# UAL-BASED SIMULATION ENVIRONMENT FOR SPALLATION NEUTRON SOURCE RING [*]

N. Malitsky[†], J. Smith, J.Wei, BNL, Upton, NY
R. Talman, Cornell University, Ithaca, NY

*Abstract*

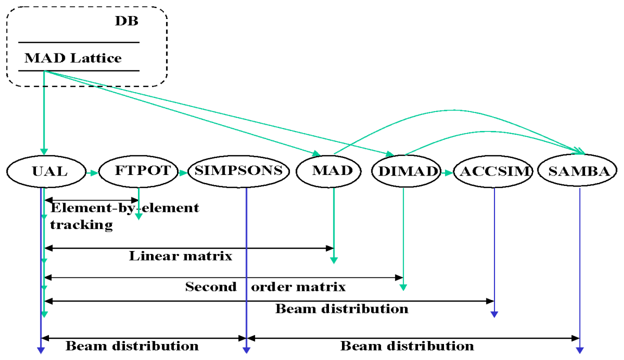
This paper outlines the major activities and applications of the Unified Accelerator Library environment for the Spallation Neutron Source (SNS) Ring.

## 1 BACKGROUND

The Unified Accelerator Libraries[1] are designed as a customizable and extendible environment for developing diverse accelerator applications. Its main architectural principle is a separation of physical entities and mathematical abstractions from algorithms. The accelerator algorithms are implemented as classes that share data via Common Accelerator Objects (Element, Bunch, Twiss, *etc*.). This highly flexible structure has facilitated selecting and implementing more appropriate software design patterns and accelerator approaches, supporting project-specific requirements, and connecting the UAL applications with heterogeneous data sources. At this time, the UAL joins several object-oriented accelerator programs: PAC (Platform for Accelerator Codes), TEAPOT (Thin Element Program for Optics and Tracking), ZLIB (a numerical library for differential algebra), and ALE (Accelerator Libraries' Extensions). The Application Programming Interface (API), written in Perl, provides a universal homogeneous shell for integrating and managing all these components and project extensions. The UAL environment has been successfully applied to several accelerator projects: LHC, RHIC, and CESR. This paper outlines the major activities and applications for the new project, Spallation Neutron Source (SNS) Ring.

## 2 SIMULATION ACTIVITIES

The SNS ring dynamics presents a complex combination of several physical effects and dynamical processes[2]. Some of them, such as field errors and misalignments, are supported in general-purpose accelerator codes. Other effects, such as space charge and collimator surface grazing, are actual only for high intensity hadron rings and distributed into a set of independent specialized programs (Table 1). The mismatch among diverse data formats, units, and notations complicates the usage of these programs and increases the risk of errors and misinterpretations. Besides, the accurate simulation of the very low beam loss ($10^{-4}$) requires the simultaneous consideration of several different effects in a single scenario. The UAL open environment addresses all these tasks. It supports the incremental development of independent components and their configurable packaging into the accelerator applications. For the SNS project, the UAL is being extended with the following features: injection painting, collimator, and space charge. To facilitate the implementation and employment of new modules we have developed a benchmark infrastructure that provides the consistent interfaces among alternative accelerator approaches:



### 2.1 *Injection Painting*

During the multi-turn injection into the SNS ring, protons are painting over a large phase space volume in order to reduce the space charge tune shifts and to minimise the number of traversals through the stripping foil. The ACCSIM code offers the most consistent approach for optimization and simulation of these dynamical processes. However, the control of the different scenarios is hidden behind of the ACCSIM input language impeding the inclusion of new physical effects (field errors, misalignments, *etc*.). In UAL, all these dynamical processes are programmed directly with the Perl API that provides an unlimited access to the UAL core packages, project-specific extensions, and a wide variety of general-purpose supporting tools and applications (Graphics, GUI, etc.).

### *2.2 Field Errors, Misalignments, Correction*

In the UAL, all accelerator elements are located in the central repository, the Standard Machine Format (SMF). The SMF supports both the hierarchy of beam lines and

[*] Work performed under the auspices of the US DOE
[†] Email: malitsky@bnl.gov

generic elements as well as parameters associated with individual elements of the as-installed machine. Magnetic errors and misalignments are implemented as fine-grained sets of element attributes and can be assigned to an arbitrary design element. The SMF structure is neutral to accelerator approaches, and the accelerator physicist can employ either the UAL core modules or local extensions (e.g. IR filter for RHIC and LHC [9]). For the simulation of nonlinear magnetic fields and misalignments, we are using the TEAPOT library that provides a rich set of simulation tools (conventional element-by-element tracker and DA integrator) and correction algorithms (tuning, closed orbit correction, chromaticity fitting, and global decoupling).

### *2.3 Collimator*

The collimator is designed to prevent spreading up beam halo into the SNS ring tunnel and localized it at a level from $10^{-3}$ to $10^{-2}$ in one controlled place. Its relative sizes and forms depend on many factors, such as an injection painting scheme, lattice parameters, and others. Then the simulation model has to be adaptable to an arbitrary combination of lattice and collimator variants. It can be achieved by implementing the collimator system as an insertion device and splitting the one-turn tracking procedure into three steps: propagating particles (e.g. using TEAPOT module) from the injection point to the collimator system, applying the collimator algorithms, and completing the turn by following particles back to the injection point. In the UAL environment, this scenario is controlled directly from the Perl script, and it is open for arbitrary representations of the collimator module. For example, this module can be implemented as a local adapter to the independent FORTRAN program (e.g. LAHET) or the HEP C++ shared libraries (e.g. GEANT 4). This solution looks very interesting from the perspective of integrating the accelerator and high energy physics software. However, its complete implementation assumes the significant overhead for this particular task. Then we are considering the ACCSIM approach that provides an optimal set of algorithms for particle-target interactions (Landau and Bethe-Bloch energy loss distributions, Moliere multiple scattering, and nuclear interactions).

### *2.3 Space Charge*

The space charge effect has a large impact on the beam dynamics and halo growth in the SNS ring and has to be included in the common model for evaluating the beam distribution and uncontrolled beam loss in the ring tunnel. The implementation of the 3D space charge effects is a difficult task because it involves the trade-off between the performance and accuracy of available algorithms. Then there is a need in a configurable module that enables the exchange of several alternative approaches. The UAL framework will address this task by providing the uniform mechanism for assembly and reuse of independently developed algorithms [11].

### *2.5 Fringe Field Models*

Since the aperture of the ring magnets is comparable to the magnetic length, fringe field impact must be considered. Taylor maps extracted from fringe field models (e.g. MaryLie[10]) will be incorporated into element-by-element tracking. This approach has been employed in previous UAL applications for simulating RHIC helical dipoles and CESR wigglers.

### *2.6 End-to-End Simulation*

The SMF structure allows one to consider several different lattices in the same process. This feature is very important for optimization of IR sections, injection and extraction systems. In the SNS project, we plan to concatenate various sections of machine lattices (e.g. HEBT line + ring + RTBT line) for the end-to-end simulation of particles with various charge states ($H^+$, $H^0$, $H^-$).

## 3 UAL FRAMEWORK

The UAL framework is a necessary and logical step in the UAL evolution. It intends to offer a single object-oriented integration environment for compatible and independent implementation of diverse accelerator applications. This will enable accelerator scientists and software developers with different kinds and levels of skill to participate in the common development process and will promote selection, sharing, and standardization of the most effective accelerator approaches and solutions. The UAL framework is being developed using the component-oriented technology and provides the following systems [11]:

- uniform mechanism for assembly and reuse of independently developed accelerator algorithms;
- uniform infrastructure for optimization and correction approaches

The off-line SNS Ring Simulation Facility is considered the first application of this infrastructure.

## 4 INTEGRATION WITH ACCELERATOR CONTROL

The value of theoretical algorithms depends to a large extent on the possibility to employ them in a real experiment. In the past, simulation programs and control system applications were developed and deployed as two independent products. We intend to merge these efforts in the single direction based on the online Accelerator Simulation Facility (Fig.1). The concept of online modelling is becoming very popular in the accelerator community because it provides them with new interesting possibilities:

- development and validation of control system applications before the commissioning stage based only on the “virtual” accelerator,
- online analysis and comparison of runtime accelerator parameters with theoretical models during commissioning and operation .

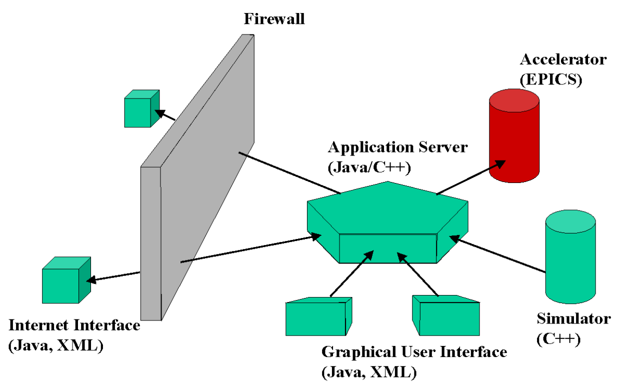

Firewall
Accelerator (EPICS)
Application Server (Java/C++)
Simulator (C++)
Internet Interface (Java, XML)
Graphical User Interface (Java, XML)

Figure 1: The UAL-based SNS Simulation Facility.

The SNS online facility will have a multi-level architecture. At this time, EPICS has implemented the lowest layer, a homogeneous interface to physical devices. However, EPICS exchange data types are too generic for object-oriented higher level applications, such as simulation and correction modules, and require an additional layer to map device parameters into accelerator domain constructions. Until recently, there has not been a portable solution for this problem. At this time, several industrial technologies address this task by providing new communication concepts, such as the Java Serializable object and the CORBA Object-by-Value semantics, that allow developers to apply seamlessly the same object-oriented models and patterns to local simulation programs and distributed control systems. We plan to evaluate these and other industrial technologies from the perspective of their integration with the UAL and EPICS environments.

We thank D. Abell, J. Beebe-Wang, M. Blaskiewicz, F. Jones, J. Galambos, H. Ludewig, A. Luccio, and S. Machida for many useful discussions.

## 5 REFERENCES

[1] N.Malitsky and R.Talman, AIP 391, 1996.

[2] J.Wei *et al*, these proceedings.

[3] L.Schachinger and R.Talman, Particle Accelerators, 22, 35(1987).

[4] H.Grote and F.C.Iselin, CERN/SL/90-13.

[5] R.V.Servranckx, *et al.*, SLAC Report 285 UC-28, 1985.

[6] F.W.Jones, TRIUMF Design Note TRI-DN-90-17, 1990.

[7] J.Galambos, *et al.* AIP 448, 1998.

[8] S.Machida, Nucl. Instrum. Methods, A309, 43(1991).

[9] J.Wei, *at al*, these proceedings.

[10] A.Dragt *et al*., MaryLie 3.0, 1999.

[11] N.Malitsky and R.Talman. ICAP98, 1998.

Table 1: Accelerator programs used in the SNS project.

| | UAL [1] | FTPOT [3] | MAD8 [4] | DIMAD [5] | ACCSIM [6] | SAMBA [7] | SIMPSONS [8] |
|---|---|---|---|---|---|---|---|
| Interface | **PERL API** | **FTPOT** | **MAD** | **DIMAD** | **ACCSIM** | **SuperCode** | **SIMPSONS** |
| MAD elements | **Yes** | **Yes** | **Yes** | **Yes** | **Yes (via nodes)** | **Yes (via nodes)** | **Yes** |
| Errors | **Yes** | **Yes** | **Yes** | **Yes** | **No** | **TBC** | **Yes** |
| Dynamic Processes | **Yes (via PERL )** | **No** | **No** | **No** | **Injection** | **Yes (via SuperCode)** | **Bρ and RF** |
| Tracking | **Thin lenses** | **Thin lenses** | **Lie algebra** | **Simplectic TRANSPORT** | **Linear matrices + node -lenses** | **Linear matrices + node-lenses** | **Thin lenses** |
| Mapping | **Any order** | **Second order** | **Third order** | **Second order** | **Linear order** | **Linear order** | **No** |
| Space Charge | **TBC** | **No** | **No** | | **3D** | **3D** | **2D and 3D** |
| Analysis (Twiss,…) | **Yes** | **Yes** | **Yes** | **Yes** | **No** | **TBC** | **No** |
| Lattice Optimization | **TBC** | **No** | **Yes** | **Yes** | **No** | **TBC** | **No** |
| Correction(Orbit,…) | **Yes** | **Yes** | **Yes** | **Yes** | **No** | **TBC** | **No** |
| Concatenation of several lattices | **Yes** | **No** | **No** | **No** | **No** | **No** | **No** |
| Support of third party extensions | **Yes** | **No** | **No** | **No** | **No** | **Yes** | **No** |
| Painting | **Yes** | **No** | **No** | **No** | **Yes** | **Yes** | **No** |
| Injection Foil | **Yes** | **No** | **No** | **No** | **Yes** | **Yes** | **No** |
| Collimator | **April 99** | **No** | **No** | **No** | **Yes** | **No** | **No** |