Online System Administration Framework

Motivation and Objectives

As part of the daily operation of any information technology (IT) facility, a variety of tasks are identified that become routine and time consuming. A good rule of thumb to follow is the “rule of threes.” Essentially this rule asserts that any given task repeated more than three times using the exact same procedure should be automated. The code supporting that automation should be placed in a protected library structure. We will refer to these code blocks as a Complete Corresponding Source Code (CCSC) unit as defined in the GPL v3 document.

As intuition would assert, the number of the CCSC units would increase in number as the IT facility matures. This increase in number of units and lack of coherent management of said units would create an administrative problem. This problem could potentially grow to the point at which the units become more troublesome that the task they were created to address. Without the benefits of a coherent framework, version control, access control, these units can become the source of delay, confusion, and ultimately data loss.

In the environment described above, work flow and efficiency are not the only victims. Audit controls are often completely ignored. Audit data from these tasks are as important as the completion of the original task itself. First, audit data provides a way to establish what and when work has been completed. This point of communication is important not only to the administrator but the consumer as well eliminating inaccurate and sometimes absent feedback. The consumer cannot use a newly created account if they do not have the knowledge it has been created or what the username and password is. Secondly, audit data from such transactions can assist in the maintenance and cleanup phase of account and privilege allocation. Some of the greatest threats to an IT framework can come from dormant accounts that are never purged.

The objective of this project is to create a web-based framework to organize, categorize, and secure the use of these utilities as well as the audit data created from these activities. The primary focus of this project is the framework and audit data collection more than the individual utilities themselves. The project will be developed on the premise that a well-design framework would accommodate any new utility provided that utility was created with input and output standards established by the framework. Attention will also focus on recording the changes made to the IT system as part of each interaction.
Beyond the development of the framework and auditing functions, one or two sample utilities will initially be ported to the framework as a proof of concept that the basis for the framework is valid. Other utilities will be ported to the framework as time allows.

**Proposed Work**

As stated earlier in this document, the primary goal of this project is to create a sound framework under which tested units can run. To be useful the framework must be kept as simplistic as possible. While this approach might appear to violate the flexibility rule stated in GHOST project, overall this simplistic approach will yield the greatest benefit by concentrating on accurate audit data.

By incorporating a collection of disconnected and non-standardized scripts currently used in the department IT administrative functions into an organized framework, creation and allocation of computing resources can be performed in prompt and more secure manner. This process will also force the individual units to follow a “well thought” path avoiding a mix of approaches (i.e. different user accounts with different access control list settings).

Automation of many of these administrative tasks inherently decreases security. Much time must be spent to assure that each state of development is performed in a secure manner. This security can be achieved through the use of SSL for all web connections in addition to the use of .htaccess files to protect core portals. It is for this reason, the client request screen, not addressed in this document, will be treated as a separate entity to avoid improper privilege overflow.

![Figure 1 Architectural Summary of System](Image)
Conceptual Level Design

Figure 2 shows the basic ER diagram for the framework. Note the simplicity of the initial model. The primary goal is to capture the event information for future auditing. The actual work that is accomplished by a unit will be self-evident in the applied domain. In other words should the unit create a new user in Active Directory, a simple lookup in the directory using the Active Directory Users and Computers tool will show that an entity was created. The framework needs only to keep track that the event occurred, nothing more.

Plan of Action

The overall system will be developed, in part at least, as a traditional AMP (Apache, MySQL, PHP) application. The CCSC units will be developed in Python. Input text parsers will be developed in Python as well. All code with the exception of elements that must run on the Microsoft Windows Server 2003 platform will be executed on Mac OS X 10.4.5.
### Schedule of Development Activities

<table>
<thead>
<tr>
<th>Date</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>3/6 – 3/17</td>
<td>Develop initial user interface and basic controls for interaction. Create initial database tables and test user interface controls.</td>
</tr>
<tr>
<td>3/20 – 3/31</td>
<td>Develop parsing and sample logic containers for populating the database and basic reporting.</td>
</tr>
<tr>
<td>4/3 – 4/14</td>
<td>Develop at least two units for integration into the framework and further develop auditing information. Begin initial testing.</td>
</tr>
<tr>
<td>4/17 – 4/21</td>
<td>Complete testing of the system.</td>
</tr>
</tbody>
</table>

### Testing Methods

Testing the system will take place on two distinct levels. The first level of testing will involve creating dummy units to attach to the framework. This will allow for testing of the framework itself to assure that events are being recorded correctly. The user interface will be tested by outside staff allow the GUI to be evaluated in an unbiased manner.

The second level of testing will be performed on the units themselves. This will be an ongoing process that will continue outside of the timeframe for the project. Once a unit has been independently tested to assure it performs its basic function correctly, the unit can then be placed inside a safe wrapper to be run within the framework.

---

1. Dr. Srini Ramaswamy, as part of a CSC 6700 lecture
2. [http://gplv3.fsf.org/draft](http://gplv3.fsf.org/draft) (current as of 2/27/06)