The History of LANDIS-II Development 2004-2005

Note: This document was assembled in August 2013 from the original LANDIS-II web site, prior to transferring our content to a new site. The information was copied verbatim except where out-of-date (e.g., manuscript dates) and the formatting was changed to enhance PDF readability.

Unified Process: The process we're using to build LANDIS-II.
See Scheller et al. 2010, Frontier in Ecology and the Environment.

Project Risks: The risks and uncertainties for the development of LANDIS-II.

- Goal: Identify the primary risks for this project. Some of these risks apply across multiple iterations and are therefore included as a separate Unified Process document. Risks can also be thought of as project uncertainties.
- Risk 1: Component Variables:
  - Will each component be able to add its own unique list of variables to the landscape?
  - Successfully resolved in Iteration 1.
- Risk 2: Plug&Play:
  - Can dynamic libraries (aka Plug&Play) be implemented in C++? If so, how difficult?
  - If C# is required, will the new C# generics (eg templates) interact well with the existing C++ generics?
  - Will be addressed in Iteration 2.
- Risk 3: User Interface and Model Interaction:
  - Need to clarify the interactions between the core model and the user interface (both graphical and command-line). This will impact the model architecture.
  - Need to complete Use Cases to resolve.
  - Use Cases will be at least 50% complete for Iteration 2.
- Risk 4: Learning Unified Process:
  - We are still learning the Unified Process and adapting it to our needs. There may be some unnecssary efforts and some valuable pieces of the Unified Process may be missed.
  - Will require us to continue up the learning curve.

LANDIS-II Features: The features and their expected version dates.

- Goal: Identify the features for this project. Features are ranked by priority and are given an estimated version release.

Foundational Features [beta 1.0]

1. Age only succession
2. 3 seeding algorithms
3. Command-line user interface
4. Plug-n-play functionality for succession & disturbance components.

Primary Features [core v1.0]

- Installer

- Wind module, Fire module, Shade Output, Max Age Output, ReClass Output.
- User's Guides for all released modules.
- Web page with User Documentation
- Web page with database of available modules.

Secondary Features

- Succession with age and biomass [module release]
- Add late mortality to Age-only Succession [core v5.1].
- Add immutable site to List
- function?

Tertiary Features

- Climate change for biomass succession [biomass succession v1.1]
- Alternate starting dates [core v5.2]
- Import/Export Imagine (*.img) files with projection [raster library]
- Import ASCII files with meta-data [raster library]
- Number of Sites per Ecoregion as an attribute of ecoregion [core v5.2]
- Relative Directory adjustment [core v5.2]
- Scenario Directory name with macro [core v5.2]

Graphical User Interface

- Ability to view files [v 1.0]
- Dynamically validated input files [v 1.0]
- Creation and editing of Scenario Project files - a single compressed file containing all necessary files for a scenario [v 1.0]
- Unique legends for each file type [v 1.0]
- Dynamic zoom in/out [v 1.1]

Core Development Cycle 1 (version 1.0)

- Cycle type: Initial
- Completion date: January 2005

Elaboration Phase:

Iterations in this phase define and create the core architecture.

- **Iteration 1** - Landscape module (site variables and iterators).
  Finished August 26, 2004 -- **Assessment**
- **Iteration 2** - C++ and Plug-n-Play (dynamically linked libraries).
  Finished October 1, 2004 -- **Assessment**
- **Iteration 3** - Investigating C#.
  Finished November 2, 2004 -- **Assessment**
- **Iteration 4** - Rounding out the Core.
  Finished December 2, 2004 -- **Assessment**

Construction Phase:

Iterations in this phase focus on building the remaining parts of the application.

- **Iteration 5** - Wind, Species.
  Finished December 22, 2004 -- **Assessment**
- **Iteration 6** - Age-Only Succession, Species.
  Finished January 21, 2005 -- **Assessment**
- **Iteration 7** - Ecoregions, Reproduction.
  Finished February 17, 2005 -- **Assessment**

- **Iteration 8** - Alpha 1 release.
  Finished March 10, 2005 -- **Assessment**
- **Iteration 9** - Performance testing, Team meeting.
  Finished April 2, 2005 -- **Assessment**
- **Iteration 10** - Performance tuning, Features from team meeting.
  Finished April 21, 2005 -- **Assessment**

Transition Phase:

Iterations in this phase focus on preparing the application for distribution to users outside the project team.

- **Iteration 11** - Beta 1 release.
  Finished May 19, 2005 -- **Assessment**
- **Iteration 12** - Release candidate 1.
  Finished June 15, 2005 -- **Assessment**
- **Iteration 13** - Official Release.
  Finished August 5, 2005 -- **Assessment**

Development Cycle 2 (version 2.0)

- Cycle type: Evolutionary
- Completion date: Summer 2005

Features

- Graphical User Interface
- Ability to download and install new and updated components from our web site.

Iteration 1

by Robert Scheller — last modified Dec 04, 2004 04:07 PM

- Phase: Elaboration
- Dates: July 19 - Aug 6, 2004 (3 weeks)
- Finished: August 26, 2004

Tasks to be Completed

+ Architecture

- Design, code & test these core elements
- Landscape object
  - Focus on the portion of its interface that a component uses to add a site variable to the landscape.
- SiteSequence class
  - Represents a sequence of site on the landscape.
  - Used by a component to create a sequence and to add sites to it.
- SiteIterator class
  - Responsibile for iterating over a SiteSequence.
- SiteVariable class
  - Represents a particular site variable associated with a SiteIterator.

+ Tools

- Setup Subversion repository on maple
- Setup MetaMill on either maple or home PC

+ Unified Process

- Start the list of project risks

Assessment

+ Code Demonstration

- Code was compiled and ran from a command prompt.
- Format of log entries needs to be more readable for users.

+ Code Walk-thru

- Code is a work-in-progress; some cleanup necessary.
- More in-code commenting of critical statements needed.
- Looked in detail at main, AgeOnlySuccession, ComponentManager, Controller - all are very clean, easy to follow.

+ Iteration 1 Hits and Misses

- Miss: Out of four classes scheduled for completion, only SiteSequence was not completion, although it has been designed. The design includes an additional class for iterating through active sites (ActiveSiteIterator) that was not fully completed.
- Miss: Subversion not installed yet.
- Miss: Mapping grid locations to the landscape vector has been designed, but not implemented. The pseudo-code is written. This will be optimized at a later date.
- Miss/Hit: More classes and elements of the architecture completed than the 4 classes initially targeted. Did not anticipate that designing the interface for those 4 classes would require work on the other classes that use them. This took time

away from stated objectives. However, some code is now completed ahead of schedule.

- Hit: Logging and Boost libraries (that are open source code) have been a huge time-savings and have added extra functionality.
- Hit: Using Remote Desktop has been very smooth; Jimm's had no problems accessing workstation in Madison from St Paul.
- Huge Hit: Architecture for allowing each component to define variables on the landscape very successful.
- Huge Hit: Architecture for active vs. non-active sites: only active sites contain data. In landscapes with many non-forested sites, this will save large (huge) amounts of memory.

+ Iteration 1 Lessons Learned

- Due Date vs. Scheduled Tasks: Rather than attempt to finish all of the scheduled tasks, we should stick to the iteration due date for a couple reasons. First, Unified Process experts recommend this approach. Second, Rob's schedule will not be as flexible going into the semester and assessment meetings in Madison need to be planned weeks in advance.
- Estimating Time: Need to allocate time for classes that will interface with the classes scheduled for completion.
- Sequence Diagrams: We decided that we will not produce sequence diagrams at this time because: a) the source code will not be released in the near term; b) creating seq diagrams would require a huge time commitment from Jimm; and c) the audience for these diagrams is very small. Other related forms of programmer documentation (e.g., class documentation) can be generated much quicker.
- Metamill: At this time, Jimm will not use Metamill. Using a CASE (computer-aided software engineering) tool like Metamill would be valuable in the long term. However, Jimm hasn't used MetaMill, and in the short term, he will focus his efforts on finishing the architecture. Metamill can be introduced at a later point because it has a reverse-engineering feature where it reads in existing source code and produces class diagrams for the code.
- The Darn Things Works! The model already demonstrates considerable high-level functionality. The architecture has quickly advanced and demonstrates the ability to achieve most of the stated objectives.

Iteration 2

by Robert Scheller — last modified Dec 04, 2004 04:07 PM

- Phase: Elaboration
- Dates: Aug 30 - Sept 30, 2004

Tasks to be Completed

+ Architecture

- Finish Remaining Iteration 1 Tasks
  - Finish site-iterator classes.
  - Finish data structure to map active-site locations to their corresponding indexes into data arrays.
- Dynamic Loading Libraries, aka Plug 'n Play
  - Locate existing C++ code for dynamic libraries. Jimm knows of several open-source projects that use dynamic libs written in C++.
  - Re-use existing C++ code for dynamic libraries. This would save time, rather than having to introduce C# into the project.
  - Demonstrate dynamic libs using two Landis components (actual or test) that will read and write data to the landscape.

+ Tools

- Finish Subversion installation

+ Unified Process

- Begin writing Use Cases Have Use Cases 50% completed by end of iteration 2.

Assessment

+ Overview

Jimm completed most of the tasks outlined for iteration 2. During the iteration, the issue of binary compatibility between C++ compilers was discovered. There are several technical challenges to getting dynamic libraries made with different C++ compilers to work together. These challenges could make development by our collaborators very difficult. Jimm was able to address most of these C++ technical hurdles (although a few remain). However, these technical fixes would limit the C++ programming of module developers. We decided to spend two weeks of the next iteration exploring C# options for overcoming these limitations. Although a delay, the information gained will be valuable for future development and has the potential to speed development by streamlining how modules interact.

+ Code Demonstration

- Code was compiled and ran from a command prompt.
- Output demonstrated:
  - how a succession or disturbance component would iterate through all the active sites on the landscape, and
  - how an output component would iterate through all the sites (active and inactive) on the landscape, in order to generate an output map.
- Format of output logs is now controlled by a configuration file.

+ Iteration 2 Hits and Misses

- Miss Use Cases not 50% complete.

- Miss Did not test two components talking to the landscape.
- Miss Binary compatibility issues between differing C++ compilers not anticipated. These issues could potentially create problems for collaborating developers.
- Hit Tasks left over from Iteration 1 completed.
- Hit Subversion and two clients (Subversion interfaces) installed.
- Hit Extracted DLL code from CppUnit (an existing open-source project). Implemented simple framework and plug-in module using this DLL code.
- Big Hit Set up CppUnit, a testing framework, to perform automated testing of our code. Tests are written for each piece (unit) of our code. Not only helps determine when a unit has been completed properly, but also detects if a change "breaks" a working unit.
- Hit Developed some classes that encapsulated the dense code needed to address some of the technical issues with using C++ and DLLs. These classes would hide messy details from component developers, simplifying their jobs.

+ Iteration 2 Lessons Learned

- Uncovered the issues with DLLs (aka Plug & Play). This issue has forced us to weigh the usability (ease-of-use), future flexibility, and the schedule. The iteration assessment allowed us to narrow the decision space so we can move forward as quickly as possible.
- Iteration assessments are a valuable tool for communicating our progress, based on project team feedback.
- We can begin publishing interface specifications before the actual interface classes are finalized or even before the language is chosen (assuming that it will be in the C family of languages). This will speed collaborations with module developers.

Iteration 3

by Robert Scheller — last modified Dec 30, 2004 05:18 PM

- Phase: Elaboration
- Dates: Oct 1 - Nov 1, 2004

Tasks to be Completed

+ Documentation

- Document Interface, including classes and functions. Post documentation on website in Developer folder.

+ Architecture

- Interface Implementation The goal is to evaluate the feasibility of using languages that compile to the Common Language Runtime (CLR). Examples include Managed C++, C#, Java.
  - Convert Interface classes to C#.
  - Test Mono compiler and/or VS C# compiler on Interface classes.
  - Demonstrate dynamic libs using two Landis components (actual or test) that will read and write data to the landscape.
- Finish Iteration 2
  - Demonstrate dynamic libs using two Landis components (actual or test) that will read and write data to the landscape.

+ Tools

- Download MS Visual C# beta with generics.

+ Unified Process

- Continue writing Use Cases Have Use Cases 50% completed by end of iteration 3.

Assessment

+ Overview

In summary, iteration 3 evaluated the risks and benefits of moving L-II to C#. The merits of such a change were quickly demonstrated and it was decided on Oct.19 to fully migrate to C#. The transition to C# is >90% complete and has been very successful.

+ Code Demonstration

The new C# code was reviewed and demonstrated. Overall, the code is much cleaner than C++ due to the ease of implementing plug-in components and the automatic memory management (no pointers or dereferencing necessary). There is new syntax that will have to be learned.

+ Iteration 3 Hits and Misses

Hit All existing C++ source code converted to C#, except the Event Scheduler. The generics feature in C# 2.0 made it easy to convert the current design of the landscape module with its ability to add new site variables on-the-fly.

Big Hit Demonstrated three plug-in modules working together, adding, modifying and sharing data.

Hit Downloaded and installed NUnit and the SharpDevelop IDE (Integrated Development Environment). NUnit is a testing framework for the .NET framework; it's the C# counterpart to the CppUnit testing framework that we had been using.

SharpDevelop is an open source IDE that will allow 3rd parties to develop L-II plug-in modules with C# without having to purchase of Visual Studio.
Hit/Miss Visual Studio C# 2.0 beta installed. Open-source Mono C# compiler with the generics feature could not be setup for Windows OS.
Hit/Miss Documentation. Progress has been made with creating html documentation using NDoc. We also have put together quite a bit of other material (Architecture docs, sample code, on-line resources) for our Developers page. However, some it is needs more polish and the html docs are not completed.
Miss We still have not finished the Use Cases.

+ Iteration 3 Lessons Learned

The migration to C# has both positives and negatives. C# will speed the development of the model architecture. The implementation of plug-in modules is much easier (on a scale of 1-5, a 4.5 for significantly easier to implement). Also, no memory management is required. Overall, the C# technology is much better and there are no obvious faults with it. The negatives are relatively minor. There is always a learning curve when using a new language and there are some new key words in C#.
Finally, we are on the cutting-edge of this technology. The compiler and many of the tools are very new or still in beta. The new technology will provide many benefits but there will be hiccups in the process.

Iteration 4

by Robert Scheller — last modified Dec 30, 2004 05:18 PM

This iteration will focus on architecture and documentation.

- Phase: Elaboration.
- Dates: Nov 6 - Dec. 1, 2004

Tasks to be Completed

+ Documentation

- A goal for this iteration is to be able to share documentation with external developers.
- Finish HTML documentation for existing classes, and publish it in the developers section of our web site.
- Finish architecture overview document, and publish it in the developers section.
- Create and publish a template for the source code for a generic disturbance module. This template will serve as an aid for developers writing plug-in modules.

+ Architecture

- Finish writing the code for the wind module. This task was started to provide insight into 1) the documentation needed by external developers, and 2) the core parts of the architecture that need further analysis & design (such as Ecoregions and Species).
- Design & document the interfaces for the architectural elements that are identified in completing the wind module. These include LandscapeRegions, Ecoregions, Species, and TextFileReader (names are subject to change).
- Continue working w/Barry DeZonia on integrating GeoSpatial Data Abstraction Library (GDAL) into L-II. GDAL is an open-source C++ library for working with raster data.

Assessment

+ Overview

Although notable progress was made during iteration 4, we fell short on some objectives because of unforeseen circumstances: limitations with software to generate documentation for our source code, challenges with combining C# generics and unmanaged code (i.e., code that uses memory pointers; needed to use with GDAL), and Jimm's illness during the last week of the iteration.

+ Iteration 4 Hits and Misses

Miss NDoc is the top software tool for generating various forms of programmer documentation from source code for the .NET platform. However, the current version of NDoc does not support latest features of C# such as generics (due to license restrictions with current beta version of C# 2.0 compiler). Jimm began modifying NDoc's code (it's open source) during the planning of this iteration. He originally estimated a few days to update NDoc, but during the 1st week of the iteration, after examining the code in more details, he revised the estimate to 1 to 2 weeks. In lieu of this, we decided it was best to focus on the other forms of

documentation: the architecture overview and the template for a generic disturbance module.
Hit Finished the first pass at the architecture overview document. It's now available in the Developers section.
Hit Finished 70-80% of the code for the wind module. This work helped determine how disturbance components will handle extra parameters for ecoregions and for species. Revised the existing interfaces to some of the core modules (e.g., landscape, raster I/O), and started the interfaces for others (e.g., logging, random number generation).
Hit Have been concurrently updating the description document for the wind module while working on the module's source code.
Miss Because the wind module's code hasn't been completed, the source-code template for a generic disturbance module hasn't been started.
Miss The interface to raster I/O module uses C# generics. Using GDAL requires the use of memory pointers. Discovered that combining C# generics with memory pointers is challenging because memory pointers must be used in code blocks that are flagged as "unmanaged" (i.e., the system's memory management is disabled). This design problem stretched the limited C# knowledge of Jimm and Barry, who are both new to C#.
Hit/Miss Jimm and Barry devised a brute force approach to combine C# generics and memory pointers in order to implement the raster I/O interface with GDAL. Barry has got an initial implementation of this approach running. Although it works, it's damn ugly in that there are over half a dozen copies of functions with very similiar code: the very situation which generics were designed to address.
Hit Just developed a better design to utilize C# generics and memory pointers more efficiently, to remove the loads of redundant code in the brute force method.

+ Iteration 4 Lessons Learned

C# generics are a powerful feature, but they differ from C++ templates in a number of ways. These differences mean it's going to take some time to become as proficient at designing with them as we currently are with C++ templates.
With the architecture overview document now available, the only remaining artifacts for the Elaboration phase are the Use Cases. If we can make substantial progress on them in the near term, we can move into the 3rd phase of the Unified Process: the Construction Phase.

Iteration 5

by Robert Scheller — last modified Jan 03, 2005 11:53 PM

This iteration is the first Construction iteration; therefore, it will focus more on code production and less on architecture and documentation.

- Phase: Construction
- Dates: Dec. 4 - Dec. 22, 2004

Tasks to be Completed

+ Core Modules

- Finish designing & documenting the interface for the Species module, and then write the code for it.
- Design and code the supporting utility modules that read input values from text files.
- Continue working w/Barry on integrating GeoSpatial Data Abstraction Library (GDAL) into L-II to implement the Raster I/O module.

+ Extension Components

- Finish writing the code for the wind module. The wind module is approximately 70% complete.
- Begin designing and documenting the Age-only and Biomass succession components.
- Begin designing the output extensions (Barry DeZonia).

+ Documentation

- Continue to produce documentation as necessary and update existing documents. In particular, work will continue on the Use Cases, which are now 60% completed. Feedback will be requested from Team Members.
- Add extensive commenting to the wind module. We have decided to publish a well-documented wind module on the Developers tab in lieu of a generic disturbance module.

Assessment

Overview: Iteration 5 was a lot of heavy coding, particularly the wind extension and the utility modules for text input. Some documentation was produced, namely a major revision of the use cases.

+ Hits and Misses:

Hit/Miss The design of Species Module is complete, but the code is not finished. The design was more complex than anticipated because of the need for two different interfaces for the module (see the section **Lessons Learned** below).

Hit/Miss The utility modules for input text files are only 90% completed. The major reason for not completing this code was the addition of unit tests. The original C++ code for the modules had no test code; as it was translated to C#, unit tests were created to ensure the new code was functioning correctly.

Hit/Miss A C# module that wraps the GDAL DLL was designed and coded; testing is currently underway. The next step in integrating GDAL into L-II is to use this C# interface to GDAL to implement the L-II Raster I/O module. However, progress on

this step was stalled because Barry had problems with the C# programming tools (the tools were crashing).
Hit The wind extension coding has been completed.
Hit How extensions add site variables to the landscape has been significantly simplified. Nicer cleaner code!
Miss Wind extension is not commented enough for publication.
Hit The design of the Age-Only Succession extension was started. All the docs are completed. Given our experience with the wind extension, we've decided to complete the code for the Age-Only Succession extension before starting work on the Biomass Succession extension.
Hit/Miss The design of output modules was started but not completed because some of Barry's time was spent on the problems with the C# programming tools.
Hit A diagram of different levels of system scope was developed and added to the use-case document. This diagram initiated a major revision of the use cases themselves to better represent how users will interact with the Landis-II software independent of the type of user interface (console vs GUI).

+ Lessons Learned:

- C# does not have explicit memory management as does C++. Therefore, we do not need to adjust our designs to improve memory usage; we are dependent on .NET framework's memory management. Given the large memory requirements of Landis-II, it'll be very beneficial to measure the actual resource usage of our working code to see how efficiently our needs are being met by the .NET framework.
- Work on the wind extension revealed that we may need to track individual data per inactive site at some future time. In other words, a developer may want to store temporary values for inactive sites; the most likely example is the spatial extent for a disturbance event. This option for site variables has been designed, but is not currently implemented.
- Work on the Species module and the wind extension revealed that a group of parameters has two interfaces. The first, most obvious, interface is how an extension uses the parameter group (for example, how the wind extension uses species parameters). The second interface includes code to validate parameter values entered by the user. If this second interface is well-designed, it can be shared by the code that reads parameters from a text file, and by the GUI. Such a design will allow more rapid implementation of the GUI.

Iteration 6

by Robert Scheller — last modified Jan 25, 2005 05:01 PM

This iteration will focus on completing the core modules from the previous iteration (Species and text input utilities) and constructing the Age-Only Succession extension.

- Phase: Construction
- Dates: January 3 - January 19, 2005

Tasks to be Completed

+ Core Modules

- Finish the documentation of the interface for the Species module, and then write its code.
- Finish the code for the supporting utility modules that read input values from text files.
- Continue working w/Barry on integrating GeoSpatial Data Abstraction Library (GDAL) into L-II to implement the Raster I/O module.

+ Extension Components

- Finish the design of the Age-only Succession component, and then write its code.
- Continue working on the design of the output extensions (Barry DeZonia).

+ Documentation

- Add extensive commenting to the wind module.

Assessment

+ Summary

In summary, we are rapidly approaching our first internal beta-release with this iteration. The only modules in the core framework left to be done are ecoregions, raster I/O, and logging; also some fine-tuning of existing code is required.

+ Hits and Misses

Hit The Species module is >95% completed, including over three dozen tests. Only minor changes to the columns are required (add a couple new ones and remove an obsolete one).

Hit The code for the supporting utility modules that read input values from text files is completed. Includes extensive test code (over 200 tests).

Hit/Miss The integration of GDAL into L-II is still not complete because of difficulties bridging the GDAL C++ core and C#. Barry has almost all of GDAL available from C#; only 4 or 5 functions remain to be bridged. However, these functions require the passing of data structures between the managed-memory environment of C#/NET framework and the unmanaged memory paradigm of C++. Barry and Jimm are still learning to master the details needed to implement this type of function calls across the managed/unmanaged boundary.

Barry estimates that he can quickly implement a library in 1-2 weeks that will read and write 8-bit and 16-bit ERDAS GIS files. So, for expediency, the GDAL integration has been postponed. Because the raster module's interface hides the implementation details from the rest of the Landis-II core framework and the

extensions, the GDAL implementation can be completed at a later date, and then added to the framework without modifying any core modules or extensions.
Hit/Miss Age-only succession was partially completed because Jimm wasn't certain which functionality belonged in this extension versus in the core framework. We had a design meeting during his visit to Madision, and resolved the issues: the cohort interfaces will move into the core, while reproduction will be implemented in the succession extensions.
Miss No comments were added to the Wind module because its code first needs to be updated to comply with the recent changes to the interfaces of the core modules.
Miss The design of output modules was not advanced because of the issues with GDAL as well as some priority system-administrative tasks that require Barry's time. Because Barry's schedule is full with system-administration, the custom raster library and IAN, the output modules have been reassigned to Jimm.

+ Lessons Learned.

- Again, we are working on the cutting edge and the GDAL effort spilled some blood, so to speak. However, this work is not a wasted effort because we're building Landis-II with well-defined interfaces among its components. This approach allows us to set aside the GDAL work while we focus on getting the first beta running. Then we can complete the GDAL work and integrate it smoothly later on.

Iteration 7

by Robert Scheller — last modified Feb 18, 2005 03:23 PM

This iteration will focus on building the core modules for ecoregions and cohorts, and implementing reproduction (seeding, resprouting) in the Age-Only Succession extension.

- Phase: Construction
- Dates: January 24 - February 16, 2005

Tasks to be Completed

+ Core Modules

- Design the interface for the Ecoregions module, and then write the code to implement the module.
- Revise the Species module to add two parameters (minimum resprouting age and serotiny) and to remove an obsolete parameter.
- Design the interface that disturbance extensions will use to specify a reproductive method (seeding, resprouting, planting) at disturbed sites.
- Design and then write the code for a new core module which will contain various interfaces to cohorts (basic cohort with age, cohort with biomass data).
- Revise the Raster I/O module to simplify its interfaces and their implementation, replacing random-access I/O with sequential-access I/O.
- Assist Barry as needed, on an in-house custom implementation of the interfaces in the Raster I/O module.

+ Extension Components

- Write the code to implement the reproductive methods of seeding and resprouting for the Age-only Succession component.
- Write the code to load the parameters for the Age-only Succession component from its input file (includes species establishment coefficients).
- Update the Age-only Wind component to use the newer interfaces to the core modules.

+ Documentation

- Update the use-case documentation with revisions from the last review meeting.
- Update the wind documentation with revisions from the last review meeting.

Assessment

+ Summary

The iteration did not accomplish as much as hoped due to some unexpected contingencies: a new build method, finishing work on the raster IO, and vacation time. Nevertheless, many critical pieces were put into place and an internal release is still expected at the end of the new iteration.

+ Hits and Misses

Miss Radically underestimated the time required to implement the NAnt build method. This build method streamlines the process of compiling executable code.
Miss Underestimated the time requried to finish the raster I/O library. Also, Barry still needs to implement ERDAS 8/16 bit read and write.
Hit Ecoregions module completed, including input text file.

Hit Species module completed, including input text file.
Hit/Miss Reproduction Interface still needs some work, although significant progress was made.
Hit/Miss Reproduction partially implemented. However, serotiny and planting needs to be added to seeding and sprouting.
Hit Age-only cohort module completed.
Miss Seed dispersal algorithm. The data structure for neighborhoods not done. Default seed dispersal (the BC Ward method) not implemented.
Miss Age-only succession input text file not completed.
Miss Wind module still not updated with appropriate interfaces, due to continued interface flux.
Hit Wind documentation updated.

+ Lessons Learned

We need to better track what the contingencies will be and allocate time for them. This will help the project leader to assess the relative merits of various tasks, given our priorities.

Iteration 8

by Robert Scheller — last modified May 13, 2005 06:28 PM

This iteration will focus on completing the first internal release of Landis-II. This alpha-1 release will just have age-only succession.

- Phase: Construction
- Dates: February 21 - March 7, 2005

Tasks to be Completed

Core Framework

- Landscape module (0.75 day)
  - Add method to Site class to fetch a site's neighbor based on its relative position.
  - Define a record structure to represent relative site locations.
  - Define interface for input data grids with sequential data access.
  - Modify the constructor for Landscape to use new interface to sequential input data grids.
- Ecoregions module (0.25 day)
  - Define class to represent pixels in ecoregions map.
- Main model module (1 - 1.5 days)
  - Write code to initialize landscape from ecoregions map.
  - Write code to create and initialize Ecoregion site variable.
  - Write code to load dataset of species parameters.
  - Define a public variable (i.e., property) by which other modules and extensions access the species dataset.
  - Write parser for scenario file.
- Plug-in module (0.25 day)
  - Add Lookup method to Manager class: it will be used by scenario file's parser to check plug-in names. Although eventually some sort of database file of installed plug-ins will need to be implemented, initially, the names of known plug-ins will be hard-wired for expediency.
- Raster-IO module
  - Landis.Raster -- Add interfaces & classes for raster metadata. (0.5 - 1 day)
  - Landis.Raster.Erdas74 -- Basic implementation of raster interfaces for Erdas v7.4 file formats (GIS, LAN) by Barry.
- Cohorts module (0.5 - 1 day)
  - Design an approach that ensures the current site being processed by succession and disturbance plug-ins is available to this module.
- Succession module
  - Initial site communities (2 days)
    - Write classes to read community definitions from text file: SiteCommunity, EditableCommunity, EditableDataset, Dataset, DatasetParser.
    - Write function to initialize site cohorts using initial-site-community map.
  - Reproduction
    - Seeding (0.75 - 1.25 days)

        - Write Neighborhood class to represent the set of surrounding sites that may seed a site.
        - Write the default seed dispersal method (based on BC Ward's work)
- Utility module (0.5 - 1 day)
    - Add random number generation (uniform and exponential distributions).
    - Locate C# code for Mersenne Twister, a widely-used RNG that is fast and robust.

User Interface

- Console Interface (0.5 - 1 day)
    - Setup new project for landis-ii-console.exe using new build software.
    - Write code that takes path of scenario file from command-line arguments, and passes it to model's main module.

Extension Components

- Age-only Succession (0.5 - 1 day)
    - Update the project to use the new build software.
    - Revise the extension's main source file to use the core succession module in the framework.
    - Write classes to read parameter file: Parameters, EditableParameters, InitFileParser.
- Output - Basic species data (0.5 - 1 day)
    - Create a very simple output plug-in component that outputs maps at specific timesteps. One map per species: the maximum age for the species at each site.

Assessment

Completed March 9.

+ Summary

Thanks to Jimm's tremendous efforts, all of the scheduled tasks (above) were completed (every task was a Hit). This was by far our most successful iteration to date. Therefore, the Alpha 1 (A1) release is completed - the first fully operational version of the model. There are some clean-up operations necessary before the beta release (adding some additional inputs, creating a beta User's guide, repackaging some of the code). However, the A1 release demonstrates all of the model capabilities and potential. I cannot begin to describe how exciting it was to see it all come together. The input files are incredibly easy to use with amazing levels of validation. Best of all, the method for telling the model which modules to use is seamlessly integrated into the input - it will be incredibly easy for Users to download and use many different modules and there are no restrictions on the number of disturbance and output modules that can be used simultaneously. Because so many pieces came together this iteration, we anticipate a beta release (to be shared externally) at iteration 10 and a public release at iteration 11 or 12 (depending on how iteration 10 goes). Please congratulate Jimm on a job well done!

Iteration 9

by Robert Scheller — last modified May 13, 2005 06:28 PM

This iteration will focus on comparing the performance of Landis-II v1.0 alpha-1 release with Landis 3.7, as well as preparing for the team meeting on April 1.

- Phase: Construction
- Dates: March 14 - April 1, 2005

Tasks to be Completed

Core Framework

- Main module
  - Add CellLength parameter to scenario file.
- Raster-IO module
  - Complete transition from v1 of interfaces (single library) to v2 (multiple drivers), for example, remove ILibrary interface.
  - Design and code the Manager class to load and manage various raster drivers.
  - Modify other modules in core and existing plug-ins to use the new Manager class.
  - Develop naming conventions for metadata so they are portable among drivers.

Raster Drivers

- Erdas74
  - Update the code to work with v2 of raster-IO interfaces (Barry).

Plug-in Components

- Age-only Succession
  - Add new parameter to initialization file that allows user to select the seeding algorithm (no dispersal, universal dispersal, Ward seed dispersal).
- Max Species Age (time permitting)
  - Add new parameter to initialization file to allow user to select which species have maps generated for them (currently all species do).
  - Add new parameter to allow user to specify a template for the filenames for maps (currently, hard-wired in).

Documentation

- Requirements
  - Update use-cases documentation based on last review meeting.
  - Integrate use-case documentation into requirements document.
  - Distribute requirements document to team members for review.

Miscellaneous

- Deployment
  - Package up alpha-1 release (core and plug-ins) for Rob.
- Performance Testing
  - Setup age-only succession run for Landis 3.7 on our model server.
  - Collect timing data for this run.

    - Write script utilites to convert some of Landis 3.7 input files into Landis-II input files.
    - Setup age-only succession for for Landis-II on our model server.
    - Run and collect timing information for this run.
- Team Meeting (April 1)
    - Prepare presentations (scientific and technical perspectives).
    - Investigate time to do a proof-of-concept for GUI window form with a text-editing area using a parser for one of the input files (e.g., scenario, species, ecoregions).

Assessment

+ Summary

The emphasis of this iteration was preparation for the team meeting and performance testing. The team meeting was regarded as an overall success. Performance testing was less successful. During the initial testing, a problem with using the memory management features of C# was encountered. In short, the garbage collector was running too frequently. The issue is imminently fixable but will require more time than was available before the team meeting. Alterations to the landscape module that will address the memory management issue and allow us to finish performance testing is the first priority for iteration 10.

+ Hits and Misses

- Miss Cell length not added to scenario input file.
- Miss The Manager class for Raster I/O not completed as planned.
- Hit Erdas 74 modified to work with v2 of Raster IO interfaces.
- Miss Age-only succession not modified to allow user selection of seeding algorithm.
- Hit Output: Max_Spp_Age completed, including species selection and naming template.
- Big Hit All the planned documentation was completed and distributed to team members.
- Hit L-II packaged for distribution, including an installer.
- Hit/Miss Performance testing nearly completed. However, memory management issues (above) prevented final completion.
- Big Hit Team meeting. The team meeting met all of our objectives.

+ Lessons Learned

The memory management issue was an unfortunate surprise. Automated memory management may be a great tool for eliminating unused objects, but the performance downside is poorly advertised. As more people use C#, these issues will likely be addressed more honestly in the documentation. Nevertheless, our lost time (~1 week) is more an irritation than a barrier.

Iteration 10

by Robert Scheller — last modified May 13, 2005 06:28 PM

This iteration will address the issues discovered during the performance testing in the previous iteration as well as the new features discussed at the team meeting. It will also include work to complete the functionality of the core framework and the succession library.

- Phase: Construction
- Dates: April 4 - 20, 2005

Tasks to be Completed

Core Framework

- Landscape module
  - Revise the site-related classes to minimize the use of temporary objects, in order to address the high garbage-collection rate which is adversely affecting performance.
- Utility module
  - Modify how relative file paths in an input file are interpreted. Currently they are interpreted relative to the directory where the model is running. However users intuitively assume that relative paths are interpreted relative to the directory containing the input file. Example: if a scenario file has the path "species.txt", users assume that this species input file is in the same directory where the scenario file is located. Currently, the model looks for the species input file in the directory where it's executing.
- Raster-IO
  - Complete transition from v1 of interfaces (single library) to v2 (multiple drivers), for example, remove ILibrary interface, switch to v2 of Erdas 7.4 driver.
- Ecoregions
  - Allow the use of 16-bit maps.
- Main module
  - Add optional parameter to scenario file which allows user to specify a starting timestep other than 1 (e.g., the year 1990).
- Plug-ins module
  - Modify the interfaces to all types of plug-ins so that the initialization method includes the starting timestep.

Plug-in Components

- Succession library
  - Allow the use of 16-bit maps for initial communities.
  - Modify the handling of the definitions of the initial communities so that ages listed for a particular species are binned into cohorts based on the succession timestep. For example, if a community has the ages 10, 15, 20, 30, and 35 for the species pinubank, and the succession timestep is 20 years, then the ages 10, 15 and 20 mean the cohort 1-20 is present, and the ages 30 and 35 mean the cohort 21-40 is present.
- Age-only Succession (time permitting)

- Add new parameter to initialization file that allows user to select the seeding algorithm (no dispersal, universal dispersal, Ward seed dispersal).

Documentation

- User Guide
  - Write the chapter that describes the format and contents of the input files associated with the core (e.g., scenario file, species file, initial communities). (Rob)
- Web site
  - Add a page that describes the project's priorities and relative timeline for various components (core, plug-ins, GUI). Sort of a "roadmap". (Rob)
- Plug-ins
  - Reorganize the conceptual document for the wind disturbance (Landis 3.7) into a new structure which will be basis for its user guide. The conceptual description will be a chapter in this guide, along with descriptions of its data files and a list of references.

Assessment

Summary

Iteration 10 was almost entirely devoted to optimization to achieve desired program performance. Our software requirements call for a run time of Landis-II that is no more than 25% (1.25x) longer than the run time of Landis 3.7. But at the beginning of the iteration, Landis-II's run time was more than 200x!

This enormous difference was largely due to a lack of understanding how certain C# language features and code designs impact memory management in the NET Framework. The Landis-II code was designed and written to utilize these powerful C# language features; however, this resulted in a significant burden on the memory management system (very high rates of garbage collection) which only became apparent with the large landscape used for performance comparison. The comparison was conducted using the BWCA landscape, which has 4.2 million cells (of which 1.8 million are active).

By the end of the iteration, performance tuning had reduced Landis-II's run time down to ~0.95x that of Landis 3.7!! Unfortunately, the tuning process took approximately a week and half longer than estimated.

Hits and Misses

Big Hit The number of temporary objects was significantly reduced and thus, garbage collection was reduced. This created the greatest gain in model performance. Other optimization changes also contributed to large gains in performance.

Miss Utility module not updated to recognize relative paths.

Hit Raster-IO updated to use v2 driver interfaces.

Miss 16-bit maps not yet allowed for ecoregions or initial communities.

Miss An alternate starting time step (e.g., 1990) not implemented.

Miss Initial community cohort binning not implemented.

Miss Seed dispersal parameter not implemented.

Hit User's Guide beta completed.

Miss Timeline with estimated benchmarks and priorities not created.
Hit Started the user guide for wind disturbances by reorganizing the structure of the associated conceptual document.

Iteration 11

by Robert Scheller — last modified May 25, 2005 02:14 PM

This iteration is the first one in the Transition phase; therefore, it will focus on the first beta release.

- Phase: Transition
- Dates: April 25 - May 13, 2005

Tasks to be Completed

Core Framework

- Utility module
  - Modify how relative file paths in an input file are interpreted. Currently they are interpreted relative to the directory where the model is running. However users intuitively assume that relative paths are interpreted relative to the directory containing the input file. Example: if a scenario file has the path "species.txt", users assume that this species input file is in the same directory where the scenario file is located. Currently, the model looks for the species input file in the directory where it's executing. [Medium priority]
- Ecoregions
  - Allow the use of 16-bit maps. [High priority]
- Main module
  - Add CellLength parameter to scenario file. This parameter value will be used if the ecoregions map is in a raster format that does not provide cell length as metadata. [High priority]
  - Add optional parameter to scenario file which allows user to specify a starting timestep other than 1 (e.g., the year 1990). [Low priority]
- Plug-ins module
  - Modify the interfaces to all types of plug-ins so that the initialization method includes the starting timestep. [Low priority]

Plug-in Components

- Succession library
  - Allow the use of 16-bit maps for initial communities. [High priority]
  - Modify the handling of the definitions of the initial communities so that ages listed for a particular species are binned into cohorts based on the succession timestep. For example, if a community has the ages 10, 15, 20, 30, and 35 for the species pinubank, and the succession timestep is 20 years, then the ages 10, 15 and 20 mean the cohort 1-20 is present, and the ages 30 and 35 mean the cohort 21-40 is present. [High priority]
- Max Species Age
  - Modify the plug-in so that it creates maps with the maximum age among all species. The maps are created every output timestep, and use "all" for the species variable in the template for map names. [High priority]
  - Create a draft user guide. [High priority]
- Wind
  - Update the plug-in to work with the current interfaces in the core framework. [High priority]

    - Complete the chapter on the input parameter file in the plug-in's user guide started during the last iteration. [High priority]
- Age-only Succession
    - Create a draft user guide. [High priority]
    - Add new parameter to initialization file that allows user to select the seeding algorithm: no dispersal, universal dispersal, Ward seed dispersal. [Medium priority]

Documentation

- User Guide
    - Expand the current draft of the document, filling in enough chapters so it is adequate for beta testers. (Rob) [High priority]

Deployment

- Beta 1 Release
    - Bundle all the necessary components into an installer program. Components include core framework, console interface, plug-ins (age-only succession, max species age, wind), various documentation files, and sample input files. [High priority]

Web Site

- Add a page that describes the project's priorities and relative timeline for various components (core, plug-ins, GUI). Sort of a "roadmap". (Rob)

Assessment

Summary

Iteration 11 was devoted to producing the beta-release. Although we were ultimately successful, we found it necessary to trim some features. This will also be true for the final release. We regret having to postpone some minor features to future versions (see below); however, we have ascertained that it is more important to meet our deadlines. In particular, we realized the need to finish v1.0 so that work can begin on the necessary extensions for the LANDFIRE project. Likewise, the timely release of the beta version will be tangible evidence of progress for people attending meetings over the summer.

Hits and Misses

Big Hit The beta release was finished and sent out to three beta-users. The beta release includes the core, age-only succession, base wind, and a maximum-age output extension.

Hit Modified the handling of the definitions of the initial communities so that ages are binned into cohorts based on the succession timestep.

Hit Modified the Maximum-age output extension so that it creates maps with the maximum age among all species.

Miss Add new parameter to initialization file that allows user to select the seeding algorithm.

Hit 16-bit maps are now accepted for Ecoregion maps.

Miss InitialCommunities were not modified to allow 16-bit maps.

Hit Add CellLength parameter to scenario file.

Big Hit All of the necessary User's manuals for the beta release were finished and posted on the web site. The Landis-II Model Description was also completed and posted.
Postponed Features These features have been postponed for future versions, and so will not be included in either the beta or final release.
Miss Utility module will recognize relative paths. Although this would have allowed for greater flexibility when using the command line, it is not necessary.
Miss Allow User's to select alternative starting year. This feature was purely a bells-and-whistles option.

Iteration 12

by Robert Scheller — last modified Jul 01, 2005 05:36 PM

This iteration will focus on the first candidate for the official release. The goal of the release candidate 1 (RC1) is to complete the implementation of all the features of the official release, so the next (and final) iteration can focus on preparing the official release for distribution.

- Phase: Transition
- Dates: May 25 - June 15, 2005

Tasks to be Completed

Core Framework

- Main module
  - Add RandomNumSeed parameter to scenario file. This parameter will allow the user to reproduce specific program behavior which is necessary for testing and debugging purposes.
- Plug-ins module
  - Add a new cleanup method to the interfaces for all types of plug-ins. This method will allow the core to shutdown a plug-in properly so it can release critical system resources (for example, close a log file).
- Cohorts module
  - Extract the age-only interfaces and classes from the core framework into a separate libary.
- Ecoregions
  - Change the ecoregion maps to use 16-bit unsigned pixel data.
- Species
  - Allow -1 for the effective-seeding-distance parameter. This special value will designate for the Ward seed-dispersal algorithm that the species is universally present on the landscape.

Plug-in Components

- Succession library
  - Allow the use of 16-bit maps for initial communities.
  - Re-evaluate the performance vs parameter-checking issue between the 2 implementations of Brendan Ward's dispersal algorithm currently in the library. This will determine which implementation should be removed from the library.
  - Modify the Ward dispersal algorithm so that -1 for a species' effective-seeding distance results in the species being considered universally present on the landscape. Therefore, no searching for sites is done.
- Age-only Succession
  - Add new parameter to initialization file that allows user to select the seeding algorithm: no dispersal, universal dispersal, Ward seed dispersal.
- Fire
  - Design and write the code for this disturbance component. (Rob)
  - Create a user guide. (Rob)
- Reclass

    - Design and write the code for this output component. (Rob)
    - Create a user guide. (Rob)
- Administration
    - Design and implement the process for installing a plug-in. This process must update the local "database" file of installed plug-ins on a user's machine, and must update the local web page of user documentation to include the new plug-in's user guide.
    - Design and implement the means by which a user can view a list of plug-ins installed on her machine. This list must include the names for the plug-ins that are used in scenario files.

Raster Drivers

- Erdas 7.4
    - Modify the driver to allow both unsigned & signed pixel data. The Erdas documentation is ambiguous about the signed issue, so for greater flexibility, the client code that's calling the driver will inform the driver to treat the pixel data as either unsigned or signed, based on the client's needs.

Documentation

- Add version #'s to existing user guides. (Rob)
- Develop a common organizational structure for user guides to improve readability. Update existing user guides accordingly.

Deployment

- Release Candidate 1
    - Research the security issues with not signing our assemblies. Signing is needed for certain internal tests to work, and yet we want to avoid having to rebuild plug-ins if only internal changes are made to the core modules.

Assessment

Summary

Iteration 12 was larger than we anticipated - there were many minor details that needed to be finished - although we did complete Release Candidate 1 on time. One of the largest obstacles was the creation of the fire and reclass output extensions. Completing Release Candidate 1 was an incredibly useful exercise - it forced us to identify and address many many details and reexamine our priorities. Completing the fire extension proved valuable for catching some earlier errors in the wind extension. Finally, the complete package has enough value that it is already attracting users.

Lessons Learned

Having better Developer's documentation would certainly ease the transition for internal and external developers.

Hits and Misses

Hit Added RandomNumSeed parameter to scenario file.
Hit Added a new cleanup method to the interfaces for all types of plug-ins.
Hit Extracted the age-only interfaces and classes from the core framework into a separate libary.

Hit Changed the ecoregion maps to use 16-bit unsigned pixel data.
Miss Allow -1 for the effective-seeding-distance parameter.
Hit Allow the use of 16-bit maps for initial communities.
Hit Evaluated the performance vs parameter-checking issue between the 2 implementations of Brendan Ward's seed dispersal algorithm in the succession library. The faster implementation does not validate site locations. When validation was added to this implementation, the run time for the BCWA scenario used for performance testing increased by 1.1% from 49:19 (mins:secs) to 49:52. This slight performance hit is acceptable, so this modified implementation with location validation was retained as the sole implementation in the library.
Miss Modify the Ward dispersal algorithm so that -1 for a species' effective-seeding distance results in the species being considered universally present on the landscape.
Hit Add new parameter to initialization file that allows user to select the seeding algorithm: no dispersal, universal dispersal, Ward seed dispersal.
Hit Designed and wrote the code for the Base Fire extension.
Hit Created a User Guide for the Base Fire extension.
Hit Designed and wrote the code for the reclass output extension.
Hit Created a User Guide for the reclass output extension.
Miss Design and implement the process for installing a plug-in.
Miss Design and implement the means by which a user can view a list of plug-ins installed on her machine.
Hit Modified the ERDAS 7.4 driver to allow both unsigned & signed pixel data.
Hit Added version #'s to existing user guides.
Hit/Miss Did some research about the security issues with not signing our assemblies, but not enough to make a decision regarding the matter.

Iteration 13

by Robert Scheller — last modified Aug 19, 2005 05:54 PM

This iteration will have two releases: the 2nd release candidate (RC2) and the official release. The goal of RC2 is to address the few issues that have surfaced as people have begun using RC1. RC2 will be completed quickly to replace RC1. Afterwards, efforts will focus on preparing the official release, and starting the Biomass Succession extension.

- Phase: Transition
- Dates: June 15 - August 5, 2005

Release Candidate 2 (RC2)

Extensions

- Reclass
  - Add validation to ensure that reclass coefficients are between 0 and 1.
- Base Wind
  - Add the wind direction enhancement that Rob developed to the spread algorithm.
- Base Fire
  - Add the wind direction enhancement that Rob developed to the spread algorithm.
  - Fix the bug that Rob identified in the neighbors foreach loop of the spread algorithm.

Raster Drivers

- Erdas 7.4
  - Optimize driver to significantly increase performance. Currently, maps with > 1,000,000 cells take ~10 minutes to write. Needs to be < 0.5 minutes.

Official Release

Core Framework

- Species
  - Allow -1 for the effective-seeding-distance parameter. This special value will designate for the Ward seed-dispersal algorithm that the species is universally present on the landscape.

Web Site

- Users Page
  - Create templates for each extension. Arrange templates into a table such that Users can quickly identify and download extensions.
- Web Site Migration
  - Migrate the LANDIS-II web site from Jimm's workstation to a new machine. Relocating the site will avoid Jimm's development activities from impacting the web site's performance.

Extensions

- Administration
  - Design and implement the process for installing a plug-in. This process must update the local "database" file of installed plug-ins on a user's machine, and

must update the local web page of user documentation to include the new plug-in's user guide.
    - Design and implement the means by which a user can view a list of plug-ins installed on her machine. This list must include the names for the plug-ins that are used in scenario files.

Deployment

- Continue researching the security issues with not signing our assemblies.

Other Tasks

Extensions

- Biomass Succession
    - Write the Users Guide.
    - Begin designing and coding the extension.

Assessment

Iteration Summary and Lessons Learned

The model is now available to all registered Users on the web site. Perhaps our greatest lessons will now come from any User feedback that we receive. Otherwise, the final iteration was relatively smooth. The next challenge will be in prioritizing additional features and formulating a careful plan for upgrading the core.

Hits and Misses

- Hit Added validation to ensure that reclass coefficients are between 0 and 1.
- Hit Added the wind direction enhancement that Rob developed to the wind shape algorithm.
- Hit Added the wind direction enhancement that Rob developed to the fire spread algorithm.
- Hit Fixed the bug that Rob identified in the neighbors foreach loop of the fire spread algorithm.
- Hit Corrected reclassification error identified by Arnie Rudy.
- Hit Optimized Erdas driver to significantly increase performance (15x previous driver performance).
- Hit Modified species module to allow -1 or "uni" for the effective-seeding-distance parameter. This special value designates that the species is universally present on the landscape.
- Hit Created templates on the web site for each extension. The templates are arranged into a table such that Users can quickly identify and download extensions.
- Miss Did not migrate the LANDIS-II web site from Jimm's workstation to a new machine.
- Miss An administrative tool was designed and partially coded (install extension and list extensions finished, uninstall extensions and modify doc unfinished). However, due to time constraints, the tool was not included in the final release.
- Huge Hit Researched the security issues with not signing our assemblies. Testing code was modified such that signing is no longer necessary. These changes will

provide significant additional flexibility for creating and deploying new extensions.

- Hit/Modification A User's Guide was written for Biomass Succession. However, in keeping with the design and philosophy of LANDIS-II, this and other future extensions will now be developed as separate projects. Progress on the Biomass Succession can be followed on the Team Page, under Extensions.