Chapter XXXVII
Governance and the Open Source Repository

R. Todd Stephens
BellSouth Corporation, USA

ABSTRACT

This chapter examines the critical task of governing the open source environment with an open source repository. As organizations move to higher levels of maturity, the ability to manage and understand the open source environment is one of the most critical aspects of the architecture. Metadata can be defined as information pertaining to the open source environment that the organization defines as critical to the business. Successful open source governance requires a comprehensive strategy and framework which will be presented through historical, current-state, and future perspectives. The author expects that by understanding the role of open source metadata and the repository within, researchers will continue to expand the body of knowledge around asset management and overall architecture governance.

INTRODUCTION

Open source continues to make inroads into the corporate environment where it is now a standard embraced by most of the top tier corporations in America (Ferris, 2003). Applications like Apache and Linux have been phenomenally successful in providing real business value (Garvert, Gurbani, & Herbsleb, 2005). However, further research is needed in how organizations should govern the open source environment which requires more than the indemnification of the product. Open source governance requires the establishment of architectural standards that each and every group can adhere to in order to deliver bottom line business value. A centralized repository for downloading certified open source products ensures that the principles of asset management are implemented and managed effectively.

The driving purpose of the architecture community is to minimize the unintended effects on the business due to technology changes. Utilizing an open source repository for impact analysis will ensure that proposed changes will not create catastrophic events within the business itself. The repository provides the mechanism for inventory management which allows organizations to see what is already acquired, deployed, and supported within the environment. In addition, efforts like domain analysis, reuse, and release

management are essential to the implementation of open source as an enterprise asset. When organizations embrace open source as a viable alternative to in-house or outsourced development, they must accept the responsibility and implications of transforming it from code to an enterprise asset.

BACKGROUND

The background section will review the core concepts that enable open source governance within a large scale deployment. Additionally, the historical precedent for a repository will set the stage for the introduction of the open source repository. Architecture governance is the practice and orientation where the technical architecture is managed and controlled at an enterprise-wide level. Maturity models provide a framework by which organizations can measure their progression of governance; software and open source maturity models will be reviewed. The higher levels of maturity define an environment where consistency, predictability, and ongoing optimization are the keys to success.

Architecture Governance

Information technology governance specifies accountabilities of technology related business outcomes and helps companies align their technology investments with their business priorities (Ross & Weill, 2005). Enterprise architecture is a set of frameworks, principles, guidelines, and standards created to guide the development and deployment of enterprise systems. The rate of change in the business is accelerating causing the cycle times allowed for implementing new systems to decrease. Existing technology infrastructure often gets in the way of rapid change and may inhibit the organization's ability to respond. By having an architectural governance program, large enterprises can respond quickly and effectively to the demands of the business. One tool that can be used to determine the road map of governance is called a maturity model. A maturity model is a method for judging process maturity of an organization and for identifying the key practices required move to the higher levels.

Software Maturity Models

In 1986, the Software Engineering Institute (SEI) was asked by the U.S. Air Force to create a systematic method of evaluating software contractors. In conjunction with the MITRE Corporation, the study group produced a questionnaire that enabled the Air Force to judge a software provider as either successful or unsuccessful in its capabilities. The questions were divided in a number of groups (key process areas) and then assigned to specific levels within the model. The resulting model was called the capability maturity model (CMM). The levels describe the path a software provider must follow in order to move to the higher levels of maturity. These paths are actually a collection of key practices that must be mastered before moving to the next level (Baskerville & Pries-Heje, 1999). Maturity implies a potential for growth in capability and indicates both the richness of an organization's software process and the consistency with which it is applied in projects throughout the organization. In addition, productivity and quality resulting from an organization's software process can be improved over time through consistent gains in the discipline achieved by using its software process (Chrissis, Curtis, Paulk, & Weber, 1993). The CMM provides five levels of maturity: initial level, repeatable level, defined level, managed level, and optimized level.

Level 1: The Initial Level

At this level, the organization has a less stable software process and management practices. The process is ad-hoc and changes as work progresses.