Toward Cloud-based Collaboration Services

David Banks
Hewlett-Packard Labs

John S. Erickson
Hewlett Packard Labs

Michael Rhodes
Hewlett Packard Labs

Abstract

In this paper, we argue increased outsourcing of non-core competencies will create demand for cloud-based platforms to address the need for content-centered collaboration between organizations. We introduce a prototype created to evaluate the suitability of current enterprise content management (ECM) technologies for this type of platform. Following from this work, we highlight several areas where we feel current platforms are inadequate and new approaches are required, particularly in multi-tenancy and user-customization

1. Introduction

As we approach the end of the first decade of the 21st century, we are witnessing a disruptive change in the provisioning of information technology: the advent of cloud computing. For most organizations, information technology is not a core competence. Until recently, their only option was to retain IT specialists on-premise, but now alternatives from the likes of Google, Amazon and Salesforce.com are becoming increasingly viable. Accordingly, out-sourced IT is now an option for all sizes of company.

This coincides with businesses seeking to operate efficiently in a global marketplace by outsourcing non-core competencies. As businesses choose to excel in a single area and partner for the rest, collaboration across organizational boundaries becomes a core part of product development. Traditional Enterprise Content Management (ECM) software has not kept up, leaving people collaborating via email—the lowest common denominator.

In response to these trends, we envision a generation of cloud-based collaboration platforms emerging to address the needs of content-centered collaboration between businesses. Although superficially similar to the best of today's ECM systems, these platforms will operate on a massive scale, simultaneously supporting thousands of organizations and millions of users. The Fractal research program [1] in HP Labs aims to design and deploy such a platform.

As part of the Fractal research program, we have built a prototype of our envisioned platform using a leading ECM system. We found several shortcomings in this system as we tried to apply it in this new context. This paper, therefore, presents where we believe research is required to bring ECM to the cloud.

The paper is structured as follows. Section 2 presents the prototype, describing our goals and giving an overview of its functionality. This provides the framework for Section 3, where we discuss our rationale for why current leading ECM systems are not a suitable base for a cloud-based, highly user-customizable collaboration platform. In Section 4, we highlight related work in this area. Finally, in Section 5 we present suggestions for future research directions.

2. Fractal Conceptual Prototype

We built the Fractal prototype to help us understand how suited current ECM technologies are for realizing Fractal's vision of a multi-tenanted, highly user-customizable collaboration service.

2.1 Key Features

We wished to demonstrate the following features of Fractal in the prototype:

Content Spaces: hosted workspaces which bring together people, content, collaborative tools, and customizable active behaviors.

Active behaviors: a way for end users to define functional extensions operating within the context of a content space involving content, metadata, automated processing services and tasks carried out by other users. An active behavior may be manually invoked as needed, or it may be automatically triggered by a change to a content space or the passing of time. An invocation may involve a single content object or many objects in parallel. Their complexity ranges from creating up-to-date PDF versions of documents as they are modified, to running workflows that automatically collate information from several collaborating organizations into a single document.

Agile configuration: the service must be lightweight, low-touch and customizable by end users without IT involvement.

Open and extensible by third parties: the platform should provide open APIs that enable third parties to develop customizations and extensions that may be obtained or purchased through a marketplace by users.

2.2 Technical Approach

We evaluated several technologies as a starting point for the prototype, including Joomla, Drupal, Alfresco, Liferay, TikiWiki and SharePoint. Our selection criteria included: strong document management features; embedded workflow; social capabilities (blogs, wikis, tagging); and user interface qualities similar to those we envisioned for Fractal. We selected Alfresco's new Share technology [2] because it satisfied these criteria.

2.3 Conceptual Prototype Overview

In this section, we give a brief overview of our prototype. For further details, we refer the reader to a series of short demo videos we have produced based on the prototype [3].

Our prototype centered on a content space supporting a collaborative pharmaceutical research project called *UTS-Alpha*. In addition to the research content in a document library, the space had a customizable set of collaboration tools (wikis, blogs and so on) with a configurable default view, the dashboard, which summarized the members and content of the space.

To demonstrate the key Fractal concepts of user-customizability and developer extensibility, we built the Fractal Extensions Marketplace where a user chooses functionality to add to their content space. To provide functionality, developers publish extensions in the marketplace. The marketplace provides a rich set of search and browse capabilities to help users find the functionality they need.

We used the Simile Exhibit faceted browser [4] to build the marketplace. Exhibit gave us several views—tabular, timeline, etc.—each of which allowed the user to narrow down their search using facets derived from descriptive, commercial and social metadata bound to listed extensions.

Adopting an extension to a user's content space requires a single click of an install button, not unlike adopting Gadgets for the iGoogle homepage.

For the prototype, we created an extension allowing the user to add functionality to the *UTS-Alpha* content space in the form of a workflow to coordinate production of a monthly report by the members of the space. This extension added a dashboard component and a jBPM workflow to orchestrate the steps necessary to build the report, including emailing members when they needed to write or review a section and archiving the complete report into the document library.

Although we successfully used Alfresco Share to rapidly prototype a functioning system with many of the end-user characteristics we envision for Fractal, our experiences building the prototype convinced us neither Alfresco nor any of the other leading ECM platforms provide a suitable base for a multi-tenanted, cloud-scale collaboration platform. The next section discusses why we feel this way.

3. Technical Challenges

In this section, we discuss where current ECM platforms have shortcomings when building cloud-based collaboration platforms. Though our discussion draws specific examples from our experience with Alfresco, we believe our conclusions apply to current ECM software in general. The issues fall into two broad topics: multi-tenancy and extensibility.

3.1 Large-Scale Multi-Tenancy

Most cloud-based services are inherently multi-tenanted; this arrangement currently seems the most efficient way to deal with the scale required to run services at this scale.

In general, the definition of multi-tenanted software is that it gives each customer the impression they have their own instance of the software, whilst in reality services share a single or a few large instances between many users. Customers see their data and settings as isolated from other customers.

This hard segregation of customer data makes it impossible for individual customers to share their data. As organizations are likely different customers from the point of view of the service provider, problems arise when applying this definition to a service intended to facilitate collaboration between organizations.

The multi-tenancy paradigm offered by a cloud-based collaboration needs to support several usage patterns. A content space may have a single user and so be a private space. It might instead have multiple users from the same company, and thus be a collaborative company space. Finally, it may have users from different organizations, and therefore be a collaborative space for inter-organizational activities. The latter two of these patterns could well have participants from different customers working together.

We therefore argue "tenant" and "customer" are not synonymous in this environment. In a multi-tenanted collaboration service, a tenant is instead a collection of distinct, collaborative activities, people and related content—a *content space* in Fractal's terminology. We use this definition of tenant in the discussion in this section.

3.1.1 Data Isolation

Data isolation ensures adequate segregation of data to prevent unauthorized access by tenants to each other's

data. In Alfresco Share, all content spaces (*sites* in Alfresco) are persisted to a single shared store. This store is backed by a relational database, together with a file system that contains content and a single Lucene index. The store provides access control, but otherwise data separation is an application level responsibility. Cloud-based ECM requires stronger data isolation for two reasons.

Firstly, the single Lucene index shared by all content spaces causes queries in any content space to slow down as the number of content spaces increases—regardless of the amount of data present in the content space being queried.

Secondly, files representing content from different content spaces reside in the same file system directories. This makes it impossible to perform efficient backups of individual tenant data to different media at the file-system level—that is, without invoking repository operations. Not only are there legal implications to this, but it also prevents the service provider offering tenants copies of their own backup media.

3.1.2 Application Isolation

Like data, customizations or functional extensions to one content space should not be visible by default to other content spaces. Furthermore, if a user adopts an application into one content space, parts of that application—dashlets, for example—should not be visible to users of other content spaces.

In Alfresco Share, developers implement applications as web scripts, written in a combination of server-side Javascript and Freemarker templates. However, all content spaces (*sites* in Alfresco) share a common search path for web scripts, so when a uses introduces a new application into their content space it is actually available to all content spaces. Users can also extend the functionality of Alfresco using custom jBPM workflows; these are also deployed globally and suffer the same lack of isolation.

Managing functional extensions as data objects within content spaces rather than a separate global space, as is currently the case in Alfresco, is required for application isolation.

3.1.3 Performance Isolation

A third type of isolation ensures resource-intensive activity in one content space does not affect the use of other content spaces. This is one of the hardest challenges when designing a multi-tenanted service because it conflicts with the goal of reducing service costs by sharing resources between tenants. In general, a multi-tenanted service should adopt several approaches to minimizing the impact of tenants on each other.

First, the service should track resource usage on a per-tenant basis. Resource usage typically includes storage, I/O bandwidth, CPU usage, and possibly memory usage. Such tracking enables identification of resource intensive tenants. It is also worth tracking resource usage against other dimensions, such as per user, organization, and application. Application resource tracking allows blocking or throttling of poorly written applications until they are improved.

Second, tenants should be charged based on resources consumed. This form of pricing (as opposed to a flat rate or fixed subscription) serves as a form of feedback to make users sensitive to what they are doing. All existing large-scale cloud platforms (Amazon Web Services, Google App Engine, etc.) use some form of resource-based pricing.

Third, the service should dynamically load balance tenants across hardware resources. Usage patterns are likely to be bursty and there will be times when resources are over-allocated causing hot spots to develop. Dynamically altering resources assigned to tenants could minimize the impact of such hot spots. In extreme circumstances, the service can throttle a tenant's resources. This is a last resort because repeatedly throttling a tenant is likely to discourage future use of the service.

Current ECM platforms, designed for use within a single organization, lack this type of fine-grained monitoring, management and billing infrastructure necessary to support these approaches.

3.1.4 Tolerance of Hardware Failures

In a cloud-scale service provisioned across thousands of servers, disk and server failures occur routinely and must not result in loss of service. In addition, continuous hardware upgrades must not interrupt the service to any tenants.

ECM platforms use a variety of techniques to support high-availability deployments. In Alfresco, servers can be clustered and share state using a transactional object cache. A single database is shared between servers, which must itself be clustered. Indexes are maintained locally, loosely synchronized to the object cache. Finally, content either is stored on a single shared file system or on local file systems replicated between servers [5].

This approach to high availability is expensive in terms of hardware, software licenses and operational costs. It also does not scale to a very large number of nodes. Following this model, a cloud-scale service capable of

supporting thousands of tenants would require many independent clusters (pods), shifting the problem of load balancing to a different level rather than solving it.

3.1.5 Per-Tenant Levels of Service

In a multi-tenanted service, different tenants may require—and be willing to pay for—different levels of service. For example, one tenant might place a premium on storing their data within certain jurisdictions. Another may highly value low latency access to their data. Providing these features requires the platform to support per-tenant data placement, redundancy and replication policies.

Current ECM platforms support policies such as these at the deployment level. Therefore, to support differing policies, a deployment per tenant is required. An instance per tenant is unlikely to be a scalable business model because of inefficiencies, not least the mechanisms used to provide high availability discussed above. Therefore, support of per-tenant levels of service within an individual deployment is required for ECM platforms to support multi-tenanted services.

3.2 Extensibility

The second major area of Fractal research centers on allowing users to customize their content spaces using functionality created by developers and other interested users. This research direction stems from our belief a cloud-based service can uniquely allow users and developers to create, share and—if desired—sell pieces of functionality.

We seek to explore a sustainable paradigm where users are able to conceive and create new functionality, leveraging social networks for discovering and propagating their wares. Achieving this requires not merely an accessible development model, but a tractable means for sharing, practical mechanisms for installation, and assurances of safe execution.

Today none of this is available close to the user. In this section, we discuss where research is required to enable this model.

3.2.1 Ease of Extension by Developers

A number of factors make a platform attractive to developers, including accurate documentation; a familiar programming language; well-designed, stable programming interfaces; effective frameworks for testing; and a solid development environment.

We found extending Alfresco to be complex, requiring us to independently modify several different aspects of the platform using several different languages and tools. Deploying our extensions involved uploading artifacts to various folders on our Alfresco server, making uploading in a single package troublesome; our Extensions Marketplace required extensive knowledge of Alfresco's inner workings to enable single-click deployment of the *Progress Report* active behaviour.

We also note that not many ECM companies have set out to create platforms as open as Alfesco. In addition, few commercial ECM platforms are sufficiently widely deployed to have attracted a large following of third party developers.

3.2.2 Ease of Customization by End Users

With Fractal, we want to empower ordinary users to tailor content spaces to their needs. We want their customizations to extend beyond simply adopting applications written by professional developers; rather, we want to create an environment where end users are able to author their own extensions that precisely meet their needs and, if appropriate, share these with the broader community.

Many ECM systems embed simple scripting and workflow capabilities that, in theory, provide an easy route to authoring simple extensions. Alfresco, like many other ECM systems, embeds the JBoss jBPM workflow engine to allow custom workflows to be developed. In our prototype we evaluated the suitability of this environment for non-technical end users. Unfortunately, our results were not positive.

Even for experienced software developers, implementing the jBPM workflow for the *Progress Report* application described in Section 2.3 was time consuming. We found several sources of complexity: users first needed to find a jBPM design environment as one was not part of Alfresco; the Eclipse-based graphical editor only gave a partial view of the workflow, requiring users write actions in code; users needed an understanding of concurrent programming concepts, such as fork and join; users needed to make XML configuration changes across the platform to support the workflow.

This complexity needs eliminating if non-technical users are to have a chance at authoring their own custom active behaviors.

A core value we see in active behaviors is allowing non-technical users to create actions that respond to changes in their content in novel ways. This is impossible in Alfresco, but we believe essential in allowing people to build compelling, timesaving functionality around their content. How to provide simple, effective means for users to produce such reactive functionality around groups of content is a specific focus area for our work.

4. Related Work

Focusing on the scientific domain, myExperiment [6] provides compelling confirmation that users can successfully author and share complex workflows, given the right tools. In the myExperiment "virtual research environment," participants are able to share, extend and even execute scientific workflows and share data sets with fellow researchers and developers from across domains of interest. We envision a similar ecosystem of development and sharing in Fractal for both user defined behaviors and developer created extensions.

The Ning social network platform [7] demonstrates how easy it can be for users to create their own customized spaces. The key difference between Ning and Fractal is that Ning is consumer focused and does not extend its powerful user configurability to document management and workflow features.

Several cloud application platforms have recently emerged that free developers from concerns over how the infrastructure supporting their applications will scale if they are successful. The leading examples are Microsoft Azure [8], Google App Engine [9] and Salesforce's Force.com [10]. The Force.com platform is especially compelling, as they have fully recognized the importance designing specifically for multi-tenancy, using a metadata-driven approach.

The Storage and Information Management Platforms Lab (SIMPL) at HP Labs has recently presented work in which they recognize the tradeoffs between consistency and availability, and provide fine-grained control of this balance at write-time [13]. We see this as a promising step toward enabling tenant-level control of similar tradeoffs in a multi-tenanted environment, control not available today.

In the area of ease of extension for end users, there are a number of works discussing ways to present BPEL service composition workflows within a graphical authoring environment [11][12]. These provide solid foundations for the environments we require for enabling wide-ranging user composition of services during extension authoring.

5. Conclusions and Future Work

In this paper, we argued for a coming need for cloud-based, highly user-customizable collaboration platforms. We overviewed the Fractal project at HP Labs that aims to create such a platform. We described the Fractal Conceptual Prototype, where we explored what we see as key requirements for a multi-tenanted cloud-scale platform focused on content-centric collaboration. We argued the current generation of ECM technologies is not a good match, and highlighted some of the improvements required.

Over the next twelve months, our research will focus on alternative implementation patterns to satisfy these requirements.

6. References

[1] Erickson J et al. "Content-Centered Collaboration Spaces in the Cloud". HPL Tech Report HPL-2009-11. Submitted Jan 2009 to *IEEE Internet Computing* special issue on Cloud Computing. http://tinyurl.com/fractal-vision

[2] Alfresco Share: http://tinyurl.com/3z7wkh

[3] Fractal Conceptual Prototype Videos: Content Spaces (http://tinyurl.com/cfzhjo), the Extensions Marketplace (http://tinyurl.com/d53jyx), Active Behaviors (http://tinyurl.com/c76ncd).

[4] Simile Exhibit http://code.google.com/p/simile-widgets

[5] Alfresco Cluster Configuration http://tinyurl.com/ct72mh

[6] De Roure, D., Goble, C. and Stevens, R. (2008) "The Design and Realisation of the myExperiment Virtual Research Environment for Social Sharing of Workflows". Future Generation Computer Systems. http://eprints.ecs.soton.ac.uk/15709/

[7] Ning Social Network Platform: http://ning.com

[8] Microsoft Azure: http://www.microsoft.com/azure

[9] Google App Engine: http://code.google.com/appengine

[10] The Force.com Multitenant Architecture: http://tinyurl.com/8uxxb7

[11] Lei Li, John Hosking and John Grundy, "Visual Modelling of Complex Business Processes with Trees, Overlays and Distortion-based Displays". 2007 IEEE Symposium on Visual Languages and Human-Centric Computing.

[12] Martinez, A., Patino-Martinez, M., Jimenez-Peris, R. and Perez-Sorrosal, F, "ZenFlow: a visual Web service composition tool for BPEL4WS". 2005 IEEE Symposium on Visual Languages and Human-Centric Computing, 20-24 September, 2005.

[13] Amitanand S. Aiyer, et.al. "Consistability: Describing usually consistent systems." Proceedings of HotDep 2008, The 4th workshop on Hot Topics in Dependability (7 Dec 2008), San Diego, California, USA. http://tinyurl.com/cnspyx