Evergreen Software Development Report
February 13, 2008

Recent Evergreen Releases:

At the time of the last Executive Committee meeting, the PINES servers were on a heavily patched version of Evergreen 1.0.6. In January, the cluster was updated to 1.2.1.1; subsequently, minor release 1.2.1.2 has been deployed, and 1.2.1.3 will be deployed this week..

1.2 marked a significant change from 1.0; though the end user experience itself is not radically different. Many of the changes revolve around stability, friendlier messaging, bug fixes, and back-end changes, though a few new features have also been added. 1.2.1 marks the first release in the 1.2 line; the subsequent 1.2.1.1, 1.2.1.2, and 1.2.1.3 releases include fixes to regressions introduced by 1.2. Regressions are errors introduced by other bug fixes or new development. Further releases in the 1.2.1 line will include bug fixes but probably very little in the way of new development (Unrecovered Debt will probably be an exception to this; see below).

Depending on bug fixes and other changes in the 1.2.1 line, 1.2.2 will probably be the next large upgrade to the servers. Among other changes, 1.2.2 currently includes an interface to export holdings records for OCLC updates, holds freezing/thawing, bill printing changes, and a fix to an issue for patrons that currently have 50 holds placed.

Recently Deployed Changes:

- **1.2.1.1** (deployed January 23rd)
  - Error messages have been made friendlier/more useful.
  - Staff Client stability has been improved.
  - Popup Windows supported from Staff Client Portal page and other embedded pages (to allow Galileo use from within the SC).
  - Multiple open circulations on an item are no longer allowed by the Staff Client. This protects against situations where an item may appear twice on a patron's record and cause an improper billing.
  - Bill Payment receipt templates can now use the %PATRON_BARCODE% macro.
  - Different Windows users can now have different Staff Client settings (such as workstation name, receipt templates, barcode checking, and other user-determined options).
  - Staff client now disallows selection of invalid libraries as the workstation library.
  - Patron Searches can now be filtered by home library at local branch, local system, or consortium levels.
  - Phone number fields can include additional information. For example, if a patron requires calls to be placed via TTL, this information can be

included in the phone number field as follows: 123-456-7890 ttl code 8925.
    - Payment Annotation: All payment types now allow notes to be included when the payment is made.
    - Support has been added for collection agencies to query for "users owing money."
    - Protection against attempts to change the destination for holds that are in-transit has been put into place.
    - Bucket List Sorting: Bucket lists are now sorted alphabetically.
    - Batch Record Deletion: Option added in record buckets to remove multiple records from catalog.
    - Permissions:
        - New permission has been added to allow deletion of records, if no volumes are attached.
        - New permission has been added to prevent importing of records with alternate TCN's.
    - The OPAC now has Multiple Host Support: Content such as images or page templates can be located on different servers, which can be used to tweak Evergreen for faster loading of pages and to reduce problems related to added-content services.
    - Stop Words: Stop words are no longer used by default, though they can be configured. This allows title searches on phrases such as 'it or not' to return titles such as 'Believe it or not.'
    - Sorting of Call Numbers – in a record details display, each location's copies are now sorted by call numbers.
    - The server-side Installation/Upgrade process is more streamlined to reduce the possibility of configuration errors that could cause problems directly after an upgrade or installation. This will not affect most Evergreen users directly, but will lower the likelihood of problems related to the server upgrade.
    - Shelving Location's "Unholdable" overrides "Holdable" status on items.

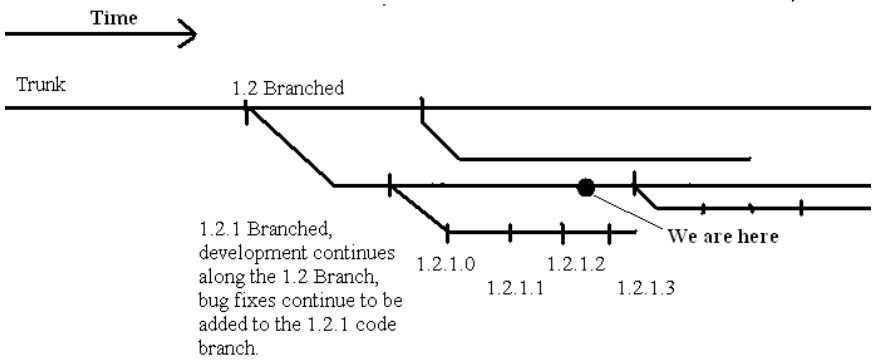
- **1.2.1.2 – Regression Fixes Only** (deployed January 26th)
    - Alphabetic sorting of regional systems in OPAC and Staff Client is reintroduced.
    - Automatic population of city and state in Patron Registration interface is brought back.
    - Authority Validation now works properly.
    - Doubling of Shelving Locations in the Copy Editor has been removed.

- **1.2.1.3 – Regression Fixes Only** (to be deployed this week)
    - Items will show in the Lost/Claims Returned/Long Overdue list upon being marked "Lost"
    - DOS Print Strategy bug fixed (users are only affected by this bug in isolated cases, but when it does manifest itself, it fairly severe for those that rely on the DOS Print Strategy)

- "Stat Cat Not Populating Error" will no longer popup upon adding Volumes/Items to a library.

Version Management Across Releases:

Once a major release branch, such as 1.2, is created, new development continues in Trunk, which is destined for a future release. The branch will also include backports of features Trunk bug fixes and new features that can be successfully integrated with code in the branch. Once a minor branch is created off of this branch (for example, 1.2.1 off of 1.2), bug fixes are added to it, the major branch, and Trunk, if applicable (sometimes Trunk can be very different than the branches, as it contains the newest and most experimental changes, so code changes to a branch or subbranch may be irrelevant to the code existing in trunk).



Currently, development is at a point on 1.2 after 1.2.1 has been branched, but before 1.2.2 has, so all bug fixes that have been added to 1.2.1 have been ported into 1.2, and will be included in 1.2.2. New development is also taking place in the 1.2 branch, as well as trunk.

Unrecovered Debt Update:

Requirements and Design:

The functional requirements for Unrecovered debt have been more or less finalized, as has the design for the feature.

Unrecovered Debt will be recorded as a flag that can be set to true, at library's discretion, on billable transactions. This flag will be set, on this iteration of Unrecovered Debt, via a CGI interface that is accessible through a web browser, rather than the Staff Client (for expediency's sake – it may be that this will be added to the Local Admin menu in the future) .

For the first round of development, only batch processing will be possible; the user will upload a file that contains a list of Transaction IDs; the interface will mark these transactions as Unrecovered and create a report containing Transaction ID, Amount Owed, Creation Date, Patron Barcode, Transaction Location. The Staff Client will also reflect the Unrecovered Debt status on applicable transactions.

Currently, the feature is designed and under construction; the database support is trivial, and the interface itself is basically complete. Right now, the middle layer to allow communication between the interface and the new field is under development; this is

expected to be completed in the next couple of weeks. The Staff Client portion is not under development yet, but will be relatively simple.
Reporting based on this status will be automatically allowed in the Reporting interface, once support for Unrecovered Debt is included in the production database. This will allow filtering and/or display of this flag on transactions in the same manner other fields on transactions can be filtered and displayed by reports.

Policy Concerns:
This feature should in no way dictate policy. It allows library directors or local administrators to mark these transactions, and reports can created to show patrons or transactions with Unrecovered debt – however, marking of Unrecovered debt is at the discretion of individual libraries, and handling of debt across different libraries or systems falls outside of the scope of the software, and should be handled according to current policies for handling such transactions.

There is some discussion about auto-barring or auto-blocking patrons when a transaction on their record is marked as Unrecovered. This will require input from the Executive Committee, so current development is going forward without this capability (it may be added later, if needed).

Version Timeline:
Depending upon timing of Evergreen releases, this will either be included in a 1.2.1.x release or a 1.2.2 release – in general, new development such as this would be held until 1.2.2; however, if it is ready and tested prior to 1.2.2, it will be included in a 1.2.1.x release as this is a very time-critical addition. The addition of Unrecovered Debt will have little or no impact on existing functionality, so integrating it into 1.2.1 will not be problematic (as could be the case with new code that does affect existing code).

Process Management Update:
We're still going forward with organizing functional groups (Catalogers only, for our first trial run) to help determine the priorities for Evergreen development. Currently, I'm working on drafting, more formally, the guidelines for the process as described below and setting up a central document repository. This repository will be used to hold documents related to process, such as a basic definition of process (version 1.0, of course – process will evolve as we learn along the way), requirements specification documents, templates, and issue tracking that groups can use to keep up with issues brought up in discussion.

The next step after this will be to set up templates for the various documentation that the process will use – most notably, requirements specifications. The briefer explanations of new features or bugs will be fairly informal, though guidelines will be created and published to give the groups a starting point.

Once the repository and necessary documents are prepared, we'll be ready to start the trial run of this process. Right now, the proposed process is loosely defined as follows:

The groups will each have a facilitator, most likely from the appropriate subcommittee. This person will be responsible for wrangling the discussion, leading the subcommittee meetings, when they occur, and keeping people on track in general. S/he will also be in charge of entering issues into issue tracking (this may prove to be too much of a job for one person; if so, then it may be broken up among various subcommittee members).

Phase 1: Initial, Informal Discussion

For a period of time, functional groups will discuss among themselves issues, bugs, desired functionality, and such. The various functional groups (again, just catalogers for the initial test run) should have some fairly private way to discuss this, and all members of the group should have access. Current thinking is that mailing lists like CAT-L would be appropriate, though it may become apparent that a separate mailing list for each group is needed. New issues will be added to the issue tracking database at this time. Discussion and introduction of issues to the tracking database will be called to a halt a week or two before the next phase begins to allow soak time, and give the subcommittees time to prepare for the next phase.

Phase 2: Prioritization by Functional Group

After the soak period, the subcommittees will discuss the issues that have been tracked and to determine their order of priority. This will happen in a combination of email discussion on listserv and an in-person meeting, lead by the facilitator. Each subcommittee will come up with a prioritized list of bug fixes and enhancement requests that directly affect the functional group represented. Bug fixes will include a brief explanation of the bug's symptoms, how often it occurs, and the degree of impact it has when it occurs. Enhancement requests will include a brief explanation of the day-to-day needs that the enhancement should meet, possibly with a description of how the need is currently met outside of Evergreen, if applicable.

Phase 3: Prioritization Across Functional Groups and PINES Team

From here, the PINES team will take the priorities lists with the information given, and discuss it with development to get rough estimates of how much effort the various issues will take (and to weed out unfeasible requests).

Using this information, along with discussion on the lists about the issues, we'll determine how to shuffle in the various requests, based upon urgency, severity and amount of development effort involved.

There will be some overlap (possibly a good bit) of priorities between functional groups – this is great, since it will give us an idea of issues that affect all users. There will also be priorities for each group that don't get reached, as there are only so many development resources that can be devoted to priorities at a given time. Priorities that are weeded at this time should be revisited and reviewed by the functional groups at future iterations of the process.

Phase 4: Formal Requirements Specification

Once changes to be implemented are chosen, the subcommittees and functional groups will be relied upon again to describe the requirements in a more detailed form. Ideally, larger features will end up having an 'owner' that is responsible for documenting the requirements, though this person will no doubt work closely with the PINES and other affected end-users, and there will most likely be input from the development team. For smaller issues, the specification or bug information may be very simple and can be gleaned from the issue tracking database.

Phase 5: Development and Deployment

Once the functional specification is completed and agreed upon by both the requestors and the development team, then we'll make request for code changes (of course, development work may start sooner, but we want to be very clear in stating what is needed from the software. It's also important that we have prior input from development to determine feasibility and to keep us honest about the differences between functional requirements and software design).