Using Selection Pressure as an Asset to Develop Reusable, Adaptable Software Systems

Stephen Berrick, Chris Lynnes (NASA/GES DISC DAAC)

GSFC Earth Sciences (GES)
Data and Information Services Center (DISC)
http://disc.gsfc.nasa.gov

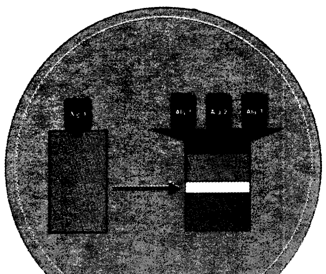
The Goddard Earth Sciences Data and Information Services Center (GES DISC) at NASA has over the years developed and honed several reusable architectural components for supporting large-scale data centers with a large customer base. These include a processing system (S4PM) and an archive system (S4PA) based upon a workflow engine called the Simple, Scalable, Script-based Science Processor (S4P); and an online data visualization and analysis system (Giovanni). These subsystems are currently reused internally in a variety of combinations to implement customized data management on behalf of instrument science teams and other science investigators. Some of these subsystems (S4P and S4PM) have also been reused by other data centers for operational science processing.

Our experience has been that development and utilization of robust, interoperable, and reusable software systems can actually flourish in environments defined by heterogeneous commodity hardware systems, the emphasis on value-added customer service, and the continual goal for achieving higher cost efficiencies. The repeated internal reuse that is fostered by such an environment encourages and even forces changes to the software that make it more reusable and adaptable. Allowing and even encouraging such selective pressures to software development has been a key factor in the success of S4P and S4PM, which are now available to the open source community under the NASA Open Source Agreement.

S4PM

The Simple, Scalable, Script-based, Science Processor for Measurements (S4PM) was a processing system conceived and developed in 2001 to support the processing of data from MODIS[1] in the GES DISC. Built upon the proven S4P[2] core, S4PM developers quickly developed a capable processing system optimized to support MODIS algorithms and meet the reprocessing schedule. In 2002, S4PM was enhanced to process data from the Atmospheric Infrared Sounder. The influx of new, instrument-unique requirements forced an S4PM refactoring toward a more modular architecture, particularly at the interfaces where algorithms are plugged in. A subsequent requirement to support multiple platforms (Sun, SGI, Linux) provided the impetus to enhance portability. And the need for S4PM to be installed, deployed, and configured quickly by multiple users resulted in better packaging and more automated configuration. This led to public release of S4PM under the NASA Open Source Agreement in 2004. S4PM was then reused by Langley Research Center to process data from CALIPSO[3], MISR[4], and FLASHFlux[5], and by EROS Data Center to process ASTER[6] data.

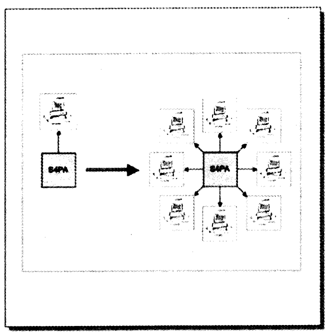
[1]MODerate resolution Imaging Spectroradiometer, [2]Simple, Scalable, Script-based, Science Processor, [3]Cloud-Aerosol Lidar and Infrared Pathfinder Satellite, [4]Multiangle Imaging SpectroRadiometer, [5]Fast Longwave And SHortwave Radiative Fluxes, [6]Advanced Spaceborne Thermal Emission and Reflection radiometer



S4PA

The Simple, Scalable, Script-based, Science Product Archive (S4PA) was designed to handle the migration of legacy data from robotic tape archives to disk. Beginning in 2006, S4PA was tapped to replace the massive EOSDIS[1] Core System (ECS) data archive with a hard deadline at the end of 2007. The schedule pressure to stand up multiple disk-based instances of S4PA to accommodate the vast data volume (~400 TB), each with its own integration engineer, drove the development of more sophisticated packaging and easier, more automated installation mechanisms.

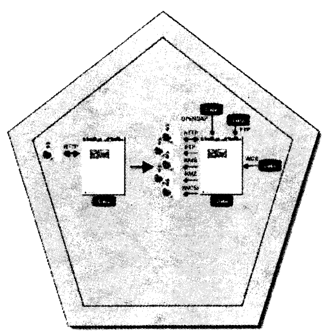
[1]Earth Observing System Data and Information System



Giovanni

Giovanni[1] is an online data visualization and analysis system that started out in 2002 as a Web-based CGI[2] application. It provided an interactive session via HTTP through which users could select data parameters, regions of interest and time ranges and generate a variety of plots and images. Data supporting Giovanni were on the machine hosting it and could be downloaded by the user via FTP. As Giovanni grew in popularity, science users provided feedback, which drove enhancements which, in turn, expanded Giovanni's popularity. Scientists seeking collaborations and interoperability with their systems followed. In 2005, Giovanni was then refactored from a CGI-based application to a services oriented architecture supported by a asynchronous workflow management system. Data no longer had to be on the host machine and could access remote data using an expanding list of protocols as well as multiple output protocols and data formats.

[1]Goddard Interactive Online Visualization ANd aNalysis Infrastructure, [2]Common Gateway Interface



The Earth Science Data Systems Reuse Working Group has developed a DRAFT description of Reuse Readiness Levels (RRL), a metric for evaluating the reuse readiness of software. The top table below shows how S4PM, S4PA, and Giovanni progressed to higher RRLs over time in each category. The bottom table indicates which selection pressures were the most dominant in that process.

	1	2	3	4	5	6	7	8	9
Packaging	[S4PA] [S4PM] **Source code available**		Detailed installation instructions available		Software easily configurable for different environments		[S4PA] [S4PM] **OS detect and auto-build for supported platforms**		GUI installation environment
Portability	The software is not portable at any cost	Some parts of the software may be portable	Portable only with significant costs	[S4PM] **Portable at a reasonable cost**	Moderately portable	The software is portable	The software is highly portable	[S4PM] **The software is easily portable**	The software is completely portable
Documentation	[S4PM] **Limited internal documentation available**	Fully commented source code available	[Giovanni] [S4PA] **Basic external documentation available**	Reference manual available	[S4PM] **User manual available**	[Giovanni] **Tutorials available**	[S4PA] **Interface guide available**	Extension guide available	Full software lifecycle engineering design available
Support	No support available	[S4PM] [S4PA] **Known contact available**	Original developers provide proactive support	[S4PA] **Infrequent updates and patches available**	[S4PM] **Informal user community available**	Centralized support available	Organized/defined support by the original developer available	Support by organization available	Large user community with well-defined support available
Extensibility	No ability to extend or modify behavior	[Giovanni] **Prohibitive costs and efforts needed to modify or extend the system**	Can be extended with the input of considerable time and effort on par with recreating the system separately	[S4PM] **Can be modified and extended via config. changes & small code changes**	[S4PA] **Consideration for future extensibility is designed into the system**	Designed from start for easy extensibility	[Giovanni] [S4PA] **Proven to be extensible internally, structured code**	[S4PM] **Proven extensibility on a major external program, clear plan for extending**	Proven extensibility in multiple scenarios with documentation
Intellectual Property (IP)	Potential owners and stakeholders identified	Relevant IP policies of potential owners and stakeholders reviewed	IP agreements proposed to potential stakeholders	IP agreements and authorship issues negotiated	Authorship, attribution and intellectual property issues agreement and approval	Authorship, attribution, and IP statements drafted	Authorship and IP statements included in product prototype	Manifestation of authorship, attribution, and IP statements reviewed	Reviewed authorship, attribution, and IP statements in product
Standards Compliance	No particular standard	[S4PA] [Giovanni] **Follows some parts of common standards and best practices**	Follows a company-wide standard for development and testing	Most components follow a complete, universal standard, but not validated	All components follow a universal standard, partially validated	Validated to follow a specific proprietary standard	[S4PA] [Giovanni] **Validated to comply to a specific open standard**	Proven by validation to comply with a "gold" standard	"Gold" standard compliance of entire system and development, independently validated
Verification and Testing	No testing performed	Software formulated and unit testing performed	[S4PA] [S4PM] [Giovanni] **Includes testing for error conditions including input errors**	Software tested and validated in laboratory environment	Software demonstrated in laboratory environment	Software demonstrated in a relevant environment	Software tested and validated in a laboratory environment	Software "qualified" through test and demonstration	[S4PA] [S4PM] [Giovanni] **Software application tested and validated via successful use of output**
Modularity	[Giovanni] **No designs for modularity**		[S4PA] [S4PM] **Modularity at major system or subsystem level only**		Partial segregation of generic and specific functionality		[S4PA] [S4PM] **Clear delineations of specific and reusable components**		[Giovanni] **All functions and data encapsulated into objects or accessible through Web services**

	Platform Heterogeneity	Target Environment Heterogeneity	Cost Efficiency	Feature Growth
Packaging	S4PM	S4PA		
Portability	S4PM			
Documentation		S4PA	S4PM	Giovanni
Support			S4PM	S4PA
Extensibility				S4PM, Giovanni, S4PA
Intellectual Property Issues				
Standards Compliance			Giovanni	Giovanni, S4PA
Verification and Testing		S4PA	Giovanni	S4PM
Modularity		S4PM, S4PA	Giovanni	