NeXus Quo Vadis

What is this?

NeXus is in existence since around 1996. This is 16 years ago. The author thinks it is high time to review the state of NeXus and have a strategic discussion on how we continue with NeXus. Times have changed and I think there are some experiences and developments which can change how we deal with data and NeXus. This strategic discussion is meant is to be held at the full NIAC meeting in september 2012. The result of this discussion can very well be that we continue how we started in 1996. But it can also be that NeXus changes.

I like to assert that I feel bound by the descisions made so far by the NIAC. Even if I write a document aimed at starting a rethink of NeXus.

Access Restriction

In order to avoid confusion among NeXus users this document is for the eyes of NIAC members only.

Review of NeXus

NeXus defines rules for structuring and storing data on top of a physical file format. Which happened to be HDF-4 at the time of conception. NeXus moved to support the newer version of HDF, HDF-5, and XML as physical file formats too.

One of the NeXus aims is to store as much information possible on any given experiment in the data file. Thus NeXus developed structures to allow for:

- Multiple experiments in one file.
- A full description of the beamline in addition to the raw data in the file.
- Easy plotting of data.

In the last years application definitions were developed which define which parameters have to be in a data file for a certain use case of NeXus. Use cases can be raw data file for instruments of a particular type or processed data of various types. Application definitions allow the expression of strict and verifiable standards.

Much effort was expended by the NeXus group to develop and define the NeXus API(NAPI). The original business case of the NeXus-API was to protect users from the complex API's of HDF-4. Later, the NAPI acquired the task to protect users from the different physical file formats on which NeXus was based. Over time, the NAPI collected more and more bindings to different programming languages and utilities for dealing with NeXus files.

The uptake of NeXus in the scientific community has been unsatisfactorily sloooowww. Newly developed instruments or software tends to use NeXus for file formats. The trouble is that few things are newly developed. Also, the NeXus files generated at different facilities implement NeXus at varying degrees of compliance. A recent development which gives rise to some hope is that the company Dectris will start writing Eiger detector data files in HDF-5 using NeXus conventions, probably in 2013.

NeXus Problems

In this section I will discuss some of the problems which are currently being experienced with NeXus. And make educated guesses on the causes of the problems.

NeXus Adoption

In order to adopt NeXus an energy curve as seen in Figure 1 needs to be passed.

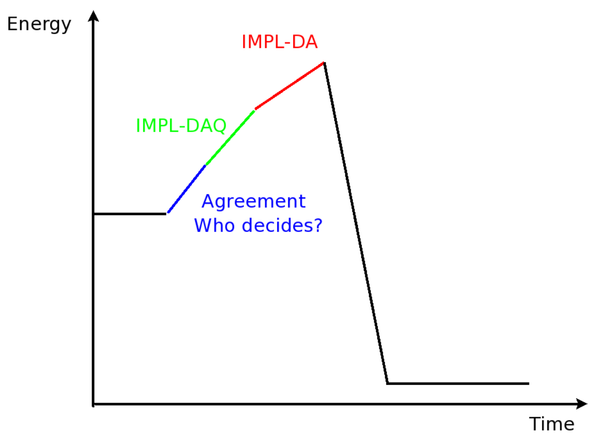

Figure 1: Stages and activation energies for adopting NeXus.

Thus NeXus adoptions requires deveral stages before the full benefits of NeXus can be realized:

1. The community has to agree on a standard.
2. The standard has to be implemented in data acquisition systems.
3. The standard has to be implemented in data analysis software.

Already the first stage, agreement in the scientific community, is a difficult thing to achieve. There are various factors which contribute to this:

- There exist no governing bodies in science with which standards can be negotiated or who are capable of enforcing standards.
- Most scientists are just not interested in anything computing. They wish to do their science. There is only a small group of scientists who have an interest in computing at all.
- Standardization would make life easier for those people who actually have to solve problems with data access. These are either graduate students or computing staff. Both groups have little political clout.

The second stage, implementation of NeXus, is also difficult. We observe that there are little resources for tackling scientific computing issues. This has a couple of reasons:

- The organisations which run science are valued according to the number of publications in high impact journals. Scientific computing is a necessity but clearly on the cost side. Which has to be minimsed. Which means that there is only minimal computing staff at many facilities.
- Not everyone is suitable to do scientific computing. Computer science graduates lack the science knowledge and tend to love to play with tools. Scientists going into computing often lack the knowledge necessary to write professional grade software.

Thus scientific computing resources are scarce. Implementing a standard directly competes with more sexy tasks like implementing that new data analysis algorithm or getting that instrument online. Often standards get neglected then.

The situation is further complicated by the fact that NeXus has reached a level of complexity which requires an experienced person to deal with it. NAPI requires many libraries. The NeXus structures try to solve a great variety of problems. This makes the NeXus rule set large and complex.

Conflicting Aims in the NeXus Design

With the advent of NeXus application definitions it became apparent that there are conflicting aims within the NeXus design:

- NeXus structures were designed to store a full beamline description (FBD). Which can easily mean hundreds of parameters. The FBD is desirable to fulfill some secondary issues when storing data: scientific integrity and the ability to deduce problems with the data from the data file.

- NeXus application definitions define the strict standards for exchanging data with other applications. Typically a NeXus application definition consists of 10 - 30 parameters, with the average probably close to 15. For this number of parameters the NeXus file structure is overkill.

Thus if the only aim is to exchange data between applications, then NeXus is not minimal.

Are NeXus Data Files Still Timely?

There have been developments in the way experiments are conducted and technical developments which affect the way data files are written and can be used.

Let us start with the technical aspect first. When NeXus was designed in 1996, a 40GB disk array was a large and expensive thing. Today the same amount of data can be fitted on a finger length USB drive. With the disk space available in 1996, a snapshot of the instrument at the time of the experiment was the only thing which could conceivably be stored. Fast forward to 2012 and cheap terabyte disks. Now it is possible to take a snapshot of the instrument at the start of an investigation and log each and any command, parameter change etc. with time stamps. This is desirable as it captures completely how the experiments were performed. Data files would then be extracted from this log to interface with data analysis software at certain time intervalls. I do not know yet of any experiment which is run like this but it is possible.

The last couple of years brought on changes how experiments are being performed:

- At neutron sources, event based experiments and data storage becomes increasingly popular. NeXus covers this with the NXeventdata but this aproach is not general.
- With the advent of high speed detectors at X-ray sources, increasingly dynamic experiments are being performed. This means that some parameter is varied and images taken on the fly. Often at high data rates and with different sampling rates. This raises again the requirement to store time based data.
- The free electron lasers currently being built will definitly ask for time stamped data storage.

Summarizing, there is a need to find a general way to deal with time stamped data in the future. This raises some questions:

1. Does NeXus whish to enter this market?
2. How would such data be stored? Make every NeXus field a NXlog?

Dictionary Based Programming Techniques

HDF-5 made a new file reading technique possible which I call dictionary based reading. This is the underlying technique of the Common Data Modell aproach from Soleil. This technique makes use of the fact that any object in an HDF-5 file is addressable by its path into the file hierarchy. Now, rather then hard coding the paths to the data items needed into the reading program, a programmer can choose to externalize these paths into a separate file in some form. This is the dictionary. Then the reading program can deal with any file containing the necessary data items. The only modification needed would be an edit of the dictionary file.

This programming technique goes a long way to remove the need to have predetermined names and hierarchies in a file.

Object Oriented NeXus

It is accepted that the NeXus base classes would become much clearer and easier to document if object oriented concepts like interfaces and inheritance would be introduced. The problem is that this breaks backwards compatability. The other problem is how to map a more refined base class description onto NeXus file structures. They are already complicated as is.

NeXus Application Programmers Interface

There is feedback from the community that the only physical file format they really want is HDF-5. This eliminates one of the main causes of the existence of NAPI. The HDF group was not lazy either and there now exist higher level HDF-5 API's which make it much easier to use HDF-5 files directly. This eliminates the second main business case for NAPI, the protection against the complex HDF-API's. This raises the question: What do we do with NAPI?

Options for NeXus

There are a variety of options for how we proceed with NeXus.

Hold On

The first obvious option is to continue NeXus as is. There are many good reasons to do so. But we ought to decide upon this consciously and not by drifting. If we decide to go this way, other questions immediatly need to be adressed:

1. Does NeXus deal with time stamped data?
2. If so, how do we do it?
3. What do we do with NAPI?
4. Do we investigate a way to implement object oriented NeXus base classes?
5. How do we address questions/criticism about NeXus complexity?

NeXus-Ultralight

Of course there is the option to simplify NeXus to various degrees. This is the ultra light version which is close to giving up. This would be:

1. Require to use HDF-5
2. Use a dictionary based approach for file reading
3. Convert the NeXus aims into a Data File Best Practices Document. Which need to include: document what you do with your data file.

NeXus-Armando-Light

A better name is required here. The Armando is for Armando Sole who more or less suggested this aproach if I read him correctly. This is another version of a lighter NeXus. I believe the approach suggested here would formalize a practice of using NeXus which is common at synchrotron sources. This version looks like this:

1. Use HDF-5 as a file format
2. Store your data in a simplified hierarchy using NXentry and NXcollection groups. Apply a lot of freedom how you do this.
3. Have NXmethod groups at NXentry level which contain links to those data items required for a specific use of the file. For example a group NXsas would contain links to all data items required for small angle scattering analysis. The content of the NXmethod groups can be derived from the existing application definitions. Just flatten the application definition hierarchy and resolve name conflicts by prefixing.

Your Option Here

Summary

This is just my views and ideas. This is also an invitation to engage in a discussion into the future of data and NeXus. Please speak up with your opinions. The only thing I am really sure about is that we should have a discussion about the future of NeXus in a changing environment.