astraZENeca and the ART of metadata maintenance

Paul A. Frost, AstraZeneca, Alderley Park, United Kingdom

Abstract

The use of metadata to drive dynamic programming is a common and well documented process; from Oracle® system tables to the SAS® SQL dictionary tables. A&RT (Analysis and Reporting Technology), AstraZeneca's first truly global reporting system, extends this concept to incorporate data standards, metadata maintenance and metadata re- use; leveraging structural hierarchy[i] , equivalence[ii] and inheritance[iii],together with the collection of metadata ahead of time to streamline the process and minimise resource intensive activities on the critical path to submission.

We will look at the how and why of the ART reporting solution, process versus technology, key touch points for any reporting solution and visions for the future. The importance of integrating a data standardisation process and organisation into the reporting technology and how adding metadata collection and import/export functionality to a reporting technology can help facilitate validation of data standards, production of key deliverables and the creation of additional, non-reporting deliverables, such as define.xml. Our roadmap for the future includes enhanced impact analysis, tighter integration between UNIX processing power and windows UI components and a lifecycle management plan to ensure our technology base remains current and our system enhancements user driven.

Introduction

The A&RT project, with several work streams, including a process definition activity, was initiated during 2003. The purpose was to improve and resolve differences in the programming, analysis & statistical business area. These differences were between sites, roles & responsibilities, competences areas and competence needs and ways of working as well as supporting tools. The project team had a remit to create a complete reporting environment with consistent processes, tools and infrastructure.

The WHY

The merger between Astra and Zeneca created a disparate reporting and analysis topography which is represented pictorially in Figure 1 - The A&RT concept: Before and After.

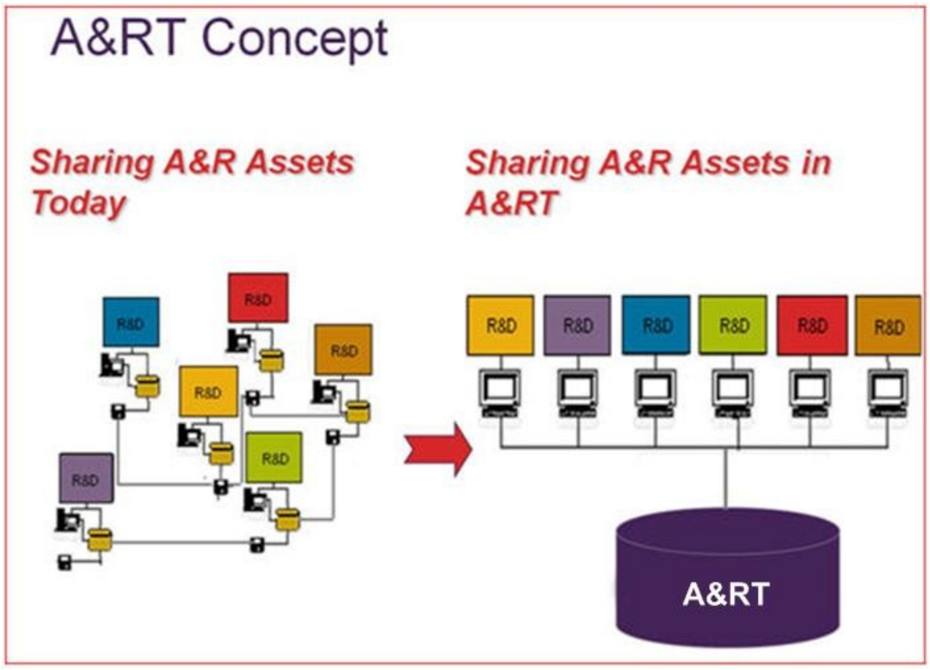


Figure 1- The A&RT concept: Before and After

As is clear from the above in 2003 there was a lack of consistent process, tools and infrastructure available to facilitate the transfer or sharing of work between different AstraZeneca sites and even regions. Previous mergers within the company had led to a global organisation with six major Research and Development sites across three continents; some of those sites having multiple legacy reporting applications. This often resulted in the same work being done multiple times at different sites to allow for the differences in technologies and process. In the UK, for example, the two R&D sites each had their own reporting systems with different architecture in terms of both technology and folder structures and a markedly different process. The process to align of all these processes and technologies was much more complex than the merger between two companies would suggest!

The realisation that significant change was vital prompted the A&RT project team to take a step back and ‘blue sky’ what they would expect from an ideal reporting environment.

There was already recognition that data throughput from early project stages down to phase III submissions was increasing and would continue to increase. To be able to handle this throughput and gain efficiencies in doing so, rapid and efficient analysis and reporting of clinical data would become key, along with robust and scalable infrastructure.

To facilitate the efficient analysis of clinical data we would also need to harmonise our data standards and in doing so why not move AstraZeneca in its entirety towards the CDISC standards, specifically SDTM?

The HOW (Process)

To focus the development of the A&RT solution we concentrated on a number of key concepts:

Minimising activities on the critical path

- ✓ Upfront planning including early consensus on the content of the Clinical Study Report or other key deliverables
- ✓ Start with the definition of any high level documents (HLD) and work backwards
- ✓ Create all data specifications as early as possible

Working in Parallel

- ✓ Data capture
- ✓ Reporting database specification
- ✓ Reporting output specifications
- ✓ Draft clinical study report with tables, listings and graphs

Common processes

- ✓ Drive towards corporate level RAW data standards and introduce SDTM as a data layer and a revised RDb definition predicated on SDTM
- ✓ Create a governance organisation to control ALL data standards
- ✓ Introduce standards for data transformations based on RAW / SDTM / RDb data standards
- ✓ Extend the governance to include definition and control of statistical and reporting outputs
- ✓ Eliminate re-work across sites

Common tools

- ✓ Implement common software for Analysis and Reporting at all sites
- ✓ Create SAS® code fragments for defined data transformations
- ✓ Develop and implement libraries of global, reusable computer programs for production of statistical output
- ✓ Creation of standardized tables and listings by using generic SAS reporting macros.

A key deliverable from all of these processes was the ability to share the same global standards, environment and tools with an enhanced possibility to move work around and to seamlessly share work across sites within any given project or study.

The HOW (Technology)

The A&RT project team developed the process part of the HOW equation and assembled a detailed design specification. There followed an intense period of technology evaluation against requirements which can be summarised as a quest for the following holy grail:

- ✓ Metadata driven
- ✓ Hierarchical representation of all components with inheritance and equivalence
- ✓ Single platform for reporting and analysis

The result of the evaluation was an acknowledgement that we could create this solution with one of our existing software partners. When reviewed in detail the A&RT application is both sophisticated and complex but the kernel can be summarised thus:

"…*a basic UNIX SAS environment with a sophisticated GUI web based front end and a standard structure for environment and libraries with global accessibility and an Oracle database for metadata storage*"

The logical architecture is described in Figure 2 - A&RT Logical Architecture.

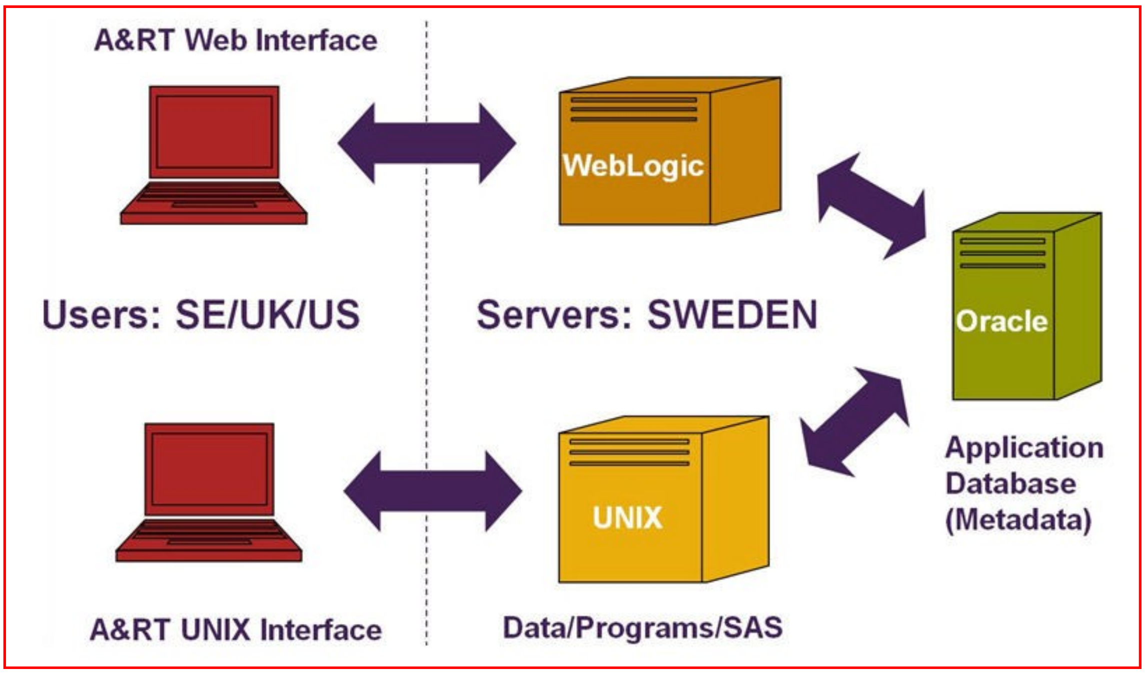
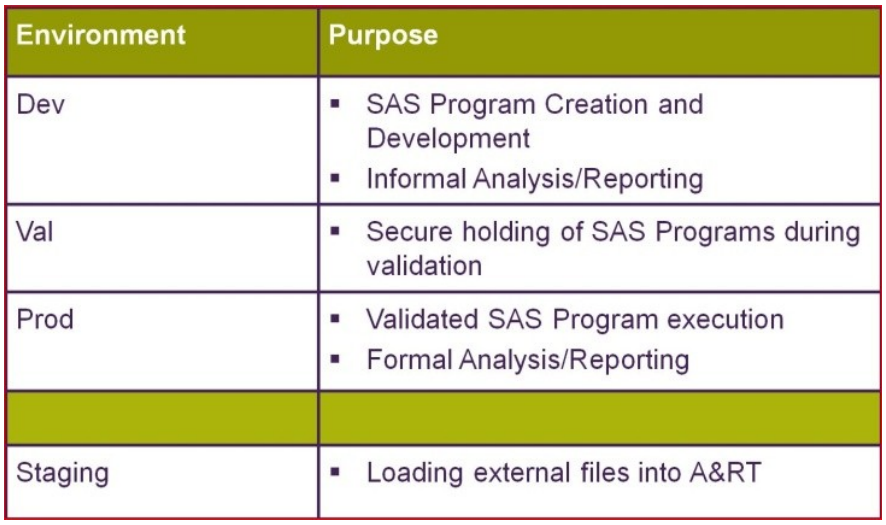


Figure 2 - A&RT Logical Architecture

In addition to the standard folder structure we introduced three distinct environments within the same physical infrastructure for development, validation and production. This was intended to both facilitate and standardise the validation process and the key concepts are highlighted in Figure 3 - A&RT Environments and Usage.



Environment	Purpose
Dev	▪ SAS Program Creation and Development ▪ Informal Analysis/Reporting
Val	▪ Secure holding of SAS Programs during validation
Prod	▪ Validated SAS Program execution ▪ Formal Analysis/Reporting
Staging	▪ Loading external files into A&RT

Figure 3 - A&RT Environments and Usage

HOW – metadata

The use of metadata to drive dynamic programming is a common and well documented process; from Oracle system tables to the SAS SQL dictionary tables. A&RT extended this concept to incorporate data standards, metadata maintenance and metadata re- use. All of the metadata relating to clinical data is made available within A&RT as the starting point of any study. RAW, SDTM and RDb data standards are all tightly controlled by a dedicated governance organisation. RAW data standards are made available as the first step in our data collection process which is designing the eCRF. At the same time these standards are imported into A&RT where they form the first source in the RAW => SDTM => RDb supply chain. For our current, standard RAW data we have accompanying definitions of both SDTM and RDb variations of this data structure. The metadata for these data structures is made available at Project/Study level by a process of inheritance from the AstraZeneca corporate level.

Where data does conform to the AstraZeneca standards but has been renamed it is possible to define an equivalence in order to continue the inherit process. An example of this might be laboratory data which is often collected over two eCRF pages and delivered in two SAS datasets. The study will see LAB1 / LAB2 which do not exist at the corporate level but can both be made equivalent to LAB. This process works not only for metadata but also for the mappings between source and target data.

For data this is a simple concept; we have imported or inherited all of the metadata for our physical data. As described above, we can action this way ahead of having physical data available for the study which effectively takes this activity off the critical path. At this point we have just a description of our data which is useful but limited. The next step in the process is to define the links between source and target data. To do this we use the same standards organisation to define the algorithms required to transform source to target data. Once these algorithms are defined they are created in A&RT in a number of ways:

- ✓ Keywords
- ✓ Simple SAS (SQL) code
- ✓ Macros

Although, in reality, this is a very complex process the underlying concepts are simple. To create a transformation, or mapping, between variables we just use standard SAS code. This code is represented either by a keyword, a simple SQL statement or a SAS macro. These transformations are created and stored at the corporate level and are then available to inherit in exactly the same way as we inherited our metadata describing our data.

We now have a number of different types of data stored in our application but so far all we have created is a way of storing, maintaining and distributing metadata. The next bit is the secret of creating complex life from single cells.

HOW – "program builder"

Now we have data describing both our physical data and the required transformations from one data layer to another we need to put all this together. We can think of all the variable level metadata and associated transformations as being single cells floating together in our application sea. In order to make intelligent life out of these single cells we have created a SAS program which interrogates the metadata and create a number of SAS programs which together will create our reporting database. This "program builder" is composed of a number of SAS macros with a host of features; we can mention a few of them just to give a flavour.

Keywords

Keywords represent the simplest and easiest to understand way to transform data. The program builder reads source and target variable metadata and applies a transformation between the two based on the keyword. An example of this is creating a character SDTM variable from a numeric source variable by applying the appropriate format. The required length, label etc of the SDTM variable are all read from the metadata and included in the created SAS code.

SQL

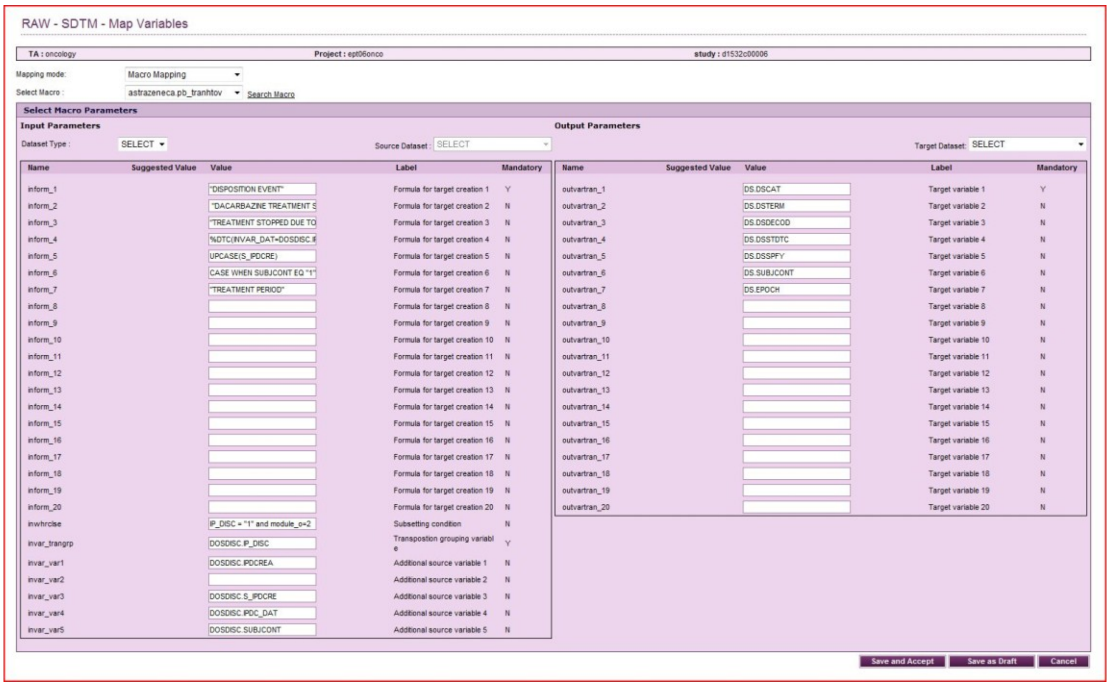
We did not want to have too many keywords so we can also write simple SQL statements which describe a transformation.

Example: `CASE WHEN CXAGENT NE '999' THEN PUT(CXAGENT,$CXAGENT.) ELSE UPCASE(S_CXAGNT) END`

This is stored as plain text in the Oracle database and when the program builder runs it is unable, in the current version, to syntax check these SQL statements. It is important that the user ensures that they write valid SQL. If the SQL code required starts to get too complex we recommend using a macro to do the transformation instead.

Macros

SAS macros are used in those cases where we need to do more complex transformations or to include multiple input or output (or both) variables. We have created many macros at the corporate level which can be used at the project/study level. The user is presented with an input screen in which to enter the macro parameters which will be written into the macro call by the program builder (Figure 4 - The A&RT macro interface).



RAW - SDTM - Map Variables

TA : oncology **Project :** ept06onco **study :** d1532c00006

Mapping mode: Macro Mapping

Select Macro : astrazeneca.pb_tranhtov

Select Macro Parameters

Input Parameters

Dataset Type : SELECT

Source Dataset : SELECT

Name	Suggested Value	Value	Label	Mandatory
inform_1		"DISPOSITION EVENT"	Formula for target creation 1	Y
inform_2		"DACARBAZINE TREATMENT S	Formula for target creation 2	N
inform_3		"TREATMENT STOPPED DUE TO	Formula for target creation 3	N
inform_4		%DTC(INVAR_DAT=DOSDISC.I	Formula for target creation 4	N
inform_5		UPCASE(S_IPDCRE)	Formula for target creation 5	N
inform_6		CASE WHEN SUBJCONT EQ "1"	Formula for target creation 6	N
inform_7		"TREATMENT PERIOD"	Formula for target creation 7	N
inform_8			Formula for target creation 8	N
inform_9			Formula for target creation 9	N
inform_10			Formula for target creation 10	N
inform_11			Formula for target creation 11	N
inform_12			Formula for target creation 12	N
inform_13			Formula for target creation 13	N
inform_14			Formula for target creation 14	N
inform_15			Formula for target creation 15	N
inform_16			Formula for target creation 16	N
inform_17			Formula for target creation 17	N
inform_18			Formula for target creation 18	N
inform_19			Formula for target creation 19	N
inform_20			Formula for target creation 20	N
inwhrclse		IP_DISC = "1" and module_o=2	Subsetting condition	N
invar_trangrp		DOSDISC.IP_DISC	Transpostion grouping variabl e	Y
invar_var1		DOSDISC.IPDCREA	Additional source variable 1	N
invar_var2			Additional source variable 2	N
invar_var3		DOSDISC.S_IPDCRE	Additional source variable 3	N
invar_var4		DOSDISC.IPDC_DAT	Additional source variable 4	N
invar_var5		DOSDISC.SUBJCONT	Additional source variable 5	N

Output Parameters

Target Dataset: SELECT

Name	Suggested Value	Value	Label	Mandatory
outvartran_1		DS.DSCAT	Target variable 1	Y
outvartran_2		DS.DSTERM	Target variable 2	N
outvartran_3		DS.DSDECOD	Target variable 3	N
outvartran_4		DS.DSSTDTC	Target variable 4	N
outvartran_5		DS.DSSPFY	Target variable 5	N
outvartran_6		DS.SUBJCONT	Target variable 6	N
outvartran_7		DS.EPOCH	Target variable 7	N
outvartran_8			Target variable 8	N
outvartran_9			Target variable 9	N
outvartran_10			Target variable 10	N
outvartran_11			Target variable 11	N
outvartran_12			Target variable 12	N
outvartran_13			Target variable 13	N
outvartran_14			Target variable 14	N
outvartran_15			Target variable 15	N
outvartran_16			Target variable 16	N
outvartran_17			Target variable 17	N
outvartran_18			Target variable 18	N
outvartran_19			Target variable 19	N
outvartran_20			Target variable 20	N

Figure 4 - The A&RT macro interface

This input screen is created dynamically when the macro is defined. The A&RT inheritance method also works with macros by checking input and output variables for a match between the source and target data and returning either a full or partial 'map' depending on whether all mandatory parameters are present in the metadata. The status of the inherited map reflects the level of confidence of the system algorithms that the map is complete and correct; suggest implies that the map should be correct and only requires checking.
Partial implies that although all mandatory parameters are available there are some missing optional parameters. In this instance the map may be accepted as-is, edited to add additional parameters or rejected.
If there are no macros present at the corporate level the user can create their own and there are a number of guidance documents available to explain how best to do this.

So, we now have lots of SAS code fragments created by our program builder. It has done some interesting things to do with ordering of code segments based on dependencies identified from within the metadata. It won't try to use a variable as an input if that variable is yet to be created by a different step. Assuming that all of the metadata has been accurately and complete defined, and there are no circular references, the program builder will order all the code fragments in the correct order and produce a number of complete SAS programs.

Execution Time

We now have lots of SAS programs which we need to run to create our database. We decided that most people wanted to have the ability to create that database as a "one stop shop" so the program builder creates a program for each domain and then an additional master program which includes all the domain level programs. It also includes all the setup macros that you would expect to find in such an application to define data locations, SAS autocall libraries etc. we run the program by creating a "job" which is executed on the underlying Solaris box. This job is executed in a special encapsulated shell called a jail which has some special security privileges. It also has a very particular functionality which is exploited in the validation and production environments; it reads all the input data and captures all the outputs and SAS LOG files and creates an archive file which we call a “bundle”. This bundle can be retained as long term storage to allow validation results to be confirmed or entire Clinical study Report outputs to be re-created at any point in the future. We have just mentioned outputs for the Clinical study report for the first time! The database creation piece is our key focus since it relies so heavily on metadata but have we tried to do anything different in our output generation?

Reporting touch points

As you might expect our use of metadata extends to the creation of SAS outputs. We do not use a program generator this time – we have bought a SAS reporting application and embedded it into the A&RT architecture. We do collect information about titles and footnotes; all the usual stuff! We also create logical groupings in preparation for generating PDFs and sub-groups to allow the same program to create multiple versions of the output for different destinations.

We also have a special Output Object Data Store which acts as a link between A&RT and our clinical publishing environment (GEL – Global Electronic Library). A schematic of how the A&RT system interacts with our publishing environment is presented in Figure 5 - A&RT publishing process.

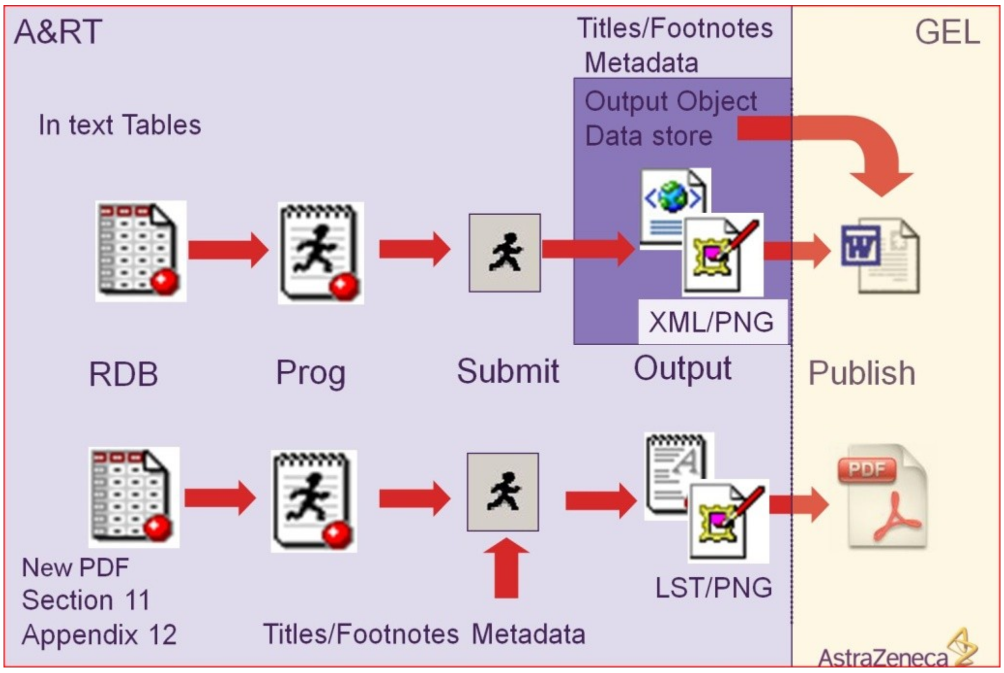


Figure 5 - A&RT publishing process

With this in place our medical communication scientists can pick an XML object representation of a SAS output directly from the data store and insert it into a Word document being edited ‘in-situ’ in our global electronic library. There are obvious advantages in this process in terms of compliance and efficiency gains but another real benefit to AstraZeneca communication scientists is that Word will automatically take care of all the usual formatting and table numbering to conform to AstraZeneca house style for regulatory documents.

When a new version of the output is created its status is updated so the study team know that the word document is now out of date and needs refreshing; just a button touch on a toolbar from within word and the whole document is re-built with the latest tables and figures while remaining in its own, compliant, publishing environment.

The key difference between inserting objects into an existing document and creating a new PDF is that we do not require titles to be included with the XML object. To allow the programming user community to review XML objects with all the title information without having to enter the publishing environment we dynamically attach titles as the object is viewed from within the object data store.

Technology Roundup

To try to make sense of how it all fits together we have a platform diagram showing how the system is underpinned by platform, applications and packages and surfaced through components (Figure 6 - The Composite A&RT Platform).

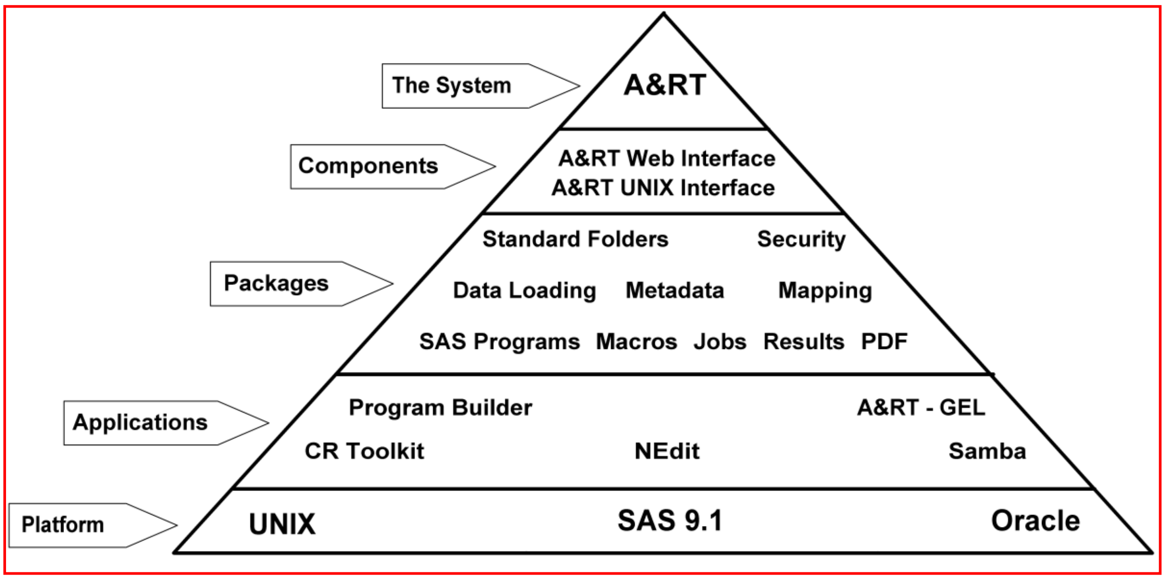


Figure 6 - The Composite A&RT Platform

The A&RT web interface (WI – developed on behalf of AstraZeneca) and the A&RT UNIX interface (SUN Secure Global Desktop) are the user interfaces with the two key parts of A&RT; the processing power of a Solaris machine and the usability of a web based interface. A key point here is that any additional SAS coding activities will currently take place via the UNIX interface rather than the web interface. This is a disconnect in the existing system and a closer integration between the UNIX and web interfaces, to allow SAS code development through the web interface, is high on our 'to-do' list.

Visions for the future

Like all major software applications we have a lifecycle management plan to define the day to day practicalities around maintenance requirements and software upgrades. We also have a dedicated team working with this plan to resource upgrades, schedule patching and conduct testing. We have recently "upgraded" the remit of this team to include the provision of a road map for the A&RT application. We have an enhancements list which is maintained by our governance team and populated by our user community via the global support teams. This is naturally concentrated on improvements to the user experience but we have also introduced an initiative to look at both new technologies and new deliverables which are possible from our existing infrastructure but not yet fully realised or implemented. An example of this is the define.xml document. Our metadata around data and transformations is sufficiently well structured to allow the automatic creation of this key deliverable. This initiative is tied into the governance and standards organisations and has its own internal work stream.

A key advantage of a metadata driven system is the ability to perform impact analysis. This is currently available through direct interrogation of the underlying Oracle database holding the metadata but plans are underway to build this into the interface.

The program builder already has a lot of diagnostic capabilities but we have a number of enhancements planned; additional syntax checking and the ability to check validity of transformations held as text strings together with some added sophistication around the ordering of code fragments.

As mentioned earlier, there is some disconnect between the web interface which controls metadata creation and consumption and the creation of SAS programs and "jobs" and the actual editing of SAS programs. With the current technology, the web interface allows read-only access to all files within the UNIX environment so we can view SAS programs, macros, outputs and logs. What we cannot do is edit the SAS programs – for that we must log-in directly through the UNIX interface. We have a dedicated A&RT

platform team who work to maintain and enhance the supporting platform and high on their list is the evaluation of a code development environment which can be embedded directly in the web interface.

We continue to investigate software and technology advances but we are also evolving our governance and standards organisations and we must remember that process and technology must be evolved together to maximise any benefits.

Conclusion

We wanted to communicate some of the challenges which led to the creation of the A&RT system and some of the thought processes which helped us define the system as it exists today. Along the way, we hoped to identify some of the benefits in using metadata to drive the clinical reporting process and illustrate how the technology can and must act as a solid platform but needs to work in harmony with the process. The provision of metadata driven reporting is of little practical use without solid and consistent data standards together with the relevant governance structures to define, maintain, enhance and police those standards. The use of metadata and pre-packaged SAS fragments to allow the automatic generation of database creating SAS code is a paradigm shift in how AstraZeneca approaches this process but must be underpinned by the same governance organisation to ensure those transformations are aligned to the data and that they evolve as the data standards evolve. None of this would enable us to "…streamline the process and minimise resource intensive activities on the critical path to submission" without the ability to inherit all of our data definitions and transformations from a controlled corporate layer down through a Therapeutic Area level and eventually to Project and Study. This inheritance is enriched by allowing the use of equivalence which not only allows differently named datasets to be mapped to an equivalent but also leverages the program builder to dynamically change data references within the standard transformations.

It seems like a long time ago that we embarked down this road but since implementation started in earnest in 2009 we have started to materially change the way we do our business and as the governance and process organisations continue to evolve we are in great shape to continue to increase efficiencies and make good on our promise to the wider AstraZeneca business to deliver more with less.

Contact Information

Your comments and questions are valued and encouraged. Contact the author at:

Paul A. Frost – SAS Consultant
AstraZeneca
Clinical Information Sciences (Programming)
90s6-4
Parklands, Alderley Park
Macclesfield
Cheshire
SK10 4TG

Telephone: +44 (0)1625 519375
Fax: +44 (0)1625 519375
Email: paul.frost@astrazeneca.com

Endnotes

[i] Hierarchy - an arrangement of items (objects, names, values, categories, etc.) in which the items are represented as being "above," "below," or "at the same level as" one another and with only one "neighbour" above and below each level

[ii] Equivalence - assigning one object to another
[iii] Inheritance - a way to form new classes (instances of which are called objects) using classes that have already been defined. Inheritance is intended to help reuse existing code with little or no modification.