Versioning WebDAV Server
Tools & Technologies Plan

Mike Fisk Daniel Wittmer

Revision: 1.5 ; Date: 2001/06/15 13:55:42

This document identifies tools and technologies that will be used throughout this project. In turn, these technologies drive the personnel skills required for the project. We show how the selected technologies address some of the overall project risks, and also detail more specific risks that result from using these technologies.

1 Skills & Technologies

In order to implement the DeltaV server we will rely on several tools, technologies and skills. These technologies have been chosen to partially address two of the risks identified in the project vision document: resource and time constraint risks and risks of the product not being accepted by the customer-base. In addition, the choice of these technologies implicitly requires that we have basic skills in these technologies or have the ability to easily fill-in gaps in our knowledge.

1.1 C development

A fundamental tool and technology that will be relying on during the course of the project is the C programming language and our comprehension thereof. C will be required in order to implement the module which will interact both with the Apache server as well as the CVS repository. Since all of the developers of the project are well versed in C, and the language is well documented, its use will aide in achieving the deadlines. Additionally since C is the predominant language for Unix open source programming, to which our program is geared, and use the of any other language would pose a barrier to acceptance.

1.2 HTTP

The HTTP protocol forms the basis of the DeltaV and DAV technologies. While we intend to reuse existing components to handle the basic HTTP implementation, a strong knowledge of the protocol is required. In particular, it is important that our implementation remains standards compliant in order to achieve the interoperability that is important to the success of the product. Further, important aspects of DAV, such as authentication and authorization are not specified in the DAV and DeltaV documents and rely on other, existing HTTP extensions. Fortunately, both primary members of the project team have written HTTP servers and/or clients.

1.3 DAV Standards

The DeltaV server will implement a significant portion, if not all, of the DeltaV specification for versioning extensions to DAV. Since DeltaV is an emerging technology with strong ties to DAV, a ”hot” technology, its use, as opposed to developing a proprietary versioning protocol, improves the chances of acceptance as well as reduces the development time. The project team has previously implemented DAV clients and is already familiar with the basic protocol.

1.4 Apache modules

Apache includes a modular architecture which allows developers to extent it’s functionality via the definition of new modules. We will utilize this technology in order implement the DeltaV server, or more specifically to add the DeltaV functionality to Apache. Using this technology will aide the development process since a well defined interface, on that Apache server side, will already be provided. Additionally due to the availability of a DAV module for Apache the development time will be reduced since we can use this module as a reference.
1.5 CVS

CVS is a free open source versioning client/server available for a large number of operating systems and architectures. Since DeltaV requires a versioning back-end, rather then implementing our own versioning server, we originally planned to rely on CVS. We felt that using CVS would provide a mature versioning back-end to the DeltaV server, greatly reducing the development time. Furthermore the use of a widely accepted versioning back-end would greatly increase the probability of acceptance due to an increased customer comfort level. Later in the design and development process we realized that the requirements for storing and accessing version metadata made using CVS very difficult in return for very little value. The primary utility of CVS in this scenario would be more compact storage of deltas between versions rather than complete versions. However, the development time to build an interface around CVS that would be usable by WebDAV make this an unwise strategy at this time. Instead, we will use a simple flat-file version repository.

1.6 Systems programming

Systems programming and an understanding of systems design will be required in order to interface the DeltaV module with Apache and CVS. More specifically both will be required in order to perform the necessary transfers of data between the two applications as well as commonly used sanity checks. Furthermore knowledge of Unix security practices will be required in order to create a system which does not severely violate common security practices e.g. running our module as root etc.

1.7 Open source development

In general, our target market is weary of non-open source development projects. This, in conjunction with the fact that our startup has no reputation, be it good or bad, drives us to use open source development in order to improve acceptance and potentially reduce the development time. It will aide acceptance since it being open source will allow any potential customers to verify our design and/or customize it to their specific needs. Being open source also allows third parties to contribute program code, bug fixes or suggestions (requirements), and documentation to the project, increasing the size of the development team without incurring significant costs on the side of our company.

2 Risk Analysis

Using the identified tools and technologies presents the following new risks that have not been previously identified or addressed:

2.1 Security Requirements

The goal of our project is to create a software component that is absorbed into the Apache distribution as ‘mainstream’ software. The community is very conscious of security concerns with high-profile, publicly accessible services such as a web server. Thus, our software must be sufficiently robust and well designed that it is acceptable to this community. Specifically, it must be able to run as a non-privileged user, and should be free of buffer-overflow and other common attacks. If software quality is sufficiently poor in early public releases, it is likely that the component will not be widely used and distributed.

2.2 High Skill-level Required for C Programming

Due to customer requirements of the Open Source community, it is necessary to implement our program in ‘C’. While we have extensive experience with ‘C,’ we are also aware that creating secure and robust code is made more difficult by the low-level nature of the language. Higher-level languages would lead themselves to rapid prototyping, type safety, and bounds checking, but would be unlikely to satisfy the customer.

2.3 Apache Reuse

The Apache server is designed with a modular architecture for supporting new back-end services and implementations, but the front-end, the HTTP protocol has not varied much. However, WebDAV and versioning introduce new operations and response codes that must be included in the front-end. We do not yet know the extent of the Apache code that will have to be modified and how time-consuming these changes will be.
2.4 CVS Reuse

CVS is widely used as stand-alone application and server, but has not been widely used by other applications. Thus there is no library interface to CVS. Our project will either have to use CVS as an external application (with security, performance, and reliability risks), create an API interface to CVS (requiring more development), or use CVS in a client-server architecture (if the normal CVS functionality is a superset of all DeltaV functionality).

2.5 CVS Documentation

CVS documentation is somewhat lacking in general, and completely lacking in terms of source code internals and network protocol. For example, reading the source code shows that the official response codes form the CVS network authentication routines are ‘I LOVE YOU’ and ‘I HATE YOU’. Most open-source projects have relatively readable code, but there is a risk that any modifications that we might make to CVS may be difficult and time-consuming.

2.6 CVS Functionality

Related to (or at least exacerbated by) the previous two risks is the risk that CVS has no internal support for copying, locking, or file access control. The latter function is performed by using underlying OS file protection mechanism and distinct accounts for clients. Thus, we may have to augment CVS with additional functionality in order to support DeltaV in a secure fashion.

2.7 WebDAV Component Dependence

DeltaV is an extension to WebDAV, which is itself an extension to HTTP. Since WebDAV is handled as an extension to Apache, it may (or may not) be difficult to use that component as infrastructure for another component. In a worst-case scenario, the existing module may have to be replaced with a new module that provides DeltaV functionality as well as basic DAV functionality. This would require additional resources and may hinder acceptance by our customers.

3 Assessment/Conclusions

While we have identified additional technical risks, the project still seems viable. We are gaining a better understanding of what the software architecture will be and what external constraints are placed on it. However, we look forward to the customer requirements analysis which we hope will produce additional use cases and focus for the product.