

An Integrated Used Fuel Disposition and Generic Repository Model

A Nuclear Engineering and Engineering Physics PhD Preliminary Report

Kathryn D. Huff

University of Wisconsin-Madison

September 1, 2011





Outline

① Introduction

Motivation

Methodology

② Literature Review

Repository Capabilities within Systems Analysis Tools

Conceptual Discussion of Disposal Environments

Models of Radionuclide Transport

Models of Heat Transport

③ Modeling Paradigm

CYCLUS Simulator Paradigm

Repository Modeling Paradigm

④ Proposed Work

Demonstration Case

Base Case

Extensions

Summary



Future Fuel Cycle Options

Domestic Fuel Cycle Options

Title	Description	Challenges
Open	Once Through	High Temperatures, Volumes
Modified Open	Partial Recycling	Both high volumes and myriad fuel streams
Closed	Full Recycling	Myriad fuel streams

Table: Domestic Fuel Cycle Options

Abstraction





Outline

① Introduction

Motivation
Methodology

② Literature Review

Repository Capabilities within Systems Analysis Tools
Conceptual Discussion of Disposal Environments
Models of Radionuclide Transport
Models of Heat Transport

③ Modeling Paradigm

CYCLUS Simulator Paradigm
Repository Modeling Paradigm

④ Proposed Work

Demonstration Case
Base Case
Extensions
Summary

Top Level Fuel Cycle Simulators





Need For an Integrated Repository Model



Clay Disposal Environments

Granite Disposal Environments



Salt Disposal Environments





Deep Borehole Disposal Environment

Waste Form Release Models



Waste Package Failure Models





Transport Through Buffer Material



Transport Through Geology



Impact of Repository Designs



Heat Limits In Geology



Analytical Models



Detailed Techniques

Lumped Parameter Technique





Outline

① Introduction

- Motivation
- Methodology

② Literature Review

- Repository Capabilities within Systems Analysis Tools
- Conceptual Discussion of Disposal Environments
- Models of Radionuclide Transport
- Models of Heat Transport

③ Modeling Paradigm

- CYCLUS Simulator Paradigm
- Repository Modeling Paradigm

④ Proposed Work

- Demonstration Case
- Base Case
- Extensions
- Summary



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.



Encapsulation

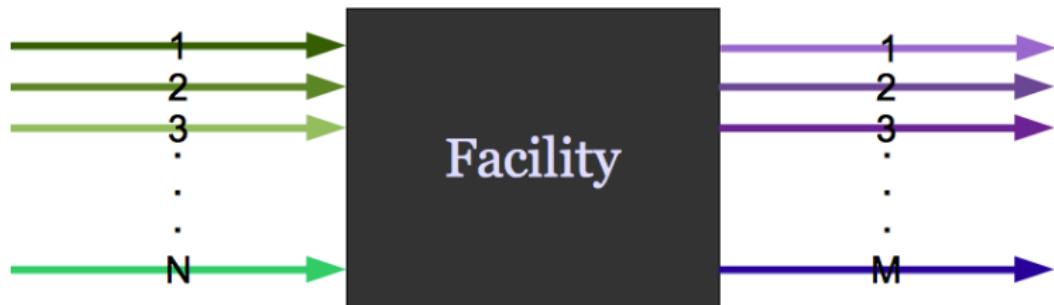


Figure: Regions, Institutions, Facilities, and Markets are all black boxes.



Module Interfaces

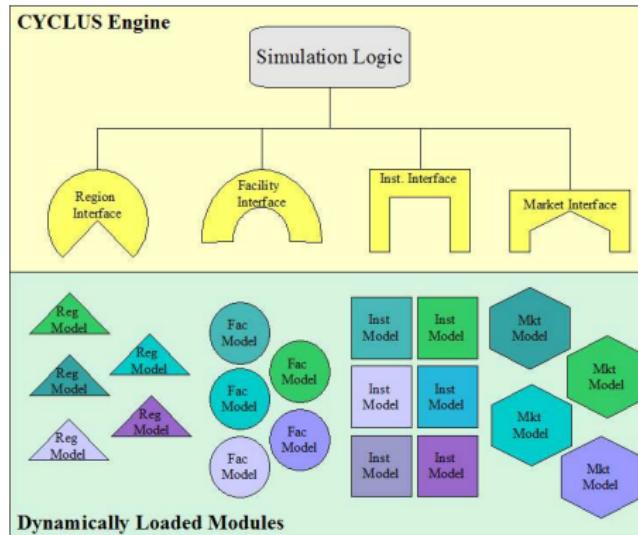


Figure: Well defined model interfaces facilitate model interchange. The user may choose the model at their desired level of detail.



Facilities Are Black Boxes

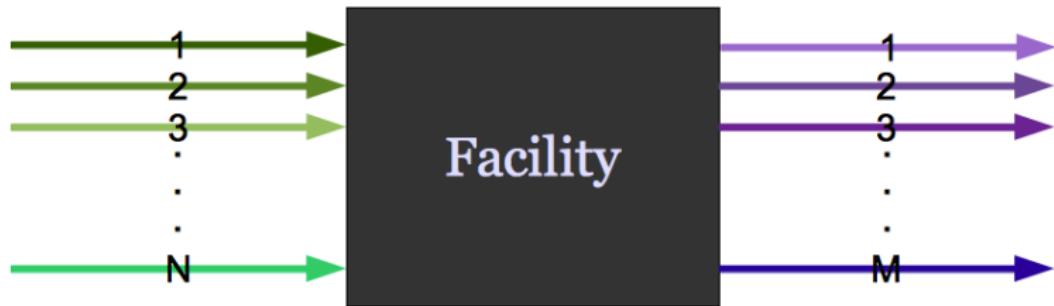


Figure: Each facility in the simulation makes requests and offers to fill its stocks and empty its inventory respectively.



Facilities Are Black Boxes



Figure: A facility might only make offers.



Facilities Are Black Boxes

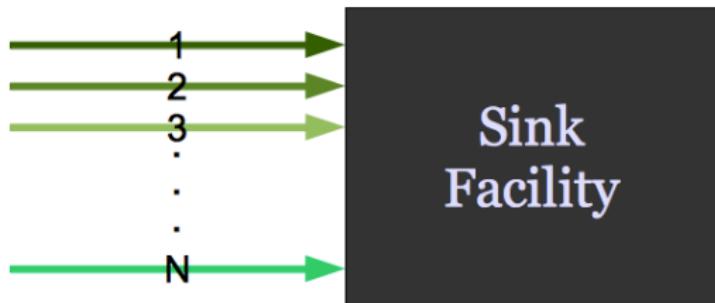


Figure: A facility might only make requests.



Each Commodity is Associated with a Market



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



The Market Solves the Matching Problem

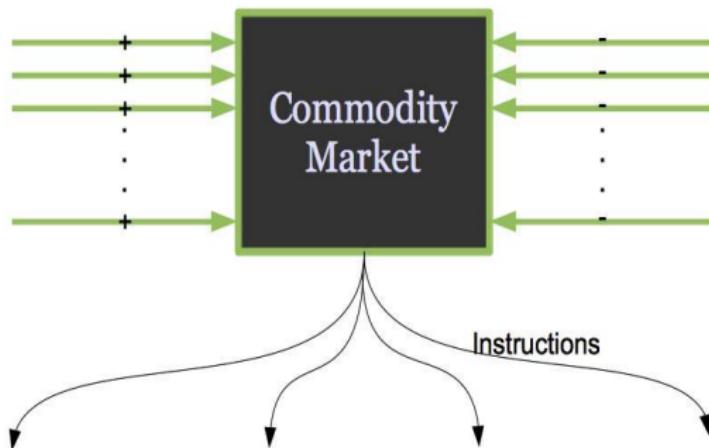


Figure: When the Market's arbitrary algorithm solves the matching problem, the Market sends instructions to the offering facilities.



A Simple Example

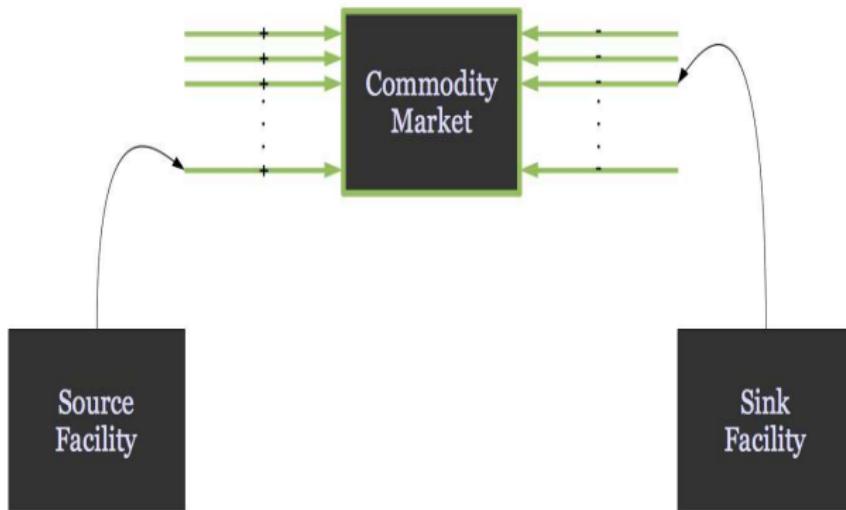


Figure: The source sends an offer and the sink sends a request.



A Simple Example

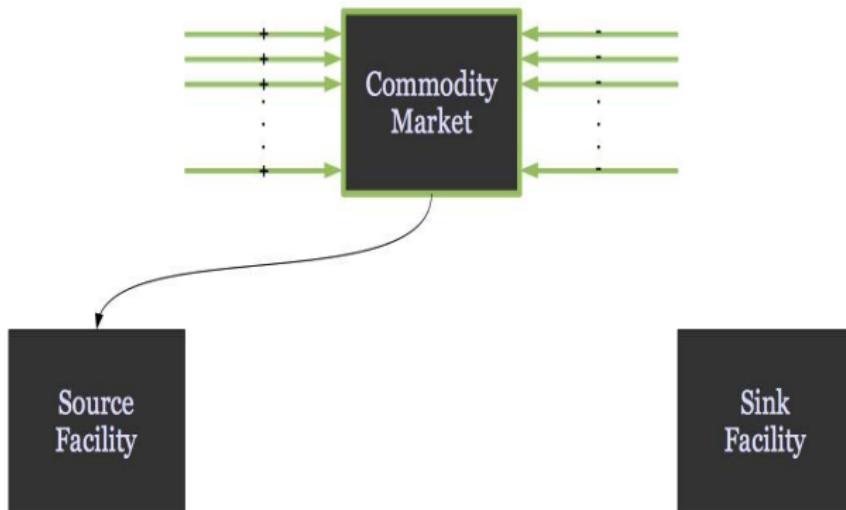


Figure: The Market solves the problem and instructs the source facility to send a certain amount to the sink facility.



A Simple Example

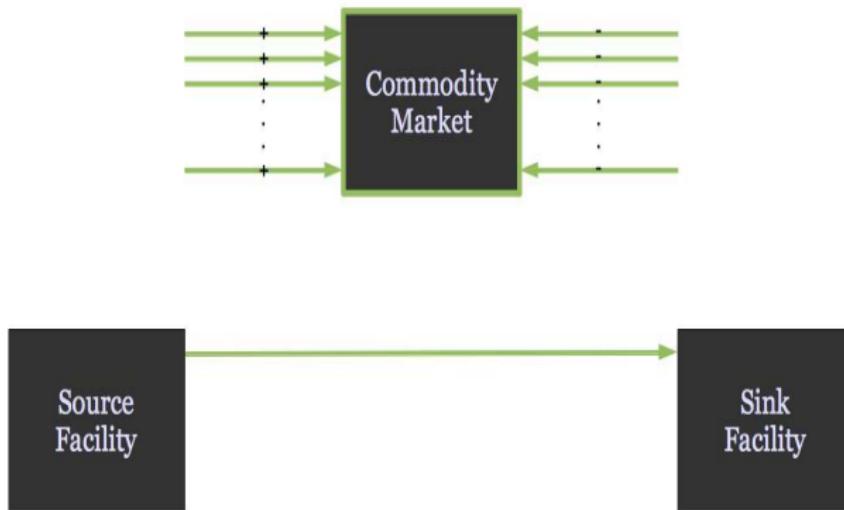


Figure: The source facility sends the material directly to the sink facility.



This Market Model Scales for Complex Systems

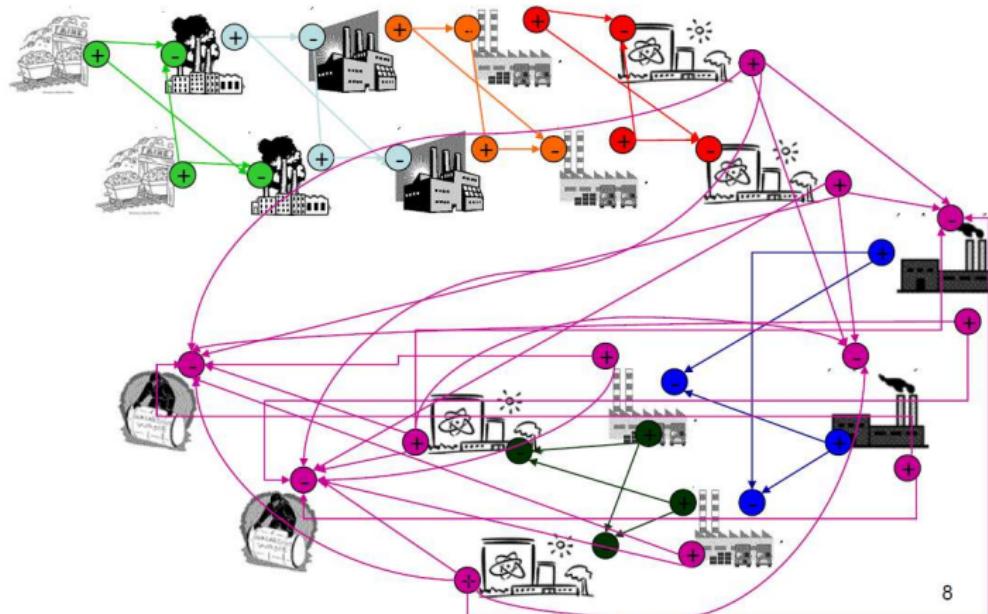


Figure: Well designed interfaces and strict encapsulation support scalability of the Market-based simulation paradigm [?]



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

Figure: Dynamic c library loading separates simulation logic from knowledge of available models, supporting extensions by developers with minimal lines of code.



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 is available for the end user to define available module implementations.



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://frit.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	katyhuff
branches	CMakeLists.txt	3.5 KB	r289	Apr 30, 2011	katyhuff
doc	Commodity.cpp	903 bytes	r111	Jul 24, 2010	katyhuff
trunk	Commodity.h	2.3 KB	r111	Jul 24, 2010	katyhuff
input	Communicator.cpp	1.8 KB	r117	Jul 29, 2010	katyhuff
src	Communicator.h	1.6 KB	r117	Jul 29, 2010	katyhuff
	InputXML.cpp	7.0 KB	r301	May 5, 2011	katyhuff
	InputXML.h	9.2 KB	r115	Jul 28, 2010	katyhuff
	Logician.cpp	8.4 KB	r240	Feb 23, 2011	Matthew.Gidden
wiki	Logician.h	7.2 KB	r166	Oct 19, 2010	katyhuff
	Material.cpp	22.8 KB	r334	Jun 4, 2011	Matthew.Gidden
	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 svn:sync can be used to upload existing code history.



'Modified Open' Source

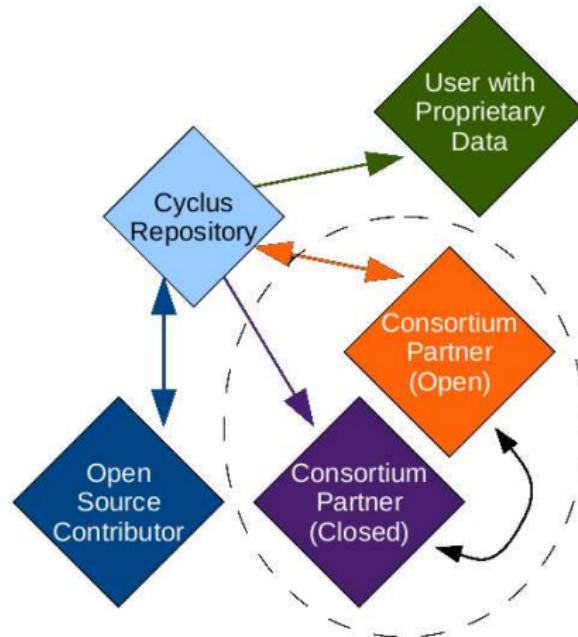


Figure: License, architecture, and development paradigm allow varying levels of code sharing and data security.



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	katyhuff
branches	CMakeLists.txt	3.5 KB	r289	Apr 30, 2011	katyhuff
doc		903 bytes	r111	Jul 24, 2010	katyhuff
trunk	Commodity.cpp	2.3 KB	r111	Jul 24, 2010	katyhuff
input	Commodity.h	1.8 KB	r117	Jul 29, 2010	katyhuff
src	Communicator.cpp	1.6 KB	r117	Jul 29, 2010	katyhuff
	Communicator.h	7.0 KB	r301	May 5, 2011	katyhuff
	InputXML.cpp	9.2 KB	r115	Jul 28, 2010	katyhuff
	InputXML.h	8.4 KB	r240	Feb 23, 2011	Matthew.Gidden
wiki	Logician.cpp	7.2 KB	r166	Oct 19, 2010	katyhuff
	Logician.h	22.8 KB	r334	Jun 4, 2011	Matthew.Gidden
	Material.cpp	14.5 KB	r334	Jun 4, 2011	Matthew.Gidden
	Material.h				

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



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.



Nested Components

Quantities Calculated Each Timestep

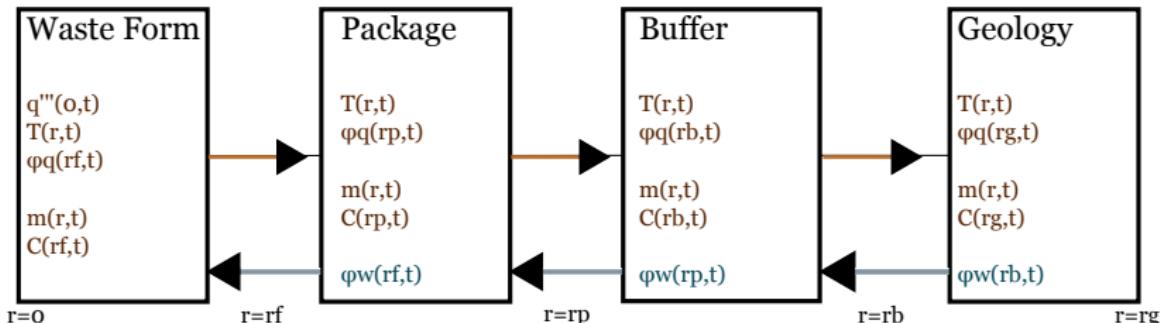


Figure: The nested components supply thermal flux and concentration information to each other at the boundaries.

Waste Stream



Waste Form



Waste Package



Buffer



Geological Environment





Outline

① Introduction

Motivation
Methodology

② Literature Review

Repository Capabilities within Systems Analysis Tools
Conceptual Discussion of Disposal Environments
Models of Radionuclide Transport
Models of Heat Transport

③ Modeling Paradigm

CYCLUS Simulator Paradigm
Repository Modeling Paradigm

④ Proposed Work

Demonstration Case
Base Case
Extensions
Summary



i++i

i++i

i++*i*



i++i

i++i

i++*i*



i++i

i++i

i++*i*



i++i

i++i

i++*j*



i++i

i++i

i++*i*



i++i

i++i

i++*i*



Demonstration Case : Concept



Demonstration Case : Testing



Base Case : Component Abstraction



Base Case : System Level Abstraction



Base Case : Concept



Base Case : Testing



Extensions : Abstraction



Extensions : Testing



References I