Project acronym: OSEPA

Project name: Open Source software usage by European Public Administrations

Project code: INTERREG IVC, 0918R2

Document Information:

Document title: OSEPA_CP3_72_policy_recommendation_24012012

Date of Delivery: 24/01/2012

Component: 3

Component Title: Exchange of Experience

Component Leader: USFD

Distribution (Restricted/Public): Restricted

Nature: Report

History Chart

Date	Changes	Cause of change	Implemented by
N/A	Initial document	N/A	USFD

Authorisation

No.	Action	Partner	Date
1	Prepared	USFD	24/01/2012
2			
3			

Disclaimer

The information in this document is subject to change without notice.

All rights reserved

The document is proprietary of the OSEPA Consortium. No copying or distributing, in any form or by any means, is allowed without the prior written agreement of the owner of the property rights. This document reflects only the authors' view. The INTERREG Programme is not liable for any use that may be made of the information contained herein.

1. Table of contents

2. Abbreviations

EPA(s): European Public Administration(s)

PA: Public Administration

FOSS: Free / Open Source Software

IT: Information Technology

OSEPA: Open Source software usage by European Public Administrations

OFSTED: Office for Standards in Education (UK)

API: Application Programming Interface

JVM: Java Virtual Machine

OGL: Open Government Licence

3. Introduction

"The OSEPA project aims to conduct a systematic debate among European public administrations supported by analysis and exchange of experience on the issue of free and/or open source software (FOSS). Consequently, it is critical to explore the main benefits and disadvantages, as well as the cost effectiveness resulting from FOSS adoption and use by public authorities." [1]

The British Government is "going cloud", and this presents an opportunity for FOSS in the UK that might be paralleled in other European public administrations. Cloud computing is a variant of the client/server model that has been around for many years. Historically, large organisations have maintained their own mainframe servers, often with custom software, connected via a network to terminals within the organisation. In cloud computing, servers are more likely to be a bank of networked low-cost computers, which are somewhere "in a cloud" – it matters not where – on the Internet, and users connect to software running on these servers via a variety of interfaces, from web browsers to smart phones.

This report is intended to provide recommendations for technical standards relating to FOSS usage. Instead of providing fixed technical standards, which would be very quickly superseded, this document aims to provide a recommended framework for the future. It firstly addresses the issue of cloud computing and then considers the current corporate situation with some description of existing solutions. It then addresses the issues of open standards and data security, the UK plans for cloud solutions with some examples for potential development using FOSS.

Finally, this report recommends that the interests of the Free and Open Source communities will best align with those of public administration by developing both server-side software *and* the client-side applications to support this software on a range of devices. This is the "Software as a Service (SaaS)" model of cloud computing discussed below. The government can best support this industry – and hence reduce costs and improve service – by ensuring appropriate infrastructure is in place, is accessible, and remains so.

4. Cloud Computing – What is it?

The catch phrase is "computing as a service rather than a product" and the idea is to use computing services over the network on an as-needed basis rather than having to buy and maintain in-house computers and software [2]. This can lead to cost savings for the PA – for example by services being hosted where electricity is cheaper – but perhaps the cloud model's greatest advantage is that these or equivalent resources are available to developers too. Thus cloud computing is an opportunity to lower the barriers to entering the software provision business, opening up competition, which in turn has the potential to dramatically improve the quality of the service provided.

Figure 1. The cloud is full of servers that need power, and a user doesn't need "a computer" any longer.

The classic example of computing as a service is a search engine. The Google search engine, for example, does not run on your personal computer (or handheld device), but on tens of

thousands of low-spec computers in places where electricity is cheap – the electricity being used primarily to run air conditioning units to keep the computers cool, rather than power the processors. The computer on your desk (or in your hand) is there primarily to run a web browser. However it is not just search engines that can run in this way. Google provide a host of applications, from word processors to real-time navigation services on their servers. The question becomes what *can't* be run on Google's servers, and the answer is, very little. The idea of *data* being stored on a central server is by no means new, but the cloud model goes two steps further. Firstly, the physical location of the server is irrelevant (and in fact “the server” may in fact be a farm of servers distributed over a range of physical locations). Secondly, and more importantly, in the cloud computing model *processing* is also handled on the server(s), and the users’ device simply provides the display.

4.1. The Technicalities

Ever since there were networks it has been possible to give users a local interface to a remote computer. For example in the 1980's users could run a VT100 emulator on a machine at home and connect via a phone line to the computer at work. Today Sun Microsystems can provide access to an OpenSolaris server on SPARC processor-based Sun systems [3] that allows anyone to run shell scripts. This type of cloud provides “infrastructure as a service” (IaaS) [1]. The true excitement about cloud computing however is in "Software as a Service," in which end users can use applications such as word processors and spread sheets, find train timetables or check the weather forecast, seamlessly from devices such as their computer browser or smart phone. Such services are often provided free of charge, from an open market of developers.

At its most basic, cloud computing requires a network, a client at the user end, and data at the server end along with the software to deliver it. The widespread use of HTML and browsers provides a new channel for service delivery; developers no longer need to produce software for particular user platforms. As soon as the browser was invented it was realised that web pages could be generated dynamically and served up to a browser to give the appearance of local interaction. This is how search engines work, and HTML forms provide a generic model for this technology. An alternate model is that software can be downloaded from the cloud and to run on the user’s device. For example, many operating systems can be configured to automatically download and install updates without user intervention, and smart phone applications can be

installed and updated through a virtual shop front. Downloading “apps” for a smart phone provides explicit but extremely simple means of installing software on one's own device and, critically, the iPhone app store [4] model provides an easy means for anyone to provide software to users (with or without charge). That software might be a stand-alone application for the phone, but equally might be a client interface to some service provided elsewhere.

There are of course security issues when it comes to running third-party software on one’s own device (or indeed providing data for software running in the cloud). A range of approaches have been taken to this issue, from the notion of “trusted suppliers,” to the use of Java applets that have limited functionality inside a “sandbox” – they cannot, for example, access a local file system.

It has become standard for third party code to run on user machines and, in combination with smart server-side processes and data access, anything is possible. What truly empowers application developers is that they can develop not only the client side interface but also the server side support for interesting apps. The technological revolution that supports this is the idea of “Platform as a Service” [1] whereby a developer is provided with not only the (possibly virtual) physical hardware and an operating system, but also software infrastructure such as revision control, backups, security, databases and of course, a server.

At the server end interesting things can be done using dynamic web pages [5], but the custom code to do that has to be done on one's own hardware. Today, when disk space is cheap enough to give away (with some advertising perhaps) other people will not only host my web-page, I can, today, for free, set up a database and a server on Google's machines. As with applet code there are restrictions, and if my server becomes very popular then Google will start charging me, but the barriers to entering the market are very low. Contrast this with the process of becoming a software supplier to the UK National Health Service (NHS) at the moment. Medical general practice’s tend to keep their records on on-site computer maintained locally and running software provided by private companies “... that are contracted to offer systems on the GPSoC Framework“ [6]. These companies have a vested interest in locking out the competition so to even look at the API for a leading system supplier requires one to become a partner, and that requires “... your product is on the market for resale” [7].

Cloud computing has the potential to enable anyone, trusted or not, to enter the market place. The app store model also provides a seamless means of charging for software that might encourage others without an interest in open source to participate in the process. The advantage of the model is that it pushes competitive software provision to the limits. This is of course not in the interests of big business who tend to use market dominance to systematically control innovation. It took Microsoft 10 years to produce a windowing system comparable to the Mac Classic, but that was 10 years of profit; it was not because Microsoft engineers couldn't do it. Similarly many database interfaces have a modern “look”, but the “feel” is of pre-windows “green screen” technology. Cloud computing can be seen as an opportunity for EPAs to set up and maintain a level playing field for the software industry from multi nationals to individuals in their garage, reducing costs and improving quality.

5. A Comparison

Although many things are being called cloud computing at the moment, there is genuine potential in one model of cloud computing and that is Google's. In order to compare and contrast possible alternatives the following set of questions can be addressed. Migrating to cloud has a social side, and these questions raise not just the technical issues but political and social issues as well.

1. What, exactly, can cloud services expect to be in front of a user?
2. Who provides and maintains the servers and storage required on the network?
3. What motivates individuals and organisations to provide apps and data?
4. What motivates individuals and organisations to
    - 4.a. Use the service?
    - 4.b. Look for new apps/data?
5. How does one ensure that apps and data are interoperable?
6. Who has access to data on the server - both formally and illegally?

7. How does one ensure that required services are still available next week, next year, next century...?

5.1. The Corporate Model

One no longer buys a computer operating system from Microsoft but rather uses an operating system provided by the company along with all the upgrades specified in the service contact. These upgrades are typically performed automatically, without user intervention. Considering the questions above, (1) this model expects a computer with one of Microsoft's supported operating systems to be in front of the user. The corporation provides the servers (2) for software downloads but it is expected that data be kept on a local disk (3). Profit and the competition keep Microsoft providing the upgrades (4) and users have little choice about using the service. The corporation of course has an interest in making new services interoperable with existing products (5) however there is on occasion a strategic reason for *not* being interoperable - the corporation has complete control over interoperability. Access to Microsoft upgrades (6) is of course a big issue with corporations very keen to stamp out “pirate” software and place restrictions on others writing software that uses its file formats. Finally (7) what would happen if Microsoft became insolvent? The capitalist model is of course that other investors would buy up valuable assets. Such a scenario could result in interoperability issues, with browsers not talking with document editors and so on. Given society's reliance on Microsoft products, it is an unlikely scenario that would be treated much the same as failure of the banks, that there would be government bail out in some form.

5.2. Apple Inc.

Apple is also a corporation with an interest in protecting investment, but in this case the “app store” – which users subscribe to upon purchasing the phone – makes it not only possible, but also easy for third parties to provide and charge for software to download. Considering the questions, this looks very much like the above. Again (1) the corporation keeps tight control over what is in front of the user, and (2) provides the servers. The provision of apps (3) is much like the provision of software in the FOSS community, but there is a clear model for making money from the process if your apps are in high demand. Provision of data seems to be primarily driven by app development. It is not clear how users are expected to find apps (4) but there has been significant and effective marketing of the iPhone and its related process, and it will be

interesting to see how it works out in the long run. At the moment, the primary method of finding apps seems to be by word of mouth and by press stories. There would seem to be very little interest in interoperability (5) between apps at the moment, and as above, (6) the corporate model means Apple has total control over access to data and would like to keep it that way. As above, it is very hard to imagine a scenario in which Apple does not exist in 5 years time although the current product is likely to be superseded.

5.3. Google Android

On the surface Android looks very much like the iPhone model, with the exception that Google has no commercial interest in the operating system itself (it gives it away) and the corporation takes no interest in what apps are available. There is nothing new here; it is a phone with the Java Virtual Machine (JVM) and a browser; so what is Google's business model? One possibility is that Google's interest is in its servers, and the aim of the Android exercise is to ensure that other corporations do not get a monopoly on the software downloaded and hence do not get control of data format and storage location. Addressing the questions above,

1. Users have a java bite code interpreter (JVM) and a browser
2. Google runs the servers and provides storage for your data
3. Apps are provided by the community - often for free but possibly with a (micro) charge based on Google's own payment process
4. Google's expertise and reputation is in search engines and there is no doubt that that is the expected means of finding Android apps
5. Google has explicitly addressed the interoperability issue with regard to doc format by providing a conversion service - a conversion service that very effectively keeps up with Microsoft's "enhancements". Other app providers are most welcome to provide converters for other formats.
6. Again the corporation has total control over the data
7. Once again Google is likely to be around for the next five years but is there any guarantee to access?

This last point is discussed further below.

5.4. The FOSS Model

The open source community has been running software over the network for years, primarily because there is no need for a means of controlling or charging for future use. The FOSS community is indeed well ahead when it comes to many of the issues associated with cloud technology and so it is perhaps worthwhile to look at the above issues as applied to Ubuntu. What is expected in front of the user (1) is IBM PC compatible hardware, X86 being the default, although there are variants for old Macs. Storage (2) has been an issue. Initially it was the servers running web pages at universities that enabled software to be made available. More recently specialist sites could be hosted on servers funded by paid advertising that is, hopefully, relevant to the sites being hosted. Motivation (3) in the FOSS community happens effectively for a number of reasons. Finding apps (4) is in part driven by search engines but primarily works via forums and mailing lists. This requires a certain level of commitment and it is hard to see how it might work for those motivated by the “cool” factor with minimal technical interest.

Interoperability (5) has been an issue with open source software. This may invoke the notion of standards (discussed below) but the alternative approach that has emerged for the community of software developers has been “the community process.” This was formalised with the Java Community Process for developing standardised APIs (with limited success). What has happened in the Open Source Community is worthy of study but in general what one finds is a set of fundamental or core packages that provide a standard base, and changing these is a slow process. Anyone can develop and release new software that uses these, but developing software upon which other code depends leads to notions of backward compatibility. Hardware drivers are an example of software that is used by other software and so has a slow development cycle. An individual may develop a new driver from scratch for the first webcam for instance. The driver is then extended, maintaining existing backward compatibility, to cover more cameras as they become available. Then, say, a new camera comes along that is not compatible and someone must write a completely new driver. Either the new hardware dies (at least in the FOSS community) or it becomes so popular that it forks to produce a new driver development programme that may, or may not, be merged at some future date with the original

driver development process. This process was formalized with Sun's “community process” for Java development [8] but many claim a suitable community of practice did not develop.

Security (6) is an interesting issue because there is none. Two strategies have evolved; the first is that open source by its very nature is well written code that is highly scrutinised. It is interesting to note that the Chinese government would only use Microsoft products if they could have the source code, and a year later the US was accusing “hackers” in China of cyber attacks on US institutions [10]. The second approach is to hide data in the public arena. Doodle [11], discussed more below, is a case where one does not need an account or anything to identify yourself to be a user because the only way someone knows about the stored data on the public website is if they have been sent the link. Similarly “the dark web” [12] is a suitably sinister name for the parts of the web that are not linked and hence not crawled by search engines. The only way to access one of these websites is to know of its existence.

Ensuring services are available next week (7) is a problem because, although one keeps a local copy of an application (unlike the cloud model) the software with which the application interacts may evolve to the point where it no longer works with the application. At an enterprise level, the solution is for entire systems to be kept in running order and run in parallel with newer systems. Note this problem is not unique to FOSS, but the problem is one for which the community has a process to manage and mitigate impact.

6. Open Standards

A huge issue for governments around the world has been the interoperability of data and the solution advocated by many, and embraced by many government departments, is to enforce interoperability by decree. Open Standards for data certainly address the interoperability issue, but can lead to stagnation. The introduction of the use of cloud computing by the UK government provides an alternative based on the Open Source community’s process.

There is a need to dramatically improve the ability of government departments and the wider community to exchange information. This is of course a claim for providing data standards typified by the work of World Wide Web consortium (W3C), which provides a set of standards

for data on the Internet [13]. Some of these will be familiar - XML, HTML and CCS being common enough to need no explanation. Another of particular interest in this context is XBML – the Extensible Business Markup Language – at the time of writing the required format for corporate tax returns in the UK [14] .

CSS	SKOS	XML Information Set
CGI	SOAP	XML Schema
DOM	SPARQL	XPath
GRDDL	SMIL	XQuery
HTML	SRGS	XSL-FO
MathML	SSML	XSLT
OWL	XHTML	WSDL
P3P	XHTML+Voice	XForms
SVG	XML	
SISR	XML Events	

Table 1. Data standards for the Internet [cf Wikipedia entry for W3C].

This set of standard data formats is not complete and in some cases the development of standards is actually driven by legislation. The GPRA Modernization Act of 2010 (GPRAMA), became law in the United States in January 2011. Section 10 requires agencies to publish their strategic and performance plans and reports in machine-readable formats.

The UK Government conducted a public consultation entitled Making Open Data Real: A Public Consultation in August 2011 [30] focused on public data transparency and the issue of standards. They have appointed a Public Sector Transparency Board to support and challenge public sector bodies in the implementation of Open Data standards; and a new Open

Government Licence (OGL) has made it easier for public service providers to publish data. When considering opportunities for improvement, despite an array of legislation and guidance, there are a number of barriers to accessing, using and re-using data that could generate economic or social value. One of these is cost resulting from historic ICT procurement and data management where information is held within government in a way that makes it costly to release, so requesters are refused on the grounds of cost. One of the proposals is to ensure that through procurement rules data collected by public service providers is stored in ICT systems that minimise the cost and difficulty of publishing data online. Currently, data requests are often refused because the data is stored in a fashion that makes it difficult to extract. Procurement rules for ICT systems could be reformed to ensure that new systems are designed in ways that make data extraction easier and cheaper.

In addition open data can be a driver of economic growth. A new market for public service information will thrive if data is freely available in a standardised format for use and re-use. At present the market for information on public services is highly underdeveloped. This new market will attract talented entrepreneurs and skilled employees, creating high value-added services for citizens, communities, third sector organisations and public service providers, developing auxiliary jobs and driving demand for skills.

So to ensure that Open Data standards are embedded in new ICT contracts it is necessary to ensure a line of continuous improvement for public service providers in achieving the highest ratings for their published data when compared against the Five Star Rating20 for Open Data:

One Star: Available on the web (whatever format), but with an open licence.

Two Star: As (one star) plus available as machine-readable structured data (e.g. Excel instead of image scan of a table).

Three Star: As (two star) plus use non-proprietary format (e.g. CSV and XML).

Four Star: All the above plus, use open standards from the World Wide Web Consortium (W3C) such as RDF and SPARQL21 to identify things, so that people can point at your data.

Five Star: All the above, plus link your data to other people‟s data to provide context."

The UK Government aims to work with data providers and the data re-user community to set standards. They intend to set general standards for data release, which will cover policy and technical measures so data can be used as widely as possible. In addition it is already considering how to maximise the potential economic gains from Open Data. In January 2011 it announced its intention to create a Public Data Corporation (PDC), which would bring together data-rich organisations with the aims of:

- Providing a more consistent approach towards access to and accessibility of Public Sector Information, balancing the desire for more data free at the point of use, whilst ensuring affordability and value for taxpayers;
- Creating a centre of excellence driving further efficiencies in the public sector; and
- Creating a vehicle that can attract private investment

In order to do this the consultation outlines a number of draft public data principles. On of these states:

"Public data will be published using open standards, and following relevant recommendations of the World Wide Web Consortium. Open, standardised formats are essential. However to increase reusability and the ability to compare data it also means openness and standardisation of the content as well as the format." [30]

Standards for data formats are a move in the right direction but committee drives these standards. As such, the process is usually slow, it is subject to manipulation [15], and potentially the resulting data format is not aligned with use. HTML worked because it was a standard that allowed many to develop browsers, and it is that relationship with the browser that led to success. Rather than extending the HTML standard, plug-ins were provided to extend functionality and the app model takes that to the extreme. Android handsets are essentially a JVM and incidentally have a browser and a telephone.

The challenge is to ensure interoperability between apps and between new apps and existing server side software. Rather than having apps exchange data in a standardised format, the

proposal is that what needs standardising is the APIs for apps and, in particular, the code running on the servers. Rather than having the data format readable by a browser or a browser plug-in, the data is accessed through an API that is usable by code running on the machine in front of the user. Rather than storing tables of data in files in an XML format, the format of the data is irrelevant, but data is *provided* by a database interface such as that of MySQL that is used by the app running on my smart phone.

This is not to say that the work of the W3C is irrelevant however. It is non-trivial to decide how data should be carved. In a classic case name and address is quite a good identifier for an individual but it is not sufficient as people change address and can, legitimately, use multiple names. A backend app that accesses XBML data is trivial and would make an excellent starting point. The standard should however be expected and allowed to evolve based on how it is used.

7. Securing Data

Many authorities are worried about open source software and the issue of security of data. Putting sensitive data on a web server can seem even more problematic. Third, having hackers write code that asks for your on-line identity seems even worse. This worry means that these issues have been addressed and indeed the real security issues are the ones we do not think about. Upgrading a computer, people rarely think about how to securely wipe the data off disk drives. Doing it properly is a time consuming process and is often omitted without explicit management directives.

Looking at these common concerns in turn. Open source software may be written by the untrusted, but it is also highly scrutinised by the community in a highly public manner. Finding holes in commercial software often results in legal action against the messenger [16]; finding security holes in open source results in kudos for the finder and a rapid fix. Putting sensitive data on the web is not trivial but of course the banks do it regularly. Having a well documented process with some level of oversight by an organisation with a vested interest in security, be it the State securing our medical data out of fear of public outrage or Google overseeing security to maintain the reputation of the service, would be better than the current practice of leaving it to the limited IT services available to, for example, a medical practice.

Access to online identity is another matter and one that has been addressed by the Open Source community. If you have a Google account for your email then you already have a digital identity under the OpenID scheme. Likewise for anyone with a Yahoo!, Facebook, Twitter, Windows Live ID or LinkedIn account. Fortunately these sites all share the open standard for handling "signing in" and so signing in to one *can* mean you don't need to sign in to the others. It is fortunate that this is open source because this means that those in the open source community know how the signing in process works, and it means that there is an explicit interest in separating the process of *authenticating* a user's id from the process of authorization to use data. The two protocols for handling these processes are OpenID for federated identity management [17], and OAuth [18], which provides a layered approach to authorizing sites to access personal data. An example of this is given below.

8. gCloud

The UK government is aiming to move to cloud computing and ideally this move would make FOSS the software choice of preference for all government departments. The gCloud (the UK Government planned cloud) is still under development so the recommendations made below are speculative in the UK context. What will probably happen is that some departments will change current practice, and that the cloud used will be Google's. The recommendation is that the Government set up its own server farms and encourage everyone to use the service. There are several reasons for having local server farms.

8.1. A level playing field

Large software providers have advantages other than economies of scale. Some of these are illegal, some may be illegal in the future, and some are simply the result of natural monopolies. The Government's vision in the UK is to create a "level playing field" that makes it as easy for small software and data providers to cater to the government's ICT needs as it is for large corporations. The model mirrors that of Google, with the server farms being run by the state, while competition is fostered and encouraged between service providers. This distinction between infrastructure and services is not new in the UK, with Network Rail being a monopoly maintaining the infrastructure while National Rail is an association of train operating companies

that provide services and, in an ideal world, compete with each other. In addition to a government app store and the servers to support apps, a means of finding apps (that is, services) needs to be provided that is unbiased and transparent. Looking at commercial search engines one does not see this. For example, Google's search algorithms are a closely guarded secret. In part that is enables them to provide a unique service, but it also means that they can bias results (for example, a company could pay to have its ranking raised for particular search terms). A government services search engine might include criteria that are valued by the open source community – licensing arrangements and backward compatibility being two examples.

8.2. National Security

As described above, it is difficult to see how Google could conceivably not exist in five or even ten years, but this does not mean access to Google services is guaranteed, either legally or in practice. Although not such an issue in the west, this could be a real issue if, national interests do not align with the US Federal Government. Google is expected to, and does, filter search results tailored not just for individuals but also for countries such as China. In 2008 there were submarine cable breakages that greatly reduced Internet access to much of the Middle East except for Israel and Iraq. This coincided with Iran's move to trading oil in Euros rather than dollars, which lead to conspiracy theories in the Wall Street Journal [19]. In 1988 Saddam Husain was allied with the United States in its war with Iran [20] and in 2009 the Libyan government of the day had come in from the cold [21]. Should Zimbabwe be using Google Docs in place of MSWord to run its infrastructure? It is probably not in their national interest irrespective of the politics. As these “resource wars” [22,23] heat up, should Europe be assuming its national interests would always align?

8.3. Carbon trading schemes

This is an aerial photo of the “chicken coop” data centre for Yahoo in Lockport, New York [24].

Figure 2. The Yahoo! data centre in Lockport, New York

These data centres are power intense, an average of 3.3 kW per square metre [25], and including the power consumption of desktop computers, laptops and smart phones on the client end, the carbon foot print of IT exceeded that of the aviation industry for the first time in 2007. As reported in the Telegraph on the 9th of January 2009 (and countered 5 days later in the same paper) two Google searches, on average, produce as much carbon as boiling the kettle.

Server farms consume huge amounts of electricity but nothing compared to factoring a desktop computer for every home, plus another for every worker. As a guide, the following figures come from the sust-it website: (c.f.[26])

Desktop	power used when idle can range from 10W to 315W
Laptop	power used when idle can range from 5W to 22W
LED monitors	power used 'in use' ranges from 17W to 127W.

Server farms are an opportunity to make economies of scale and indeed an opportunity to distribute IT costs and utilise IT infrastructure. Imagine if every rural community had a mini server farm instead of a telephone exchange. The network traffic would be highly distributed and remove bottlenecks, providing a highly "robust" IT infrastructure, and providing those communities with an economic justification for being connected with high-speed Internet connections. Indeed, in the same way as waste heat from steel production has been used to heat houses, perhaps server farms could be distributed even further becoming a local industry with servers being built into the very fabric of our buildings and heating them.

9. The One Recommended Standard: Enforce Published APIs

The vision is that the FOSS community be given the resources to develop unlimited apps. The FOSS community could thus compete with commercial interests in the provision of software for public administration. Rather than impose standards on the data stored in the cloud, the one requirement for data stored on the gCloud would be that the API is public and clear. Note that such a scenario is not in the interests of many existing software developers. These corporations can maintain market share despite a poor product because they control access to the data. The primary recommendation presented here is that control of State run server farms is leveraged to ensure that data stored on them is accessible to all, with the right to access it: freedom of information.

The APIs might mirror existing W3C standards but, following the practices of the open source community, these standards should be allowed, even expected, to evolve via a community process. For instance VoiceXML provides data in "forms" and "fields" that would seem to have very little to do with voice browsing but a draft API for voice based access to data should start with an API that has methods called "getForm" and "getField".

10. An Example

Booking an appointment with the doctor can often be carried out on the phone or from a website, but the interface is often less-than-perfect. A typical practice would like to have a better interface, but their software provider has little interest in improving the interface because that practice has their standard off-the-shelf product. The better interface that the practice administrator has in mind looks very much like the Doodle interface [11] – a widely used cloud service. There is little work in writing the app that would provide such a service; the challenge is the server-side software. What are the blockages now, what are the issues using Google cloud, and what would an ideal situation be, as visualised for the gCloud?

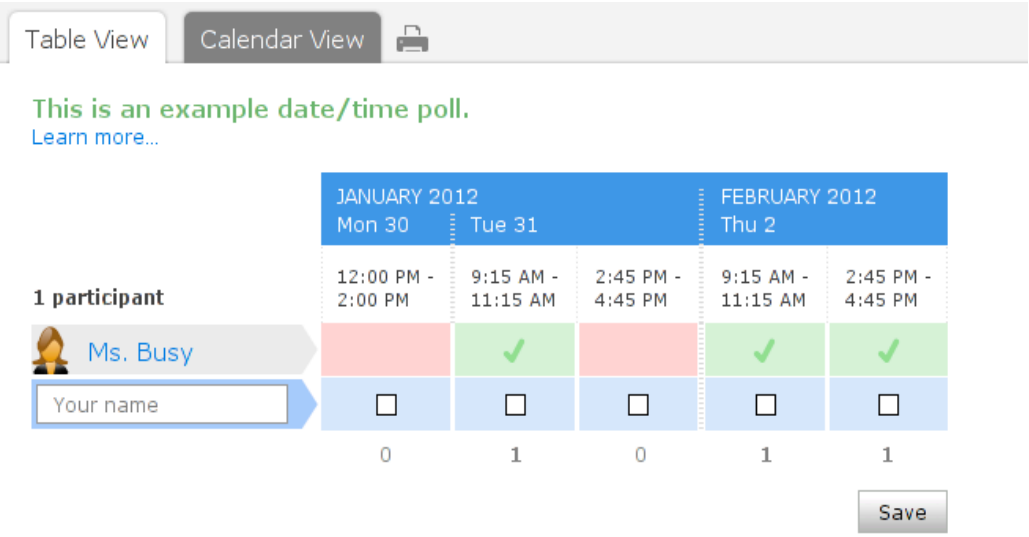
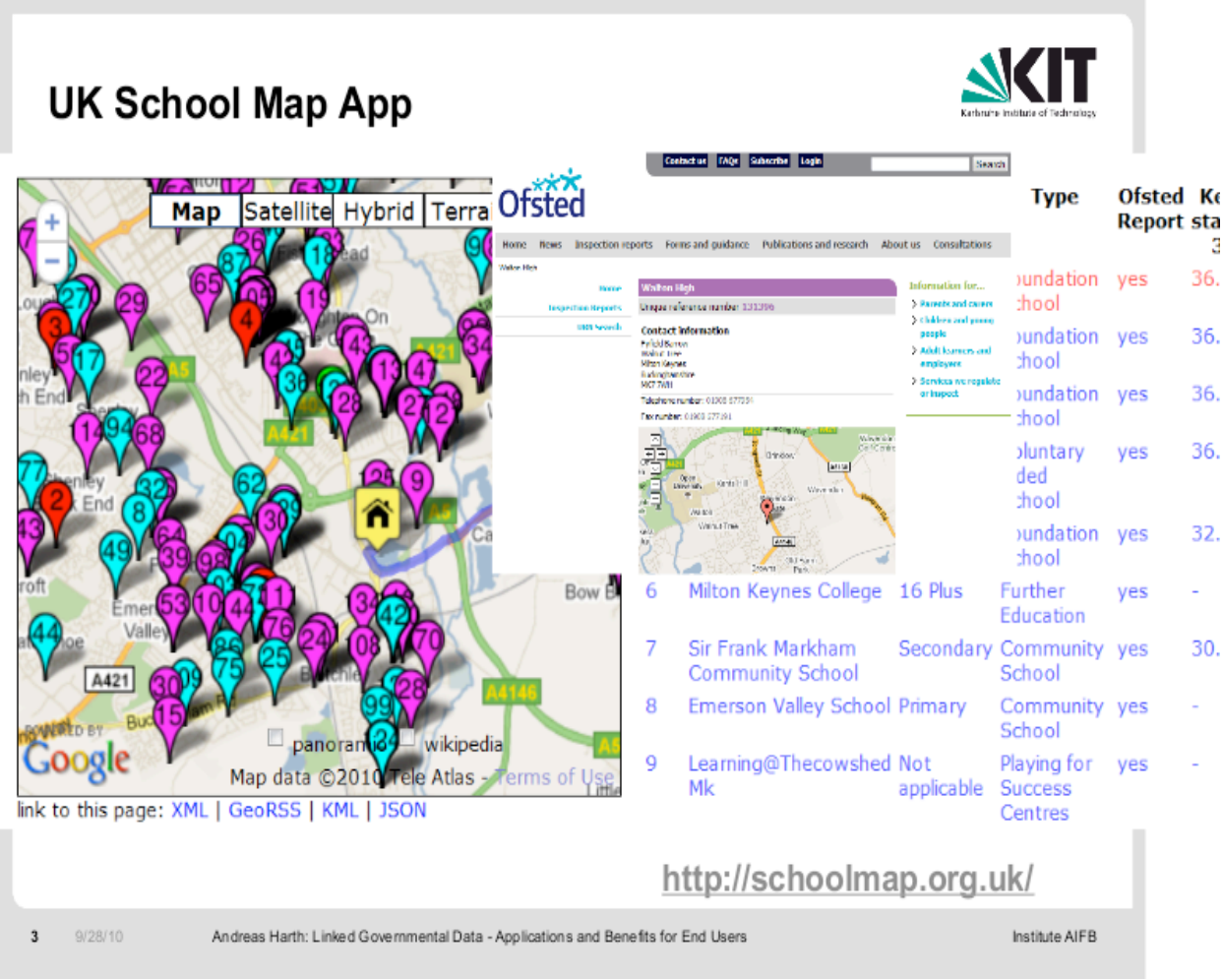


Figure 3. The Doodle meeting-scheduling interface.

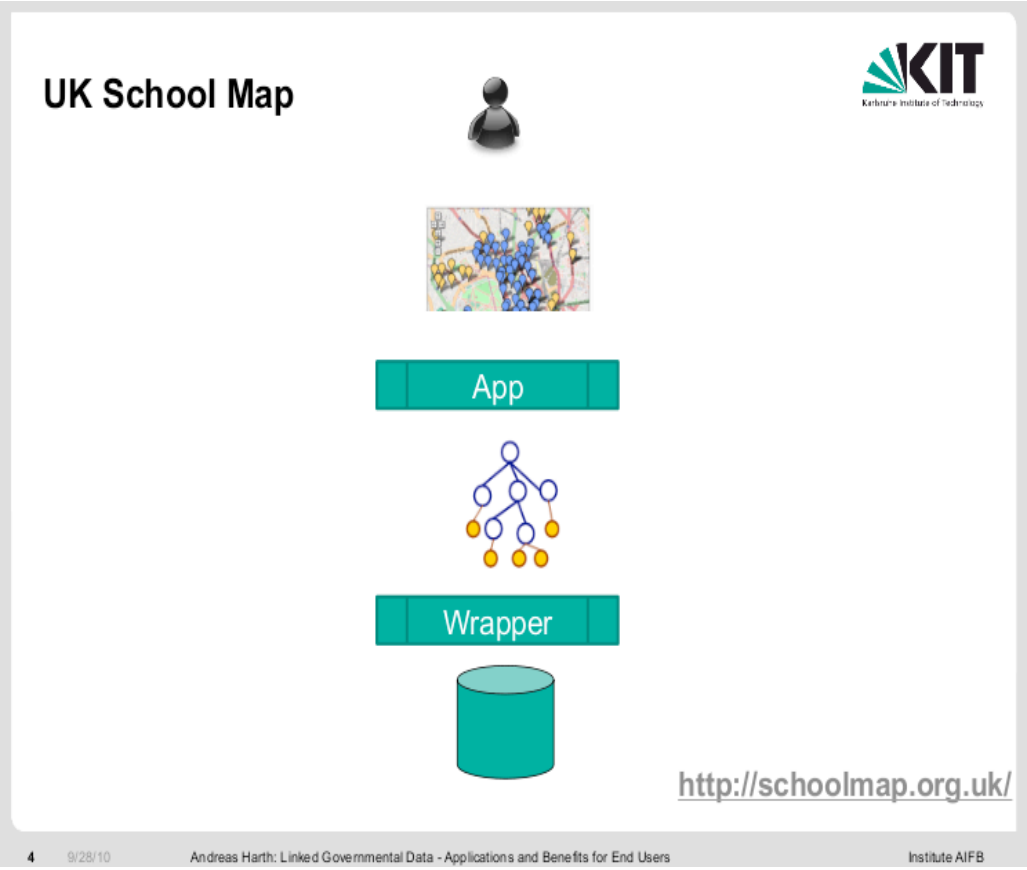
Using the Doodle interface to arrange a meeting with Ms Busy, I would add my name and tick the box in the square under one of the green boxes in her row, and then press submit. That is it. For the medical practice an app on a user's phone would need to know the doctors' names and when they are available. Upon the user clicking the submit button, the app would add an entry in the server-side database storing the practice's appointments. There would need to be some checking – a rival software company or practice might set up a denial of service attack adding bogus appointments – as will be discussed later.

Providing such an app is currently not possible because, at the moment, the appointments data is stored on a computer at the practice itself, along with all the patient records and employee data. In order to access the data for appointments, I would have access to patient records and so need to gain permission to be trusted with such data. To gain permission I need to be a software provider partner. To become a partner, as mentioned above, I need to have an existing customer base. So much of the competition for supplying the practice's software is locked out.

In the proposed scenario the practices data would be held in the UK Government's gCloud and be accessible to all through an API. As an example, consider another app [c.f. 27] that uses OFSTED [28] data to provide data about which schools are available in your local neighbourhood.



This app allows you to type in your postcode and choose your house, and then it provides a Google map with the local schools displayed along with a table of relevant statistics for each. The data is a view on an existing OFSTED database that has been “wrapped” to provide an interface to the Internet.



The primary recommendation above is that the Application Programmer Interface, or API, that describes what that wrapper can do is written and documented in a standardised format, and that the publishing of such data on the gCloud is enforced. Of course not all government data should be available to everyone and this raises the issue of trusted authentication and authority, issues that can be handled at the Platform level.

Lets assume that the smart phone I am using is mine, and I have set it up to, on request, give my full name as a means of identification. The server side API for the existing practice has a list of people registered with the practice. To make an appointment, only requires that my name (which is supplied by the phone) match one on the list of registered clients. Likewise to cancel

an appointment; but to see all my past appointments might require a further level of security such as my National Insurance number for instance, which my phone was not authorised to surrender automatically. From a public terminal in a library, I could make an appointment, but I would need to enter my full name by hand. Whatever the device, some data might need to be typed in every time for security purposes, and, ultimately, some level of security may require dedicated hardware such as the finger print scanner available on some laptops or an electronic one-time-pad as being used by some banks.

This is the ideal scenario and relies on data being held at an *appropriate* level of security. This requires government intervention. Left to the goodwill of the data owners, the pressure is on them to continue to use access as a means to lock out the competition. It is not enough for Google to make it *possible*; access to the government's gCloud is an opportunity to provide a level playing field without legislation.

11. Recommendations

The open source community has evolved many of the tools and practices that would make cloud computing a viable future, and Google provide a model for the infrastructure. The Open Source community has tended to focus on operating systems and desktops because has been the underpinnings of what computers *are*. In ten years, hacking operating systems will go the same way as building computer hardware from discrete components. The next generation of open source developers will hopefully carry on the current open source ethos but move to new areas, which are likely to look more like Google's ventures.

The recommendations are that the FOSS community should:

1. Embrace the “Software as a Service” model of cloud and look to supporting public administrations by investigating and providing "apps" for mobile devices and desktops.

2. Encourage Government to foster free and open server farms (infrastructure) that provide a platform (PaaS) for open source development, including sound models for data security and user identity.

3. The FOSS community should continue to develop its existing community process to encourage interoperability, rather than developing and using standards for machine-readable data (the W3C model).

4. The FOSS community should actively set out to identify the use of “unfair advantage” by software vendors, and actively encourage the Government as owner of the servers to remove them from the cloud.

5. As a means of finding apps, the Open Source Community should set up its own search engines to ensure that big companies don't get bigger just because they are bigger (such as by Google's PageRank [29] algorithm).

12. References

1 - OSEPA short summary

2 - http://en.wikipedia.org/wiki/Cloud_computing

3 - http://developers.sun.com/cloud/

4 - http://www.apple.com/uk/iphone/from-the-app-store/

5 - Philip Greenspun,"Philip and Alex's Guide to Web Publishing", Morgan Kaufmann, ISBN/ASIN: 1558605347 (2003)

6 - http://www.connectingforhealth.nhs.uk/systemsandservices/gpsupport/gpsoc/summary

7 - EMIS partner programme enquiry, June 2011.

8 - http://en.wikipedia.org/wiki/Java_Community_Process

9 - http://www.informationweek.com/news/software/operating_systems/225400063

10 - http://www.guardian.co.uk/world/us-embassy-cables-documents/214462

11 - http://www.doodle.com/main2.html

12 - http://www.guardian.co.uk/technology/2009/nov/26/dark-side-internet-freenet

13 - http://www.w3.org/

14 - http://www.hmrc.gov.uk/ct/ct-online/file-return/switching.htm

15 - http://en.wikipedia.org/wiki/Embrace,_extend_and_extinguish

16 - http://www.theage.com.au/technology/security/evil-the-hacker-refused-bail-over-nbn-plot-20110727-1hzdk.html

17 - http://openid.net/

18 - http://oauth.net/

19 - http://www.marketwatch.com/story/middle-east-internet-interruption-looks-fishy

20 - http://en.wikipedia.org/wiki/Halabja_poison_gas_attack

21 - http://en.wikipedia.org/wiki/Gadafi#cite_note-224

22 - http://www.amazon.co.uk/Resource-Wars-Michael-T-Klare/dp/0805055762

23 - John Reid (Defence Secretary address to Chatham House, 2006)

24 - http://www.datacenterknowledge.com/archives/2011/12/12/inside-yahoos-chicken-coop-data-center/

25 - http://cuppcomputing.com/global-pc-power-consumption/

26 - http://www.ethicalconsumer.org/buyersguides/computing/desktopcomputers.aspx

27 - Andreas Harth http://www.w3.org/2010/09/egov-session-ict2010/

28 - http://www.ofsted.gov.uk/

29 - http://en.wikipedia.org/wiki/PageRank

30 – HM Government, August 2011, Making Open Data Real: A Public Consultation