Alfresco – *A Fresh Approach to Content Management*

Open Source Enterprise Content Management

White Paper

A Fresh Approach to Enterprise Content Management

Alfresco brings a fresh approach to enterprise content management delivering convenience, scalability and very low cost through open source. The team at Alfresco has been working on content management for over 15 years and has designed a new system for performance, modularity and ease of use. Using the open source model Alfresco makes enterprise content management affordable for many applications that were not possible before. Alfresco simplifies content management through intensive user design, rule-bases that automate work, and interfaces that make managing content as simple as "SaveAs". By simplifying and modularizing content management, the Alfresco system is much faster than many commercial content management systems.

Alfresco was founded in June 2005 by John Newton, co-founder of Documentum, and John Powell, former COO of Business Objects, and funded by Accel Partners. Alfresco's mission is two-fold: to open up content management through open source to people and uses that could not afford enterprise-level content management before; and to increase innovation in the sector through community participation and free access to source code and designs. Alfresco's team has extensive experience in designing content management systems and interfaces, including Documentum's server, Java web development kit and portal integrations.

This experienced team has applied the lessons of the last decade in content management to create the Alfresco system, an open-source, open-standards content repository and portal framework. The Alfresco system has a modern architecture that utilizes Aspect-Oriented Programming for modularity and adaptability along with the latest open source tools for performance and scalability. Alfresco provides components designed to integrate into standard portals and authoring tools that are easy for end-users to understand and are quickly productive. Alfresco makes content management much easier for users by making the content management look like a shared file system and automating tasks, such as versioning and classification, through business rules. Alfresco is licensed as open source through the LGPL which puts no restrictions on the use as part of applications, even commercial systems.

The Alfresco content management system is targeted at professional developers of portals, content and compliance applications, and online services struggling to meet the needs of end-users creating and sharing content. For J2EE-based professional portal projects, the Alfresco system has enterprise content management capabilities at a much lower cost than commercial systems. For organizations that wish to replace shared drives for storage of information to improve sharing and increase compliance through content control, Alfresco provides file share interfaces to simplify content capture and portal interfaces to facilitate searching and sharing. For ISVs and developers who wish to enhance their applications, such as CRM or ERP, with content management, Alfresco provides a low-cost, royalty-free, super-fast content management system. Alfresco is also a low-cost, open source alternative to Microsoft® SharePoint™.

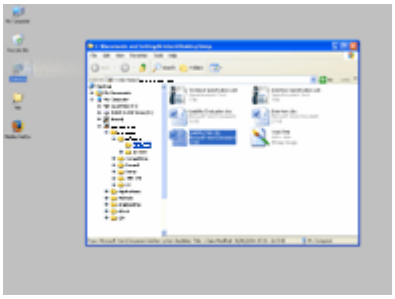
The Alfresco product is being developed incrementally and completely transparently. A technology preview will be released at the end of June 2005. This release will include a scalable repository with meta-data and data dictionary support, fulltext indexing and retrieval, rule-based processing, collaboration capabilities, and an extensible aspect framework that includes versioning, locking and transformation. The Alfresco user interface in this release will include an adaptable portal framework based upon JSR-168 portlets and JSR-127 Java Server Faces, including portlets for repository browsing with multiple views, wizards for administration, content handling, and in-line editing of web content. The Alfresco repository will also include a Microsoft Shared File System-compatible interface that can be use with file-based tools, support offline replication and allow the repository to act as a replacement for shared file stores.

The Alfresco product will be available for production use by the end of 2005. During the months following the technology preview, Alfresco will be incrementally integrating additional functionality toward creating a complete enterprise content management system and repository. Some of the following functionality intended for future releases are: a new advanced query language, web site management, replication and federation capabilities, and a BPEL4WS integrated workflow.

Alfresco Solutions

The Alfresco system will evolve significantly over the next few years, but we are focusing on enabling high-value applications that have, in the past, required expensive commercial systems. The Alfresco platform provides a modern content repository, pre-configured portlets that provide application components for common content operations and a set of standards-based APIs. Out of the box, configured or implemented on top of the platform, the Alfresco platform provides the basis for the following solutions.

J2EE-based Professional Portal Projects – Portal projects can be very expensive to implement with commercial software that includes portals. Alfresco content management can manage content for portals and provide search and categorization services for targeted delivery of content within portals. Alfresco implements its portal components using the JSR-168 standard to fit into open source portals such as JBoss Portal 2.0. Using open source, you can build an entire professional portal without the cost of commercial software.



Controlled Shared Drives – Due to compliance regulations and out-of-control content growth, organizations are seeking to replace shared network drives that have become the dumping ground of enterprises. Users have resisted content management implementations because they are too hard to use. The Alfresco system provides a shared drive interface for the storage of information augmented with automated rules to simplify the categorization and organization of information to improve searching. It facilitates sharing through portal interfaces and increase compliance through content control.

Content-enhanced Applications – ISVs and application developers who wish to enhance their applications, such as CRM or ERP systems, can provide content management and search services in their applications with a low-cost, royalty-free, super-fast content management system. The Alfresco system provides a web service interface so that no matter what language your application is implemented in, you can integrate enterprise-level content management capabilities. The Alfresco LGPL means that the license status of your code is not affected.

Compliance Applications - With regulations such as Sarbanes-Oxley, the Patriot Act and numerous financial and control rules, even the smallest organizations are affected by the need to control paperwork and documentation. As an open source enterprise content management system, Alfresco provides an affordable alternative for putting controls into place. With simple-to-use shared drive integration that requires no client install, Alfresco provides the easy way to get control of paperwork without adding a greater burden on employees.

Low-Cost, Open-Source Alternative to Microsoft SharePoint™ – If you have been considering using Microsoft SharePoint because it comes free with Windows 2003, think again. In order to use the capabilities of SharePoint to search and use content in a SharePoint repository, you need the added-cost Microsoft SharePoint Portal™. Alfresco provides desktop integration, portal access and simple-to-use interfaces and programming interfaces combined with integration to professional open source portal products. Alfresco gives you an all-encompassing portal and content management solution without any license fees.

Why Alfresco?

Alfresco was designed to solve the most pervasive and difficult problems in content management. Alfresco is perfect for developing enterprise and departmental portals, compliance applications or replacing uncontrolled shared file drives. Alfresco's architecture is modular and standards-based allowing you to use only the functionality you want or add new functionality as requirements grow. Alfresco's modularity and execution paths offer greater performance and scalability.

Easy to use – Alfresco has employed a user-centered design that has packaged easy-to-use portal components for managing and organizing content. Alfresco's integrations with file and content management standards makes authoring content as easy as "SaveAs".

Built by the people who invented content management – The Alfresco team led by John Newton has unrivaled experience in building enterprise scaleable content management systems. Our team of computer scientists and engineers, having built the leading Java Content Management systems of the 1990's, saw how the open source movement could help them deliver on their dream of letting content management become a tool for all.

Open source, open standards – Alfresco is open source and open standards, which means that you do not have to buy it to try it. Open standards means that you are not locked into any vendor, including us. We intend to make everything we do transparent and open.

Faster than existing content repositories – Our team has created a new architecture based on their experience of client-server and the first generation of web content management applications. Using Aspect-Oriented Programming techniques we literally shorten the running distance (in programming terms) making response time many times faster as well as more predictable as you add new users.

All the content capabilities and applications you will ever need – By making the open standards content repository open source and free to developers we transform the economics of building content management applications. The Alfresco developer network will rapidly become the most popular platform for content management applications for software vendors and enterprise programmers alike.

Much lower cost of deployment – Open source minimizes Alfresco's sales and marketing costs. This re-focuses Alfresco's funds and management effort on product development, engineering and quality which delivers a lower cost, better designed solution than that which closed source companies provide.

Adapts to your changing requirements – Alfresco's distributed, aspect-oriented architecture allows you to plug and play new security modules, storage options, workflow engines and more: it is completely customizable in every aspect possible. Alfresco will offer different deployment options based on your company priorities, IT support, and business needs.

Quality and experience – Alfresco offers a fast implementation out of the box content portal that enables you to start receiving the benefits within minutes of downloading the application. It is so intuitive it's as easy as 'SAVE AS' to have your users collaborating on content contributions in seconds.

Alfresco Product

Alfresco is the first open source enterprise-scale content management system that includes a modern content repository, an out-of-the-box portal framework for managing and using content designed to work with standard portals, and a groundbreaking Common Internet File System (CIFS) interface that provides Microsoft Windows file system compatibility. Alfresco has taken the lessons of building content management systems for the last 15 years and applied them to build an open source content management system that is easier to use, more scalable and more adaptable. The Alfresco system is developed using the latest Java technologies including JBoss 4.0, JBoss Portal 2.0, Spring 1.2, Hibernate 3.0, MyFaces 1.0, Lucene 1.4 and Java 1.5.

The first preview of this technology is available at the end of June 2005. This release demonstrates a new approach to content management, such as the ability to connect to the repository as though it were a shared drive in your network as well as access and control that content through an easy to use portal interface based upon specifications JSR-168 Portlet and JSR-127 Java Server Faces. By making the source code available, it will be possible to see our approach to Aspect-Oriented Programming as we use the popular Spring framework for introducing incremental functionality to content objects. This lean, modular approach to content management architecture is fast and adaptable and will allow us to add new functionality rapidly. Some of the first features available in this release include:

Modern Repository – The Alfresco repository is an enterprise-scale repository designed to be distributed, federated and scalable. The system is architected to use aspects and Aspect-Oriented Programming wherever possible. This makes it possible to streamline communication between components and to introduce new capabilities incrementally without re-coding. Aspects allow the collapsing of communication barriers, introduction of web services, simplification of security, transactions, caching and version control. Use of aspects is controlled using a flexible rules engine that can add new functionality to content automatically, such as classification, meta-data, versioning, locking and translation. Aspects make all facets of the repository configurable including the data model, versioning, process separation and security. Configurable aspects allow shorter, faster communication paths and controllable caching that make the repository extremely fast.

The capabilities of the Alfresco repository are those found commercial enterprise content management repositories.

- Aspect-oriented – including the ability to add new aspects
- Hierarchical folder structures
- Document types with standard document behaviors
- Metadata – with extensible types and complex properties
- Classifications – Browse or search classifications based upon global definitions
- Rule-driven processing of content to add or modify data or move content
- Automated content processing – including auto-classification, auto-indexing and workflow handling
- Global dictionary – Namespace driven dictionary allows hierarchies and standardization of definitions
- Open authentication – Configuration-driven to use enterprise standards for authentication, such as LDAP and Microsoft Active Directory™.
- Fulltext indexing and retrieval – Using the Lucene 1.4 engine and content transformation to search many different content types
- Team collaboration spaces
- Versioning
- Locking – a configurable aspect to add locking if necessary
- Content-streaming
- Configuration control – a central point of configuration for all Alfresco features

Standards-based Portal Components – The Alfresco system includes an out-of-the-box portal solution to simplify the development of enterprise and departmental portals. This solution includes a framework of portlets for interacting with the repository. The portlets are developed using the JSR-168 standard and integrate with any JSR-168 compatible portal, such as the JBoss 2.0 Portal. The Alfresco portlets are reusable and extensible to be used in portal applications. The portlets are developed using

the JSR-127 Java Server Faces standard, which is a tag-based interface for adding user interface capability simply without programming the user interface presentation. The Alfresco JSF tags can be used in the development of custom portlets. Some of the pre-packaged portlets that come with the Alfresco system are:

- Browsing content hierarchies with multiple, configurable views
- Wizards for administration and creation of content spaces
- Wizards for content handling and uploading
- Interfaces for version handling and version history
- Interfaces to manage properties of content
- Team collaboration spaces interfaces
- In-line editing of web content
- Desktop integration for seamless launching and saving of content
- Bread-crumb support for navigation
- Easy Navigation model
- A clipboard for saving content work and components
- Query interface to allow users to search and browse content

Shared File System Emulation – To provide easy access to the repository for end users, Alfresco has included emulation of the Common Internet File System standard. CIFS allows users to access the Alfresco repository as though they were accessing a shared drive, permitting off-line synchronization, drive mounting and access from any application. A rules-based engine provides automatic versioning, classification and control, eliminating the manual work associated with content management systems. This interface was developed by one of the Alfresco team who developed of JLan, the only Java-based CIFS engine, over the last seven years.

Application Programming Interfaces – The Alfresco system is designed to make programming of content management applications easy. As standards emerge, Alfresco will add these interfaces to the repository and make them interoperable with existing APIs. The main API is service-oriented to support web services, other languages such as Perl and PHP, and to provide a stateless, scalable application base. The web services and WebDAV interfaces will be available in the near future. Information on these is available in the preview release.

There will be multiple query models to support different types of searching and information access. The first supported in the preview release are the JCR/Xpath query specification and an extended version of the Lucene query language that allows easier mixing of meta-data and classification searches with full-text searching. We will also support a SQL-based query language in the near future.

Relational Database Support – The Alfresco system uses Hibernate 3.0 to support the mapping of its internal objects to the underlying database. The first database system supported is MySQL. In the future, we will support multiple databases as part of a simultaneous release.

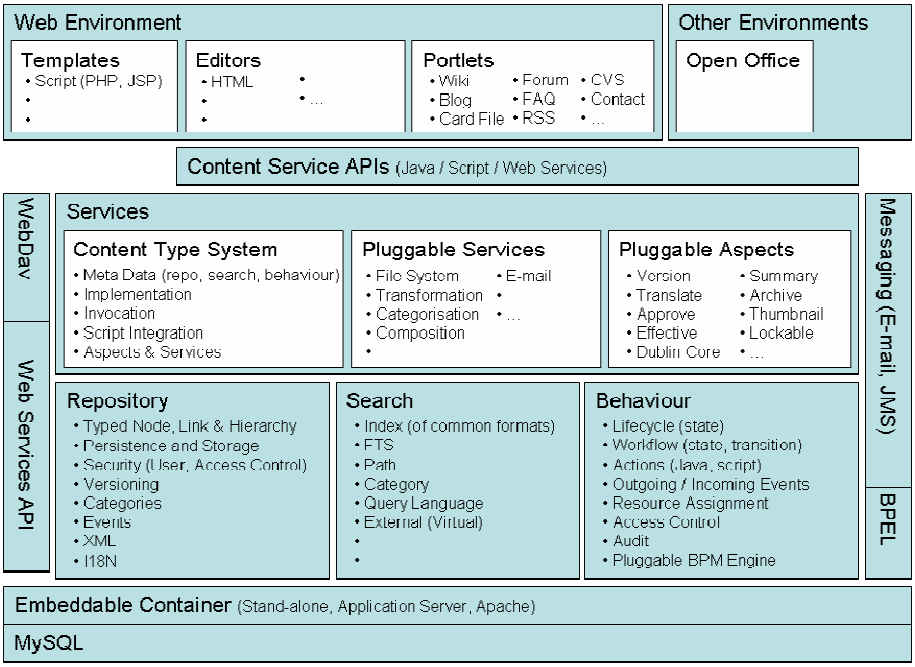
Alfresco Roadmap

The technology preview is only a first release of the Alfresco system. It is not a complete system, but a demonstration of what we are building and the approach that we are taking. Over the coming months we will be adding a lot of new functionality. We look forward to working with the community to prioritize the work ahead and to provide continuous feedback on how various components are performing.

- Glossary Support – The glossary is the data dictionary for a space and all of its folders. Contained in the glossary are types, rules, policies, categories and aspects. The portal will support the specification of all of these although many are already supported by the repository.
- Types Administration – Present a simple user interface for users to create types, aspects, features and categories. Currently these are only supported through modeling tools.
- Personalization – The portlets in the portal will support preferences for how items are displayed as well as remembering what information and content a user is interested in.
- Dashboard view – Provide the user with an easy-to-configure process view on a space including a micro-site view of the space for outside users.

- Security and User Management – Using role-based access control, provide a complete security model and management of access control and user profiles. Indication of the capabilities can be found in http://acegisecurity.sourceforge.net/.
- Automated Services – Using the aspect-oriented capabilities of the repository, add the ability to automatically classify, summarize and transform content, along with automated creation of associations between content objects and business objects. These services are pluggable and possible for others to override.
- Complex Workflow – Using BPEL, support complex business processes for managing compliance and other content processes.
- WebDAV – Complete WebDAV and DeltaV support.
- Aspect Management – Provide a user interface and infrastructure for the management and control of aspects to be introduced into content types. This will allow developers to plug in their own aspects and have the portal and repository identify this capability.
- Scripting Language Support – The ability to create aspects (e.g. classification) using languages such as PHP or Perl. Provide access to the Alfresco object model to those languages as well as callback mechanisms in Java.
- Advanced Search User Interface – The Alfresco query language is very powerful with the ability to combine complex expressions with powerful categorization techniques. The user interface will provide users with a simple interface to exploit these capabilities.
- Federated Search – The ability to search multiple repositories in a single search and to specify complex search scopes.
- Fail-safe Content Services – The ability to back-up, restore, archive and purge content in combination with administration interfaces.
- Associations and Link Management – Provide a user interface for the creation of named, well-known associations between various content objects (e.g. translation-of, copied-from, references) and business objects.
- Complex Branch and Merge – The ability to branch and merge entire trees of content with complete transaction control (either it all happens or it doesn't).
- Distributed Synchronization – Add the ability to synchronize two repositories that are physically separated and resolve any conflicts. Uses the branch and merge capability.
- Thumbnails – Provide a thumbnail view for image galleries including the transformation of images to smaller images for a light-box view of images.

Architecture Diagram



Modern Architecture and Standards Support

The Alfresco system uses the latest technologies available for open source. These are primarily Java technologies, but we intend to make the Alfresco system available to many other languages. These technologies are on the leading edge of performance and power, but we have chosen them for their reliability and quality as well. Many of these are also implementations of standards, reference implementation of standards or de facto standards in their own right.

- Java 1.5 including annotations and generics - http://java.sun.com/j2se/1.5.0/docs/guide/language/
- JSR-168 Java Portlet Integration - www.jcp.org/en/jsr/detail?id=168
- Spring 1.2 Aspect-Oriented Framework – www.springframework.org
- ACEGI Aspect-Oriented Security Framework – http://acegisecurity.sourceforge.net/
- MyFaces 1.0.9 JSF Implementation – www.myfaces.org
- Hibernate 3.0 ORM Persistence – www.hibernate.org
- Lucene 1.4 Text Search Engine – http://lucene.apache.org/
- CIFS/SMB Microsoft File Share Protocol – www.microsoft.com/mind/1196/cifs.asp
- JLAN – www.starlasoft.com
- WebDAV / DeltaV – www.webdav.org
- JBoss Application Server 4.0 – www.jboss.org
- JBoss Portal 2.0 – www.jboss.org/products/jbossportal
- POI File format conversion – http://jakarta.apache.org/poi
- PDFBox – PDF conversion – www.pdfbox.org
- Open Office 2.0 – www.openoffice.org
- JSR-223 Java Language Integration - www.jcp.org/en/jsr/detail?id=223
- MySQL – www.mysql.org

Content Management Standards

A rising tide of content management applications has not yet appeared comparable to that found in relational databases in the late 80's and early 90's. Prior to that time, RDBMS vendors all had proprietary APIs. With the publishing of SQL89 and some level of interoperability, the client-server application industry was launched (including content management). With the creation of SQL92, a usable set of functionality and common language binding driven primarily by Microsoft's ODBC, client-server had yielded a huge increase of applications, SQL-literate developers, the basis for the enterprise application industry, and laid the foundation for the World Wide Web and internet services. That client-server revolution required three elements:

- A common query and data manipulation language
- A common set of language bindings
- A common model for managing queries, results sets, updates, transactions, triggers, procedures, security, data definition and data dictionary elements

Once these were in place, there was a sufficient basis to create data-intensive applications. Legions of developers were trained by universities and corporations to create these applications. The whole database industry continued on an exponential growth curve for the next decade. The creation of a more comprehensive eco-system did not really level the playing field, but opened up the dominance of Oracle, IBM and Microsoft with a much larger market to play in.

A similar opportunity is available now to create a similar revolution in content-intensive applications. An opportunity to create an environment for developers to build applications with confidence and with a common development paradigm that will promote education programs and a whole ecosystem of developer tools and toolkits to emerge. A standard based upon web services can take advantage of the shift toward interoperability, loose coupling and platform and language independence.

It is reasonable to expect the possibility of millions of developers of content management applications in the next 5 years. The following chart indicates what has been accomplished in other areas of development in the last decade or so just in North America:

Do you spend any development time on any of the following TODAY:	Population	Percent of Population
Database	2,415,431	86.5%
Web / Internet	2,290,126	82.0%
Web services (Service Oriented Architecture)	1,475,858	52.9%
Open Source	1,147,490	41.1%
Embedded systems	695,093	24.9%
North American Developer Population Study 2004: © 2004 Evans Data Corp		

The question is whether JSR-170 creates that opportunity or hinders it. JSR-170 has taken the approach to try to define content management, rather than try to find common ground between the different existing content management systems. Therefore, the common model that was created for SQL doesn't really exist yet for content management. Also, the fact that there is no data model or data definition language, as existed in SQL, means that developers can make no assumptions on what information is being acted upon, which is a critical element in creating packaged applications. There is also a demand for content management interfaces in other than Java. We expect that a web services-based standard that is more inclusive, rather than prescriptive, will emerge soon that will either compete or complement JSR-170.

JSR-170 has some interesting concepts, many of which can be found in the Alfresco system, such as the node and type interface and the JCR query language. However, we felt that the interface itself was inappropriate for building scalable, adaptable applications. Internally, we have taken a more service-oriented approach that allows us to make scalable applications and to keep the connection stateless – absolutely critical in building enterprise systems. Our services-oriented approach also allows us to

introduce new Aspects through Aspect-Oriented Programming. This services-based interface does not preclude a JCR interface.

We will be taking our cue from the community on providing a JSR-170 interface. If there is demand for it, the interface will be created. We are actually in a better position to support it than most content management systems. We already support the JCR query language, but we don't believe that there will be a huge uptake for an XML and XPath-based query language. Most content management is SQL and relationally oriented.

One standard that is more mature and widely adopted is WebDAV. WebDAV was initially developed in the mid-1990s to simplify the versioning of content over the web. WebDAV provides navigation and locking models and a rudimentary meta-data model. The DeltaV extensions to WebDAV provide a more complete versioning model. File explorers provide interfaces to WebDAV for document navigation, although this interface is not as comprehensive as the Alfresco CIFS support. Microsoft Office, Open Office and most web authoring tools support the WebDAV interface. WebDAV does not yet support a typing mechanism or query language.

Aspect-Oriented Content Management

Aspect-Oriented Programming (AOP) was invented in the mid-1990s at Xerox PARC, the same place where Object-Oriented Programming (OOP) was invented. Although OOP provided simplicity and re-use of code over previous generations of programming techniques, the researchers at PARC realized that some functions were being duplicated in many classes because of the inability of OO languages to introduce these common functions dynamically into other classes. Because of common routines being used in many classes, programmers would add all of these utility functions into over-burdened super-classes or "Überclasses" that became slow and overweight. This breaks the principles of modularity that OOP was trying to espouse.

Thus was born the notion of "Cross Cutting Concerns" that allowed routines of common utility to cut across many different classes. These crosscutting units became known as Aspects and the discipline of using them became Aspect-Oriented Programming. Aspects modularized behaviors and made reuse across different classes much easier. Special languages were developed to support the notion of Aspects, the first of these being AspectJ from Xerox PARC. Although AOP has been developing for nearly a decade, it has only been usable for the last couple of years. Java, with the reflective nature of its runtime system, has been the primary language to which aspects have been added. Rod Johnson, author of J2EE without EJB, built an Aspect Oriented Framework called the Spring Framework, which has gained a lot of popularity with developers with its ability to incrementally add new functionality and switch in different implementations of services. The Alfresco™ system uses Spring as the foundation of its Aspect-Oriented approach.

Content Management is a perfect example of a class of systems that can use AOP Content Management is used for many different types of applications: web content management, document management, records management and image management. These applications use a common set of functionality that includes such capabilities as versioning or content storage management. Some behavior, such as locking, may be used by some, but not all of these applications. Meta-data, such as the industry standard Dublin Core set of meta-data, might be useful for document or records management, but not necessarily for image or media management. Some functionality may be optional and useful for some applications, such as categorization and auto-categorization, but add too much of a performance drag for high-performance content management applications such as some types of simple records management. In a purely object-oriented approach, all this functionality is made available to all types of content that might use it, adding a space and time burden to applications that do not need it.

In the AOP approach, a system uses a pattern of development called Dependency Injection. Dependency Injection is to take method on an interface, such as fetch or save, and to inject a handler that does additional processing either before, after or instead of the standard call. For instance, to add a read lock on a content object, we could make a lock call before a fetch and an unlock call after a save or release call. The calling application should not be aware whether a read lock has been added or not, so the dependency injection should be automatic and not require any additional programming. The Spring Framework provides us with the ability to wire in these automatic behaviors as a configuration exercise rather than add code at all the places we need to use it. Transaction control is an example of a capability that can be added in without programming and providing us with robust control of content. In addition, we use a number of essential services by merely configuring the service aspects in, such as: transactions, n-tier partitioning, caching, web service calls, authentication, and service call authorization.

The first advantage of the AOP approach is performance. The Alfresco system does not use code or manage data that are not needed. This eliminates a lot of unnecessary data transferring and persistence. It also lessens the amount code that needs to be executed access or save content. We can also configure remoting calls or remove them as necessary. This means that we do not need a three-tier architecture if it is not necessary for applications such as web applications or portals. If a portal runs in the same JVM as the content repository, it is likely to run much faster without compromising security. However, if applications require a multi-tier architecture for consolidation or process partitioning purposes, the Alfresco system can layer its architecture simply by configuration.

The second advantage is the adaptability of the content repository. We can add new capabilities such as new automatic processing or new access controls without recoding the repository. We use dependency injection to add it at well know locations such as at content access and save points. This same mechanism also makes it possible for us to add new web services without a great deal of effort. Alfresco has also added the ability to drive these aspects using a rule-base and making them meta-data driven. New aspects can be introduced to a content object simply by changing its meta-data. Since there are clear boundaries and introduction mechanisms for aspects, there is no reason why these aspects need to be implemented in Java. The Alfresco system will be introducing the ability to add new aspects using PHP, Perl and other popular languages.

The Alfresco architecture has been organized into service layers to take maximum advantage of this Aspect-Oriented approach. We will be adding new aspects on a regular basis that should not require changing the underlying system. Some of the early aspects include categorization, versioning, transformation and auto-classification. Our intention is to encourage others to create aspects as well and use the open source community to make it available to all Alfresco users. The Alfresco system will allow users to add new content capabilities dynamically without changing their entire infrastructure. This will allow users to adapt to new business requirements and rely on a stable, performant basis for content management.

Why Open Source?

We have been involved in enterprise software for a long time. In that time, we have seen many business models come and go. Some have come and gone very quickly. However, the tried and true model that we have known for so long, the old model of expensive sales force, big ticket sales and a distribution system that convinces customers that they want software, is now dead. The evidence is everywhere as software companies continue to struggle. Except for open source…

Ingres, one of the first relational database systems where co-founder John Newton was one of the original engineers, used some of the first open source software available. In fact, the Ingres system ran on the first Unix system outside of Bell Labs. US government funding for the Ingres project required that the University of California make the source code freely available except for the cost of production – making a 9-track tape. Ingres was widely distributed to the academic and military community and actually pulled along what would become the Berkeley Software Distribution of the Unix operating system. This was arguably the start of the open source movement.

However, in the early 1980's, this was not considered an effective model for commercial software distribution. The cost of development was extremely high and required license revenue to support it. People could not possibly know that they required a relational database, so they needed a sales force to educate them that they needed it. Technical support required a long chain of people writing things down and tracking issues. In other words, the industry needed a big commercial organization to fulfill unmet demand. These were times when the internet ran on 1200 baud modems, a megabyte of memory cost over $10,000, a 300 Megabyte drive was the size of a washing machine, and amazingly, the right for a single person to use a relational database was over $2000.

That kind of money was addictive to the commercial companies. A 500-seat deal was a major cause for celebration since it was worth a million dollars. Champagne flowed regularly. Larry Ellison's magnificent houses were not built with inherited money. Content management in the early days was similar, but starting at a more modest $500 per seat. Even today, it can be $200 per seat in even big enterprise deals.

The Internet and Moore's law have since intervened. Customers no longer require a sales force to either discover or be convinced that they need any particular software. In fact, they do not like being told that they need software, especially if they are cutting costs elsewhere. Customers do not need a long and complex supply chain to provide them with software when they can download complete solutions themselves from their desk. More importantly, the chain of customer requirements to ultimate creation of software does not require armies of people to record ideas, wants and complaints and to translate them for software developers to turn them into software fixes and releases.

The enterprise software industry must now face the fact that the world has changed. Customers have a choice and they will exercise it. The largest players are protected by the brand that they have developed over the years. The smaller players must adapt or disappear. Customers are just not willing to accept the old model and enterprise content management is no exception. It is inevitable that new forces change the nature of an industry in the midst of technological change, such as easier distribution and cheaper means of production and use.

Harvard Professor Clayton Christensen in his book, *The Innovator's Dilemma*, noted that mature industries hit a point where functionality is good enough. What customers want is convenience, reliability and low cost. The automobile industry has hit this point, so has the airline industry. The software industry is not immune to this cycle. Open source has become the new means of production and innovation that puts pressure on all, but the largest manufacturers.

Enterprise Content Management has had a good run over the last decade. Usage has grown dramatically, but customers are not willing to spend as much as they had in the past. A recent survey by the Institute for Information Architecture asked what the barriers to adoption to content management were. The survey listed the high cost of commercial software, the flexibility of systems and ease of use. This is a clear indication that the Innovator's Dilemma is in effect.

Open source as a distribution model and an innovation model is the natural evolution of the enterprise software industry. It establishes a very efficient sales, marketing and distribution mechanism that allows customers to discover software that they need for themselves. They can access, try and test the software before they invest in it. They can also establish a much more efficient conversation between users and developers that introduces changes, fixes and innovations in a fraction of the time. There is just no way that the old model can compete.

Alfresco Software, Inc. Licensing Policies

At Alfresco, we have made a decision to participate in the open source movement, and so we want the software development community to understand what we intend to contribute to the community, and how we hope the community will work with us. This page is intended to help you understand our mission and our intention in working with the community, and the intellectual property policies we have developed to enable our goals.

We license our product under LGPL.

Alfresco has developed a content repository system designed to improve the collaboration, control and compliance of business documents and associated business processes. Our mission is to be the custodian of this repository standard, and to propagate this standard through the industry. We want to enable any developer to use the Alfresco repository model to maintain and manage content, whether that developer is an open source developer or a commercial developer. For that reason, we have chosen the LGPL as our license. The LGPL enables the free software community because it allows developers to re-license our code under the GPL, if they choose to do so. The LGPL also enables the commercial developers by allowing developers to integrate our software into commercial products, while maintaining dynamically linked portions of those products under commercial licensing terms.

We license our product under other licenses upon request.

If you are a commercial developer, and would prefer to receive our product under a commercial license agreement, or a different free software license (such as CDDL, CPL, or Mozilla), we are happy to discuss this with you on a case-by-case basis. Because we have made our software available under LGPL, and we hope that will be acceptable to most commercial developers, we have not posted a standard commercial license on this page.

If you are not sure whether LGPL is right for you, you can always test our software under the LGPL and inspect the source code before you contact us about purchasing a commercial license.

Our commercial products include support and services.

While you may download our software product under LGPL terms free of charge, we also offer commercial products. At this time, our commercial product is identical to our free software product, but the commercial product includes services such as support, maintenance, professional services, integration, or the like. To purchase these services, customers enter into agreements with us describing our services and the fees we charge for them.

We may change our licensing policies.

Over time, we may need to streamline our licensing policies to respond to developer needs and to ensure the stability and commercial viability of our products. Any changes to our policy will be posted on this page. But of course, if we change our policy, and you have received a license to our code under LGPL, we will not terminate the rights you have already been granted. Any changes will only apply going forward to new releases of our code.

We welcome contributions to our source tree.

We welcome you to submit fixes, improvements, or other contributions to our code base. If you release any changes under LGPL, as may be required by that agreement, we would appreciate your letting us know, so we can evaluate whether they would benefit the community as a whole. Of course, to maintain the integrity of our products, we will need to exercise discretion as to whether contributions you offer are included in our official source tree.

If you make contributions, and we accept them into our source tree, we want to make sure we have the flexibility to use those contributions under LGPL, other terms we may use in the future, or commercial terms. Of course, nothing will prevent you from making your contributions available under any open source licensing terms you choose. We simply ask that you allow us this flexibility, so our intellectual property position does not become too complicated. As a company, we have made a decision to make our intellectual property available to the community, so we want to make sure our resources are

devoted to making our products the best they can be, rather than fighting legal battles over contributions. Therefore, if you wish to contribute code, we ask that you do so under our standard contribution agreement [link here], which gives us this flexibility.

Our trademarks protect us, and the community

While the LGPL gives you freedom to change our software, we need to make sure this does not conflict with our responsibility to maintain our trademarks. Our trademarks are important to us as a company, of course, but also to the consuming public, which will associate our trademarks with the quality and reputation of our business. We need to strictly enforce our trademark rights to keep them valid, and to protect the public. This portion of our policy describes what you can do with our trademarks, and when you must seek our permission to use them.

Do not remove or alter our trademarks.

If you use or redistribute our binaries, you must not remove our trademarks from them. When you use our software in the exact form we provide it, we allow – and encourage – you to state that the software is Alfresco software, as long as you include the notices below. However, if you are distributing unmodified copies over a Web site, please do this by providing a link directly to our site, rather than offering it on your own server.

If you are redistributing unmodified versions of our software, you should use a notice like the following in a prominent place on the packaging and about boxes for your software: "[Your product name] includes the Alfresco **[content repository software]."**

All this is necessary to be sure that you are using our trademarks correctly to identify us as the source of our software. However, if you have modified our software, you should instead include a notice (in a prominent place on the packaging and about boxes for your software) stating: "[This product includes software used under license from Alfresco Software, Inc., but it is not an Alfresco product and has not been tested, endorsed, or approved by Alfresco Software, Inc. or any of its affiliates]." In this case, you should remove any of our logos from the software. To make this easier for you, our source downloads – as opposed to our binaries – do not include our logos.

In all cases where you use our trademarks – including our name and our logos – you should also follow the guidelines below.

Do not use our trademarks in a way that states or suggests that we sponsor, endorse, or are otherwise affiliated with your project or product, without a formal license from us. We will consider such licenses on a case-by-case basis, but we may not grant you such a license. No formal license agreement is necessary if you wish to make purely informational reference to our trademarks (such as in a product review), but even informational references should follow our trademark guidelines. You must also follow these guidelines if you link to [www.alfresco.org]. However, you must make it clear that you are not endorsed by or affiliated with us, but are simply linking to our site.

Trademark Use Guidelines

The first or most prominent mention of our trademark Alfresco should be accompanied by the letters "TM" in superscript and an asterisk immediately to the right, ALFRESCO™*. Nearby in typeface large enough to read, you should include an asterisk next to the following: "ALFRESCO™ is a trademark of Alfresco Software, Inc., and is used under license."

We may register our trademarks in the future, at which time it would be appropriate to replace the "TM" symbol with the registered trademark symbol (®). You do not need to use the TM symbol with every use of our name, only on the first and most prominent reference.

Trademarks should only be used as adjectives followed by a generic noun – for instance, ALFRESCO™ software. Our trademark should never be used as a noun, a verb, or in a possessive or plural form – it would be improper to refer to say "Alfrescos" or "ALFRESCO it." Never vary the spelling and syntax of our trademarks. Please use no abbreviations or acronyms.

Contact

John Newton
Alfresco, Inc.

Top Floor
Park House
Park Street
Maidenhead, Berks SL6 1SL
United Kingdom

Phone: +44 1628 860 639
E-mail: john.newton@alfresco.org