The Harvard Integrated Management System - Project Integer[1]

History/Background

For more than 25 years, networks and the elements that comprise them, such as routers, firewalls, load balancers, name servers, middleware, database servers and the applications that use them have grown in number and complexity.

Little fundamental has changed in the way we configure and monitor each of these systems. Protocols and data formats for the configuration and monitoring of these systems come and go without producing a meaningful improvement in our ability to manage our complex environments. While methods of automation have begun to help us with scale a bit, separate monitoring and control systems are still the rule. The tasks of configuration and monitoring are in stove-piped domains and each type of technology and vendor has a separate set of tools and instrumentation. For example, we tend to configure and monitor our routers and network devices with one set of tools and our servers with another set.

Tools for the management of these systems come from one of several sources:

1. Home grown scripts and programs that often were created in isolation and when the engineer that wrote leaves the organization, it becomes difficult to evolve and maintain them. These scripts also tended to be single purpose, solving a specific problem and not integrated with other information.
2. Open source solutions began to pop up in the 1990's but these too were limited in scope. For example, monitoring basic server or network element statistics. Open source solutions are available today that do some configuration, especially of servers, but they too are limited in scope and do not integrate well with other systems.
3. Commercial management software. Commercial management software is driven toward implementations that are targeted at low hanging fruit in terms of function, leaving the more complex and difficult problems unsolved. This is particularly true in the case of configuration. This class of software includes software from various public and private consortia. These organizations make assumptions about the ability to drive a single access method, set of management objects or approach, or they ignore all outside the range of things over which they have direct control.
4. Management software from vendors that produced the hardware or software system (e.g., server, router, storage or database system) - this management software was frequently the most complete and was targeted at the products

[1] The project, Integer, is an attempt to create a unified whole from the separate protocols, data elements and software systems we use to operate our increasingly complex computing environments. See: http://www.thefreedictionary.com/integer. Also see: http://en.wiktionary.org/wiki/integer#Latin

produced by that vendor[2]. The reality is that organizations are serviced by multiple hardware vendors so this approach which is costly, creates stove pipes of information. For example, information about servers is separate from network elements in these vendor solutions.

With the evolution of new technologies such as virtualization of everything from servers and networks to entire data centers, a new set of tools is emerging that is also different. To make this more difficult, many applications/environments will be of a hybrid nature. That is, some of the systems will exist in data centers owned by the enterprise and others will exist 'in the cloud'. This further fragmentation will make worse the problems many already have in the areas of coordinated configuration management and monitoring. Imagine the scenario where it is necessary to permit access to/from a server in the cloud to a server in the enterprise data center. In this case, multiple devices (virtual or physical) will need to be configured, including the servers for the access permissions to work correctly. In addition, many different elements of the infrastructure will have to be monitored in a coordinated fashion to be able to accurately identify when problems arise and isolate their root cause.

The Failure of Integration

The approach most often taken to address the stove-piped environments that result from these separate tools has been to attempt a variety of forms of integration. Since the tools were not designed to be integrated, the costs are often high and the results less than optimal. Integer takes a different approach, the construction of a system from the ground up that is broad in scope with regard to functions, vendors and technologies managed. It is believed that this approach will cost less than the myriad of tools that are usually acquired from the sources identified above and glued together on and ad-hoc basis or with expensive in-house or consulting resources.

Scope of Project Integer

The system described in this document is designed to integrate all of the major aspects of the management of the systems described above across all the hardware, software and vendor environments identified. Additionally, it does so in a way that makes it possible to configure, control and monitor these elements in their relationship to the applications/services they support. For example understanding how the impact of a firewall failure may impact a payroll or personnel application, or what systems in the environment must be (re)configured to allow access to an application from systems outside the core enterprise. In short, over time, the system will incorporate all of the following:

1. All the major areas of management:
    1. Fault
    2. Configuration
    3. Accounting

[2] The situation is not even this good. Some vendors through acquisition or organizational stove-pipes, sometime provide more than one management system. Often these systems are not well integrated.

4. Performance
5. Security

2. It will operate on a wide range of types of hardware and software systems:
    1. Network elements such as routers, switches, firewalls, name servers, load balancers and variations associated with different cloud technology vendors.
    2. Server/Operating system environments
    3. Middleware such as JBOSS
    4. Databases of various types
    5. Storage technologies
    6. Network/infrastructure services like the DNS
    7. Application layer software
    8. A variety of cloud/virtualization technologies
3. The starting place for Integer is the hand off from a development process. Indeed it will be designed to integrate with traditional tools found in development like source code control and build systems. This is necessary so that the system can manage the 'promotion' of code and configuration as it moves from test to production.

Assumptions/Design Goals and Values

All engineering projects make assumptions. By enumerating them, the logic behind various decisions and approaches to additional priorities and technical choices is clarified.

1. Integration of functions, system types, and environments, is a fundamental objective. This integration will serve to reduce errors, facilitate more rapid resolution of errors when they occur and improve efficiency of the operation. This is based on observation of the types of errors that occur. They are often ones of logic that span multiple systems. It is believed that integration of information across systems will enable more efficient/accurate configuration, improve productivity and improve overall availability of the services delivered by the environment.
2. Not all data and the functions enabled by their integration into this system are equal. That is, while there is a strong desire for integration; implementation priority is driven by a reasoned analysis of the cost/benefit of the integration of each type of data/system[3].
3. Specialized tools will remain - it is not cost-effective to integrate into Project Integer highly specialized analytic tools. Examples of these tools would be those that perform deep packet inspection or tune a specific vendor database technology. It is desirable to try to provide a loose coupling to some of these tools where possible. For example, if a performance problem with an

[3] One of the reasons for running this project in the Open Source community it that different organizations will have different resources and priorities, by providing a solid framework for the integration of different types of data to that community, everyone will benefit from having data integrated and functions available that their own resources would not permit.

application can be isolated to a Database system and the data available to the integrated system does not sufficiently answer why there is a problem, then the integrated system might be used to launch the specialized tool and pass a few relevant parameters (like server and DB names).

4. Orchestration and Automation Tools - Project Integer assumes that it can/will make use of/integrate with many extant orchestration/automation tools such as Puppet.
5. A single layer of abstraction is better than two, or three - for a system like Integer to work, it must abstract all of the different formats of data like information from Puppet Manifests, SNMP data of all types, some log information[4] and various cloud automation frameworks such as those found in AWS, etc. It will avoid intermediary tools/frameworks that have their own abstraction layers. Multiple layers of abstraction have proved problematic is the past - including those provided the various standards consortia.
6. Open Standards and Frameworks - whenever there is an option of using an open standard/framework for a function or data exchange format, that will be used if at all practical as opposed to creation of new formats/protocols.
7. No technology that requires per user licensing will be incorporated into the system unless is can be 'unplugged' and reasonably substituted.

An Open Source Development Effort

As noted previously, this project will be an open source effort. Efforts more limited in scope than Integer have a long history in the IP management domain[5]. The objective is to leverage the combined operational and engineering effort represented in that community (particularly in higher education) as soon as a functional framework exists.

Component Strategy

The general approach will be to use open source technologies that are 'reasonably' stable, and do not restrict other design choices. Possible examples include (actual choices will be made as the engineering effort requires:

1. Logging facilities for the system.
2. SNMP Libraries.
3. Database technologies.
4. Facilities for the creation of user interfaces and user interface frameworks.
5. Libraries for the integration to cloud based configuration and monitoring environments.

Initial Phases

The initial phases of the project will focus in two areas:

---

[4] How much log information is integrated versus externally referenced will be made on a case by case basis.

[5] Two well known examples include: OpenNMS that was started in 1999, http://en.wikipedia.org/wiki/Opennms#History, and CMU SNMP in 1992, http://en.wikipedia.org/wiki/Net-SNMP#History,

1. Creation of enough basic infrastructure so that one key function may be implemented from top to bottom.
2. Extensive and flexible discovery and inventory management. Knowing what is in your environment and what has changed is fundamental to all other aspects of management. The initial release will collect information from all types of server and network resources including layer 2 and 3 topology with the ability to annotate layer 1 elements. Recognizing that integration with extant systems is important until Integer has a broader range of functions, export of this information in JSON format is available.