

Open Architecture and Modular Paradigm of Cyclus

American Nuclear Society Annual Conference 2011 Hollywood, FL.

Kathryn D. Huff
Paul P.H. Wilson
Matthew J. Gidden

University of Wisconsin-Madison

June 26-30, 2011





Outline

① Top Level Simulators

Introduction

Lessons Learned

② Cyclus Modular Architecture

Model Hierarchy

Modularity

Example : Market-Facility Interface

Extensibility

③ Cyclus Development Paradigm

'Modified Open' Source

Quality Control

Testing Framework



Top Level Simulators

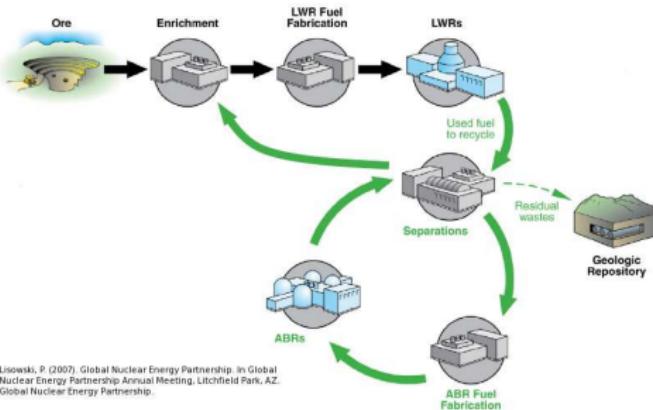


Figure: Top level simulators are intended to model the collective behavior of various fuel cycle decisions and strategies.[3]



Lessons Learned

Areas of unmet need in previous simulators include:

- **Usability** directed toward a wide range of user sophistication.
- **Performance** solving simple problems in interactive time scales
- **Fidelity** accommodating a range of levels of detail commensurate with user sophistication

Valuable software development practices have been identified [2][1]:

- Modularity allows the core infrastructure to be independent of proprietary and/or sensitive data and/or models.
- Extensibility with a focus on both robustness and flexibility allows for myriad potential developer extensions.
- Openness supports collaboration, verification, validation, and code sustainability.



Cyclus Modular Architecture and Open Development

The combination of modular encapsulation within the software architecture and an open development paradigm allows for collaboration at multiple levels of simulation detail and data security.



Outline

① Top Level Simulators

Introduction

Lessons Learned

② Cyclus Modular Architecture

Model Hierarchy

Modularity

Example : Market-Facility Interface

Extensibility

③ Cyclus Development Paradigm

'Modified Open' Source

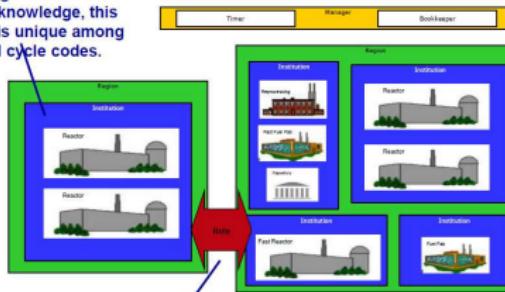
Quality Control

Testing Framework



Region Institution Facility Hierarchy

Institutions represent companies and government entities. To our knowledge, this modeling layer is unique among comparable fuel cycle codes.



User-specified rules might state whether one fuel cycle state can send materials to another, or define some special fuel-trade contract.

3

Figure: The Region, Institution, Facility Hierarchy of the Cyclus paradigm was a success of the GENIUSv2 code.[4]



Encapsulation

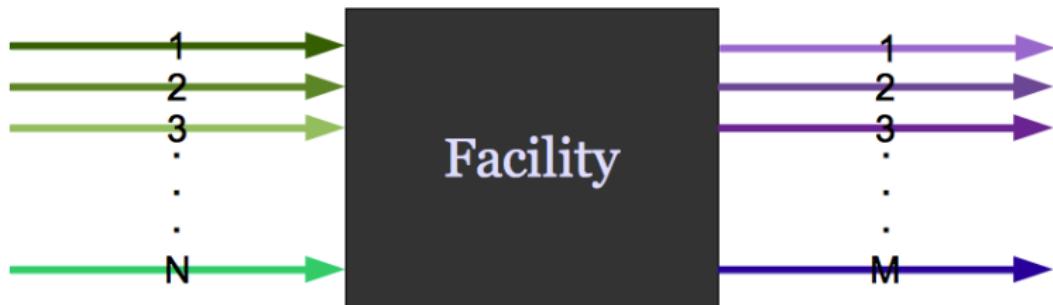


Figure: Regions, Institutions, Facilities, and Markets are all black boxes.



Module Interfaces

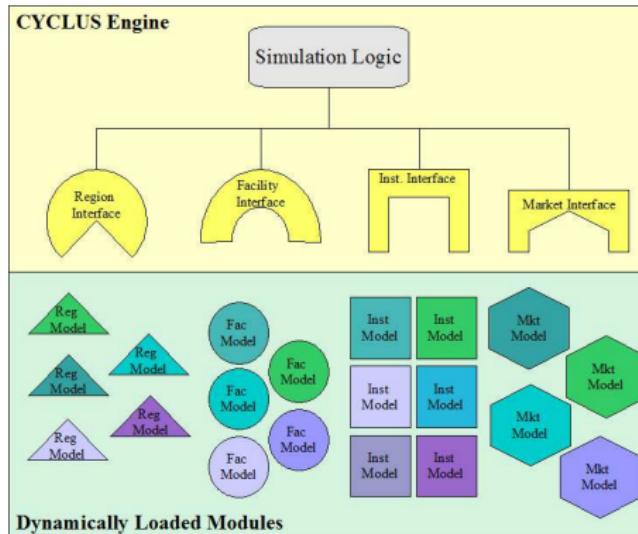


Figure: Well defined model interfaces facilitate model interchange. The user may choose the model at their desired level of detail.



Facilities Are Black Boxes

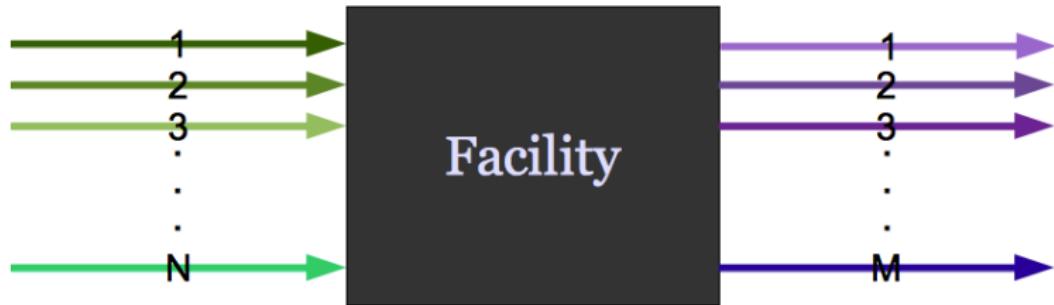


Figure: Each facility in the simulation makes requests and offers to fill its stocks and empty its inventory respectively.



Facilities Are Black Boxes



Figure: A facility might only make offers.



Facilities Are Black Boxes

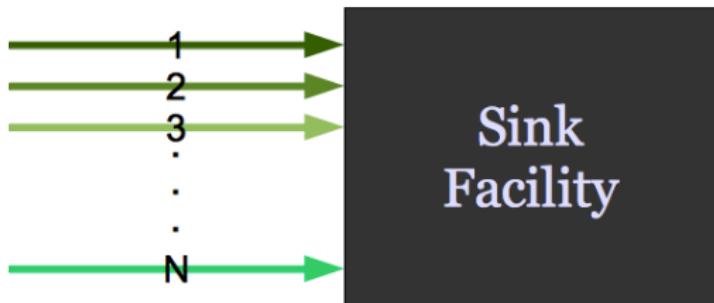


Figure: A facility might only make requests.



Each Commodity is Associated with a Market



Figure: A market receives offers and requests concerning its commodity.



The Market Solves the Matching Problem

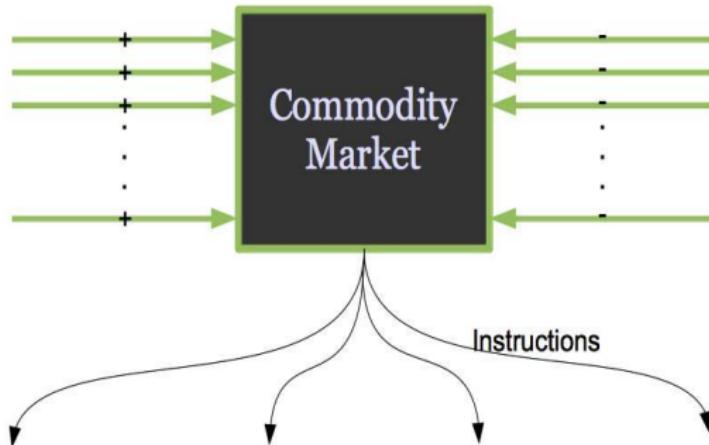


Figure: When the Market's arbitrary algorithm solves the matching problem, the Market sends instructions to the offering facilities.



A Simple Example

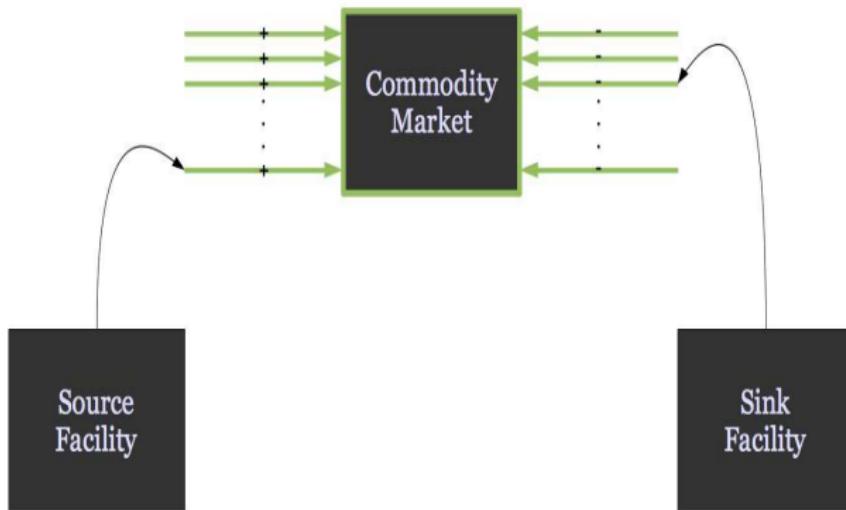


Figure: The source sends an offer and the sink sends a request.



A Simple Example

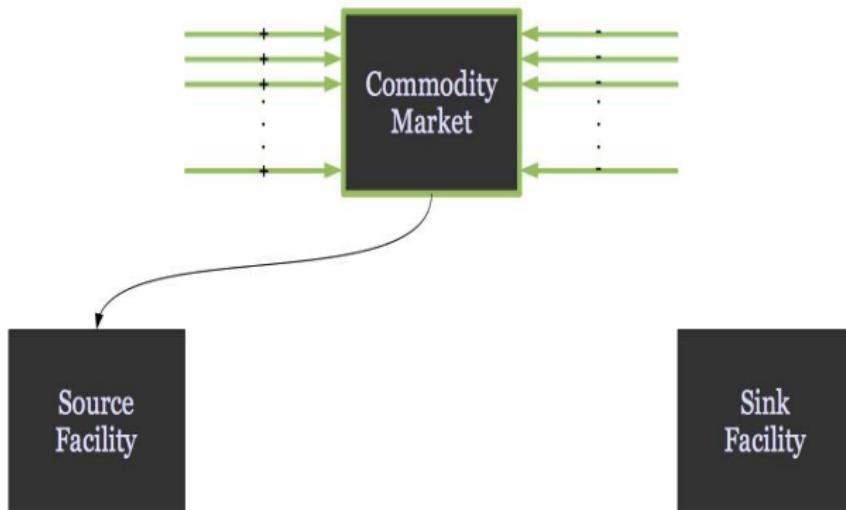


Figure: The Market solves the problem and instructs the source facility to send a certain amount to the sink facility.



A Simple Example

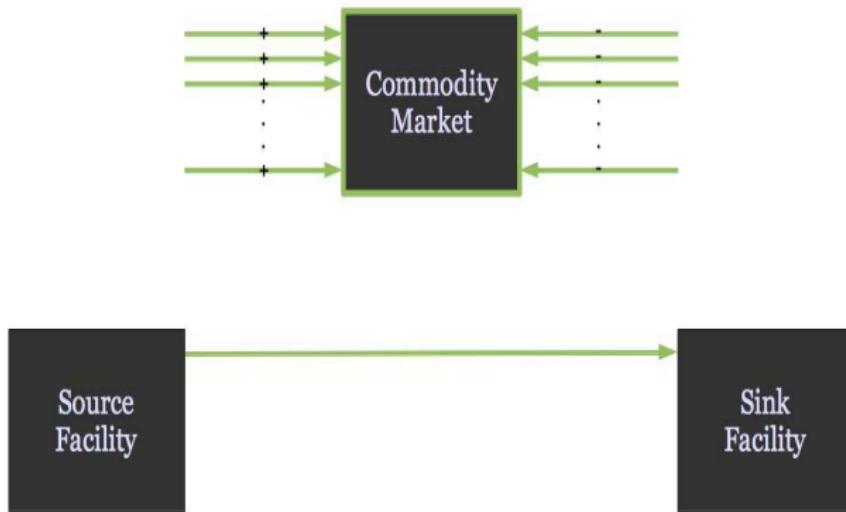


Figure: The source facility sends the material directly to the sink facility.



This Market Model Scales for Complex Systems

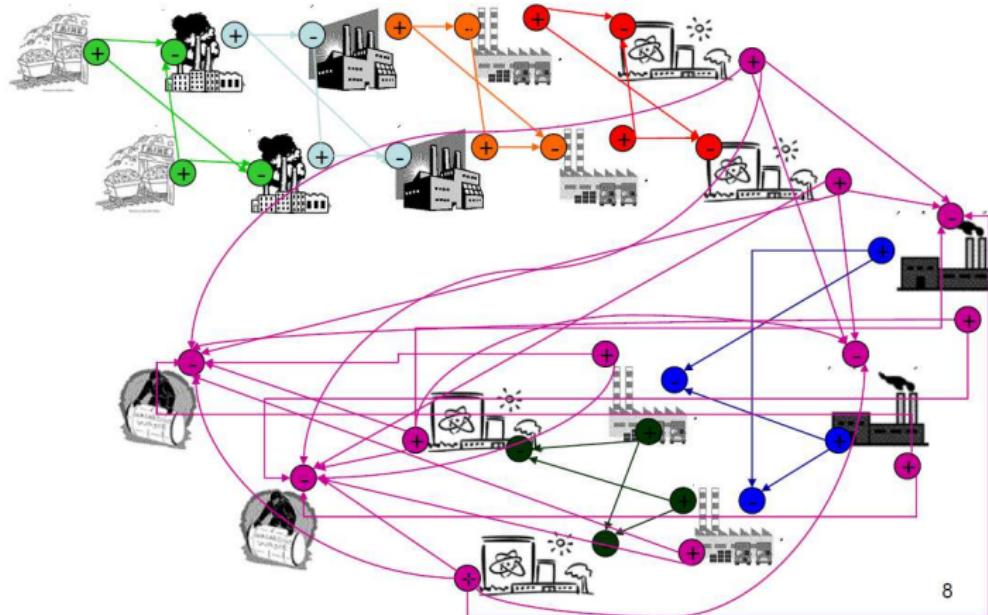


Figure: Well designed interfaces and strict encapsulation support scalability of the Market-based simulation paradigm [4]



Dynamic Module Loading : User

With a dynamic, plug-in implementation, the simulation logic is independent of the available models and models are loaded as shared libraries at runtime.

	input.xml
	<pre><simulation> <startYear>1962</startYear> <duration>1200</duration> <region> <name>UChicago</name> <DeployRegionModel> <deployment> <facility> <name>ChiPile1</name> <model>RecipeReactor</model> </facility> <year>1942</year> </deployment> </DeployRegionModel> . . </region> . </simulation></pre>

Figure: XML input parsing and a relaxNG schema provide a simplified XML interface



Dynamic Module Loading : Developer

With a dynamic, plug-in implementation, the simulation logic is independent of the available models and models are loaded as shared libraries at runtime.

Model.cpp	
	#include Model.hpp #include dlfcn.h mdl_ctr* loadModel(name){ void* model = dlopen(name.c_str(), RTD_LAZY) mdl_ctr* new_model = (mdl_ctr*)dlsym(model,"construct"); return new_model; }

RecipeReactor.cpp	
	#include RecipeReactor.hpp extern "C" Model* construct() { return new RecipeReactor(); }

Figure: Dynamic c library loading separates simulation logic from knowledge of available models.



Cross Platform, Multi-language

CMake (Crossplatform Make) build system is cross platform. Thus, users may build and run Cyclus on Windows, Mac, or Linux operating systems.

CMake is also multilingual, so modules may be written in any language or tool that can be made into a shared object library or wrapped with C++.

These include (but are not limited to):

- Fortran
- Python
- R
- Java
- Lisp
- IDL
- Matlab
- Perl
- etc...



Outline

① Top Level Simulators

Introduction

Lessons Learned

② Cyclus Modular Architecture

Model Hierarchy

Modularity

Example : Market-Facility Interface

Extensibility

③ Cyclus Development Paradigm

'Modified Open' Source

Quality Control

Testing Framework



Open Source Repository

This open source repository provides a centralized location for documentation, developer history, and unhindered developer access.

code.google.com/p/cyclus/source/browse/#svn%2Ftrunk%2Fsrc

Apple | Yahoo! | Google Maps | News | Wikipedia | Popular | proxy | Note in Reader | http://frt.iss.wisc.edu | NEUPFC6 | Office - Windows Live | Other Bookmarks katyhuff@gmail.com | My favorites | Profile | Sign out

cyclus

A Nuclear Fuel Cycle Simulation Code from the University of Wisconsin - Madison

Project Home Downloads Wiki Issues Source Administrator

Checkout Browse Changes Search Trunk Request code review

Source path: svn/

Directories	Filename	Size	Rev	Date	Author
svn	App.cpp	2.0 KB	r314	May 19, 2011	kathyhuff
branches	CMakeLists.txt	3.5 KB	r289	Apr 30, 2011	kathyhuff
doc	Commodity.cpp	903 bytes	r111	Jul 24, 2010	kathyhuff
trunk	Commodity.h	2.3 KB	r111	Jul 24, 2010	kathyhuff
input	Communicator.cpp	1.8 KB	r117	Jul 29, 2010	kathyhuff
src	Communicator.h	1.6 KB	r117	Jul 29, 2010	kathyhuff
CMake	InputXML.cpp	7.0 KB	r301	May 5, 2011	kathyhuff
Models	InputXML.h	9.2 KB	r115	Jul 28, 2010	kathyhuff
Testing	Logician.cpp	8.4 KB	r240	Feb 23, 2011	Matthew.Gidden
Utility	Logician.h	7.2 KB	r166	Oct 19, 2010	kathyhuff
doc	Material.cpp	22.8 KB	r334	Jun 4, 2011	Matthew.Gidden
wiki	Material.h	14.5 KB	r334	Jun 4, 2011	Matthew.Gidden

Your project is using approximately 7.0 MB out of 4096 MB total quota.
You can [reset this repository](#) so that svnsync can be used to upload existing code history.



'Modified Open' Source

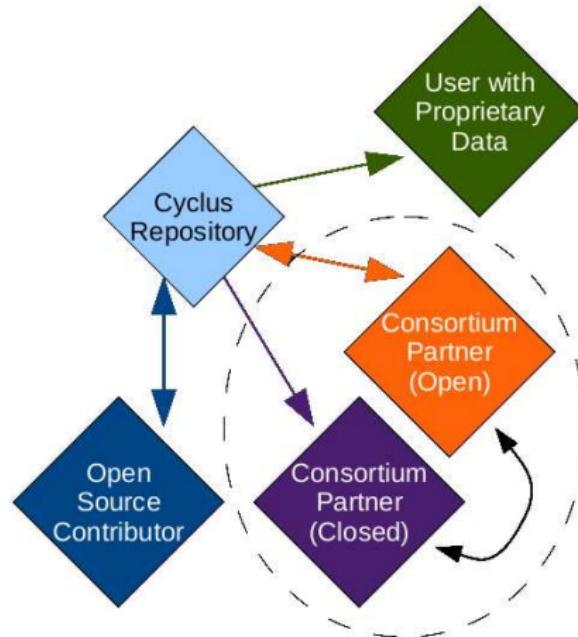


Figure: License, architecture, and development paradigm allow varying levels of code sharing and data security.



Version Control

This open source repository employs a version control system for provenance, developer access, and reproducibility of results.

code.google.com/p/cyclus/source/browse/#svn%2Ftrunk%2Fsrc

Apple | Yahoo! | Google Maps | News | Wikipedia | Popular | proxy | Note in Reader | http://frt.iss.wisc.edu | NEUPFC6 | Office - Windows Live | Other Bookmarks | katyhuff@gmail.com | My favorites | Profile | Sign out

cyclus

A Nuclear Fuel Cycle Simulation Code from the University of Wisconsin - Madison

Project Home Downloads Wiki Issues Source Administrator

Checkout Browse Changes Search Trunk Request code review

Source path: svn/

Directories	Filename	Size	Rev	Date	Author
svn	App.cpp	2.0 KB	r314	May 19, 2011	kathyhuff
branches	CMakeLists.txt	3.5 KB	r289	Apr 30, 2011	kathyhuff
doc	Commodity.cpp	903 bytes	r111	Jul 24, 2010	kathyhuff
trunk	Commodity.h	2.3 KB	r111	Jul 24, 2010	kathyhuff
Input	Communicator.cpp	1.8 KB	r117	Jul 29, 2010	kathyhuff
src	Communicator.h	1.6 KB	r117	Jul 29, 2010	kathyhuff
CMake	InputXML.cpp	7.0 KB	r301	May 5, 2011	kathyhuff
Models	InputXML.h	9.2 KB	r115	Jul 28, 2010	kathyhuff
Testing	Logician.cpp	8.4 KB	r240	Feb 23, 2011	Matthew.Gidden
Utility	Logician.h	7.2 KB	r166	Oct 19, 2010	kathyhuff
doc	Material.cpp	22.8 KB	r334	Jun 4, 2011	Matthew.Gidden
wiki	Material.h	14.5 KB	r334	Jun 4, 2011	Matthew.Gidden

Your project is using approximately 7.0 MB out of 4096 MB total quota.
You can [reset this repository](#) so that svnsync can be used to upload existing code history.



Issue Tracking

A built-in issue tracking system allows ticketing of development milestones and public bug reporting.

Screenshot of the Cyclus issue tracking system interface:

The URL in the browser bar is code.google.com/p/cyclus/issues/list.

The search bar contains "for" and a placeholder "Tip: Type ? for issue tracker keyboard shortcut help."

The main table displays the following issues:

ID	Type	Status	Priority	Milestone	Owner	Summary + Labels
42	Enhancement	Assigned	High	NullSimulation	Matthew.Gidden	Possible Issue Passing Material During Market Matches
21	Enhancement	Accepted	High	INPRO_01	Matthew.Gidden	Growth region - on the fly
22	Enhancement	Accepted	High	INPRO_01	Matthew.Gidden	Flesh out INPRO required pre-processing
23	Enhancement	Accepted	High	INPRO_01	Matthew.Gidden	Flesh out INPRO required post-processing
25	Enhancement	Accepted	High	INPRO_01	Matthew.Gidden	Determine the remaining capabilities for INPRO after deployment, pre-, and post-processing
28	Enhancement	Accepted	High	INPRO_01	katyhuff	Output Database - Target Date 4/4/2011
9	Design	Started	High	INPRO_01	katyhuff	Define Fundamental Data Unit for Output
38	---	Accepted	Low	INPRO_01	Matthew.Gidden	Add XML input for BuildRegion Vector Capability
41	---	Accepted	Critical	NWTRB_01	Matthew.Gidden	Document and complete NWTRB simple test case
26	Enhancement	New	High	NWTRB_01	Royal.Elmore	Create NWTRB Facility Catalog - Target Date 4/11/2011
27	Enhancement	New	High	NWTRB_01	Royal.Elmore	NWTRB Recipe Book - Target Date 4/11/2011



Collaborative Verification

The **transparency** inherent in this type of open source development path also **facilitates code review** by exposing available content to verification and validation **by collaborators** with diverse areas of specialization and levels of expertise.

A screenshot of a web browser displaying the Cyclus project page on Google Code. The URL in the address bar is `code.google.com/p/cyclus/`. The page title is "cyclus" with a subtitle "A Nuclear Fuel Cycle Simulation Code from the University of Wisconsin - Madison". Below the title is a navigation menu with links to "Project Home", "Downloads", "Wiki", "Issues", "Source", and "Administer". A secondary navigation menu below it includes "Summary", "Updates", and "People". On the left side, there is a sidebar with sections for "Project Information", "Code license", "Labels", "Members", "Your role", "Links", and "Groups". The "Project Information" section shows that the project was started by 7 users, has high activity and many project feeds, and is licensed under New BSD License. The "Labels" section lists "nuclear", "energy", "simulation", "university", "python", "c", and "fuel.cycle". The "Members" section lists "katyhuff", "uwgonzuki", "code.scrivener", and "4 committers". The "Your role" section shows "Owner". The "Links" section lists "External links" with "cnrg" and "Groups" with "Cyclus Developers List". The main content area contains sections for "Project Information", "Code license", "Labels", "Members", "Your role", "Links", "Groups", "Internal code development", "External code development", and "Links". The "Internal code development" section discusses the next-generation fuel cycle systems analysis simulation tool, Cyclus, and its successor to GENIUS. It mentions that the modeling paradigm will let users reconfigure the basic building blocks of a simulation without changing the s of a simulation will be a commodity market that collects offers and requests and matches them according to some algori able to select which type of algorithm is used for each market by selecting a MarketModel and configuring it with a parti defined by that MarketModel. Changing the parameters of a market will change its performance and selecting a different completely change its behavior. The "External code development" section lists replicating GENIUS2 functionality, developing a unit test suite, and deploying a library of basic data and a catalog of market and facility configurations.

As a successor to **GENIUS**, the next-generation fuel cycle systems analysis simulation tool, **Cyclus** will preserve many a software architecture that provides a great deal of flexibility, both in terms of modifying the underlying modeling algorithm different levels of complexity to different users.

The Cyclus modeling paradigm will let users reconfigure the basic building blocks of a simulation without changing the *s* of a simulation will be a commodity market that collects offers and requests and matches them according to some *algo* able to select which type of algorithm is used for each market by selecting a *MarketModel* and configuring it with a *parti* defined by that *MarketModel*. Changing the parameters of a *market* will change its performance and selecting a *different* completely change its behavior.

- [Cyclus Introduction](#)
- [Getting and Building Cyclus](#)
- [User Documentation](#)
- [Developer Documentation](#)

Internal code development

Most internal code development at this time will be led by [Katy Huff](#), [Paul Wilson](#), [Matthew Gidden](#), and [Royal Elmore](#), group. Near term goals include :

- replicating GENIUS2 functionality
- developing a unit test suite
- deploying a library of basic data and a catalog of market and facility configurations.

External code development

Interested developers are welcome and encouraged to contribute, but will experience significant code instability in the *e*



Testing Framework

A testing framework built on a cross-platform, multi-language build system (CMake) allows developers to incorporate unit and integration tests into their code before it is committed.



References I

- [1] Kathryn Huff and Brent Dixon.

Next generation fuel cycle simulator functions and requirements document.

Technical Report fcrd-sysa-2010-000110, July 2010.

- [2] Kathryn Huff, Paul P. H. Wilson, and Matthew Gidden.

Cyclus: A nuclear fuel cycle code from the university of wisconsin madison.

<http://code.google.com/p/cyclus/>, 2010.

- [3] P. Lisowski.

Global nuclear energy partnership.

In *Global Nuclear Energy Partnership Annual Meeting*, 2007.

- [4] K. M Oliver.

GENIUSv2: software design and mathematical formulations for multi-region discrete nuclear fuel cycle simulation and analysis.

PhD thesis, UNIVERSITY OF WISCONSIN, 2009.