Lessons learned developing the Nuxeo EP open source, component-based, ECM platform

Stefane Fermigier, Thierry Delprat, Olivier Grisel,
Florent Guillaume – Nuxeo SA*

November 22, 2010

Abstract

ECM, a concept that emerged in the early 2000s, represents the integrated enterprise-wide management of all forms of non-structured (and sometimes, semi-structured) content, including their metadata, across their whole lifecycle, supported by appropriate technologies and administrative infrastructure.

Nuxeo EP is a commercial open source ECM platform developed in an open and collaborative fashion since 2006, with the goal of enabling the whole spectrum of ECM applications, from horizontal applications such as Document Management and Digital Assets Management, to more specific applications focused on the needs of a given industry or corporation.

This paper presents the main original business-driven goals for the project, their impact on the functional and technical requirement for the platform and its architecture, and the solutions that were devised to address them. It reflects upon the benefits and challenges of developing the project both based on open source libraries and frameworks, and itself in an open source fashion.

We found that existing open source Java frameworks, libraries and tools provide sufficient building blocks for enterprise software, as long as a proper architecture and development process are set in place to address the software quality needs of this kind of project.

keywords: enterprise content management, electronic document management, open source, software architecture, OSGi, component-based software engineering.

1 Introduction

According to AIIM[1], "Enterprise Content Management (ECM) is the strategies, methods and tools used to capture, manage, store, preserve, and deliver content and documents related to organizational processes. ECM tools and strategies allow the management of an organization's unstructured information, wherever that information exists."

*18 rue Soleillet, 75020 Paris, France. Tel: +33 1 40 33 79 87. Fax: +33 9 59 10 59 63

[1] http://www.aiim.org/. AIIM describes itself as "the leading non-profit organization focused on helping users to understand the challenges associated with managing documents, content, records, and business processes."

ECM platforms, or their predecessors, Electronic Document Management Systems (EDMS) have been around since the early 90s. Highly regulated industries, such as pharmacy, were their first adopters [Lal05]. As of 2006, the leaders in this market were large American companies, such as EMC/Documentum, Open Text or IBM/FileNet. On the other hand, open source enterprise software was at the time emerging as a credible competition to incumbent companies, with for instance Red Hat in the enterprise open systems market, or JBoss in the application servers market[2].

In early 2006, Nuxeo decided to engage in a major rewrite, using modern open source Java technologies, of its existing ECM platform, Nuxeo CPS [AFG+05] that had been developed by the company from 2002 to 2005 using the Zope open source framework based on the Python language.

An initial vision statement was written at the project inception. It stated the following business goals for the new platform:

- An initial set of functionalities focussed on Document Management, to match those of incumbent ECM platforms.
- "Enterprise-grade" robustness and scalability (e.g. 10000s of users, millions of documents and terabytes of storage), which is a prerequisite to compete in the mid to high-end ECM market.
- An easy and pleasing to use graphical interface, to ensure end-user adoption and word-of-mouth marketing.
- A good experience for developers, so as to make it easy to recruit open source contributors and third party application vendors.
- The use of existing and popular (or promising) standards in order to ensure a wide adoption by systems integrators and third-party application vendors, and to leverage existing third party tools, documentation and frameworks, thereby lowering development costs.
- The possibility to create horizontal and vertical applications on top of the platform, both by Nuxeo and by its partners, with the end goal of creating a rich ecosystem of third-party vendors.
- Ease of deployment, both for testing and production purposes, on entry-level servers (running Linux or Windows), and ease of management for IT operation teams, thereby lowering operations costs.

These goals where chosen to minimize initial capital investment and to ensure quick adoption of the new platform by the market, by providing both higher value and lower implementation costs than incumbent technologies, and using the open source development and commercialization model as a disruptive factor.

[2] JBoss was acquired by Red Hat in 2006.

2 Functional Requirements

2.1 Core ECM Services

The following services were identified as forming a cohesive set of functionalities that cover the basic needs of a content management system:

- Document types definition and management
- Storage of the documents and their associated metadata
- Document life cycle and versioning
- Access control
- Indexing
- A query language, expressive enough to enable complex queries on both full-text and metadata

As we will see below, these functionalities form the "core" of the Nuxeo platform. They are used both as the foundation for the full-stack platform, and to embed a "lightweight ECM core" into third-party products, e.g. for onboard plane maintenance documentation management in avionics, or in high-end scanner/copier devices.

2.2 Other ECM Services and User Interaction

The core services are complemented by higher-level services usually needed by most traditional document oriented enterprise applications:

- Workflow
- Transformation and rendering
- User management
- User interface
- A rich set of HTTP-based APIs exposed to third-party developers and integrators, conforming to either the WS-* or the REST paradigms.

User interfaces usually need customization to adapt to the needs of business users. We provide a generic user interface, that can be customized using "actions" (dynamically computed menu entries), "layouts" (that allow declarative rendering of content information, including creation and editing forms and their validation) and "content views" (flexible document listings).

We also eventually developed, for the needs of specific projects or clients, several alternative user interfaces, for rich clients (e.g. desktop application based on the Eclipse Rich Client Platform, or Flash) or for mobile clients (native mobile applications or HTML5, using a RESTful JSON API).

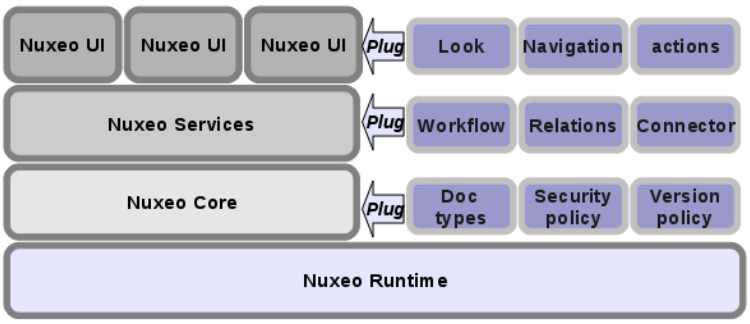

Figure 1: The Nuxeo layered architecture with services as pluggable components

2.3 ECM 2.0 and 3.0 functionalities

"Enterprise 2.0" features, oftentimes described using the SLATES and FLATNESSES acronyms [Hin07], were out of the original scope of the project, but were nevertheless envisioned as important features to be added later. They include:

- Tagging and folksonomies management
- Lightweight collaboration (wikis) and publishing (blogs)
- Social networking ("friending" or "following" colleagues or business partners, user timelines)
- Collaborative filtering
- Mobile and disconnected access
- Semantic content categorization and named entities extractions

It was therefore important to adopt an architecture that would allow to easily plug new features such as these into the core of the platform, without needing major rewrites of the existing code base.

3 Technical Challenges

3.1 Standards Choice

As the saying goes, "the problem with standards is that there are too many to choose from".

With Nuxeo CPS, we had chosen a programming language (Python) and application platform (Zope) that were considered exotic by the mainstream systems integrators. With Nuxeo EP, we wanted instead to bet on existing, widely adopted standards, but also on technologies that were in line with modern, agile development practices, and with our own focus on open source technologies.

In 2006, after a somewhat rough start in the late 90s, the open source Java ecosystem was starting to mature and provide many of the building blocks

that were needed to build our platform. This ecosystem was already providing many alternatives, with first-generation toolkits such as Struts[3] being quickly displaced by newer projects. The Java Community Process[4] was also becoming more open to agile and open-source led innovations and less relevant on heavyweight tools vendors to foster the development of the new enterprise Java standards.

After evaluating the most popular and promising standards and projects in this ecosystem, we came up with the following initial choices:

- The brand new (at the time) Java EE 5 standard, as the structuring general framework for the server-based application (but not for the core services).
- OSGI[5], as a packaging model for components.
- The JCR (Java Content Repository), also known as JSR 170[6], as the API to manage content and metadata at the most basic level, and Apache Jackrabbit[7] as its open source implementation.
- JBoss Seam, a web presentation framework that was only emerging at the time, and certainly not standardized, but that we felt would provide a much improved developer experience over the "pure Java EE 5" model.

In retrospect, we've been very happy with most of these choices. Java EE is still the enterprise Java standard[8]. OSGi is becoming the *de facto* standard for Java modularity, albeit at a slower pace than we expected. Seam has now (mostly) evolved into CDI[9], a specification that has been integrated into Java EE 6. We eventually dropped the JCR in late 2010[10], because we had needs that weren't addressed by the JCR model and Jackrabbit, but adopting it early in the project was still the right decision to enable us to bring a product to market as early as 2007.

3.2 Components and Reusability

While it is widely argued that one of the advantages of open source is that one can take an existing open source project and modify it by changing its source code to adapt it to one's own needs, our 10 years of experience with open source development is that this causes several problems: as soon as you modify code in a project, you end up practically "forking" it, hence having to maintain *yourself* a branch separated from the main codeline[11].

[3] http://struts.apache.org/

[4] http://www.jcp.org/

[5] http://www.osgi.org/

[6] http://www.jcp.org/en/jsr/detail?id=170

[7] http://jackrabbit.apache.org/

[8] Java EE 5 is now supported by all major application server vendors. We still have to wait for 2011 or 2012 to see wide adoption of Java EE 6, which was ratified by the JCP in December 2009.

[9] JSR 299: Contexts and Dependency Injection for the Java EE platform, http://jcp.org/en/jsr/detail?id=299

[10] We're now focussing our efforts on CMIS (the Content Management Interoperability Standard), a model and a set of web services API that have been ratified by OASIS in 2010.

[11] During the course of the Nuxeo EP project, we've had not other choice but to do it for some third-party libraries, to address some of their bugs or shortcomings, but we had to pay a price for this in terms of maintenance.

To achieve the business goals of enabling a wide range of business applications built on top of the platform, we therefore had to devise a system that would be configurable and extensible by third parties without touching our source code, and where application developers could create applications by assembling components that are provided either by the platform, by third-parties (as "plugins") or developed specifically for the purpose of their project.

This approach is also a great way to manage complexity: an application developer only needs to understand the roles of the components he's planning to use, and only the interfaces for these components is useful to develop against them, not their whole implementation.

4 Architectural Solutions

Our solution, called the *Nuxeo Runtime*, for the technical challenges described above is three-fold: a *component system* based on OSGi, *extension points* to enable configuration and extension of these components, and an *event bus* to provide loose coupling between these components.

OSGi [MVA10] is a mature standard initially developed for set-top boxes and other consumer or industrial systems, but eventually adopted by the enterprise platforms developers to provide modularity to their applications and solve some of the problems (dependency management, service isolation, etc.) that they face.

It allows creating components as "bundles" of Java classes, hidden behind one or several interfaces, and that can provide services to other components when they are activated inside an "OSGi container" (or OSGi-aware application server).

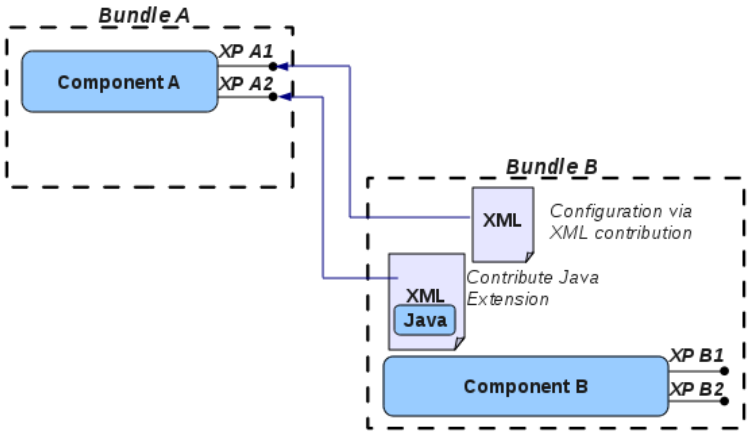

Figure 2: Extension points between components in the Nuxeo Runtime

Inspired by the Eclipse runtime system [MLA10], we've extended OSGi with *extensions points*, XML configuration files that express declaratively (among other things) how a particular implementation class can be plugged into the interface of another component, thereby extending it.

Event-driven architecture (EDA) [Cha06] is a well-known architectural pattern that can be applied to the design and implementation of applications and systems comprise loosely coupled software components and services. By transmitting events over an event bus managed by the Nuxeo platform, components can communicate without a priori knowledge of their existence or specific interfaces, hence helping the creation of applications out of diversely sourced components.

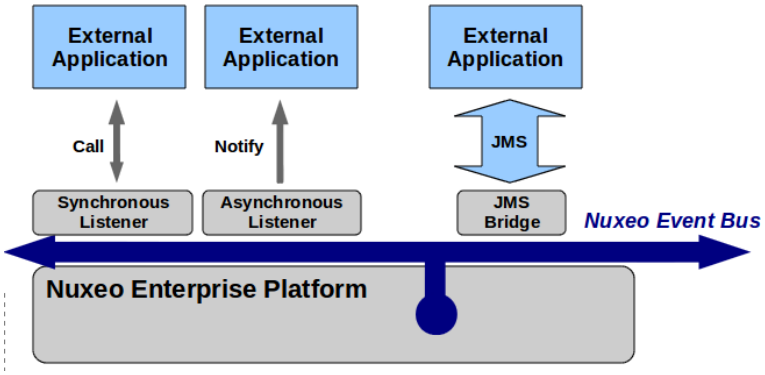

Figure 3: The Nuxeo event bus

5 Open Source Issues

5.1 Participation

One of the most important aspects of an open source platform project is to foster collaboration with external developers or organizations by creating an *architecture for participation* [O'R04].

While managing an open source community is an activity in itself that needs dedicated care and resources (see [Nea09] or [Bac09]), we'll only note here that by splitting a project into a solid *core* managed by a small group of highly skilled and dedicated developers, a larger set of *maintained components*, managed by a larger group including community contributors with *committer* rights, and a wider set of *plugins* managed by external groups, we solve the main issue of open source projects, which is to bring together people with highly various levels of skills and dedication to the project.

5.2 Open Source Libraries Choices

As for choosing a standard, choosing an open source library is a decision that needs a lot of attention. In our case, and given our priorities, factors that need to be taken into account included:

- Licensing: our choice of the LGPL as the main open source license for the project meant that we had to reject libraries that were licensed under the GPL, or licenses that are incompatible with the LGPL. This still allows us

large choice of licenses, such as the BSD "three clauses" license, the MIT license, the Apache Public License or the Eclipse Public License [Ros04].

- Maturity and quality: we want to use libraries mature enough to be well documented and with already stabilized APIs, but still close enough to the innovation stage of their lifecycle to bring a technical edge to our platform. We can note here that as customers of open source software, we have the opportunity to look into their source code and make an informed opinion on the quality level their authors have chosen to aim for.
- Popularity and reputation: open source libraries with more users, and more contributors, have a higher probability of being both high quality and maintained over the long term.

As of November 2010, Nuxeo EP includes, at runtime, 219 jars from approximately 150 different open source projects. These dependencies are either direct, or indirect (i.e. Nuxeo EP depends on project A which itself depends on project B). This can lead to conflicts between versions (that we choose not to address using the "pure OSGi" approach, because of the additional work needed to do so) that need to be carefully watched for and addressed if needed.

5.3 Quality Assurance

Just as for traditional software, open source software development needs rigorous quality assurance. This is even more important when you are including in your platform contributions by external developers, either as patches or by making them committers or module owners.

To address the needs for quality, we implemented the following practices, to catch as early as possible quality issues:

- Architecture and code reviews
- Automated builds
- Continuous builds
- Unit tests
- Integration tests
- Functional (web) tests
- Performance tests
- Static code analysis

Scooting the web and blogs for experience reports and best practices, and after trying many potentially interesting solutions by ourselves, we settled on the following list of major QA-related tools, that range from the good to the outstanding: Maven[12] for automating builds, Hudson[13] for continuous builds,

[12] http://maven.apache.org/
[13] http://www.hudson-ci.org/

JUnit[14] for unit tests, Selenium[15] for functional tests from a real web browser. We also developed a tool called Funkload[16] to run performance tests.

Our build infrastructure has grown from a single machine to a farm of 10 servers that builds and tests throughout the day more than 200 different projects, tracking dependencies between our different sub-projects and plugins.

On the negative side, like many other developers[17], we found out that Maven could introduce hard to track bugs on complex projects (such as ours), but that no other alternative exists at this point. We also couldn't find satisfactory solution for running *unit* tests on the JSF web interface without running the whole application in its application server and running the tests in a browser, nor could we find an open source solution for testing our Eclipse RCP client.

6 Related Work

6.1 Open Source, Java-based

Alfresco[18] is another open source ECM platform. It is based on Java and uses the Spring framework[19] to wire services statically. For this reason, it lacks the runtime configurability available in the Nuxeo platform. Alfresco can also be scripted at the web-level using JavaScript.

Jahia[20] is a Web Content Management platform, and XWiki[21] is a collaborative platform (a Wiki); both are developed in Java using their own custom component model. XWiki can be scripted using the Groovy dynamic language.

Day Software[22] has a range of proprietary products (mostly WCM and DAM) based on an open source project called Apache Sling[23], which is based on OSGi components and content stored in a JCR content repository (they are using Jackrabbit, of which they are also the main developer), and a lightweight web framework programmed in JavaScript.

6.2 Open Source, not Java-based

There are hundreds if not thousands of open source content management systems (CMS) written in dynamic languages such as PHP, Python or Ruby, but most of them fall into the category of "web content management", blog engines, or wiki engine, which is a subcategory of ECM but very far from what we are trying to accomplish with Nuxeo EP.

A few platforms deserve a mention though, is the context of this study: Plone[24] (a Zope-based platform for building intranets, with a rich ecosystem of extensions called "Products"), WordPress[25] (the most widely used open source

[14] http://www.junit.org/
[15] http://www.seleniumhq.org/
[16] http://funkload.nuxeo.org/
[17] Google for "maven *derogatory word*" and you will easily find many such rants on the Web.
[18] http://www.alfresco.com/
[19] http://www.springsource.org/
[20] http://www.jahia.org/
[21] http://www.xwiki.org/
[22] http://www.day.com/ – Day software was acquired by Adobe in October 2010.
[23] http://sling.apache.org/
[24] http://www.plone.org/
[25] http://www.wordpress.org/

blog engine, written in PHP and extensible by literally thousands of plugins) and Drupal[26] (another PHP platform, which provides richer functionalities at the expense of more complexity).

These platforms have succeeded in attracting wide extensions ecosystems. However, a noticeable drawback of their choice of dynamically typed languages and simplistic component systems is that they can't guaranty, except by extensive manual testing, that API changes in newer version of their platform won't break existing extensions, or that a given set of extensions will work together as expected.

6.3 Proprietary Platforms

Proprietary software is by definition harder to study for an outsider than open source. From the prior experience of some of our team members, and conversations with our systems integrators partners, we can nevertheless assert the following facts about our main competitors, Documentum, FileNet and Open Text:

- They were created in the 90s, and as such, have a mature and well-tested code base, but are also hard to evolve and extend, and don't leverage fully the software and usage innovations of the last decade.
- Their respective companies were grown by numerous acquisitions of non-compatible technologies that they didn't try to rewrite using a common framework or architecture.
- They carry an expensive (albeit frequently discounted, due to market pressure) price tag, are hard to procure, and hard to install.
- Their user interfaces are not perceived as pleasing as modern Web 2.0 applications.

7 Conclusion and Perspectives

The Nuxeo EP project has succeeded in creating the foundation for a rich range of vertical and horizontal applications, in meeting its other initial business goals, and in evolving according to unforeseen business and technical needs that have emerged since its inception, thanks in great part to the efforts that have been spent on the design and development of its modular and extensible architecture.

Being in a very competitive market, we still need to improve and enrich the platform to stay ahead of the competition. Ongoing work includes:

- Work on simplifying the developer experience: faster turnaround between writing code and testing it, simpler web front end development using the more recent JAX-RS[27] standard or Google Web Toolkit (GWT)[28].
- Development of business-oriented RESTful APIs to allow high-level interaction with the content, and eventually business application development by non-technical users.

[26] http://www.drupal.org/
[27] http://jcp.org/en/jsr/summary?id=311
[28] http://code.google.com/webtoolkit/

- Work on replication, both in a LAN (for scalability and fault tolerance) and WAN (for replication between remote data centers, or between a server and a desktop or mobile client) contexts.
- Social integration (using the OpenSocial standard[29]).
- Further work on semantic technologies, developed in part within of the IKS[30] and SCRIBO[31] collaborative projects.
- Porting the platform to the Cloud, and exposing it as a PaaS service.
- Work on mobile ECM, with clients for platforms such as the iPhone/iPad, Android and Blackberry operating systems.
- Bringing Nuxeo EP to Java EE 6 and full OSGi compliance.

References

[AFG+05] Julien Anguenot, Stefane Fermigier, Florent Guillaume, Jean-Marc Orliaguet, Lennart Regebro, and Tarek Ziadé. Nuxeo cps: an open source framework for the development of enterprise content management and collaboration applications. In *Proceedings of the 2005 EuroPython conference*, 2005.

[Bac09] Jono Bacon. *The Art of Community: Building the New Age of Participation.* O'Reilly Media, 2009.

[Cha06] Mani Chandi. *Event-Driven Applications: Costs, Benefits and Design Approaches.* California Institute of Technology, 2006.

[Hin07] Dion Hinchcliffe. *The state of Enterprise 2.0.* The Enterprise Web 2.0 Blog, 2007.

[Lal05] Rajiv Lal. *Documentum Inc. Business Case Study.* Harvard Business School, 2005.

[MLA10] Jeff McAffer, Jean-Michel Lemieux, and Chris Aniszczyk. *Eclipse Rich Client Platform (2nd Edition).* Addison-Wesley Professional, 2010.

[MVA10] Jeff McAffer, Paul VanderLei, and Simon Archer. *OSGi and Equinox: Creating Highly Modular Java Systems.* Addison-Wesley Professional, 2010.

[Nea09] Dave Neary. *Community Building – Barriers to Entry.* dneary.free.fr, 2009.

[O'R04] Tim O'Reilly. *The Architecture of Participation.* oreilly.com, 2004.

[Ros04] Lawrence E. Rosen. *Open Source Licensing: Software Freedom and Intellectual Property Law.* Prentice Hall, 2004.

[29] http://www.opensocial.org/
[30] http://www.iks-project.eu/
[31] http://www.scribo.ws/