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

Top Level Simulators

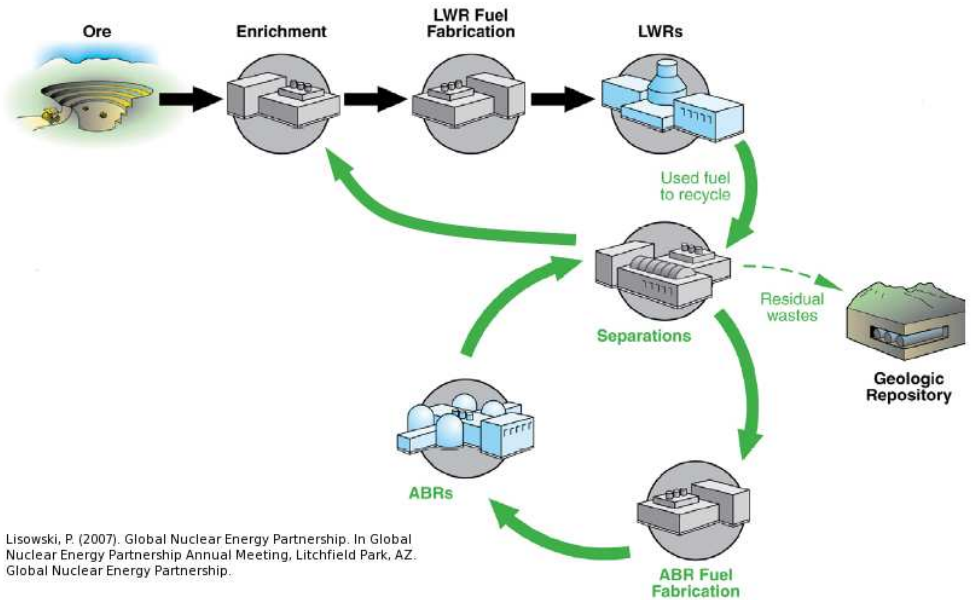


Figure: Top level simulators are intended to model the collective behavior of various fuel cycle decisions and strategies.[3]

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

Region Institution Facility Hierarchy

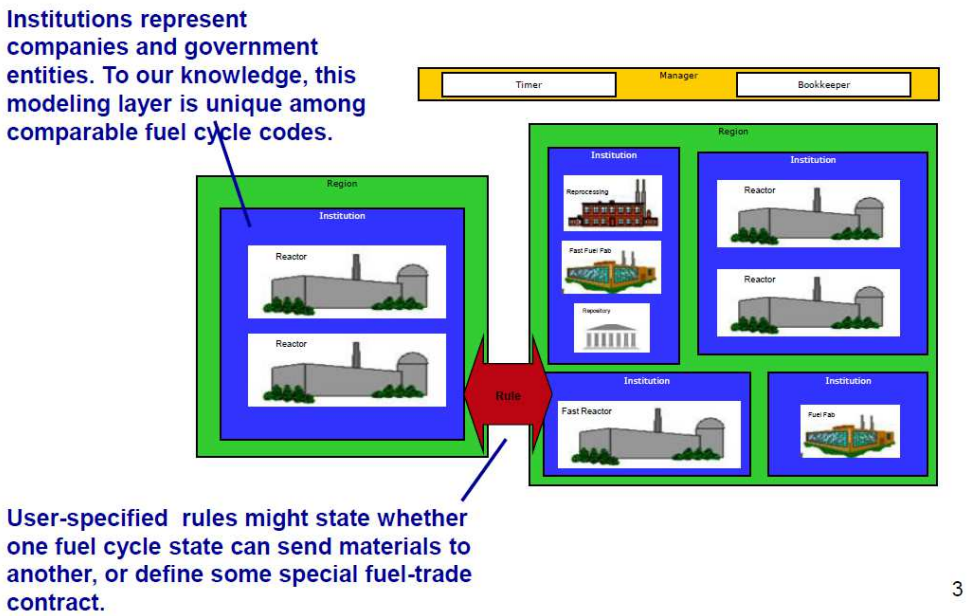


Figure: The Region, Institution, Facility Hierarchy of the Cyclus paradigm was a success of the GENIUSv2 code.[4]

Encapsulation

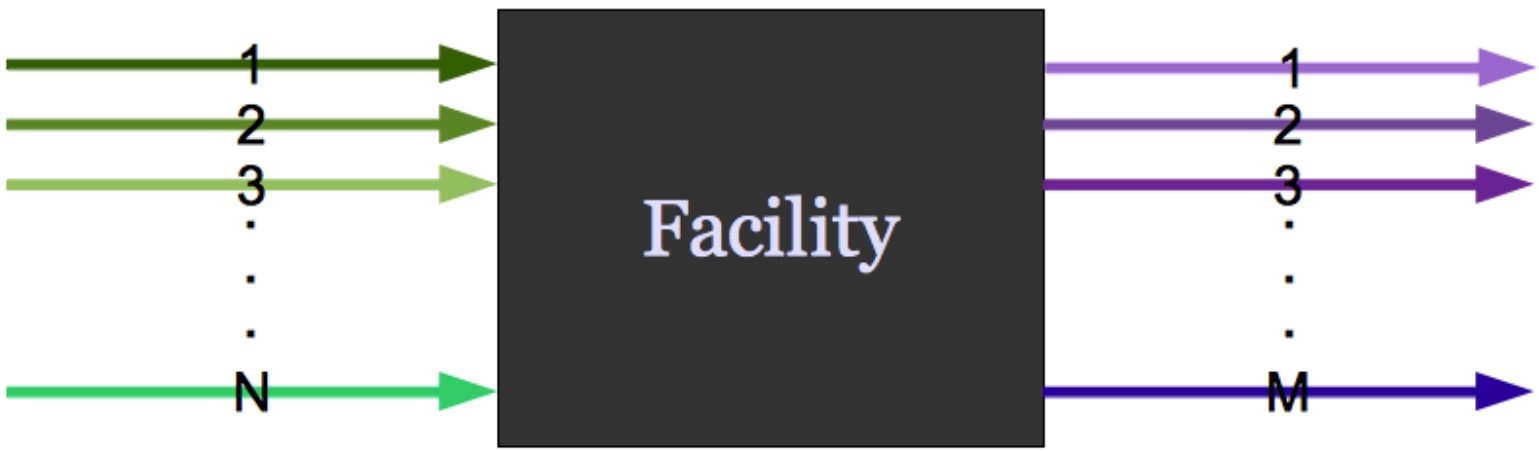


Figure: Regions, Institutions, Facilities, and Markets are all black boxes.

Module Interfaces

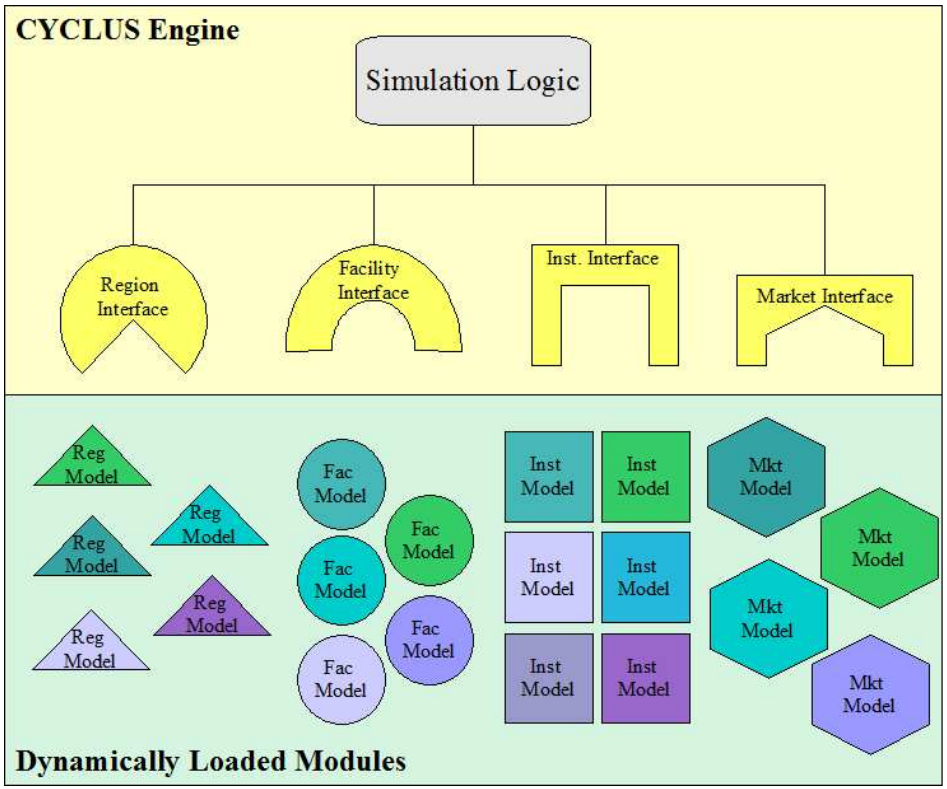


Figure: Well defined model interfaces facilitate model interchange. The user may choose the model at their desired level of detail.

Facilities Are Black Boxes

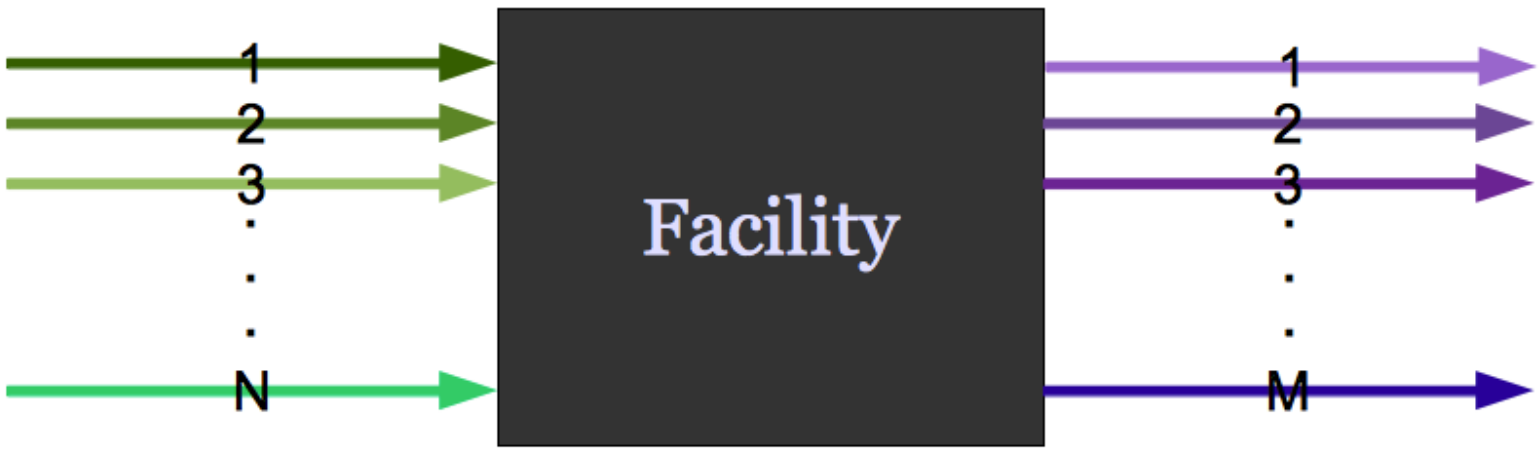


Figure: Each facility in the simulation makes requests and offers to fill its stocks and empty its inventory respectively.

Facilities Are Black Boxes

Figure: A facility might only make offers.

Facilities Are Black Boxes

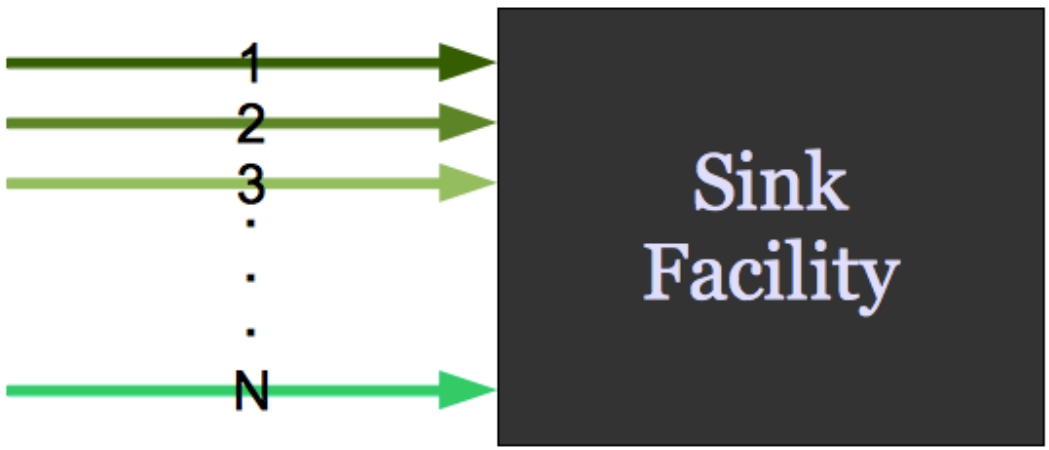


Figure: A facility might only make requests.

Each Commodity is Associated with a Market

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

The Market Solves the Matching Problem

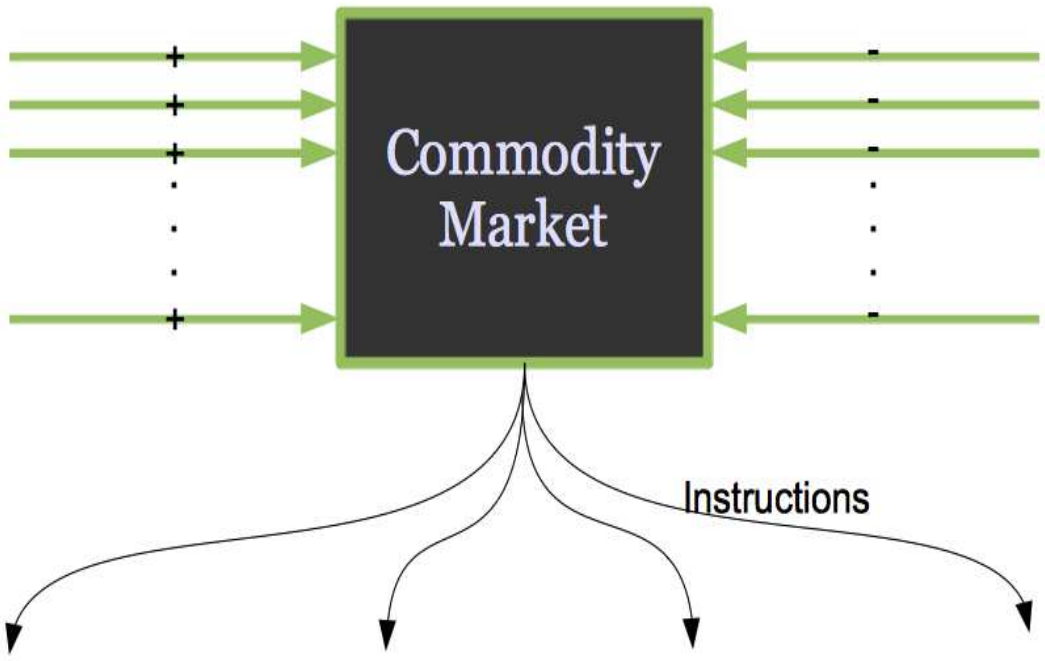


Figure: When the Market's arbitrary algorithm solves the matching problem, the Market sends instructions to the offering facilities.

A Simple Example

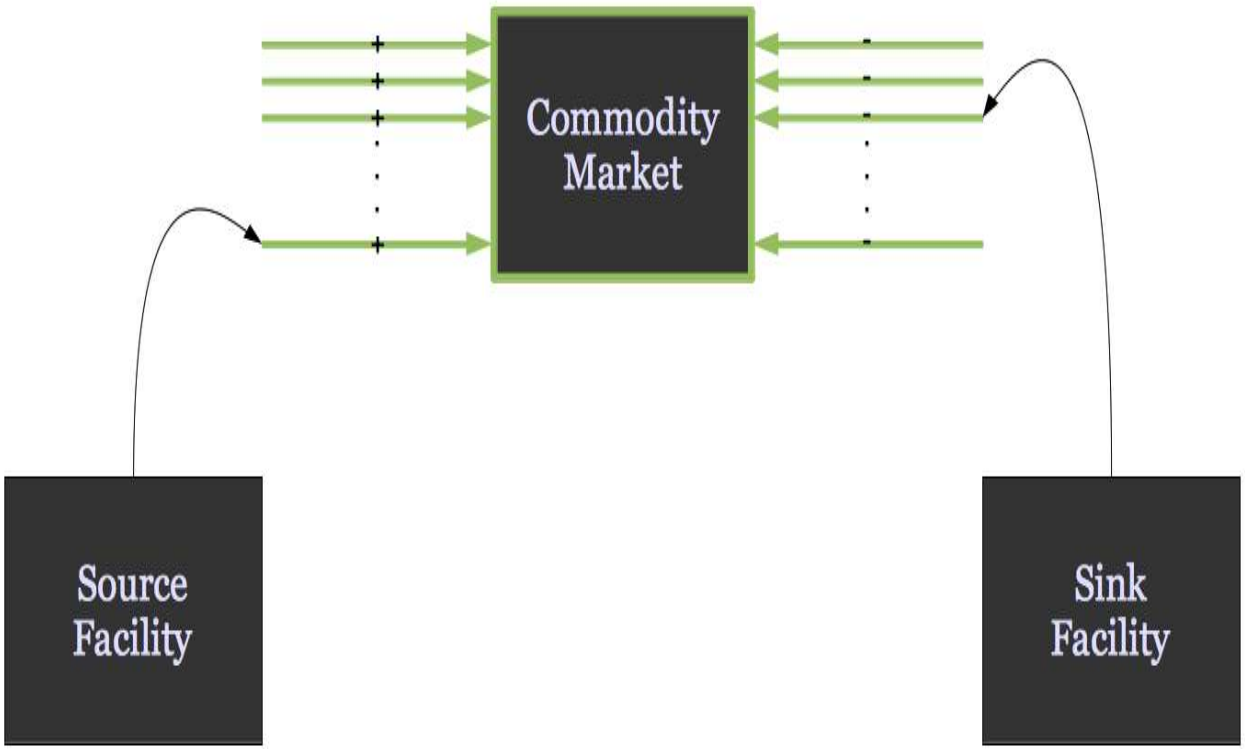


Figure: The source sends an offer and the sink sends a request.

A Simple Example

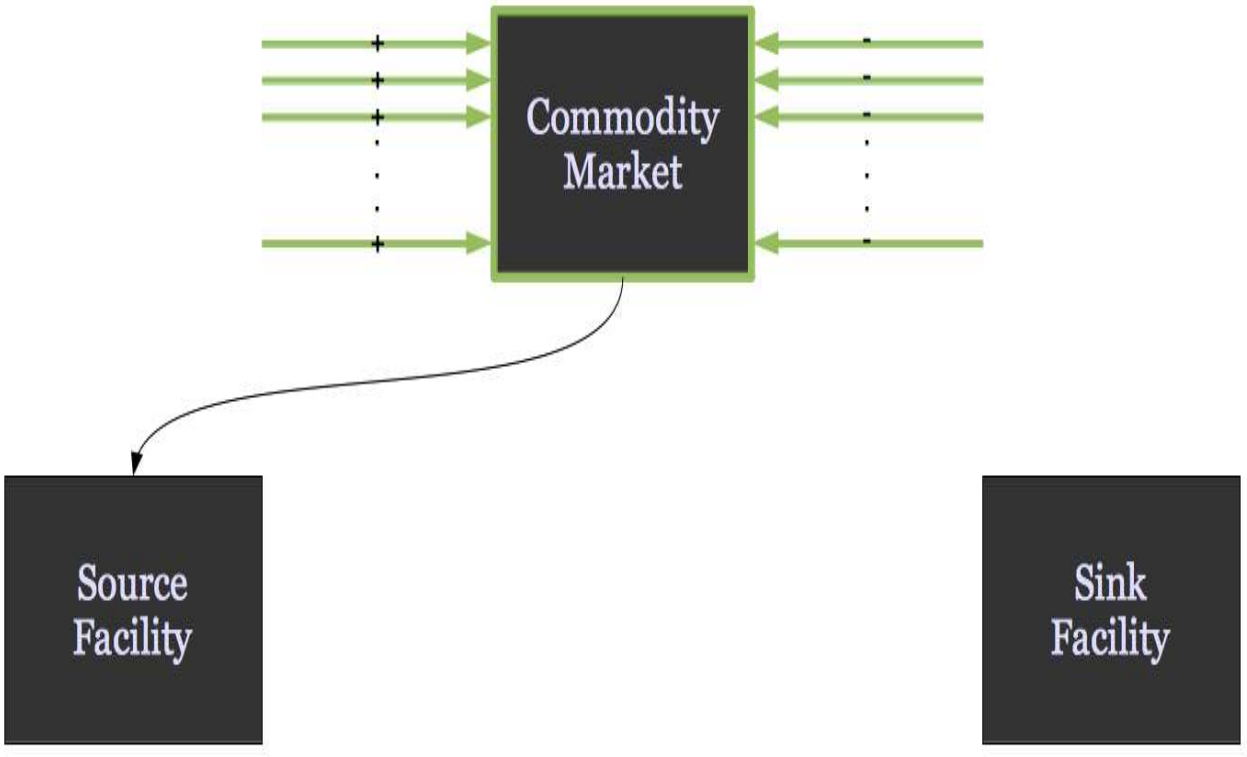


Figure: The Market solves the problem and instructs the source facility to send a certain amount to the sink facility.

A Simple Example

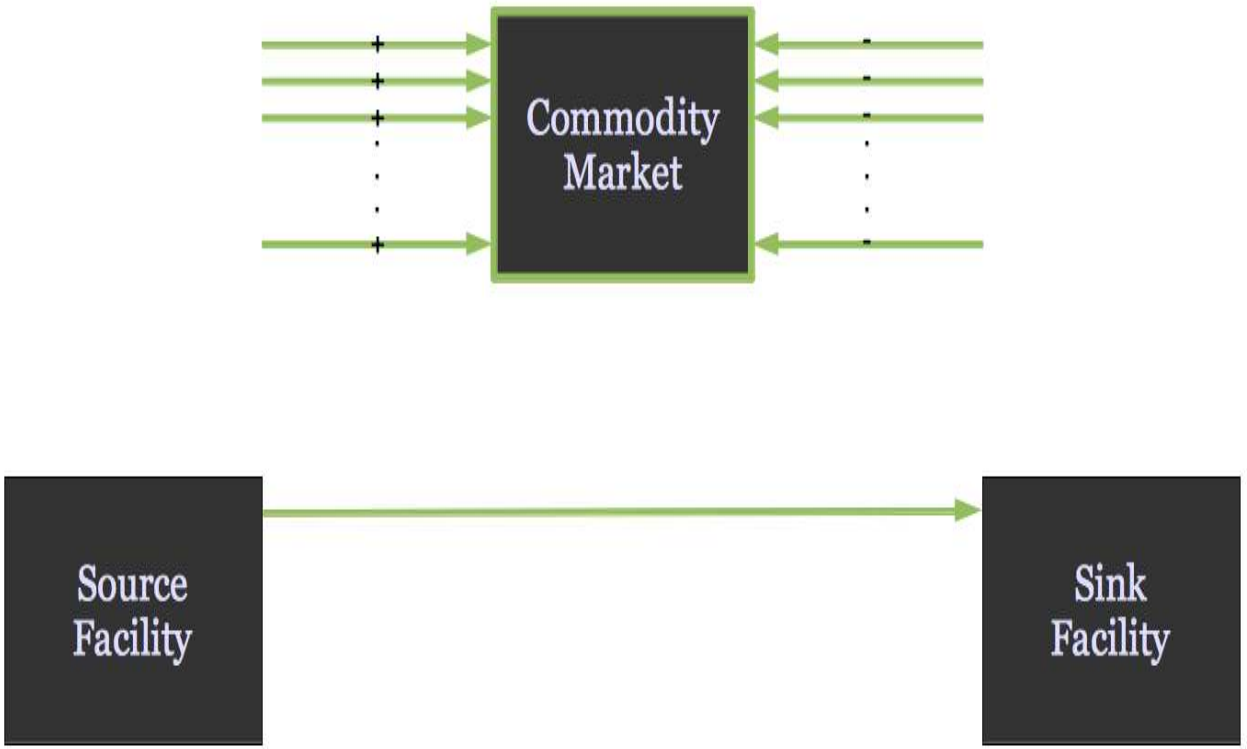


Figure: The source facility sends the material directly to the sink facility.

This Market Model Scales for Complex Systems

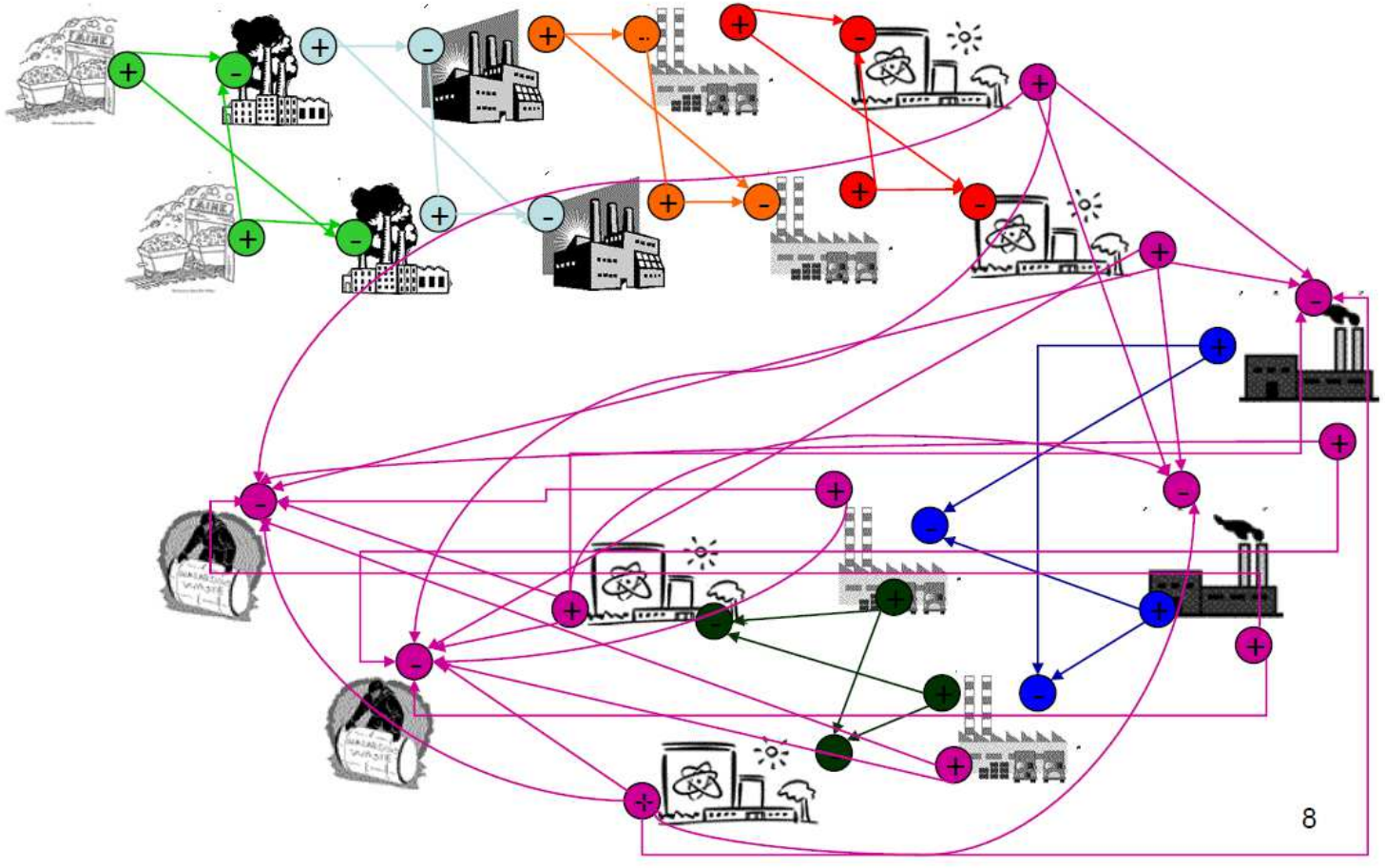


Figure: Well designed interfaces and strict encapsulation support scalability of the Market-based simulation paradigm [4]

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

```
<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>
```

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_ctor*) 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

Cross Platform, Multi-language

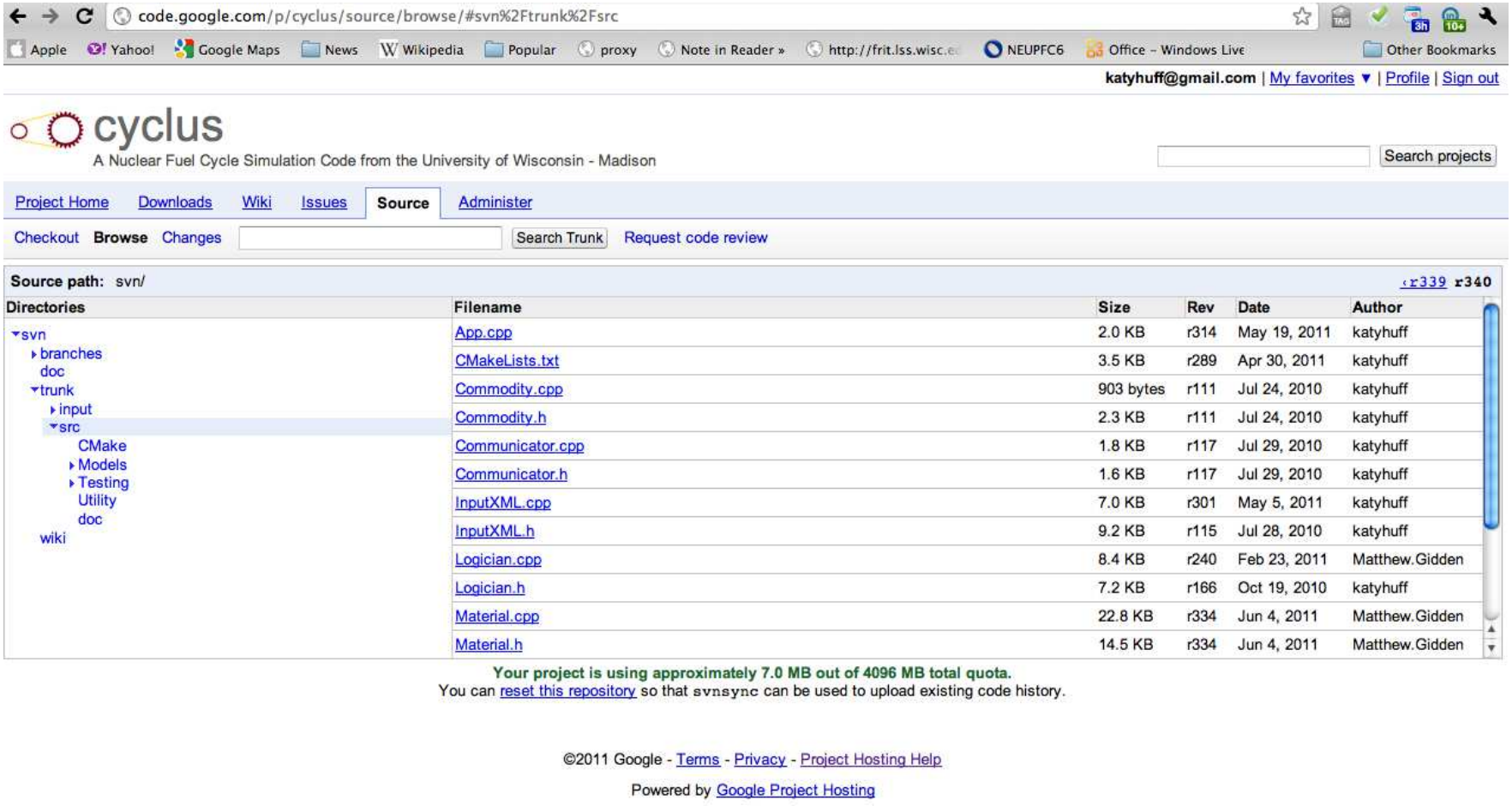
CMake (Crossplatform Make) build system is cross platform. Thus, users may build and run Cyclus on Windows, Mac, or Linux operating systems.
CMake is also multilingual, so modules may be written in any language or tool that can be made into a shared object library or wrapped with C++.
These include (but are not limited to):

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

Outline

Open Source Repository

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



‘Modified Open’ Source

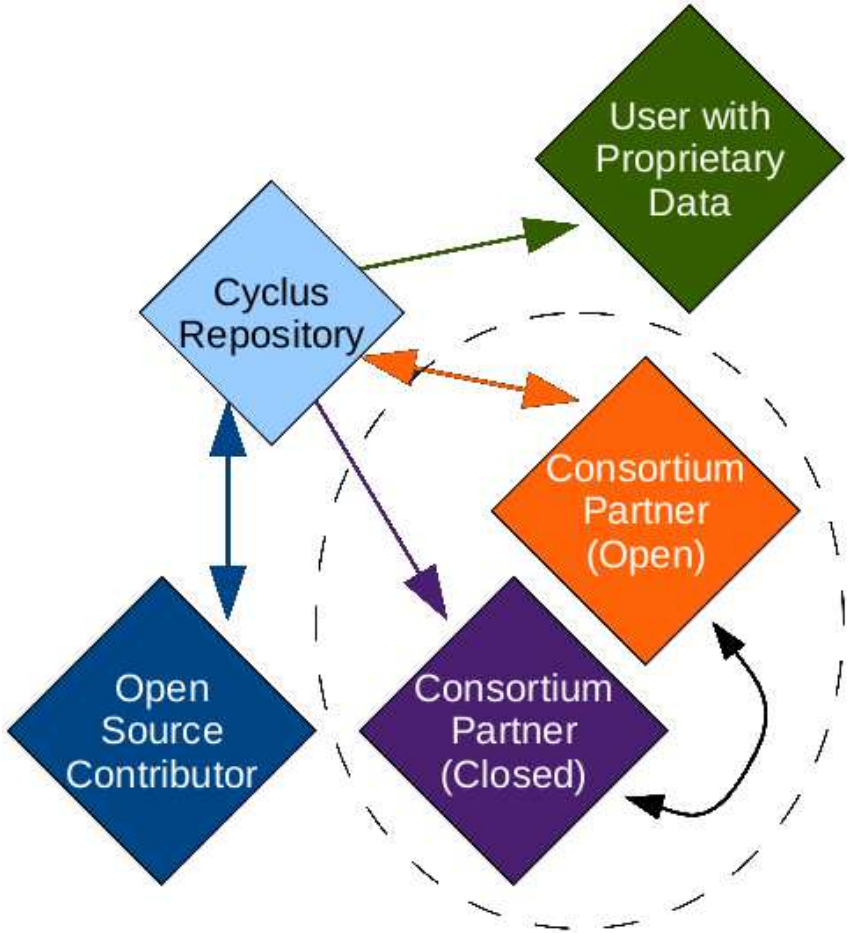
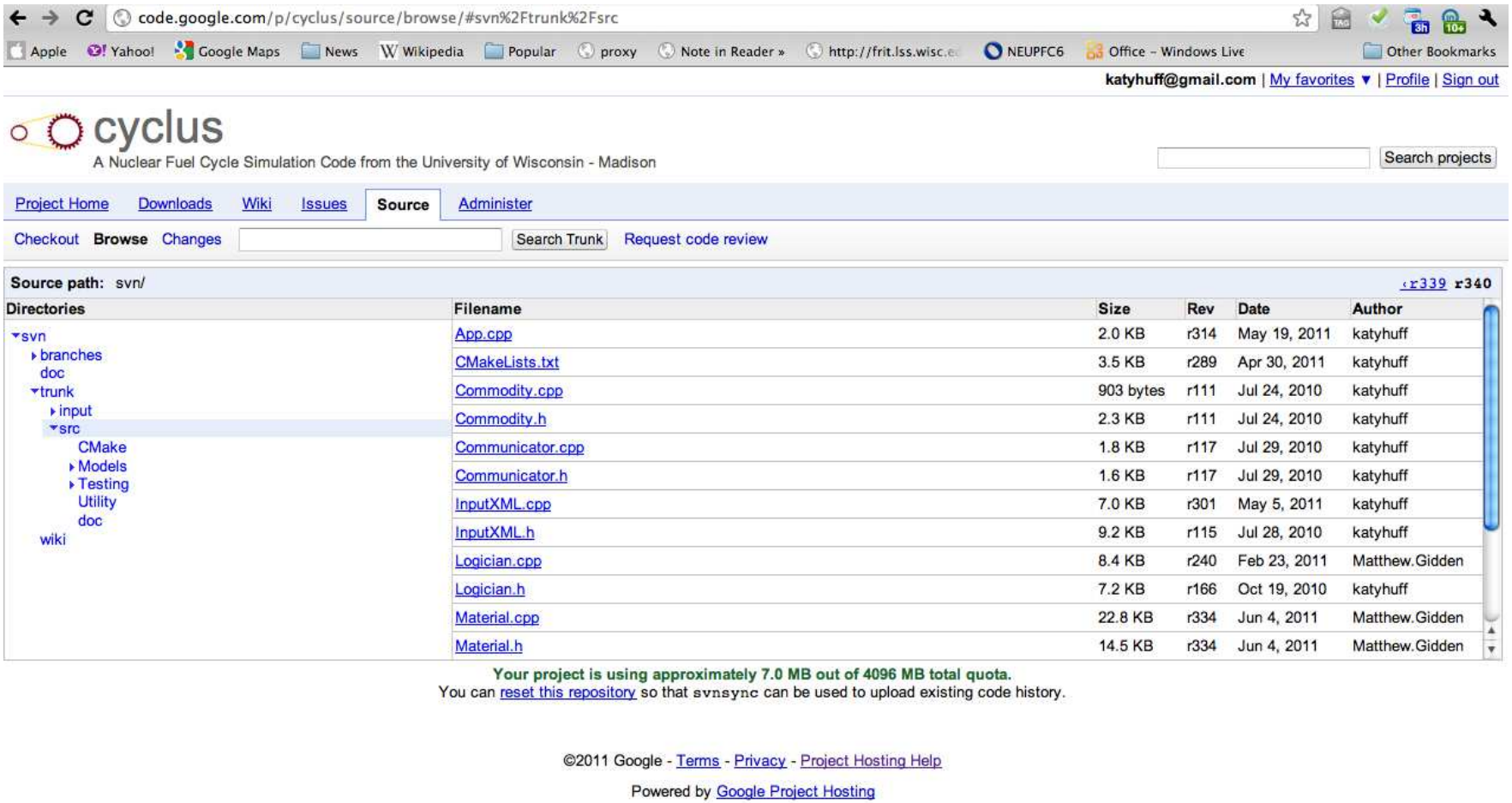


Figure: License, architecture, and development paradigm allow varying levels of code sharing and data security.

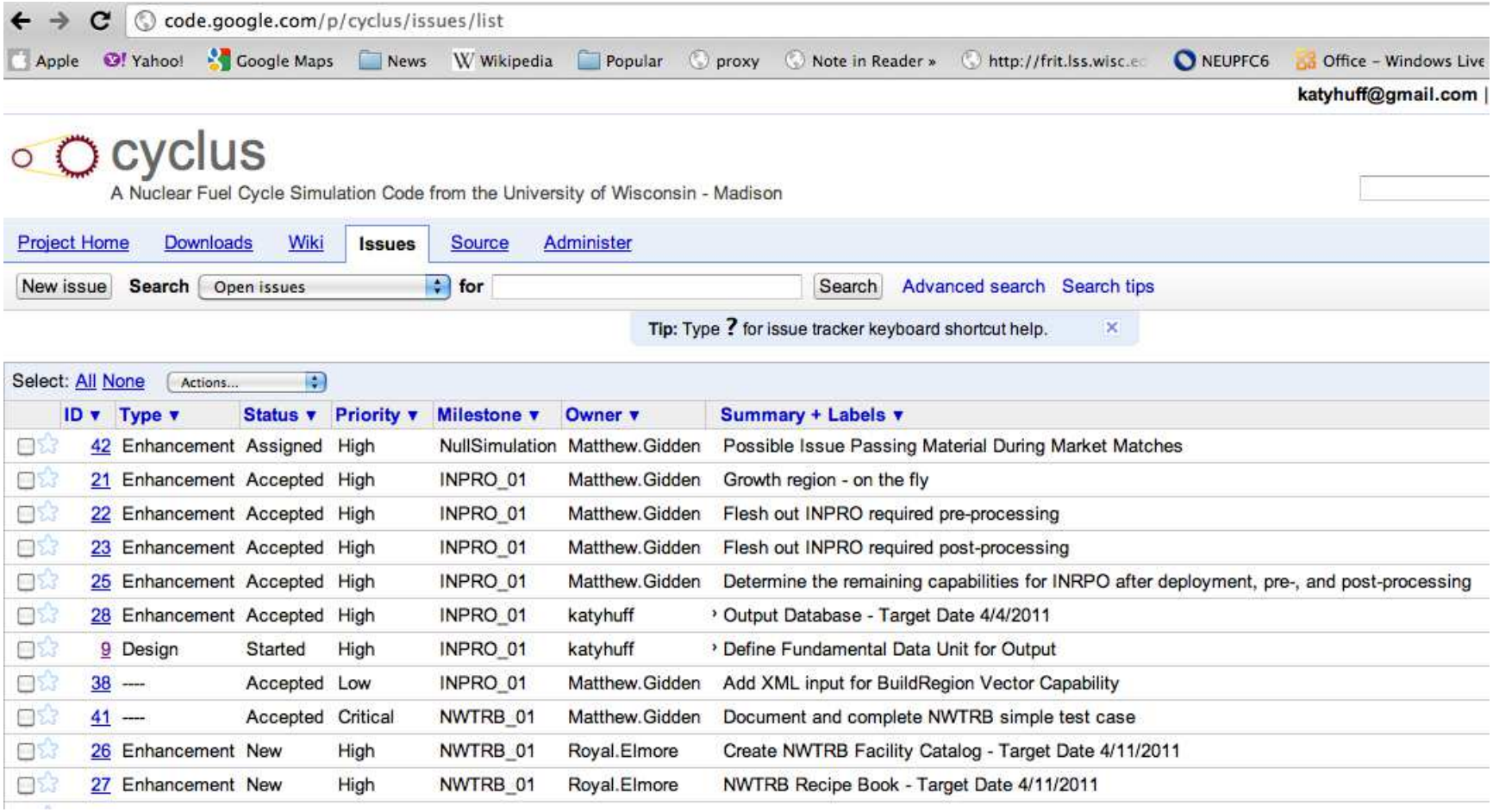
Version Control

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



Issue Tracking

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



code.google.com/p/cyclus/issues/list

Apple Yahoo! Google Maps News Wikipedia Popular proxy Note in Reader » http://frit.lss.wisc.e NEUPFC6 Office - Windows Live

katyhuff@gmail.com |

cyclus

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

Project Home | Downloads | Wiki | **Issues** | Source | Administer

New issue | **Search** Open issues **for** | Search | Advanced search | Search tips

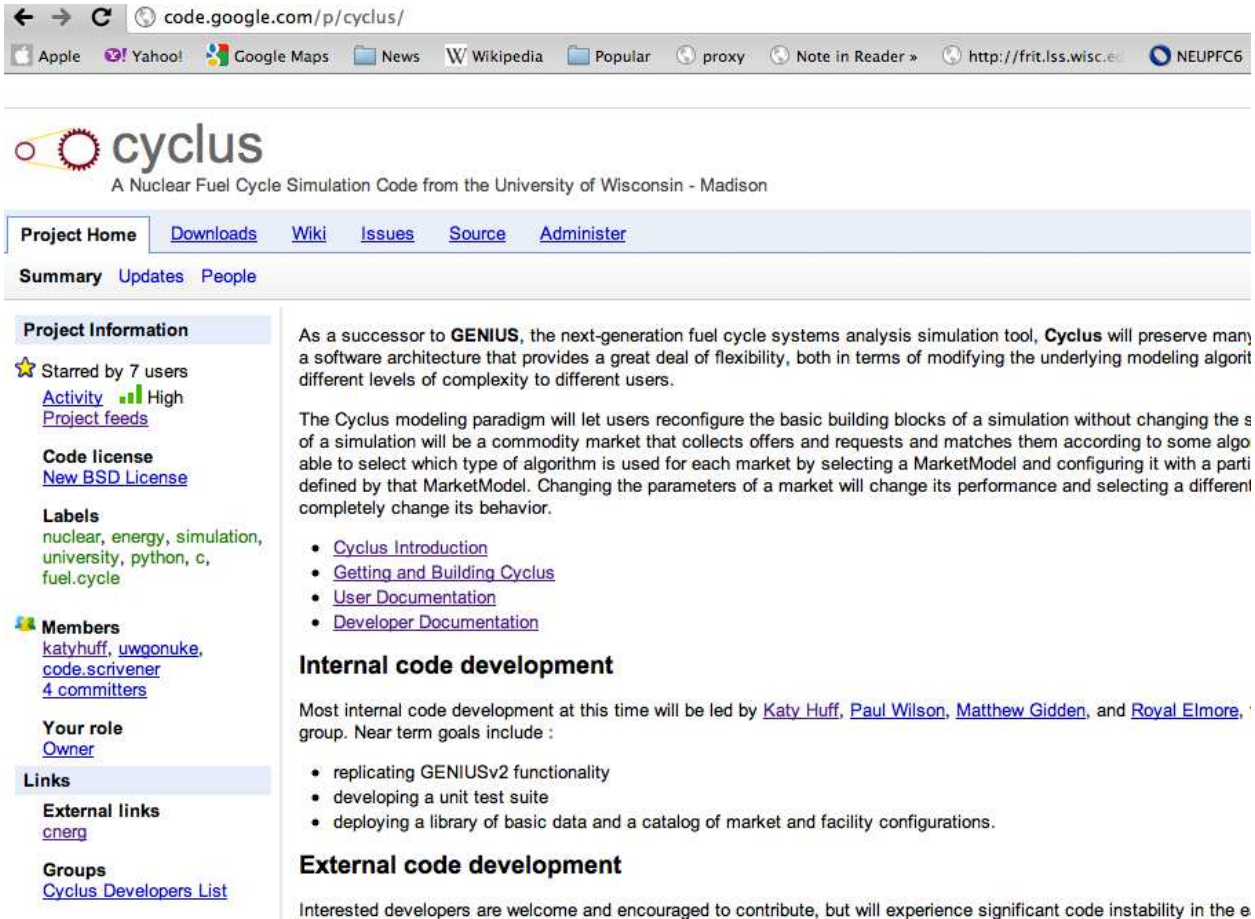
Tip: Type **?** for issue tracker keyboard shortcut help.

Select: All None | Actions...

	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 INRPO 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.



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 commited.

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.