



Project Plan

Mike McKerns, Caltech

-
- Building the Project Plan
 - Managing the Project Plan
 - Updates to Plan for Common Algorithms

Goals & Objectives

- The goal of DANSE is to build a software system for neutron scattering research that:
 - integrates the basic data reduction capabilities that are available today
 - enables new types of science in all major subfields of neutron scattering research
 - provides a coherent framework onto which software components can be added by scientists
 - is maintainable by the SNS software group before the end of the project

Identifying the Project Scope

- **Planning begins: January, 2002**
 - requirements collection with neutron scientists
 - software workshops & reviews
 - email polls & surveys
- **First draft for five-year project plan: November 2003**
 - identified project tasks
 - problems with estimation of effort, risk, & duplication of effort
- **First draft for Scope Baseline (WBS): August 2004**
 - risk assessment & mitigation plan
 - resource leveling, task dependencies, & external drivers
- **Project Descoped (\$16.7M – \$11.9M): March 2006**
 - identified critical tasks & scope contingency

Building a Scope Baseline (task-by-task)

Level 5 Task DURATION

Version 1.1a

Task Name MMTK

Task Description

WBS 5.3.5.4

Task Risk Factor 1 (enter as an integer)

Risk Multiplier 0.30

Task Damage Factor 30% (enter as decimal)

0.30

0.26 4.42 1.17 2.60

less Scope
Scope weeks

	Allocation as decimal	Annual Rate	Hourly Rate	Number of Resources
Sr. Scientist				1.00
Postdoc	100%			1.00
Programmer	100%			1.00
Tech Writer	50%			1.00
Grad Student				1.00
Undergrad	75%			1.00
Other	75%			1.00

History:

Sr Sci, Postdoc, Programmer, Tech Writer, Grad were 4%, 75%, 75%, 75%, 75%

Enter time in weeks

0.26	4.42	1.17	2.60			
------	------	------	------	--	--	--

8.45 8.45 275.60 \$13,355 less Scope

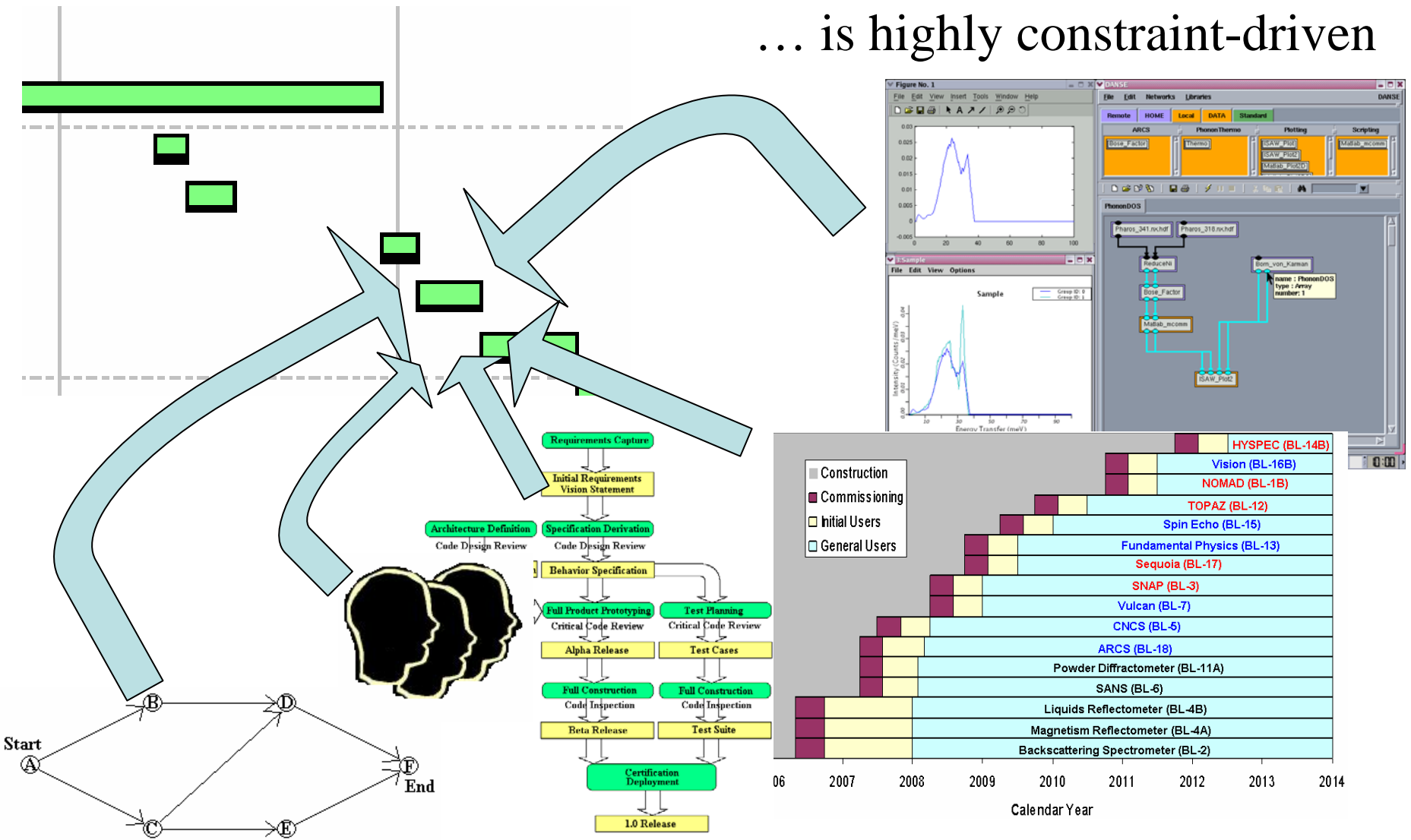
Maximum Duration (weeks)	Minimum Duration (weeks)	Total Allocated Hours	Total cost
8.45	8.45	275.60	\$13,355

Scope Contingency

Task Name	Sr. Scientist	Postdoc	Programmer	Tech Writer	Grad Student	Undergrad	Other	Raw weeks	Risk Adj. Maximum Duration	Risk Adj. Minimum Duration	Risk Adj. Allocated Hours	Cost
Modify input file writer			0.4					0.40	0.52	0.52	20.80	\$1,100
Write bindings to modules			0.1					0.10	0.13	0.13	5.20	\$275
Modify output file reader			0.4					0.40	0.52	0.52	20.80	\$1,100
Test/use case 1: energy minimization		0.2						0.20	0.26	0.26	10.40	\$540
Test/use case 2: constrained molecular dynamics		0.2						0.20	0.26	0.26	10.40	\$540
Test/use case 3: normal mode analysis		0.2						0.20	0.26	0.26	10.40	\$540
Test/use case 4: operations on dynamic trajectories		0.2						0.20	0.26	0.26	10.40	\$540
Test/use case 5: point charge fitting		0.2						0.20	0.26	0.26	10.40	\$540
Test/use case 6: molecular surface calculations		0.2						0.20	0.26	0.26	10.40	\$540
Write tutorial text		0.60						0.60	0.78	0.78	31.20	\$1,620
Write tutorial code		0.60						0.60	0.78	0.78	31.20	\$1,620
Write and integrate reference documentation				2				2.00	2.60	2.60	52.00	\$1,700
Scientific review	0.20							0.20	0.26	0.26		
User testing and subsequent maintenance		1.00						1.00	1.30	1.30	52.00	\$2,700

The Scheduling Process

... is highly constraint-driven



Management at the Package Level

- **Software Production is incremental, w/ phased development**
 - Inception [$< 5\%$ complete]: mostly planning and assessment
 - Elaboration [5-20% complete]: mostly requirements capture
 - Construction [20-90% complete]: mostly code and test implementation
 - Transition [$> 90\%$ complete]: documentation, deployment, and certification
- **Reviews are integral to effective software production**
 - reviews ensure quality
 - review by subproject leaders to ensure requirements capture
 - review by developers to guide iterative development
 - review by quality assurance to affirm standards adherence
 - review by management to assess software production status

Reviews in the Software Process

- **Planning Review**
 - vision statement summarizing purpose, functionality, and acceptance criteria
- **Design Review**
 - UML & auto-generated interface documentation for architecture specification
 - refinement of requirements and use cases into behavior specification
- **Code Review**
 - both component/application and test code inspected
 - automated test for quality standards, manual inspection for code functionality
- **Reviews involve Architect, PM, QA, SubPI, developers**
 - reviewers should be stakeholders in the code
 - review action items entered to project tracking system
 - reviews are critical in developing the most commonly reused tasks

The Project Plan Must be Agile

- The Project Plan must accommodate:
 - the talents of current developers
 - new developments in available software
 - refinement of ideas, scope, requirements, and design
- Minor adjustments to schedule and tasks in August 2006
 - architectural design of several common algorithms packages improved
 - created new tasks as code design became more extensible
- Baseline plan was good; however, not detailed enough...
- Difficulties encountered with planning occurring at task start
 - drive to design more reusable code lessened when “clock is ticking”
 - new requirements make Central Services & Common Algs. schedule unstable

Solidify Initial Scope & Design...

- Ensure a clear definition of all deliverables
 - identify all of the project deliverables (especially the flagship applications)
 - break deliverables into well-defined conceptual components
 - specify information to exchange between the components
 - draw UML diagram of components and data objects for each deliverable
- Ensure identification of all tasks
 - list our objectives and our requirements for each subgroup
 - associate each requirement with at least one task
 - associate each deliverable with at least one task
 - identify if task is well-defined, digestible, and meets design requirements
 - identify if software has the desired interfaces

...then Rebaseline

- **Building tasks with narrower scope**
 - creates additional tasks
 - tasks tend to be shorter duration with better-defined scope
 - code tends to be more reusable
 - development is more efficient and there is less rework
 - creates less surprises
- **Schedule tasks once WBS dictionary is updated**
 - developers focus on more than one task
 - more tasks run simultaneously, thus durations can be lengthened
 - more akin to development in a small dynamic group
 - less well-defined tasks must be scheduled later in the project



Project Plan

Mike McKerns, Caltech

-
- Building the Project Plan
 - Managing the Project Plan
 - Updates to Plan for Common Algorithms

Integration Strategy for Data Reduction

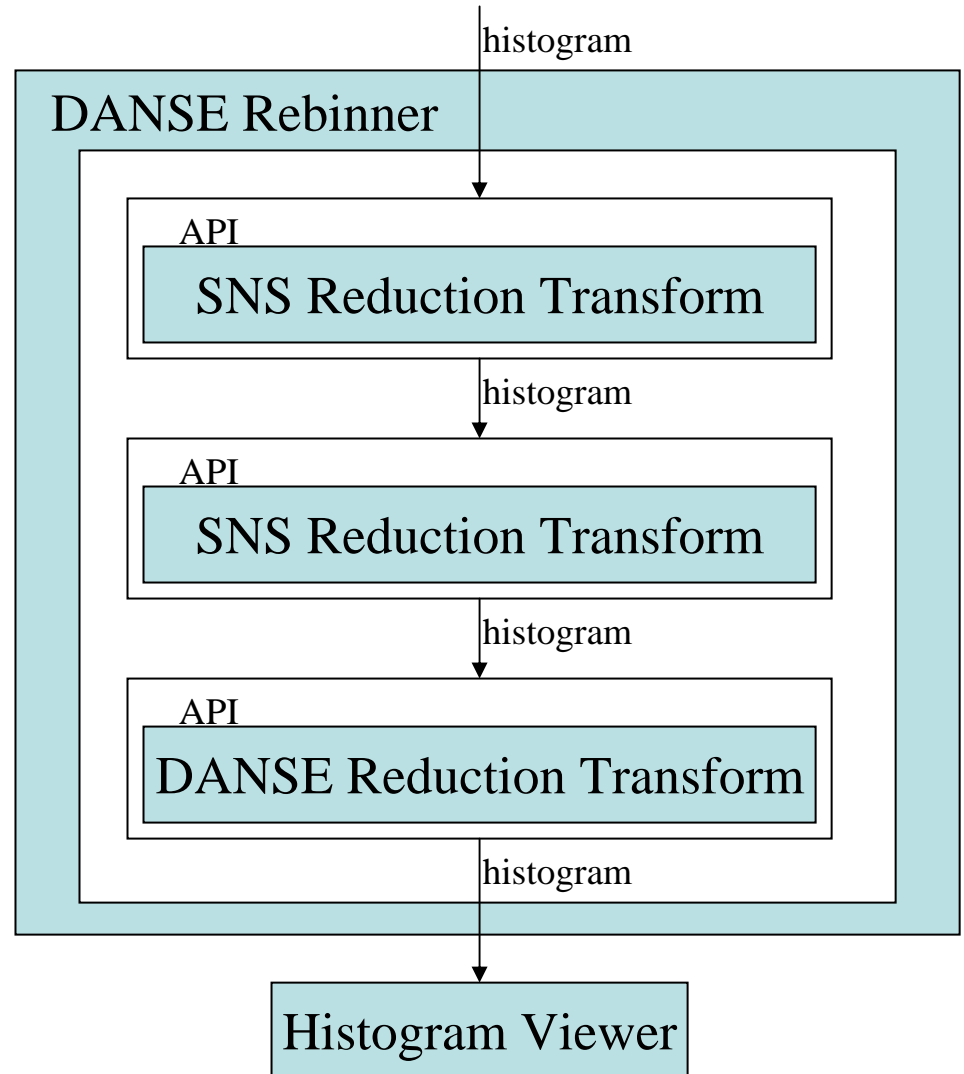
- In October 2006, DANSE & SNS teams began to work together to define a strategy for data reduction and analysis
- SNS to provide basic data reduction for all SNS instruments
- DANSE will couple reduction to analysis and simulation
 - post-reduction corrections
 - analysis tools used in reduction
- DANSE & SNS must work together to
 - ensure each other's requirements are met
 - standardize data structures & define a clear interface
 - leverage each other's effort

Several Tools Requested

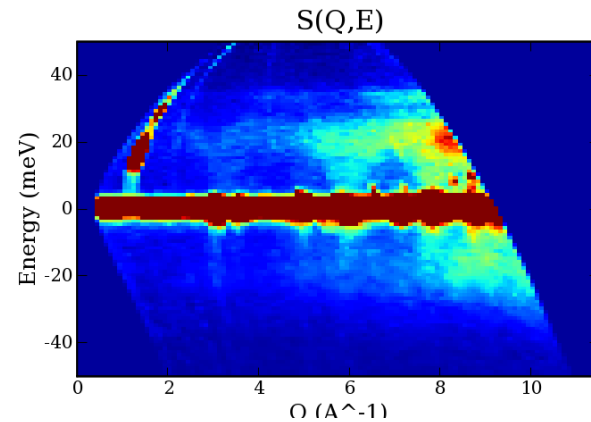
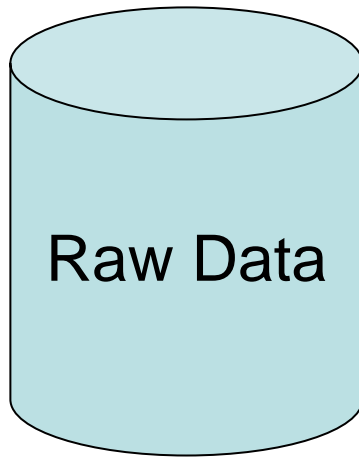
- In November 2006, DANSE representatives met with several SNS instrument scientists to collect input on desired analysis tools.
- GUI Builder
 - extract user-settable fields and defaults from applications & components
 - configure view & widgets with simple XML description
- Peak Fitter
 - obtain peak information from noisy data
 - allow selection of fitting methods
- Detector Viewer
 - examine raw data
 - look for bad detectors, create masks
- Histogram Viewer
 - provide physical representation at each reduction step
 - aid visualization of slicing and complex operations

Collaboration Through Modular Design

- API creates standardized interface, and facilitates modular design
- Transforms from either SNS or DANSE library
- Standardized data objects (histograms) passed between each transform



Extensibility by Design Abstraction



Histogram

Mapping



Histogram

- Histogram is a fundamental data structure for data reduction
- Reduction can be represented by mapping an $I(\text{pixel}, \text{tof})$ histogram to a $S(Q,E)$ histogram over the instrument information [inelastic]

Making Reduction Instrument-Agnostic

- Can we separate instrument-specific parts of data reduction, so that transformations are reusable across instruments?
- First, we identify & isolate instrument-specific pieces
 - instrument definition migrated to **instrument** data object
 - extracting histograms from instrument run files dissociated to **measurement**
- Then, we reconnect with abstract interfaces
- Reduction & simulation become instrument-independent
 - supporting a new instrument now may be very simple...
 - add an instrument description & new class to extract data from run files

Standardized Reduction Data Objects

5.4.1.1 Basic Reduction Data Structures: *Standardized runtime data structures will be developed for DANSE to satisfy two needs: to provide efficient iteration for high performance components, and to provide convenient access to scientists. It has often proved difficult to reconcile these needs, especially with histograms. DANSE will achieve these goals for histograms by decoupling iteration and access from the underlying buffer.*

5.4.1.1.x 1D Arrays

5.4.1.1.x nD Arrays

5.4.1.1.x Histograms

5.4.1.1.x 1D Pixels

Comment [mm86]: Also accessors and convenience functions. These are all abstractions to the underlying implementation, thus an API is needed for each along with the different bindings we'll use. Basic manipulations covered in 5.4.1.3.

5.4.1.2 Experiment Metadata Containers: *Efficient runtime structures will be developed to serve as containers of metadata on the instrument, sample, sample environment, experimental run, and so on. We will provide runtime analogues to NeXus files that parallel the NeXus hierarchies.*

5.4.1.2.x Instrument Description

5.4.1.2.x Run Description

5.4.1.2.x NeXus Data Object

5.4.1.2.x NeXus File Parsers

Comment [mm87]: Also accessors and convenience functions.

Comment [mm88]: How is NeXus parser different than 4.4.2 or that in 5.1.1.2? Maybe this doesn't belong here.

5.4.1.3 Common Array Manipulations: *We will provide fundamental transformations of our basic reduction data structures.*

5.4.1.3.x ???

Comment [mm89]: Only for data objects in 5.4.1.1.

Comment [mm90]: Develop to an API, so works like math with python objects.

5.4.1.4 Advanced Array Manipulations: *We will implement our basic reduction data structure manipulations for a parallel environment, to utilize streaming data, and to use iterators.*

5.4.1.4.x Iterator Implementation

5.4.1.4.x Streaming Implementation

5.4.1.4.x Parallel Implementation

Define Reduction & Standardize Interface

5.1.1.1 Data Reduction: *We will develop rebin drivers and reduction transformations of data arrays and histograms, and provide support for basic reduction of single crystals, powders, and liquids. By utilizing standardized data objects, reduction components and transformations built in DANSE should freely interoperate with those built at the SNS.*

5.1.1.1.x ???

Comment [mm56]: Develop some common utilities and transformations here... many subgroups or the SNS will develop drivers as other tasks. We will need an API for each 'level' of reduction transformation or driver, and also effort for standardization with the SNS.

5.1.1.2 Measurement: *The parsing of data files and associated metadata generated by experiments on neutron instruments is very often instrument-specific. We will provide a mechanism for parsing NeXus files into standard data objects, and a means of identifying data as a calibration, a background correction, a sample run, and so on. By adding instrument-specific parsing information, this package can be extended to instruments that do not use NeXus files.*

5.1.1.2.x ???

Comment [mm57]: Generic rebin drivers and API, rebin utilities (?), and standardization with SNS.

Comment [mm58]: What common work can be done here? Parsing of most data files are instrument-specific, so that will be done in a subgroup task. Parsing of NeXus files is done in 5.4.1.2. Maybe this task only identifies and groups the data by data/run type?

5.1.2.1 Advanced Data Reduction: *We will extend the DANSE reduction code to work in a parallel environment, to utilize streaming data, and to use iterators.*

5.1.2.1.x Iterator Implementation

5.1.2.1.x Streaming Implementation

5.1.2.1.x Parallel Implementation

Comment [mm59]: Specification of how to group runs and provide histograms.

5.1.2.2 Post-Reduction Corrections: *We define reduction as in the SNS document: "Data Reduction Library Requirements and Specifications." Further transformations and corrections to data are to be considered as post-reduction corrections and reduction support. We will provide some post-reduction manipulations of backgrounds, detector calibration, and other common corrections.*

5.1.2.2.x ???

Comment [mm60]: Again this is only the subset of common post-reduction tools, with other post-reduction tasks being developed by the subgroups. Close inspection of the SNS specifications are needed to distinguish between tasks here and in 5.1.1.1... maybe 5.1.1.1 and 5.1.2.2 should be combined? We'll also need standardization with the SNS here.

5.2.1 Texture Analysis and Visualization: *The software package MAUD is a fitting program for diffraction patterns that allows for calculations of pole figures and diffraction line broadening. The functionality of MAUD will be provided as DANSE components with a plan for interoperation with mechanics models. The core methods of*

Extend Reduction to Inelastic Instruments

10.1.0.1 Inelastic Reduction Drivers: We will provide rebin drivers and transformations that are specific to inelastic neutron scattering.

10.1.0.1.x ???

Comment [mm110]: Standardization with the SNS should be taken care of in 5.1.1.1; this should just be a collection of drivers.

10.1.0.2 Inelastic Post-reduction Corrections: We will provide support for post-reduction manipulations of background, detector calibration, and other corrections common to inelastic neutron scattering.

10.1.0.2.x ???

Comment [mm111]: Basic corrections only. Other 'advanced' corrections are listed as separate tasks.

Comment [mm112]: Same note on standardization as in 10.1.0.1.

10.1.1 Support for SNS Direct Geometry Chopper Spectrometers: We will ensure SNS-built reduction applications for the **CNCS** and **SEQUOIA** instruments are available to DANSE. The reduction code for the **ARCS** instrument will be fully extended to DANSE. Monte-Carlo simulations will also be supported for the above-mentioned instruments.

10.1.1.x Instrument Descriptions

10.1.1.x Instrument-Specific Geometers

10.1.1.x Instrument-Specific Measurement Classes

10.1.1.x Adaptation of existing reduction drivers

Comment [mm113]: Is this list complete? We need to ensure that SNS applications have programmatic access.

10.1.2 Support for other SNS Inelastic Spectrometers: We will ensure SNS-built reduction applications for the **HYSPEC** and **Backscatterer** instruments are available to DANSE. Monte-Carlo simulations will also be supported for the above-mentioned instruments.

10.1.2.x Instrument Descriptions

10.1.2.x Instrument-Specific Geometers

10.1.2.x Instrument-Specific Measurement Classes

10.1.2.x Adaptation of existing reduction drivers

Comment [mm114]: Same as comment for 10.1.1.

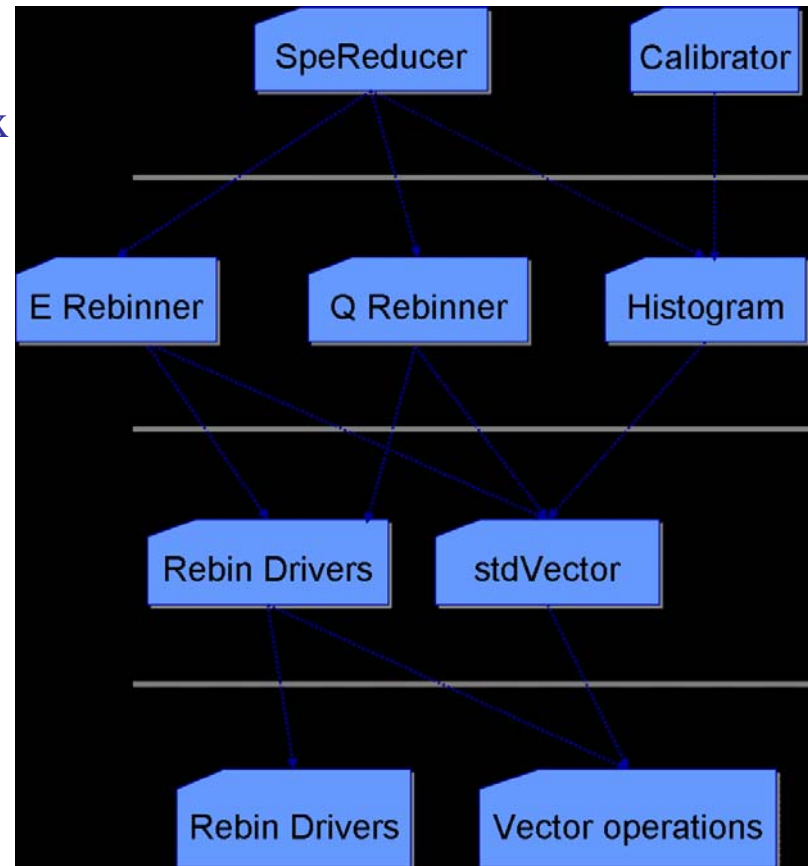
10.1.3 Support for Existing Inelastic Spectrometers: We will provide reduction applications and Monte-Carlo simulation of the **Pharos** and **LRMECS** instruments.

10.1.3.x Instrument Descriptions

Comment [mm115]: Extend support for these instruments tested under ARCS

Example Reduction Components

- **detCalibrator**
 - computes calibration constants and calibrates histograms
- **normalizer**
 - normalizes histograms by incident neutron flux
- **timeBG**
 - removes time-independent bg from histogram
- **speReducer**
 - reduces data to $S(\phi, E)$
- **Spe2Sqe**
 - converts $S(\phi, E)$ to $S(Q, E)$
- **e_iSolver**
 - computes neutron incident energy
- **darkAngleFinder**
 - determines dark angle for vanadium run

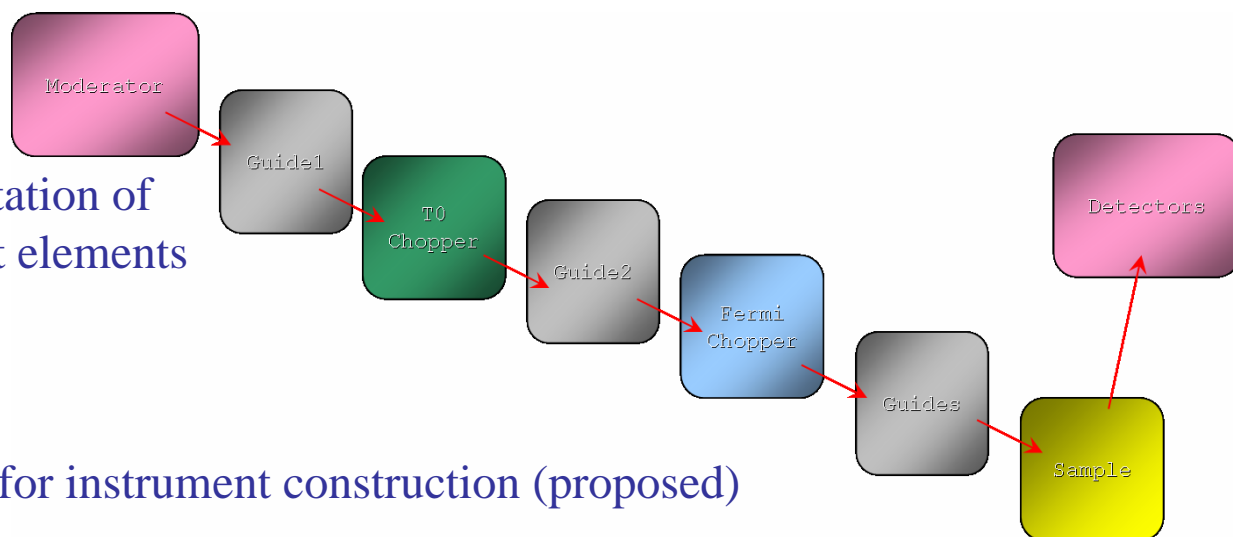


A Closer Look at Instrument

- Instrument is a reusable instrument definition
 - is built from available instrument elements and their relative positions
 - element descriptions are maintained separately from element position
 - instrument description may be obtained from NeXus file
- Instrument description is a hierarchy of elements
 - FermiChopper, T0Chopper, Moderator, Monitor, DetectorArray, DetectorTube, Guide,
 - SampleEnvironment: Cryomagnet, Cryogoniometer, ...

- Instrument Viewer

- provides a 3-D representation of instrument or instrument elements



- Instrument Builder

- drag-and-drop interface for instrument construction (proposed)

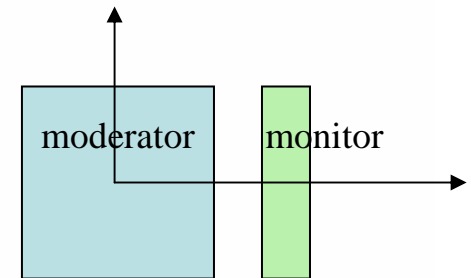
Building an Instrument

```
>>> from instrument.elements import *
>>> instrument = Instrument.Instrument( "instr_name", version = "0.1" )
>>> from instrument.geometers.ARCSGeometer import Geometer
>>> geometer = Geometer()
>>> moderator = Moderator.Moderator( instrument.getUniqueID(), instrument.guid(),
                                     100, 100, 100 )

>>> instrument.addModerator( moderator )
>>> position = [20000.0, 90.0, 180.0]
>>> orientation = [0.0, 0.0, 0.0]
>>> geometer.register( moderator, position, orientation)

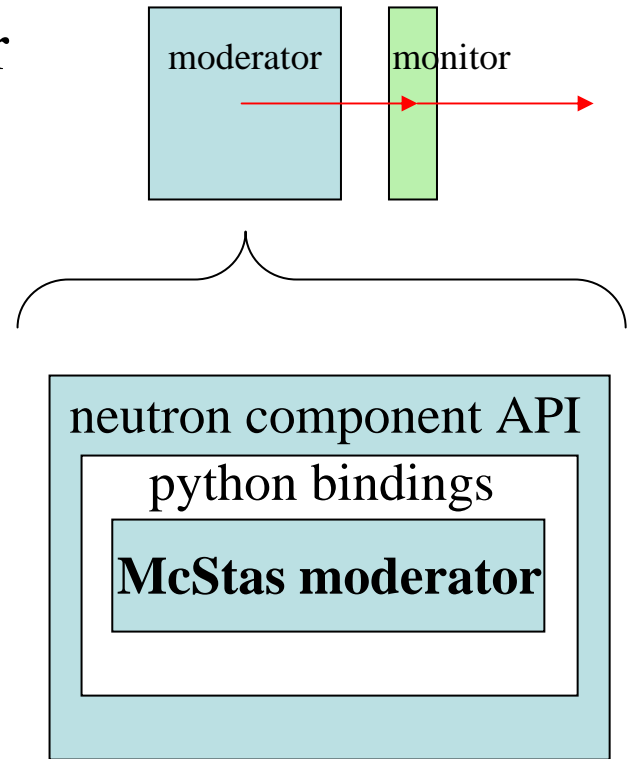
>>> monitor = Monitor( instrument.getUniqueID(), instrument.guid(),
                       xLength=30, yLength=0., zLength=50,
                       monitorNumber=1,
                       name = "monitor1")

>>> position = [2300.0, 90.0, 180.0]
>>> orientation = [0.0, 0.0, 0.0]
>>> geometer.register( moderator, position, orientation)
```

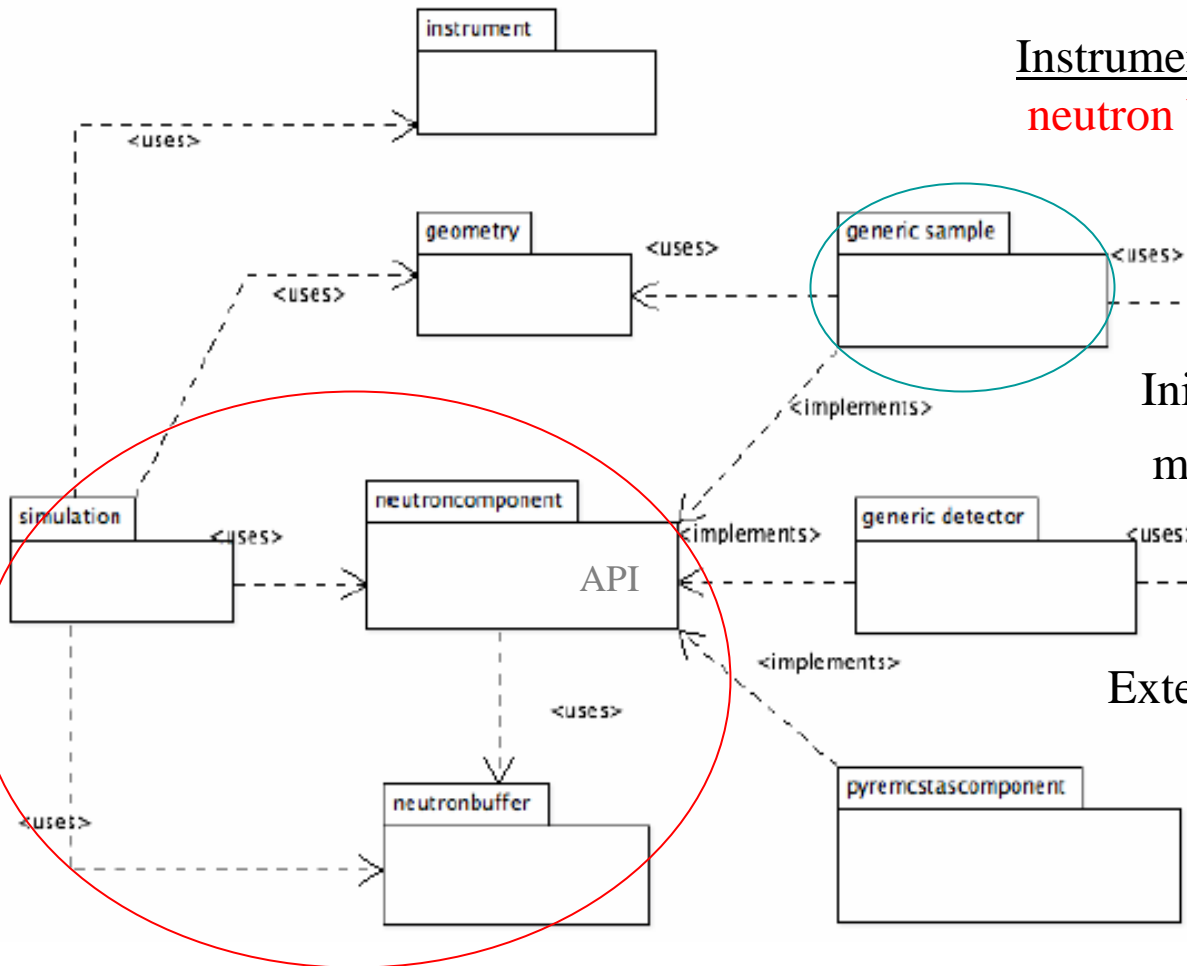


Monte-Carlo Instrument Simulation

- Simulation framework connects sources, samples, detectors, and other instrument elements
 - by providing a buffer to handle neutron flow between components
 - by providing an engine to drive the neutrons
- Connecting through the neutron component API allows exchange of newly built components with those provided by McStas, Vitess, ...



Scope Definition and Package Evolution



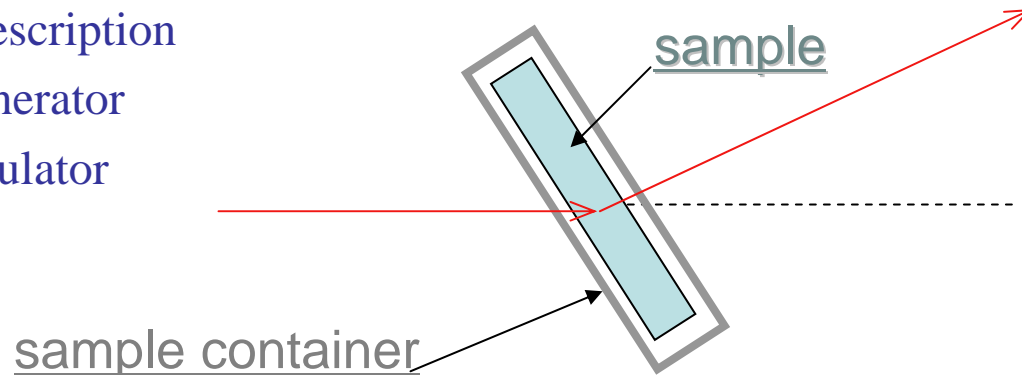
Instrument simulation is **simulation** engine, **neutron buffer**, and **neutron components**... and uses instrument and geometry

Initially, McStas components provide monitors, guides, samples, detectors

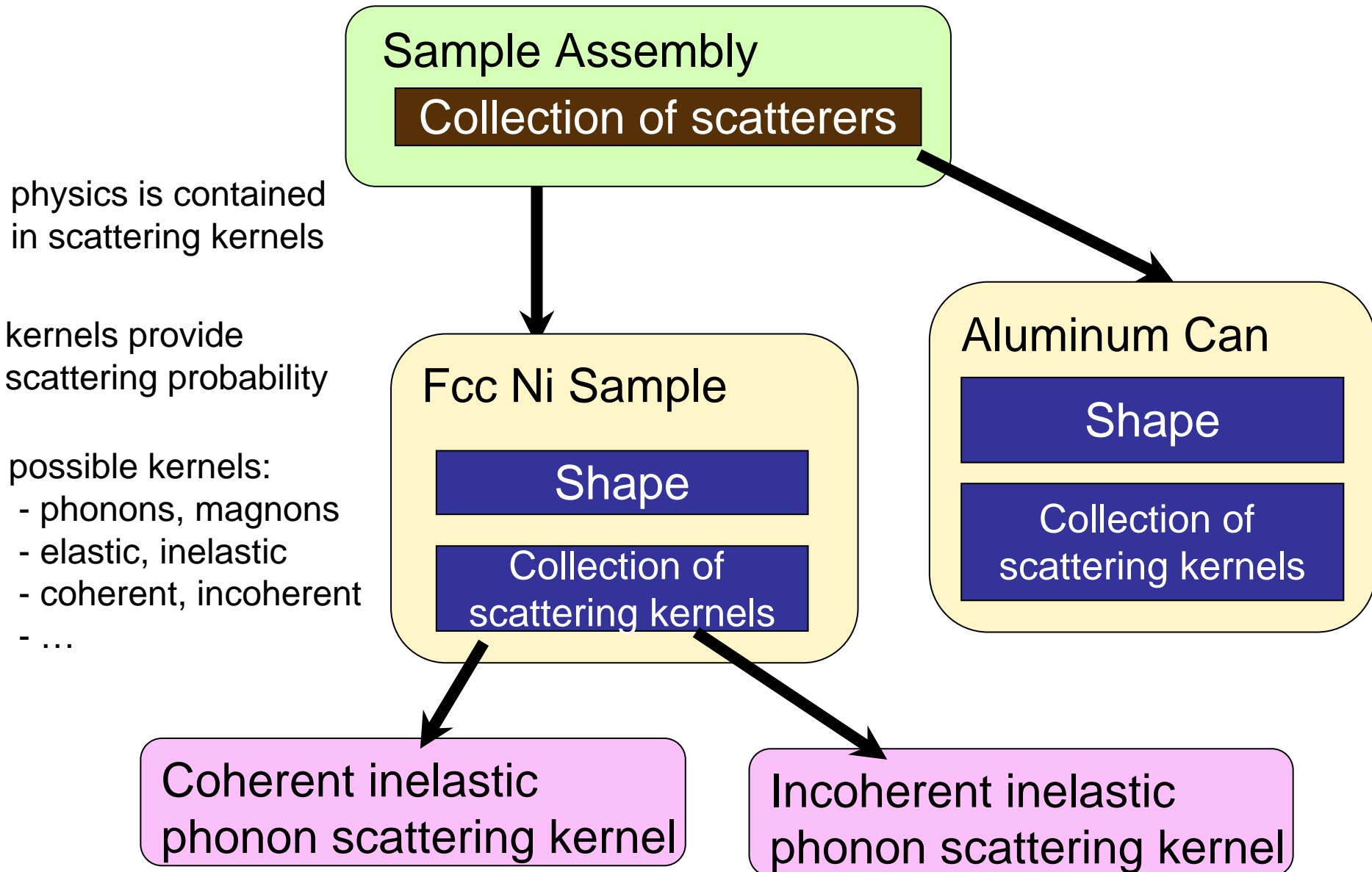
Extend by building neutron component with **generic sample** construction

Sample Simulation

- composite sample assembly required
 - a sample is usually inside some kind of container
 - a sample simulation needs to take into account a collection of scatterers including the sample and other objects
- composite scatterers also required
 - available sample simulations usually focus on one kind of scattering mechanism
 - a full simulation should take into account all possible scattering mechanisms with similar scattering strength
- clean separation of physics properties and geometry
 - reuse geometer as coordinate environment
 - shapes & sample description
 - random number generator
 - scattering path calculator



Building a Composite Scatterer



Define Instrument & Sample Simulation

5.3.1.1 Monte-Carlo Instrument Simulation Framework: *In its purest form, a full experiment simulation would track individual neutrons from the moderator, into the instrument, through the sample, and into the detectors. We will produce an instrument simulation framework that provides a stock set of extensible instrument components, and allows the integration of components from common Monte-Carlo simulation packages used to design neutron instruments for optimum combinations of resolution and flux.*

5.3.1.1.x Simulation Engine

5.3.1.1.x Neutron Buffer

5.3.1.1.x Neutron Storage

5.3.1.1.x Geometers

5.3.1.1.x Instrument Factories

5.3.1.1.x Generic Instrument Components

5.3.1.1.x Basic Sample Components

5.3.1.2-5 Bindings to Common Monte-Carlo Instrument Simulation Packages: *We will provide bindings to McStas, NISP, VITESS, and Ideas.*

5.3.1.n.x ???

5.3.2.1 Sample Simulation Framework: *A goal of the DANSE project is to integrate the structure and dynamics of the sample into Monte-Carlo instrument simulations, and calculate the scattering from the sample into the detectors. More traditional scattering kernels from geometric shapes will also be supported. Sample components are fully extensible to simulate an arbitrarily complex sample and sample environment.*

5.3.2.1.x Generic Sample Component

5.3.2.1.x Path Calculation for Scattering by Geometric Shapes

5.3.2.1.x Couple Scattering Kernel to Scattering Path

5.3.2.2 Coherent Elastic Scattering Kernel: *We will provide a mechanism for representing coherent elastic scattering from crystals and polycrystals from arbitrarily complex geometric shapes.*

Comment [mm64]: This includes a python module with convenience functions. A simulation launcher UI and a instrument builder UI will be developed.

Comment [mm65]: Launcher requires simple parameters: # neutron packets, chopper speed...

Comment [mm66]: Building an instrument from elements requires: coordinate environment (geometer), instrument elements, and instrument description (instrument). Also instrument element API.

Comment [mm67]: Python access to individual instrument elements only. Configure and launch of full instrument simulation to be handled by 5.3.1.1, not through bindings.

Comment [mm68]: Building a instrument element from samples requires: coordinate environment (geometer), geometric shapes (shapes), and a sample description. Also, a sample element API.

Comment [mm69]: Requires scattering kernel API.

Comment [mm70]: Calculation of scattering probability based on periodicity of the crystal, or crystal mosaic (in polycrystals). Overlaps 5.3.6?

Comment [mm71]: Meaning from shapes and density function as done in SANS, or from a crystal / molecular potential? Or both? I don't know what the input is here.

Comment [mm72]: At minimum, we will deliver components that launch a full

Summary

- Project Plan is good, and always improving
- First major rebaseline is underway
- Some requirements gathering & scope clarification needed
- Design and planning sessions have been successful
- Collaboration with the SNS has accelerated

End Presentation