C code
  - 68 FSW EH&A channels, 1 FP Monitor, 1 Data Product, 25 Commands, 6 EVRs
- Later in development FSW determined that several EH&A channels were needed to track 1553
  - FSW developer coordinated with FSE (with oversight from FSW Cog'E and FSE Operability Lead)
  - 74 FSW EH&A channels, 1 FP Monitor, 1 Data Product, 25 Commands, 6 EVRs
- FS Engineer completes specifying ground derived channels based on FSW EH&A
- FSW completes development and FSW I&T Team tests dictionary elements
  - FSW feeds back any self generated EVRs to FSE for review and inclusion in DMS
- Dictionary released in sync with FSW release containing SRU content

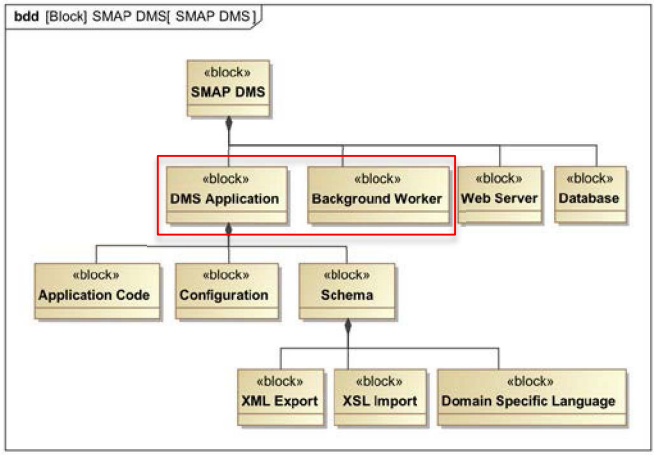
V&V Information "One Click Away"

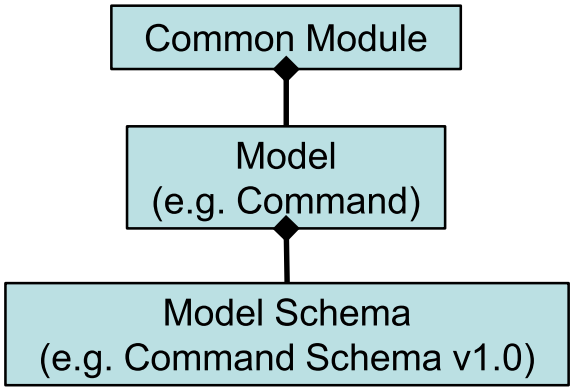
- Verification and validation is a major undertaking and is often made even more onerous by tools used to manage it.
- Given the wealth of information already located in the DMS database the inclusion of V&V results and meta data made a lot of sense.
- The visibility and availability of the detailed results and reports to the wider team was very helpful.
  - Mapping to V&V owner and V&V activity
  - Easy "stop light" indication of V&V status
  - One click away from pointers to actual test results / scripts / procedures
  - Interactive summary reporting at any level.



The DMS Application

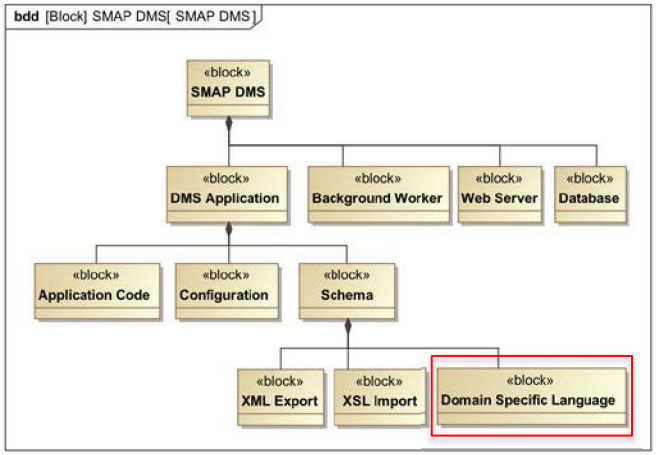
- The **DMS Application** has a model-view-controller architecture
  - **Models** define dictionary elements
    - A common *module* is inherited by all dictionary element *models*
      - Defines shared attributes – fields, validations, display names, instance methods, etc.
      - Easy to add new dictionary element types
    - Schema definitions build on what is defined in the model
      - Multiple schema versions can be defined for the same dictionary element type
      - We'll discuss this in depth on the next slide
  - **Views** are dynamically generated for each dictionary element based on its schema
  - **Controllers** perform instantaneous tasks (typically rendering forms, displaying indexes etc.)
- **Background workers** perform tasks that take some time
  - Import
    - Change request import
    - Metadata/VNV import (test import, finalize import)
  - Dictionary compilation (test build, finalize build)
  - Mass update change requests
  - Mass apply change requests to beta dictionary





Schema and Validation

- Dictionary elements are validated against a schema
  - DMS uses a domain-specific language to define the schema's structure and validation rules
  - Allows us to define multiple schema for the same dictionary element type (e.g. Channel)



- Some examples of schema validation rules:

In the Channel schema definition, we define Channel ID as follows:

```
required_key :channel_id, String, {
   label: "Channel ID", format: DataTypesSmap100::CHANNEL_ID, xml: {format: 'attribute'}
}
```

In the FP Monitor schema definition, we describe its dependency on Channels via their names:

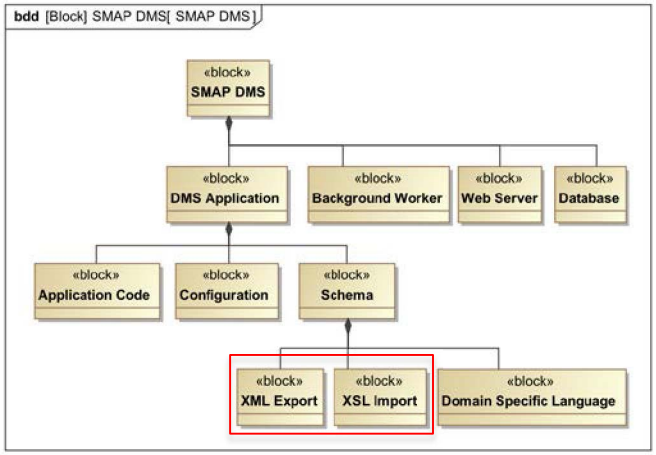
```
{class: "ChannelSmap100", conditions:{
  "monitor.associated_telemetry[n].associated_telemetry" => "channel_name"
}}
```

More complex validations not included in the language can be defined as Ruby code, e.g.

```
validate :index_range
def index_range
  if index and (index < 0 or index > 7)
    errors.add(:index, 'index must be a number between 0 and 7, inclusive.')
  end
end
```

Importing and Exporting XML/TSV

- DMS can import change requests from XML
  - Several of the XML formats used are slightly ambiguous about ordering and internal structure, and so require further clarification
  - An administrator provides XSL files that translate ambiguous XML formats into a structure DMS can parse and understand
  - Change requests are added in bulk to the DMS change store for review and approval
- DMS can export official dictionaries to XML, or subsets of the dictionary as tab-separated values (TSV)
  - The domain-specific language defines how each field in the dictionary is to be exported as XML: whether or not it is to be an attribute, node, or a property of a parent node.
  - An additional file is provided by an administrator to determine how the dictionary is to be broken up into individual XML files. For instance, all EH&A go into one long file, but each Data Product gets its own separate file.
  - Individual dictionary element types can be filtered before export, giving individuals the ability to simply get the EH&A XML file instead of everything.
  - Individual dictionary element types can also be exported as TSV files.



Dictionary Compilation

- DMS's primary product is a compiled dictionary that is syntactically and semantically valid.
- The very first version of a dictionary in DMS is built entirely from change requests. From that point on, all dictionaries are typically built from a combination of change requests and a prior "base dictionary"
- Dictionary compilation in DMS is a multi-step process:
  1. User submits target version, base dictionary, and compilation options
  2. DMS gets all elements from the base dictionary and figures out which change requests to apply. A change request constitutes an addition, modification, or removal of a dictionary element
  3. DMS cross-checks all of these elements against each other to see if there are any omissions or validation errors
  4. DMS displays the results of this effort to the user *for review*. If the user doesn't like these results, they can act on them by making additional change requests, followed by recompilation.
  5. The user can then direct DMS to deliver the results as a beta or official dictionary

Optimizations

- The algorithm used to perform dependency validations often makes heavy use of the database, which means that performance can be impacted if not careful. Several optimizations were used to increase performance:
  - Indexing fields that are commonly referenced
  - Limiting number of columns returned from queries
  - Consolidating writes to a bulk insert at the end of the algorithm
- Similar optimizations were applied to background job algorithms