Data Acquisition System Design for ESS October 2012

Mark Könnecke
Paul Scherrer Institut

Switzerland

Introduction

This is a design study for a data acquisition system for the ESS spallation neutron source. This design study is based on the ESS DAQ requirements document and a comparison of existing data acquisition solutions detailed in another separate document.

The term data acquisition means different things to different people. Thus it is necessary to define how this term is used in this document. For the authors data acquisition software includes:

- The actual collection and processing of neutron event data
- The slow controls needed to tune the instrument to various scientific use cases
- Sample environment controls
- Data reduction to a level where a scientific model can be fitted to the data. In other facilities, this task is left to the scientists. But data rates at ESS will be such that professional programming techniques and parallel processing will be required to cope. With this, scientists need help.

Recapitulation of Requirements

An extended study of the ESS data acquisition requirements can be found in another document. This is just a recapitulation as an introduction to this design document.

The most challenging aspect of ESS data acquisition is the requirement to process 10^8 or possibly 10^9 neutron events/second. In addition, neutron event data needs to be correlated with other information like chopper values or sample environment parameters. Which means that such data has to be collected as fast as possible up to a rate matching the timescale of neutron data collection. This also means that a timing system needs to exist which allows different data collectors to use comparable time stamps.

Beyond the neutron data acquisition aspect, the ESS data acquisition software requirements are quite normal. The most important points:

- Stable operation over long periods of time
- Long term maintainability
- It should be value for money
- Open to changes in requirements and hardware
- Friendly user interface
- Work flow ready
- Drive real instruments and different kinds of simulations
- Online display of data as it gets collected
- Include data reduction to physical quantities

For a more detailed requirement analysis, see the ESS DAQ requirements document.

Design Considerations

In the following sections four approaches for the ESS DAQ software will be presented and discussed. This section documents some of the thoughts which went into these suggestions. This section will also cover aspects which are common to all four approaches.

Histogram versus Event Mode

There are two basic modes of doing neutron data acquisition:

- Event mode: In event mode the position and time stamp of each neutron event is stored. All processing happens later. Event mode has the advantage that correlations and time binnings can freely be done after the experiment.
- Histogram mode: in this mode neutron events are summed into data arrays already early on in the electronics. But the time binning cannot be changed anymore once the histogramming process has been configured.

It is clear that all ESS instruments need to be able to run in event mode in order to support certain kinds of dynamic experiments. However, event mode can come at a cost in terms of storage infrastructure and later processing effort. Several cases have to be distinguished:

- For large detector arrays aiming at covering 4 pi and weak scattering, event mode may actually save disk space. Histograms would be very large and sparse and bigger on file. Event mode is recommended here.
- For instruments with medium scattering rates event mode will imply bigger data files. But not so big as to pose a significant problem. Event mode can be the standard mode of operation for such instruments in order to take advantage of the possibilities for later post processing realised by event mode.
- Then there will be instruments with strong scattering and a lot of data being collected in a relatively small detector. Here event mode may imply a significant storage overhead and histogram mode may actually save money on storage infrastructure and processing overhead. The ESS management has to decide how to handle such cases. The arguments would be largely ones of economy here. ESS could also decide to run event mode throughout in order to have a homogeneous

processing pipeline.

In the following text it is assumed that most ESS instruments will run in event mode.

Neutron Data Acquisition

For neutron data acquisition the author suggests the new solution currently being developed at SNS. This system is known as ADARA. This solution consists of:

- A Streaming Management Service (SMS)
- A Streaming Translation Service
- A Streaming Reduction Service
- A histogram service

The heart of the system is the Streaming Management Service (SMS). The SMS:

- Collects neutron event data from the electronics
- Collects auxiliary inputs from slow controls and the DAQ system
- Packages all that information into a combined event data stream
- Forwards the combined event data stream via TCP/IP to connected clients

One notable client to the SMS is the Streaming Translation Service(STS). This service receives the data from the SMS and writes it to an event NeXus file on a parallel file system.

The Streaming Reduction Service(SRS) is basically a copy of Mantid which reads the combined event stream from the SMS and directly corrects and reduces the data. Mantid is a component based data analysis framework developed jointly between ISIS and SNS. Mantid has been heavily parallelized in order to enable it to process more then 12 Million events/second. And there is still room for further improvements.

Data for online live displays is generated by another, differently configured, copy of Mantid connected to the SMS.

The adoption of this system minimizes the effort required for ESS: The only thing ESS really needs to do is to implement a new SMS which reads data from the ESS electronics and slow control system and packages the data into a compatible event data stream to publish to clients.

Data reduction and online display are handled by Mantid. Mantid is already an impressive product at the time of writing (10/2012) and will mature even more until ESS becomes operational. Mantid is jointly being developed by SNS and ISIS. Which implies it has strong support. It is expected that ESS will only have to configure Mantid for its use; may be write a couple of additional modules. Some glue code will be required to integrate Mantid pipelines into the overall data acquisition software.

This solution also implies that ESS will write ISIS/SNS style NeXus-HDF5 event data files as raw data. For which, with Mantid, a data processing tool exists.

Neutron Data Transfer and Timing

The real problem with neutron data acquisition is to get the neutron event data out of the electronics through some communication link. This step has to be optimized. The first step in this optimization is to reduce the amount of data to transfer per neutron event. These forces imply to adopt the SNS/ISIS style of handling neutron event transfer:

- Per event: 4 byte detector ID, 4 byte relative time stamp
- In addition: a per pulse event to denote new data frames

This also implies to handle all the timing through the use of the pulse signal from the neutron source. It is advisable to send an absolute time (64 bit) per pulse. The necessary infrastructure has to be provided to forward this pulse signal to any interested party in the experimental halls.

With this scheme and an 10 GB ethernet link 10^8 neutrons/second can be handled. If the need arises to go faster, there are two ways to scale:

- Use a faster communication link: 40 or 100 GB ethernet or Infiniband may be candidates. The stability of such links has to be evaluated.
- Use slower links in parallel by separating the detectors into banks with individual communication links. This is easy as neutron event data can easily be processed in parallel.

Neutron DAQ Electronics

This section may be a bit of trespassing as electronics is clearly the task of a different group in ESS. But as part of looking at DAQ the author also looked at neutron DAQ electronics a little and a pattern emerged. Moreover a few demands on the neutron detection electronics should be stated.

First the pattern: it appears to be industry practice to use a small number of standard electronic cards/components to cover all detectors. Differences in detector types are accommodated for by detector specific I/O cards which in themselves do very little. This approach reduces cost by producing electronics in larger quantities and facilitates maintenance.

Second some requests for the electronics:

- The electronics should have some memory of its own in order to even out neutron event bursts. Due to the pulsed nature of ESS many neutron events might arrive in a very short period of time, faster than they can be transferred. But then there is a short break between pulses in which buffered data can be transferred.
- Great care has to be taken to have a well performing uplink from the electronics to the data acquisition computers. Ethernet technology is advisable because of being well known and ubiquitous. The fastest, reliable technology should be choosen here. It may be a good idea to make the uplink exchangeable when better technology arrives.
- It should be possible to add user I/O through special cards into the event stream. This in order to be able to correlate neutron events with other data.

- The ISIS electronics puts the timing input for a detector bank or even for the whole detector onto one input. Even if several cards are required to process the detector. The advantage of this approach is that possible clock skews and flightpath lengths problems can be resolved by calibrating against a well known sample.

Slow Control

For the slow control and coordination part of the control system the first choice to be made is between a distributed control system or a central server/single process control system. Central server/ single server control systems tend to be more compact and usually require less manpower to maintain. However such control systems hit a limit when instruments become too large. Then distributed control systems excel: they scale very well to very large installations. This comes at a cost however: such systems require more manpower to maintain.

For ESS, with its instruments which can physically extend across hundreds of meters a distributed control system will be recommended.

When selecting a distributed control it is wise to join one of the large collaborations. This has the following advantages:

1. Share driver code
2. Share expertise
3. Leverage a well tested foundation
4. Others will have found and solved all the gotchas you cannot even imagine
5. Increase staff exchange possibilities

There are two large collaborations to choose from: EPICS or TANGO. Both will do. ESS is in the favorable situation to be able to choose freely:

- EPICS knowledge is available on site as the ESS accelerator group has already decided upon EPICS
- With Maxlab a facility using TANGO is on the same campus

Even with a distributed control system it is advisable to keep the number of computers on which it runs low in order to lower the maintenance load. A modern computer is capable of running several device servers comfortably. Add additional computers only when necessary for balancing system load or for reasons of locality.

One of the drawbacks of a distributed control system is, well, that it is distributed. Left alone, there is only a bunch of device servers running on different computers. A user should not see this. What is needed is a middleware which pulls all the different parts together and presents them to the user. Let us call this middleware a sequencer for the purpose of this document. A suitable sequencer allows the user to discover and configure all the devices available at the instrument. It also allows to discover and use available measurement procedures and edit experiment meta data. It is highly recommended to use just one type of sequencer for all ESS instruments. Maintaining multiple different sequencers will become a maintenance nightmare. It will also make the life of users more

miserable as they will encounter different systems and user interfaces on each instrument.

Given a sequencer added to the distributed control system, the question arises how the division of labor should be between device servers and sequencer. The recommendation is to use device servers for hardware access and components shared between groups of instruments. Instrument specific stuff should be implemented as script programs in the sequencer.

Maximum benefit can be achieved from using a collaborative control system when the availability of a suitable driver for the control system of choice is one of the criteria for decision making when acquiring hardware.

Network and System Setup

Network

It is a good idea to give each instrument its own TCP/IP network segment consisting of one or more TCP/IP subnets. The instrument network ought to be protected by a firewall. Instrument subnets have the following advantages:

3. Isolation. A rogue network device in one network will not bring down all instruments. Accidental cross talk between instruments is minimized.
4. Security. Sometimes, special computers are needed in DAQ which cannot be subjected to rigorous security update schedules as rightly demanded by network security officers in a public network.

Access to an instrument in an instrument network should happen exclusively through a designated instrument computer. This computers task, besides the interface task, is to coordinate data acquisition.

Due to the large distances foreseen at ESS it may be necessary to have a designated chopper network segment which is only to be accessed by the instrument networks.

Due to the large amounts of data expected at ESS the best and fastest network equipment available should be acquired. Just one level below the bleeding edge.

Storage

Currently there is some uncertainty where the storage for ESS will be located. For political reasons the idea is to situate data management at the Danish Data Management Center in Copenhagen, 40 km away from the ESS at Lund. The last state of the discussion is that data will be mirrored between Copenhagen and Lund. This is very expensive as two storage centers need to be built and maintained. On the other hand it is impractical to transfer all data to Copenhagen before doing any processing. Then ESS operations would be interrupted if something happens to the data link to Copenhagen. Thus the minimum sensible solution is to have a local cache at Lund which is capable to hold at least 3 months of data. Then data processing can happen on local data while the experiment is running. But all further processing can be in Copenhagen.

At the current state of affairs where the actual instruments to be built are not decided upon and it is not clear how much detector area will be affordable it is difficult to prognose the amount of storage to be provided. Another aspect is that the amount of data to be produced will grow slowly as instruments come online. As a guess I would expect:

- 100TB/year in the first three years
- 100-200TB/year in the following 3 years
- 300 - 500TB/year when all instruments are finished

The suggested approach is to design for several petabytes of storage, start with a petabyte and add storage as needed. How much storage will be required in total is directly dependent on a policy decision the ESS has to make: The data retention policy. This is how long experimental data from ESS is to be kept in storage and in which form: online, on tape, raw data, reduced data?

Something which is easily forgotten is that the budget for the data management center must include the fact that the storage infrastructure has to be renewed every 3-5 years.

Operating Systems

The preferred operating system for servers and data acquisition computers will be Linux. Everything else will be more expensive, not only in license costs but also in maintenance and not be stable enough. Linux is also the industry standard for data acquisition.

In terms of client access to the DAQ system, all common operating systems should be supported.

Tango Based Solution

Of the two available distributed control systems the recommendation is to use a combination of TANGO and Sardana for ESS. A couple of reasons:

- TANGO compared to EPICS is the more modern system, based on object oriented technology.
- TANGO has a flatter learning curve than EPICS. Apparently a half weeks course is enough for a reasonably competent programmer to become productive.
- TANGO device servers can be programmed in C++, Java and python. This broad range of languages helps when hiring staff. With python, a scripting language is included. This allows to provide or prototype device servers very rapidly.
- TANGO has a more open architecture than EPICS. Thus it will be easier to integrate neutron data acquisition into the system.
- TANGO can be configured at one central point, the TANGO database
- Sardana adds a powerful sequencer, scripting and an extendable GUI to TANGO
- Sardana provides its own device abstractions. This is an additional path for very rapidly providing hardware access for lesser or user provided hardware.
- Sardana has originally been developed at ALBA. But it has a growing user base at other synchrotrons.
- Choosing TANGO and Sardana opens up the possibility to provide a similar user

experience for users of ESS and Maxlab.

For neutron data acquisition the SNS system described above is to be used.

The following things need to be done by ESS when adopting TANGO/Sardana

- Learn TANGO/Sardana
- Integrate neutron data acquisition
- Integrate online data display
- Integrate data reduction
- Deploy the system

This is about the minimum required. As an option, ESS could consider to develop a more NOMAD like user interface on top of the Sardana servers.

EPICS Based Solution

As the ESS accelerator division has already decided to use EPICS, there is a strong drive to use EPICS for the ESS instruments too. There is no question that this can be made to work. A disadvantage is the steeper learning curve of EPICS and the more restricted choice of programming languages. EPICS supports only C/C++. Which is a good choice but not the fastest language to develop in. Another drawback is the distributed configuration of an EPICS system.

The other problem is that a convincing standard middleware on top of EPICS is missing. Such a middleware is necessary at instruments to allow the casual user to discover the available devices at the instruments and the available measurement procedures. And not only discover but also configure and use them. There are candidate solutions for such a middleware:

- One is the GDA from Diamond. However, from a trial undertaken at SLS it is known that the GDA as it is now has code management issues and is very difficult to build. Some abstractions are missing. Thus it will be costly in terms of man power to adopt GDA.
- It is possible to make Sardana work with EPICS IOC's. However then there are two network protocols, CORBA and EPICS CA to look after and this may not be the most stable solution.
- The SNS is developing a scan server. This is promising but not stable yet. Also the device discovery aspect is missing.

Summing it up, ESS will have to invest into the development of a suitable middleware when choosing EPICS. ESS may not need to do this alone. Possible partners include SNS, SINQ and SLS at PSI and ISIS when ISIS decides to go for EPICS as their next control system. Such a development is, of course, a risk.

Thus adopting EPICS means for ESS:

- Learn EPICS
- Integrate neutron data acquisition

- Integrate online data display
- Integrate data reduction
- I advise to develop a EPICS-IOC scripting adapter which would allow to script both device drivers and other EPICS record types.
- Develop a suitable middleware for instruments
- Devise a procedure for managing the distributed configuration files of an EPICS system.
- Deploy the system

Ideas for a new Middleware

This section contains a sketch how a suitable middleware could look like. This section draws on ideas from Sardana and other control systems. Logically, such a middleware could consist of the following components:

A Device Pool

> This component holds all the instruments available hardware and pseudo devices. It enables the user to configure and monitor devices. Some instruments have variable hardware; thus an interface to enable and disable devices must be provided. This component also provides device abstractions for common hardware. Thus it fixes one of the shortcomings of EPICS. And own device abstraction at this level also opens a back door through which to implement hardware access quickly when this is necessary. Having stated this, it should be clear that EPICS would be the preferred hardware access solution. This component should also have some query facilities in order to locate devices with specific characteristics.

A Macro Service

> This component is responsible for measurement procedures and system and user batch files. The prime concern of this component is locating and executing such codes. Facilities for editing and loading scripts are another requirement. This component allows the user to browse for existing procedures and to view documentation and parameters.

A Data Service

> This service holds experiments, samples, the data and data files and logs collected in measurements. This service serves as the single entry point for clients to locate data.

Some properties of this approach:

- The approach implements a common division in scientific software: an algorithm (measurement procedure) is run against some data structure (device pool)
- Simulation modes can be accounted for by having different device pools and just manipulate the address to the device pool.

Implementation wise the services described above should be backed by a NoSQL or SQL database. The data base is to hold configuration information, values for bump less restarts and whatever else needs to be stored persistently. It can be discussed if the macro service should not be split into two parts: the actual macro service and a queue service. The

macro services then would compile scripts or procedures into an intermediate form, a domain specific language (DSL). The queue server would then execute blocks of such DSL statements. The advantage would be that measurement procedures would not need to bother anymore with many low level details. Many checks can already be done in this cross compiling stage in order to ensure that the procedure or script actually runs without error. The queue server would open up another way to implement simulation modes. The disadvantage is additional implementation overhead.

WWW-Service Based Solution

This solution is the development of a data acquisition system based on WWW technologies. WWW technologies in themselves are mature and well proven. Another advantage of this approach is user friendliness: it is achievable to run the system from a WWW browser as well as from computer clients through a WWW-service API.

Modern WWW-service are built around the REST architecture: REST means REpresentational State Transfer. REST services are built around the Resource Oriented Infrastructure (ROI) paradigm. This paradigm is based around the following statements: Resource Oriented Architecture (ROI)

- Everything is a resource
- Address everything by a unique URL
- Statelessness: every request is on its own
- Connectedness: resources are linked with each other
- Uniform interface: Do everything through HTTP GET/POST/PUT/DELETE methods. More clearly this maps to: CREATE/RETRIEVE/UPDATE/DELETE (CRUD)

Applied to data acquisition this becomes:

- Every device or parameter has its own URL
- Every DAQ operation becomes a resource
- Driving/counting: logs of what happened
- Scans/Counts obviously create resources (data files)
- An experiment is a resource too
- The user becomes something to link to

This is a very neat way to organize an instrument and data acquisition. This option made it into this report because ESS expressed an interest in this and because ESS may be ambitious enough to go beyond what is commonly done today. Besides a very little prototype at PSI, this has never been tried before. At SINQ at PSI there is also some very good experience in using HTTP servers for device support. But a large body of code would need to be developed.

In order to develop this ESS would need to develop:

- A REST device server infrastructure
- Develop all drivers
- A REST middleware and scripting service

- Test and refine the whole system at a test instrument
- Deploy the system to ESS instruments

As a large body of code, standards and procedures would need to be developed this is the most risky option.

Distributed Control System plus WWW Middleware

Another medium risk option is to use a distributed control system like TANGO or EPICS as basis and add a WWW based middleware on top. The latter would need to be developed. It is expected that the integration steps in the task list are easier to solve with such a middleware. And the distributed control system provides for a solid basis.

What ESS needs to do then comes down to:

- Learn the distributed control system
- Integrate neutron data acquisition
- Integrate online data display
- Integrate data reduction
- Develop a WWW middleware on top of the distributed control system. This is a major development effort.
- Devise a procedure for managing the distributed configuration files of an EPICS system.
- Deploy the system

There is no reason why the middleware described in the section above cannot be implemented as a REST WWW-service.

Resources

Hardware for running the DAQ System

In terms of computers ESS will need for each instrument:

- A server class instrument computer to serve as interface to the outside world and for coordinating data acquisition. This computer will also run most device servers.
- A server class computer per detector bank for online data reduction and analysis
- Further computers to run device servers as necessary

At current (2012) prices a good estimate is 20K Euro/instrument for computers. A few high data rate instruments may require more.

Concerning the cost for storage it is very difficult to make estimates at this time. More information on the expected data rates, the actual instruments to be built and the size and nature of the detectors which can be afforded needs to be known. Taking the experiences at SNS as an example ESS needs to be able to write in excess of 4-5 GB/sec of data. Storage systems capable of handling such data rates are not cheap. And require a network infrastructure to match too. At 2012 prices suitable storage systems come at 800K Euro per 500TB. In order to give an estimate I need to make some assumptions:

- Cost of storage: 800K Euro/ 500TB
- Disk based storage
- A data rate of 200TB/year during commissioning of ESS for the first 5 years
- A data rate of 500TB/year for the following 5 years
- A 10 year data retention policy
- Short term storage at Lund
- Long term storage in Copenhagen

Then ESS arrives at storage costs of:

- 400K Euro in Lund for 200TB temporary storage
- 4PB Storage in Copenhagen: 6400 K Euro

If Lund implements a mirror of the Copenhagen data center, then replace the 400 K Euro for Lund by 6400 K Euro. Given these number ESS is well advised to invest into research on better compression algorithms for event data.

Some money can be saved when older data is stored on tape rather then on disk. The drawback is that it may take a couple of minutes or hours to retrieve data from a tape library. A petabyte tape storage is 200 K Euro/year. If we now assume 1PB on disk storage and 4PB tape storage then we end up with a storage cost of 2200 K Euro for storage. The distribution of storage between tape and disk storage is a political and economical decision to be made by ESS.

Add to these numbers the cost of buildings, electricity supply, cooling etc.

Required Staff

From the numbers given by other facilities I recommend that ESS should strive to have one data acquisition software person per two instruments. This would make ten productive personnel plus administrative overhead in the form of a boss and a secretary, twelve people all together. Staff for scientific software development is a difficult issue: People with a degree in informatics usually know how to write software professionally but lack science knowledge. Scientists moving into computing understand the science but lack knowledge of professional software development technologies. Both skill sets are necessary for successful scientific software development. Thus in an ideal team both sides are represented and support each other.

All team members should be able to tackle all tasks. But of the ten productive persons:

- Two should specialize in system management
- Two should be able to configure and develop for Mantid
- Two should be able to write device drivers
- Two should specialize in setting up and maintaining a DAQ infrastructure

- Two should specialize in user interfaces

This is the staff for data acquisition alone. The assumption is that basic office support, network management and basic services is provided by another group.

Risks

From a computing point of view the data acquisition requirements of ESS can be met. Especially if we take into account technological advances in the next few years. Risks associated with DAQ are mainly in the management domain. Known risks include:

Failure to standardize Hardware and Software

This would cause extra delays through the need to write additional device drivers. Depending on the choosen technology this can be bad to very bad. Moreover a failure to standardize will also increase the software maintenance load throughout the whole operating time of ESS. The risk of standardization failure is high. Other facilities running contributed instruments by different partners found it very hard to enforce standardization and failed in this respect. ESS is in a double bind here:

- ESS needs external partners for building instruments and political support. This limits ESS ability to enforce issues.
- The temptation for instrument builders to cut corners by using cheaper non standard components and hiring temporary staff are very high.
- What will happen when a partner brings an instrument with non standard parts to ESS? Will she be sent home? Or, more likely, will the technical staff be asked to make it work?

Unavailability of suitable personnel

Depending on the economic situation it may be difficult to hire suitable software staff.

Unattractive work environment

Care has to be taken that ESS creates an attractive work environment for computing staff. Otherwise frustration and high staff turnover may occur. This can break any schedule.

Political technology decisions

It is not very likely but nevertheless possible that for some political reasons technical decisions like the use of commercial operating systems or commercial programming environments are forced upon ESS. Such decisions can add powers of ten both in schedule and cost. And may result in a system with reduced functionality.

Wrong choice of technology

There is a risk that based on this document a bad decision is being made.

Failure to develop suitable software

Software projects do fail. Many examples in business and science stand in evidence for this statement. ESS can protect itself against this risk by starting development

early and run prototype implementations on instruments at partner facilities.

Summary

Though the neutron flux at ESS poses some challenges, the data acquisition needs of the ESS can be met with current technology.

The most important point to stress is that a uniform system across all instruments is to be implemented. Everything else will become a maintenance nightmare.

A summary of the recommendations:

- Use event mode data acquisition wherever appropriate
- Use the SNS solution for neutron data acquisition and reduction.
- Use the NeXus SNS/ISIS event storage format for raw data files.
- Standardize detector electronics as far as possible
- Use a distributed control system like TANGO or EPICS as a basis.
- Use the combination TANGO/Sardana to implement data acquisition. This is the low risk option as most required software is already available.

Using EPICS requires investment into the development of a suitable sequencer for instrument integration. This is a medium risk option for ESS data acquisition. The sequencer could be a WWW middleware.

If ESS decides to be ambitious and go beyond what is currently being done, ESS would consider the WWW service based DAQ option. This option poses some risk and it would be advisable to build a prototype fast and test it on an actual instrument. Based on the outcome of such a test ESS can still decide to fall back on one of the safer options.

Decisions to me made by the ESS

1. Policy regarding event mode versus histogram mode
2. Select an option for a control system
3. Long term data retention policy
4. Decide upon storage location and layout

Information needed from ESS for more detailed Planning

- Detailed information about the instruments which will be built
- Instrument simulations of expected data rates
- Instrument simulations of the reachable resolution on the detector
- Information about the neutron detection efficiency of the detectors which will be choosen
- Data retention policy

The Next Steps

There are two paths along which this project can continue:

1. Develop and test a prototype for a to be specified instrument using both TANGO and EPICS. Unfortunately a comparison of DAQ systems has to rely on input from other people. But the real issues and problems with any system can only be found when trying to apply them to a problem at hand. This step aims at getting better data on learning curves and experience on both systems.
2. Build a prototype WWW service based middleware on top of either TANGO or EPICS.

Either step can help ESS to base their decision on a data acquisition software on experimental data and not only a comparison alone. Both tests ought to be run against two instruments:

- A simple scanning one as a proof of concept.
- A more complicated one to find another set of hidden problems