IN53A-1609

Towards a Domain Specific Software Architecture for Scientific Data Distribution

Anne Wilson, Douglas M Lindholm, LASP, CU Boulder

LASP
LABORATORY FOR ATMOSPHERIC AND SPACE PHYSICS
UNIVERSITY OF COLORADO AT BOULDER

The Utility of a Reference Architecture: the Best of Best Practices

A Reference Architecture

- is a generic and somewhat abstract blueprint-type view of a system.
- includes the system's major components, the relationships among them, and the externally visible properties of those components.
- is not usually designed for a highly specialized set of requirements. Rather, architects tend to use it as a starting point and specialize it for their own requirements

Consider a residential architecture. The implied model for a house is composed of the following components:

- A foundation and/or other subfloor structure to connect the house to the underlying environment, whether it is earth or a body of water
- Floors to stand on
- Exterior walls to keep out the elements of nature and to provide structural support for the roof
- A roof to protect the dwelling's contents and occupants from the elements of nature and to provide privacy
- Some form of entry and exit (possibly implemented as a doorway)
- External links to some form of consumable energy (an interface to connect to the electricity grid, a windmill, or some other electricity-generating device)

The reference architecture can add details based on requirements, e.g. "a residential dwelling in Denver for a family of five" or "a one-bedroom apartment in Berlin for an elderly person."

A reference architecture plays an important role as a starting point upon which more specialized instances of a class of thing can be built, with particular purposes and styles addressed as needed. Paraphrased from [Hinchcliffe].

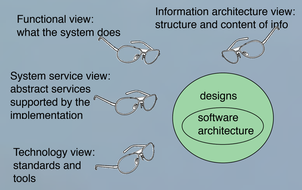
A **reference architecture** provides
- a template solution for an architecture for a particular domain,
- a common vocabulary with which to discuss implementations, often with the aim to stress commonality.

A **reference software architecture**
- is a software architecture where the structures and respective elements and relations provide templates for concrete architectures in a particular domain or family of software systems.
- often consists of a list of functions and their interfaces
- can be defined at different levels of abstraction
- "The best of best practices"
- An iterative, community process

Examples
- Java EE platform for server programming: provided servlets, portlets, JSPs, web services
- Web 2.0 Reference Architecture: provided the resource, service, and client tiers

Detailed reference architecture for Web 2.0 application architects and developers [Hinchcliffe]

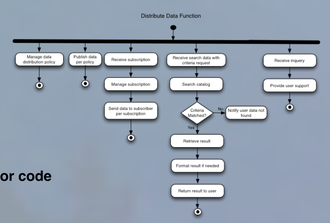
NASA's Earth Science Data Systems Working Group (ESDSWG) is defining a Reference Architecture
Motivation: handle explosion in data volumes, increased system complexity, diversified data sources, greater user expectations, increased collaboration
Use cases, scope: receive sensor data, develop products, distribute products, develop predictions, manage remote sensor/instrument, data stewardship
Stakeholders: system architect, data architect, program manager, project scientist, policy research, research scientist, NASA management



Here we focus on RA use case class 3, "Distribute Products": "Generated products are made available to end consumers. This includes discovery and access... using tools, as well as actual delivery... to end points."

A software architecture is a design that clearly articulates its components: interfaces, etc.
It can have multiple views. Relevant views may vary for different stakeholders.

Functional description: "The ultimate purpose of the ESDS is the generation of data sets that enhance Earth Science knowledge and applications. Distribute Data is the fundamental process that enables users to identify, locate and receive the data sets they need for their investigation and research. Distribute Data manages when, how and to whom data are to be distributed, as well as provides users with the capability to query the data holdings to find desired data. Access provides the ability to extract specific pieces of information from a data holding, then repackage that information in a different format and make it accessible to a user. The Distribute data function also provides resources to answer user's questions regarding issues of the data access." [Figure and text from ESDSWG_RA]

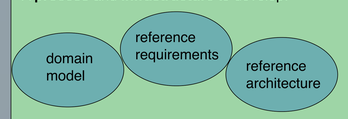


ESDS RA Distribute Data Function Activity Diagram

But this level of detail is does not provide sufficient guidance for code development. How to provide greater detail?

Domain Specific Software Architecture (DSSA)

A **process** and **infrastructure** to develop:



Goal: support generation of applications within the domain

A DSSA is
- a software architecture with reference requirements and a domain model,
- the infrastructure to support it,
- and a process to instantiate or refine it.

A DSSA forces consensus at the architectural level and modeling of the domain for additional insight and understanding. Requirements need to evolve based on feedback. [DSSA FAQ] [DSSA Ped Ex]

Here we focus on a DSSA for scientific data access and distribution.

The Domain: Scientific Data Access and Distribution

Domain Model — a framework to capture application domain knowledge, the problem space

- Modeling tools: scenarios, vocabulary, context/block diagrams, entity/relationship diagrams, data flow models, state transition models, object model
- Analysis involves domain experts, customer inputs, others familiar with the application

Goal: provide an unambiguous understanding of aspects of the domain

Use cases/scenarios — provide basis for vocabulary, requirements.

Read SORCE TSI data into IDL array.
Get times in ISO 8601 format.
Find solar irradiance data for a given time range.
Find wind profiler data over a particular region.
Find data centers that serve solar flare events.
Find irradiance measurements for a particular wavelength.
Get data in a format compatible with a particular visualization tool.
Cite a dataset used in a research project.
Acquire a dataset cited in a publication.
Release a new dataset.
Release a new version of a dataset.
Get notification when a new version of a dataset has been released.
Comment on a dataset.
Find data related to Hurricane Katrina.
Add sea surface temperatures to a model.
Locate observed and predicted wind speeds for comparison purposes.
Combine irradiance measurements from multiple instruments into a single dataset.
Give authorized users immediate access to proprietary datasets.
Provide public access to a dataset after a given time interval.
Understand the quality of a dataset: assumptions, errors, uncertainties, etc.
Understand the appropriate use of a dataset.
See what holdings a data center has.
Etc, etc.

Issue
How to find temperature?
"temperature"? "temp"? "t"?
Many names for same phenomena

Object model — describes an object in the system

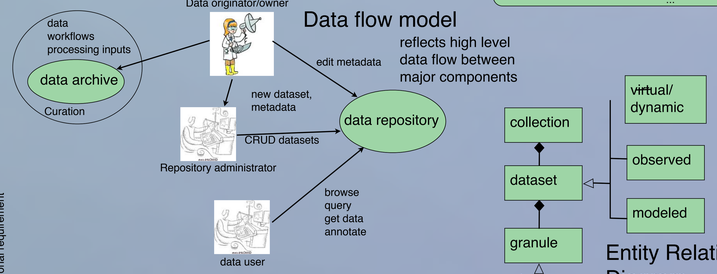
dataset
name
version
description
domain (lat/lon, time, wavelength...)
range
units
PI/author
publisher
...

Issue
what pieces of information about a dataset must be captured?

Vocabulary

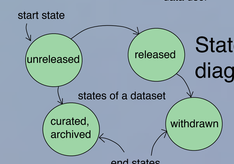
subset
aggregation
publish
request, response
release dataset
withdraw dataset
data point
measurement
observed data
reference data
format
parameter
field
variable
dataset
virtual dataset
remote dataset
local dataset
granule
collection
inventory
repository
metadata
coverage
history
provenance
units
quality
version
identifier
...
query
browse
discover
notification
subscription
stakeholder
data user
data owner
policy maker
data provider
repository
administrator

Data flow model — reflects high level data flow between major components



Entity Relationship Diagram — describes how entities are related to one another

State transition diagram — describes events and states that take place in this domain



Mapping between problem space and solution space: trace scenario to functional requirement

Reference Architecture — the solution space

For each component provide values for this template:
name
responsibility
constraints: ranges of parameter values, relationships between parameter values or components (exclusion, dependency, etc.)
dependencies
rationale: rules of thumb, lessons learned, etc.
interface

Documentation repository
responsibility: Provide browse and query access to dataset documentation
constraints: Provide a RESTful interface
dependencies:
rationale: Be able to crosswalk to standard metadata schemas
interface: browser and service API

Data Repository Front End
responsibility: provide user interface, call services appropriately, integrate services into coherent response, etc.
constraints:
rationale:
dependencies:
interface: browser

Ontology
responsibility: map from natural language to closed vocabulary for purpose of locating and using data
constraints:
dependencies
rationale:
interface: browser and API

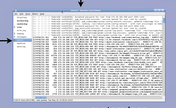
Issue
What is a dataset? Is a proper subset of a dataset a new dataset requiring unique identification?

Issue
how should dynamically created datasets be managed? What, if any, metadata should be provided?

Data server
responsibility: Provide access to datasets and dynamic information about current holdings. Access dataset granules. Be able to select variables/fields/parameters. Be able to specify output format.
constraints: Provide a RESTful interface. Be OPeNDAP compliant. Use standard output formats where ever possible, e.g. OPeNDAP, WCS, etc.
dependencies:
rationale:
interface:

Issue
Interfaces need to be clearly and unambiguously defined. As interfaces are contracts, determining the correct interface is critical to the utility and success of the RA.

Administration
responsibility: CRUD (create, read, update, delete) datasets, generate usage reports, send change notifications, ...
constraints:
dependencies
rationale: provide useful service, justify funding
interface: browser


system logs


data storage

Mapping between problem space and solution space: trace functional requirement to architectural component

Requirements — define functional, non-functional, design, and implementation requirements

Functional: "what"

Allow **data users** to:
- see available datasets
- learn of new datasets, changes
- discover datasets
- query on dataset details
- retrieve a dataset or a portion of a dataset
- retrieve a dataset in a designated format
- e.g. date, file format (ascii, NetCDF, FITS, HDF...)
- get dataset information regarding acquisition, provenance, usage
- annotate or comment on a dataset
- be able to fuse or merge datasets
- cite a dataset
- unambiguously identify dataset components

Generalizations derived from use cases, scenarios, customer needs

Allow **data providers** to:
- CRUD (create, read, update, delete) a dataset
- enter and correct metadata
- track and report dataset and site usage
- restrict usage to authorized users within authorized time ranges
- send notifications
- get credit for a dataset

Technical/implementation: "how"

How fast, often, large, accurate
What protocols, standards, best practices to use
E.g., use a service oriented architecture
Platforms: UNIX, Windows?
Support
- authentication and authorization for access to selected datasets
- authentication and authorization for dataset owners to edit metadata
- a browser interface
- a service interface

Be able to easily integrate a broad range of datasets
Support code reuse as much as possible
Uniquely identify dataset components
...

A **Service Oriented Architecture** (SOA) is a set of principles and methodologies for designing and developing software in the form of interoperable services. It aims to allow users to string together large chunks of functionality to form ad hoc applications built almost entirely from existing software services. A SOA provides the commonly adhered to programming principles of: abstraction, autonomy, composability, formal contract, loose coupling, and reusability. It is probably the best approach for the RA.

Conclusions

Summary
There are many efforts underway to build reusable infrastructure to acquire, manage, and use scientific data, such as ESDSWG's reference architecture, NSF's EarthCube, INSPIRE, GSDI, RSDI, etc. A reference architecture can provide a high level plan for how to proceed.

The DSSA process provides a structured method to capture domain information, establish requirements, and ultimately provide a reference architecture that meets established requirements.

Scientific data distribution is a key component of science data infrastructure. Scientific data distribution is well enough understood and sufficiently limited in scope that it would be possible to define a DSSA for it that could actually aid in building a system. A DSSA for scientific data distribution would help data providers specify, design, validate, package, and deploy a useful data distribution system.

Presented here are first steps towards a DSSA for scientific data distribution.

This effort is clearly incomplete. Community input, debate, collaboration, and consensus is needed.

Benefits to having a DSSA for scientific data distribution
- A reference architecture provides a documented high level design for a family of applications and places constraints on implementations. This generalization supports code reuse, which helps to minimize time and effort in development.
- By Biggerstaff's rule of three and empirical evidence, the overhead to generalize to a RA is worthwhile if at least 3 systems are generated from the architecture.
- Mapping between problem space and solution space allows tracing of requirements to responsible components.
- The documentation generated by the DSSA process aids in system maintenance and evolution.

References:

[Armitage, 1993] Armitage, James, *Process Guide for the DSSA Process Life Cycle*, Software Engineering Institute, Paper 240, http://repository.cmu.edu, December, 1993.

[Burnett, et al, 2011] Burnett, Michael, Weiss, Barry, Law, Emily, *NASA's ESDS Reference Architecture*, AGU Fall Meeting, San Francisco, CA, December 2011.

[ESDSWG_RA] *ESDS Reference Architecture for the Decadal Survey Era*, V 1.0, Nov. 2, 2011.

[Hinchcliffe] Web 2.0 Architectures, Governor, Hinchcliffe, Nickull, O'Reilly Media, May 2009.

[Tracz, 1995] Tracz, Will, *DSSA (Domain Specific Software Architectures) Pedagogical Example*, ACM SIGSOFT Software Engineering Notes V20 N3, July 1995.

AGU Fall 2011 Meeting