Environment Maintenance Tool
emt

1. Goals of emt

The concept of an application like emt was brought about by the current difficulty that exists in maintaining and releasing applications within several different environments. It was quickly realized that the tools needed for such maintenance were available, but the process of using them was large, time consuming, and complex. This lead to errors because the maintainer easily lost track of where in the process he thought he was. To be careful often meant to backtrack the steps that had been taken, costing the maintainer more time. The goal of emt is to relieve as much of this mundane and tedious work as possible, by providing a common application that will provide all the needed tools and allow not only the maintainer, but also the developer and user to monitor the status of the environment.

1.1 The User

The user will be able to take advantage of emt for inquiries on the state of a tree (what is different between beta and gamma release environments in the local tree?) and the state of an application. The state of an application will include a list of developers, what versions are in which release environments, and the current and fixed bug list for each version. The bug list will be made available through the bugs database, which will also allow users to submit possible bug reports to the developer(s).

1.2 The Developer

The developer will be able to perform all the tasks the user can, and several secure functions. Developers will be able to place requests for space for their application and for moving certain versions of their applications to different release environments (move version 012 into gamma). The developer will also be able to access the bugs database for handling reported bugs. After reviewing potential bug reports from users, new bugs can be opened. When bugs are fixed or worked on, they can be closed or updated. This will allow for a useful log of current, worked on, and fixed bugs.

1.3 The Gatekeeper

The environment maintainer should be thought of as the gatekeeper of all operations performed within the environment. It is the gatekeeper that will make any actual changes to the environment, through the tools provided by emt. After the developer request queue has been gone over to ensure there are only legitimate requests, these requests could be processed by one simple command. In addition to developer requests, there are automatic functions that generate reports on the environment and requests for changes. An example would be an automatic function that determines a volume in the omega area has been archived and then requests the volume be purged (unmounted and deleted). The gatekeeper would be able to review this list of automated requests, and process them with another single simple command.

1.4 The Tools

To provide the maintainer with the ability to perform his work within the framework of a single application will require the inclusion of the functionality of several other applications through a common interface. To increase the speed at which the maintainer can perform his tasks, several high level tools will need to be created that carry out the steps of his tasks using the base tools combined into a single step. In addition, since a large number of the base functions used require admin access, it would be desirable to perform these functions through requests to an "admin server."

1.5 The Interfaces

The user, developer, and gatekeeper will all be able to access the functions provided through emt through two interfaces. The first interface will be a command line version which will allow the use of all functions within a shell environment. The second interface will be motif based. All functions will be available through menus and buttons. Databases and request queues will be easily scrolled through.

1.6 The Databases

To allow the maintainer the confidence that all is going well, and to provide him with the current status of the tree, logs of all activity will be kept. The tree database will provide information on each release environment. The application database will contain information of the current state of each application. The bugs database will be part of the application database that contains the current bug status of each application.

2. Functionality of emt

2.1 Command Line

The command line version of emt will allow access to the commands the user is authorized for within a shell environment. When the program is initiated, the user will be presented with an emt prompt. At this prompt, the user will be able to issue the commands available, get help on the interface and commands, and perform simple character commands (previous command, history, modify last line, etc). The command line version will be the simplest front end to emt.

2.1.1 To The User

The user will be able to make inquiries as to the state of the tree through the tree commands that will be available in the emt shell. The results of the inquiries can be displayed directly or sent to a file.

The user will also be able to make inquiries as to the state of a particular application. These inquiries will be available through commands within the emt shell. Each inquiry will produce output that contains the id(s) of the maintainer(s), the version that is in each release environment (what is in alpha, beta, gamma), and a scrollable list of known bugs and their state (are they fixed in a certain release, or are they pending). This output can be viewed immediately, or sent to a file.

The bugs database will be available through commands within the emt shell. With these commands, the user will be able to inquire just about the bugs in a certain application. There will also a command to send in probable bug reports to the developer.

2.1.2 To The Developer

The developer will be able to perform all the items the user can, and a few more. In fact, certain developers could be given the ability to perform operations normally restricted to the gatekeeper, such as creating new volumes.

A developer will be able to place requests for space for their application with commands available to the developer in the emt shell. The developer will be required to supply the version, the system types, and the quota for the request. If the developer has been given the authority by the gatekeeper, these requests will be automatically honored. Normally, however, the request will be placed in a queue for the gatekeeper to review and process.

Requests for moving out a new version of an application to the next release environment will be a common action taken by developers. Once a move command has been issued, the developer will be required to provide the name of the application, which version, and to which release environment the move is desired. Some users may be given the ability to make some moves directly. Such users would be able to move their application out to beta, and pull it out of beta if there was a problem. Most users will simply have their requests placed in a queue for the gatekeeper to review and process.

The bugs database will be an aid to the maintenance of an application by the developer. With the bugs commands, the developer will be able to review the possible bugs submitted by users, open new bugs, update any work done on open bugs, mark bugs as being fixed, and purge old fixed bugs.

2.1.3 To The Gatekeeper

The gatekeeper will be able to perform all the tasks the user and developer can, and all other functions provided by emt within the tree(s) and release environment(s) for which he has been given authority. The gatekeeper will be able to give the special powers mentioned above to the developer. The purpose of the gatekeeper is to ensure that only valid requests are performed within the area he oversees.

The gatekeeper will be able to make the actual calls for creation of new space for an application through the same interface that the developer had access to, but since he is the gatekeeper the request will be immediately acted upon. Those requests that developers have sent to the gatekeeper will be available through the gatekeeper commands in the emt shell. Here, the requests can be displayed or sent to file. Each request will be have a reference number unique to it and the request queue it is in. The gatekeeper can then use this unique id in other commands to set each item to be selected for processing and purging, or deletion if it is an invalid request. If the gatekeeper prefers, he can choose not to select a request at this time, and it will remain for a future session. Once all selections are made, the gatekeeper issues a single command process the events marked for processing.

An identical system of processing will be used for the move requests received from developers. The gatekeeper will simply supply the move request queue name to the command to display the queue. He will be able to use the same commands to make his selections, since each id is unique. Again, he will be able to make his selections and process them with the push of a single button.

Yet another set of requests will be available through selecting the automatic request option of the gatekeeper menu. Within this area, requests from automated systems, such as archiving timeouts, will be presented within the same layout as that for developer requests. These requests will be generated when the gatekeeper runs the automated run command. This command will activate all the automated tests that generate the automatic requests for the gatekeeper. Again, he will be able to make his selections and process them with a single command.

The gatekeeper will receive notification of all changes made to the release environment(s) he maintains. He will also be able to add and remove from the list of people (or groups) to be notified of changes to particular trees and release environments. This will be available through the notify command.

2.2 Motif Version

The motif version of emt will present the user with push button access to emt. The initial window will have a textual logging area and pull down menus for each of the command areas that are available to the user. Different commands will also bring up new windows. Inquiries from the program will come in the form of pop up windows.

2.2.1 To The User

The user will be able to make inquiries as to the state of the tree through a tree menu that will be provided on startup. The results of the inquiries will be brought up in a pop up window, in which the information will be able to be scrolled through. A new information window will pop up for each inquiry about an tree.

The user will also be able to make inquiries as to the state of a particular application. These inquiries will be available through the base window's application menu. Each inquiry will produce a pop up window that contains the id(s) of the maintainer(s), the version that is in each release environment (what is in alpha, beta, gamma), and a scrollable list of known bugs and their state (are they fixed in a certain release, or are they pending).

The bugs database will be available through the base window application menu. Within this area, the user will be able to inquire just about the bugs in a certain application. There will also be the ability to send in probable bug reports to the developer.

2.2.2 To The Developer

The developer will be able to perform all the items the user can, and a few more. In fact, certain developers could be given the ability to perform operations normally restricted to the gatekeeper, such as creating new volumes.

A developer will be able to place requests for space for their application within the application menu of the base window. The developer will be required to supply the version, the system types, and the quota for the request. If the developer has been given the authority by the gatekeeper, these requests will be automatically honored. Normally, however, the request will be placed in a queue for the gatekeeper to review and process.

Requests for moving out a new version of an application to the next release environment will be a common action taken by developers. These requests will be found under the tree menu of the base window. Once a move has been selected, the developer will be required to provide the name of the application, which version, and to which release environment the move is desired. Some users may be given the ability to make some moves directly. Such users would be able to move their application out to beta, and pull it out of beta if there was a problem. Most users will simply have their requests placed in a queue for the gatekeeper to review and process.

The bugs database will be an aid to the maintenance of an application by the developer. It will be available through the application menu of the base window. Within the bugs area, the developer will be able to review the possible bugs submitted by users, open new bugs, update any work done on open bugs, mark bugs as being fixed, and purge old fixed bugs.

2.2.3 To The Gatekeeper

The gatekeeper will be able to perform all the tasks the user and developer can, and all other functions provided by emt within the tree(s) and release environment(s) for which he has been given authority. The gatekeeper will be able to give the special powers mentioned above to the developer. The purpose of the gatekeeper is to ensure that only valid requests are performed within the area he oversees.

The gatekeeper will be able to make the actual calls for creation of new space for an application through the same interface that the developer had access to, but since he is the gatekeeper the request will be immediately acted upon. Those requests that developers have sent to the gatekeeper will be available through the gatekeeper menu of the base window. Here, the requests will be brought up in a scrollable menu format. They will be able to be selected for processing and purging, or deletion if it is an invalid request. If the gatekeeper prefers, he can choose not to select a request at this time, and it will remain for a future session. Once all selections are made, the gatekeeper pushes a single button to process the events marked for processing.

An identical system of processing will be used for the move requests received from developers. The gatekeeper will simply select a different option from the gatekeeper menu in the base window, and he will be presented with the move requests in the same format. Again, he will be able to make his selections and process them with the push of a single button.

Yet another set of requests will be available through selecting the automatic request option of the gatekeeper menu. Within this area, requests from automated systems, such as archiving timeouts, will be presented within the same layout as that for developer requests. These requests will be generated when the gatekeeper selects the automated run option of the gatekeeper menu. This option will activate all the automated tests that generate the automatic requests for the gatekeeper. Again, he will be able to make his selections and process them with the push of a single button.

The gatekeeper will receive notification of all changes made to the tree(s) and release environment(s) he maintains. He will also be able to add and remove from the list of people (or groups) to be notified of changes to particular trees and release environments. This will be available through the notify option of the gatekeeper menu.

3. Design of emt

3.1 Notification

Emt will have the ability to notify its users and those people affected by its operations. When operations are performed that modify the tree(s) and release environment(s), messages will be sent to appropriate bboards andusers related to them. For example: When a new version of an application in the local tree is released to the beta environment, the beta gatekeeper, the beta bboard, and operations are three possible targets for notification by emt.

4. Pseudo-Code of emt

4.1 Data Structures

4.1.1 Environment Database

1. Which versions are mounted in which environments.
2. When each version was mounted in each environment.
3. Generate purge notices for gatekeeper approval (e.g. volume mounted in omega for 30 days).
4. Flag alpha volumes which haven't moved in 6 months (dead space or misused space).
5. Track all pertinent information, who is the developer, old releases, what has been purged to tape, etc.

4.1.2 Bugs Database

1. Track bugs on a per-version basis.
2. Queue reports from users for the developers to review.
3. Allow developers to file bugs, assign priorities, mark bugs as closed.
4. This implies that certain users have "privileged" access for certain applications.
5. Dump a history of bugs per application per version.
6. Honor a request for a release notice, stating which bugs were fixed in the new version - have this generated from the database.

4.2 Procedures

4.2.1 Create A New Application

1. Get name of application.
2. Get list of developers.
3. If not authorized, queue as a request to the gatekeeper.
4. Create access group for the application.
5. Create new volumes for the application.
6. Create a new entry to the application database.
7. Notify appropriate people of creation of a new application.

4.2.2 Create New Volumes for an Application

1. Get application name.
2. Get version of application being created.
3. Check to see if this is a valid request. If it isn't, notify concerned.

4. Get quota needed for source area.
5. Get list of systems for which to create volumes.
6. Get quota needed for system volumes.
7. If not authorized, queue as a request to the gatekeeper.
8. Create, mount, set quota, and set access for the source volume.
9. Create, mount, set quota, and set access for the RCS volume.
10. Create, mount, set quota, and set access for the system volumes.
11. Update the application database for this application's entry.
12. Notify appropriate people of creation of new application volumes.

4.2.3 Display the State of an Application

1. Get application name.
2. Load information for the application from the application database.
3. Display information loaded for the application.
4. Display the bugs of this application.

4.2.4 Send a Possible Bug Report on an Application

1. Get application name.
2. Get text from user regarding the possible bug regarding the application named.
3. Place this text in the possible bug list for the developer(s).

4.2.5 Display the Bugs of an Application

1. Get application name.
2. Load bugs information for the application from the bugs database.
3. Display the information shown to the user.

4.2.6 Open/Update the Bugs of an Application

1. Get application name.
2. Get name of new bug or id of existing bug.
3. Get text on the nature of the bug or efforts made to correct the bug.
4. Create new id if this is a new bug.
5. Place text for the bug prefixed by the id of the bug and the date into the bugs database.
6. Notify appropriate people of creation of new application volumes.

4.2.7 Move a Version of an Application to a New Release Environment

4.2.8 Pullout of a Previous Version Move

4.2.9 Display the Difference between two Release Environments

4.2.10

4.2.11 Release To Gamma

Gatekeeper acting on a developer's request. (Note it is possible to release for a subset of architectures)

1. Remove old mountpoints from gamma (if they exist).
2. Mount old volumes in omega.
3. Purge old omega volumes (if they exist. This means unmount the volume, request a tape snapshot, then delete the volume when the tape request has been honored).
4. Mount new volumes in gamma.
5. Run depot in gamma.
6. Replicate new volumes (if beta is not replicated).
7. Release parent volumes.
8. Generate release notice (Developer will supply the details. The bugs database can also be accessed for any outstanding bugs, or for a list of repaired bugs between this version and the previous)
9. Notify Operations.
10. Update environmental database and if necessary bugs database.
11. Fix access lists on released volumes (deny developer access).
12. Unmount volume from alpha (default behavior)
13. Clone src volume (and obj volume if mklink is used).
14. Mount src clone in new environment.
15. Generate request for src snapshot to tape.
16. After tape image is created (of src), delete clone and remove mountpoint.

4.2.12 Back-out from gamma

Note it is possible to back out for a subset of architectures.

1. Recover omega volumes (from tape if necessary).
2. Move gamma to alpha (unmount new alpha volumes - mount somewhere temporary).
3. Replicate omega volumes.
4. Move omega volumes to gamma and beta.

5. Run depot in gamma and beta.
6. release parent volumes.
7. Handle Src control - unclear how to do this currently.
8. Update environmental and bugs database.
9. Notify operations.
10. Generate and post back-out message.}

4.2.13 Release To Beta

Gatekeeper acting on a developer's request. (Note it is possible to release for a subset of architectures)

1. Remove old mountpoints from alpha.
2. Mount new volumes in beta
3. Run depot in beta.
4. Replicate new volumes (if beta is replicated).
5. Release parent volumes (if beta is replicated).
6. Generate release notice (Developer will supply the details. The bugs database can also be accessed for any outstanding bugs, or for a list of repaired bugs between this version and the previous)
7. Notify Operations.
8. Update environmental database and if necessary bugs database.
9. Fix access lists on released volumes (deny developer access).
10. Unmount volume from alpha (default behavior)
11. Clone src volume (and obj volume if mklink is used).
12. Mount src clone in beta src environment.
13. Generate request for src snapshot to tape.
14. After tape image is created (of src), delete clone and remove mountpoint.

4.2.14 Back-out from beta

Note it is possible to back out for a subset of architectures.

1. Move gamma to beta,
2. Run depot in beta.
3. release parent volumes (if beta is replicated.)
4. Handle Src control - unclear how to do this currently.
5. Update environmental and bugs database.
6. Notify operations.
7. Generate and post back-out message.

4.2.15 Request for new volumes from a developer

1. Create obj, src, rcs, alpha volumes.
2. Create access list.
3. Entry in bugs and environmental databases.
4. Mount all volumes.
5. Generate notice to developer.

4.2.16 Age Omega Volumes

Age Omega volumes and purge to tape after a certain time.

1. Generate a request to Archive Server.
2. Update databases.

4.2.17 Inquiries

1. What is different in (beta gamma omega)?
2. List outstanding (closed) bugs on application in environment.
3. Who is working on what and for how long.
4. History of each application.

4.2.18 Requests

1. Queue all requests for the gatekeeper.
2. Simple interface so the gatekeeper can honor or deny request easily. Give reasons for denial.
3. Only allow developers to queue requests on their own projects.
4. Provide a template so that all information is available.}

4.2.19 Retrieval - Source Control

1. Be able to retrieve any source for any released version.
2. Be able to retrieve depot volumes for any version.

4.3 External Services Needed

4.3.1 Administrative Server

1. Handle requests to replicate volumes.
2. Handle requests to unreplicate volumes.
3. Handle requests to create volumes with specific space requirements.
4. Handle requests to delete volumes.
5. Handle errors and denials intelligently (e.g. Not enough space on the system to honor request).

6. Balance space usage, move volumes.
7. Handle requests to clone volumes on a priority basis so development won't have to be halted when a release is requested. This is for the source snapshot.
8. Be able to check for appropriate authority to complete requested task. The requesting user need not have an admin token, just have been registered with this server as having authority to perform these tasks.
9. Set access lists and quota for new volumes.

4.3.2 Archive Server

1. Handle requests to archive a volume.
2. Answer queries as to whether request has been honored.
3. Send information which will allow easy recovery (tape number, sequence number, header information).
4. Handle requests to retrieve any volume on tape.
5. Answer queries as to whether a retrieve request has been honored.

Table of Contents