The Dataware Manifesto

Derek McAuley, Richard Mortier, and James Goulding

***Abstract*—In this paper we concern ourselves with Service-Oriented Architectures (SOA) in the "business to consumer" (B2C) arena. In particular we consider the services required to enable consumers to combine data they possess with data held about them by businesses and government. We introduce the concept of *Dataware* as the logical federation of data sources containing "my data" and discuss an SOA to deliver new and compelling services and applications able to reap the benefits of *value-in-use* for consumers.**

***Index Terms*—federation, distributed computing, value chain.**

I. Introduction

TODAY, individuals have unprecedented access to information over the Internet, whether public information from government and businesses, or private information such as bank accounts, utility bills, and email. Much of this information is derived from the data that governments and businesses have collected and hold about and for us.

Modern life involves each of us in the creation and management of data. Data about us is either created and managed by us (e.g., our address books, email accounts), or by others (e.g., our health records, bank transactions, loyalty card activity). Some may even be created by and about us, but be managed by others (e.g., government tax records). All of these data sources are commonly referred to as "my data," which can itself lead to debate over what it means to "own" data; notwithstanding this, we will continue with this common usage and refer to "my data" throughout.

Increasingly my data is available to me through web pages accessed via the ubiquitous web browser, whether on a desktop computer, smartphone or other device, and destined for human consumption. The holder of the data chooses how to represent the data to me and presents it through a graphical interface — a fundamental architectural design principle in many services built using a three-tier architecture. Most B2C transactions are performed through this interface and it has been very successful.

However, the very same network technologies underpinning a user's Web experience have also facilitated "business to business" (B2B) transactions, allowing automation of a large range of mundane, everyday operations.

Manuscript received November 18, 2010. This work is supported by Horizon Digital Economy Research, RCUK grant EP/G065802/1.

D. McAuley, R. Mortier and J. Goulding are with Horizon Digital Economy Research at the University of Nottingham UK. (firstname.lastname@nottingham.ac.uk).

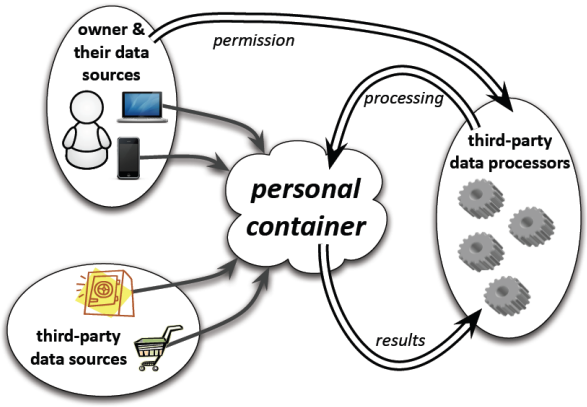

Figure 1. The *Personal Container*, a virtual entity enabling users to manage their digital contextual footprints.

A large fraction of worldwide commerce is now performed directly between suitably authenticated computer systems operating on behalf of businesses, with people involved only at critical decision points. Here SOA is the guiding principle, as the communication must be performed between computers without human intervention.

We are interested in enabling the same transformation for consumers and B2C transactions; that is, we are looking at the architecture and protocols that will underpin a new wave of personal digital services and applications enabled for individuals. This is made possible by my data becoming available to software and services running on my behalf, rather than simply being presented to me via a graphical display. For clarity we refer to these applications and services as *third-party data processors*.

Personal Containers [1] is a project investigating how to build an ecosystem around my data, supporting provision of novel, desirable applications and services by new and existing businesses. The Personal Container concept is depicted in Fig. 1. Achieving these aims poses a number of human, innovation and technology challenges; in this paper we focus on the technology, where we face two important challenges: (*i*) how to federate these disparate data sources, and (*ii*) how to build infrastructure that enables *me* to control use of *my* data.

To elaborate, the key technical problem in supporting an ecology around my data is not one of containment ("how can I archive all of my data?") but of access ("in what format is my data presented, and how do I authenticate to use it?") and control ("how can I control what happens to my data?").

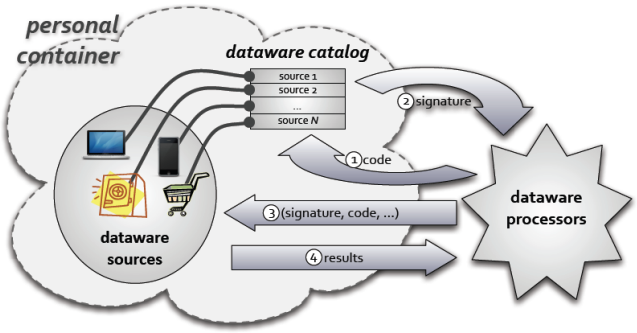

Figure 2. Dataware architecture, depicting the main interactions between third-party dataware processors and a user's personal container.

Aspects of this problem certainly relate to who should store the data and how, but the focus is on controlling: who gathers, processes and distributes my data; when and to what purpose this occurs; and the means by which I can access it and enable processing applications and services to access it on my behalf. The matter is further complicated by the basic property of digital data, that it can be infinitely copied without loss of fidelity: once my data escapes my immediate purview, I cannot easily exercise further control over it. However, in order to generate significant value from my data, I must allow others to access it. Retailers benefit from recording customer loyalty data, and in return offer me in-store vouchers, as they can extract knowledge and value from my purchasing behaviour.

II. BECOMING DATAWARE

So we have a situation where my data is collected and held by third party data sources and, being digital, can be distributed widely at little cost. One way to give me back control over my data would be to force these data sources to send all the data they collect to me, and forbid them from keeping or distributing it themselves. Although this approach would certainly work in the sense that I would recover control of my data, it would not work very well! In practice while I could maintain, for example, my own detailed shopping records, across all stores and providing more semantically rich detail of my purchasing than is available to an individual retailer such as Tesco or Sainsbury, I don't. The reason for this is that it would place a significant and – in the case of such things as financial records – possibly legally unavoidable burden on me to store and manage my data. In addition it makes it significantly more difficult for those collecting the data to extract sufficient value to justify the costs of that collection.

An alternative is to take a federated approach made natural in such a distributed system: provide interfaces and other software mechanisms to enable me to exercise control over my data and its use, while not requiring me to take sole ownership of it. We refer to this as being *data-aware*, and the software systems engendered as ***dataware***.

This need to share responsibility arises for a variety of reasons:

- I do not want to put all my data in one place lest it be attacked or stolen.
- Furthermore, I do not want to put all my data in the hands of a third-party cloud as they may place it in jurisdictions that have little data regulation in place.
- I do not want to put all my data in my home as that means I need to worry about maintaining it and backing it up.
- Many of the organizations holding my data (e.g., financial records) are required to do so by law, so why should I waste disk space and other resources when I have paid for them in my monthly bill!

I may *choose* to archive my data on a device or devices in my home or to entrust it all to a single cloud provider. However, given that third parties collect the data in the first place, the basic functionality I require is the ability to control processing of my data wherever it is stored. This general architecture is depicted in Fig. 2, with the Personal Container shown as a virtual "cloud" entity through which access to a user's data is controlled.

In summary, being dataware means providing mechanisms that allow me to become involved in the processing of my data by third-party processors. This enables existing data sources to continue to collect and process my data without requiring that I take sole responsibility for my data. It also allows existing third-party processing of my data to continue, while also enabling the development of new processors that make use of the broader range of data sources now available, and engendering so-called "competitive co-creation of value" [2].

III. SUPPORTING DATAWARE

To support dataware requires several infrastructural elements, which we will now describe in turn.

A. Formats

Data must be presented in formats and through interfaces that are standardized and open, in order to process it efficiently with software, in line with the guiding principles of Linked and Open Data movements [1]. This refers not only to the specific formatting of the data itself, but also the need to provide common means for user identification and access control.

This transformation from human-centered web pages to software accessibility for the underlying data is already well underway in other areas: for many years this has been the trend in the academic and commercial sectors. The data.gov.uk and data.gov repositories, in the UK and USA respectively, are great examples of this trend in the public sector. In pursuit of the goal of freedom of information and transparency of government, these sites release data gathered or generated by government or on government's behalf, in forms that can be directly processed, enabling new and innovative uses and representations. The availability of open data is enabling the wider community to bring their creativity and ingenuity to bear, whether it be in applications such as the

[1] See http://data.gov.uk/ and http://linkeddata.org/

estate agent's nightmare, the ASBOrometer,[2] which shows anti-social behaviour statistics overlaid onto a smart phone map, or in sites such as www.seeclickfix.com, which provide reporting to government on issues of concern to citizens. The same creativity and ingenuity can also be brought to bear to build very personal applications that process our private information, if only we can arrange to supply the software with the data and to ensure that the software runs somewhere that we trust.

Of great benefit to the implementation of the dataware architecture would be the use of standards formats (e.g., RDF[3]), and the definition of a set of common data models appropriate for different types of information.

B. Owners

I need mechanisms to help me federate access to the many data sources that create and collect my data, and to ensure that access by data processors is safe.

In more detail, I wish to exercise control over my data even though it is collected and stored by many organizations, and may be processed by many other organizations. Thus I not only need to maintain a catalog of my data sources, but I also need the ability to delegate and record access to my data sources. In order to do this, there also needs to be some standard way of describing this access, i.e., the computation that will be carried out, and my policies on such access so that an informed decision can be made on my behalf.

C. Processors

Data processors need to know where my data can be found, how to ask me for permission to process my data, and a means to express computation across my data so that I am happy about its security.

In more detail, this means that there needs to be some way that a data processor can come to me to request specific access to my data, i.e., they need to be able to locate and interact with the catalog described above. At such time as they request access, they also need to be able to represent to me what access is required by the computation that they wish to carry out. In most cases this clearly cannot be simply showing me a concrete representation of the computation, i.e., a piece of code. Thus, some mechanism is required to represent the effects of a piece of code, and to tie that representation to the code itself. Further, this signing process had best be automatable as manually determining the effect of a piece of code is complex, likely to be error-prone and to become a bottleneck to the growth of the ecosystem if carried out manually.

D. Sources

Data sources need mechanisms to enable them to notify me that they are sources of my data, to verify access to my data by third-parties, and to support third-party processing of my data.

In more detail, this means that there needs to be some protocol enabling a service (or services) that can lodge details of my data sources. Such details need to include some information about the data source itself, both syntax and semantics, in order that processors can write code to process the data. These, or other, services also need to be able to verify access delegated to processors. Finally, there needs to be some mechanism for hosting computation that presents a standardized interface so that processors can write such code.

IV. Using Dataware

Once we provide for access by individuals to their personal data, we enable a vast array of applications across many sectors. For example:

- *Government taxes*: traditionally filling your annual tax return was, and for many still is, an operation of transcription from a motley collection of pieces of paper on to the government supplied tax form. Today, with access to online bank accounts, personal employee records on the company intranet, accounting software and the web-based tax return systems, we have reduced the paperwork. However, the process is still quite tedious and nearly as time consuming for the citizen as information is transcribed from one program to another by "cut 'n' paste — the drudge work of the information age! In order to empower individuals, society at large needs to undergo the same style of transformation that has swept through the commercial sector and use computing technology to perform the tedious clerical work.
- *Personal healthcare*: my personal healthcare assistant might enable me to scan barcodes of retail medication using my mobile phone camera, offering me advice and warning me of potentially dangerous or simply unpleasant interactions with my prescription medications, even before I buy. Doing so requires coupling my personal medical record and prescription information, together with a record of my purchases (already recorded by many vendors within their customer loyalty programs), and a service offering information on drug interactions from a reliable pharmacy company.
- *Financial security*: a personal service that knows my location, derived from a multiplicity of sources (cell phone, GPS, travel itinerary, calendar, etc.), could be configured to respond to specific appropriate enquiries from my credit card company. For example, when a credit card transaction is attempted in Tokyo, I might permit the very specific request "Is Derek in Tokyo?" If I am in Tokyo using my credit card legitimately, I am already implicitly conveying my location to the credit card company, as they know the merchant's location; on the other hand, if it is a fraudulent use then sharing the information that I am not in Tokyo does not excessively infringe on my privacy.

Importantly, making available the data to software running on the individual's behalf avoids many legitimate privacy issues and concerns; by focusing on bringing the data together from disparate sources under the control of the user, we provide the opportunity for genuine informed consent.

For example, I may have different sign-up details with several retailers, on- and off-line, with my bank and credit

[2] http://www.absorometer.com
[3] http://www.w3.org/RDF/

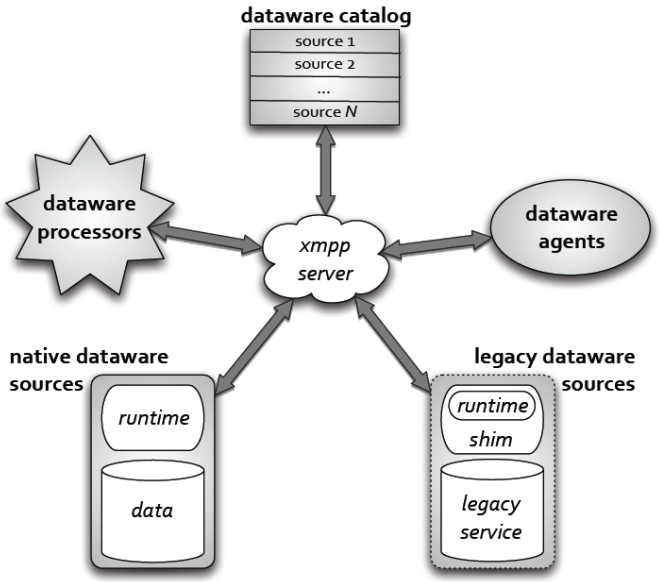

Figure 3. Dataware implementation framework, showing inter-connection of the major components via XMPP.

card companies, with my health provider, and so on. I not only need to manage all these accounts for myself, I also need to be able to delegate permission to process my retailer accounts (but not financial or health accounts) to the "OptimizeMyShopping" app, and to my financial accounts (but not retail or health accounts) to the "OptimizeMyBanking" app. This delegation needs to be sufficiently fine-grained to distinguish the two apps even if provided by a single organization.

The decision whether to permit this delegation must be informed (I must understand what each app will do with my data) and, preferably, permit automation (having expressed a policy, I need not be interrupted every time access is requested or takes place). Finally, I must be told periodically, say every month, what access to and computation across my data has taken place, by whom and what happened as a result. This information should be presented usefully, i.e., aggregated when uninteresting but with sufficient detail available that I can drill down into the details when necessary.

From the point-of-view of the processor, in order to express such a computation I must be able to discover what data sources are available for this user and verify that those sources are supported by my application, i.e., have suitable syntax and semantics to be process by the code I have written. I also require some way to express in my computation the privacy impact it will have so that the owner's policies can be applied.

Finally, from the point-of-view of the source, I need to be able to provide an interface allowing users to sign-up to have their data made available to third-parties, and to be able to host data streams for signed-up users. I also need to provide a runtime environment for processing users' data that enforces the claims made about computations it hosts, whether those computations are written by a third-party or by the source itself, and provides an acceptable audit-trail for users and possibly others such as government agents.

V. BUILDING DATAWARE

This section outlines technical details of our design, as well as describing progress to date in building an instantiation of the dataware architecture. The implementation framework is depicted in Fig. 3. Each user's dataware exists to give them the ability to selectively and controllably delegate access to their data to third-party processors without giving away the data itself. This allows applications to be built that help the user to understand, visualize and control their many sources of digital data. Their dataware consists of a single catalog and a collection of sources, interconnected using XMPP. As a secure, extensible, real-time communications protocol supporting messaging, presence,[4] as well as a simple RPC mechanism, XMPP provides an ideal transport protocol for this federated system [3].[5]

A. Catalog

The catalog maintains presence subscriptions with each source so that if either party goes down, the other will immediately be notified and, upon recovery, reconnection will be automatic. The dataware catalog can be a hosted service or run on a privately owned server, depending on the degree of trust a user has in the relevant service providers. XMPP communications are secured and one task for the runtime is to control outgoing communications to the XMPP network so as to prevent information leakage.

Each catalog maintains a selection of tables concerning the user's sources, installed applications, and permitted consumers. The *jid* is the user's Jabber ID, their identifier for XMPP communications. These tables include the following:

- ACCOUNTS: *jid*, *service-id*, *secret*, *metadata*. This enables the catalog to track the user's accounts on their various dataware sources. Account metadata includes details such as the activity, last access, etc.
- APPLICATIONS: *application-id*, *component*, *signature*. This enables the catalog to record the details of each installed application, including the particular signature computed for each application component.
- INSTALLS: *jid*, *processor-id*, *application-id, signature*. This enables the catalog to track the applications that the user has installed from different dataware processors.
- SERVICES: *service-id*, *metadata*. This enables the catalog to provide a discovery service to dataware processors enabling them to find out whether or not this user has any dataware sources providing data of interest to the processor.

In addition, the catalog maintains an audit log of operations, so that it can provide the user with a complete history of the interactions that third-parties have had with the user's data.

B. Sources

Each source performs four functions for each user:

[4] E.g., available (online), offline, busy, etc.
[5] http://www.xmpp.org/

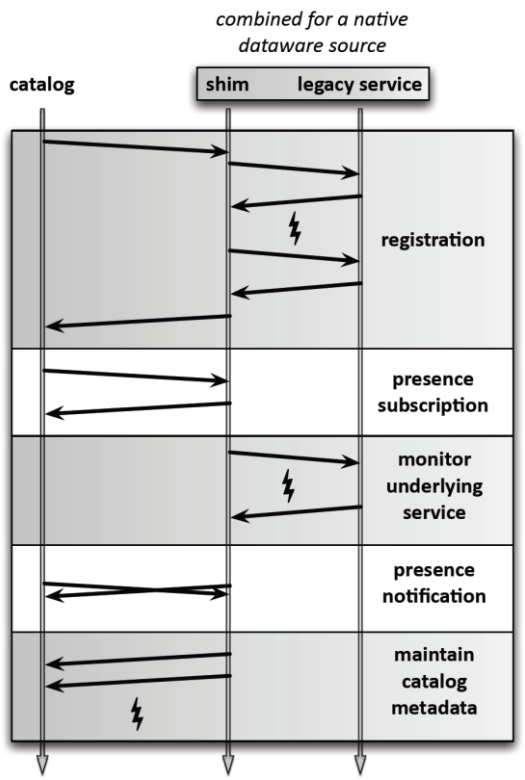

Figure 4. Dataware protocol interactions between the catalog and a legacy dataware source.

- It maintains data from their service, whether or not it is a native dataware service or a shim over a legacy service such as Facebook.
- It supplies metadata concerning the data they hold to the catalog, describing its liveness and content.
- It maintains a shared secret for securing communications on behalf of that user.
- It provides a runtime for execution of consumer application components.

Sources come in two flavours: *native* sources are built to support dataware from the ground-up, allowing them to be more efficient. In contrast, *legacy* sources consist of a shim built against an existing online services such as Facebook or Twitter. These shims may be hosted by the user or by some third-party service providing them for many users. The protocol interaction between the catalog and a (legacy) dataware source is depicted in Fig. 4.

C. Processors

Dataware processors are entities that wish to process users' data. They do so by submitting applications to users' catalogs for installation; if a catalog allows an application to be installed, the processor subsequently submits the application and its cryptographically generated signatures to the relevant sources. The source then executes the application until it is uninstalled. A wide variety of processors are envisaged, with some interested primarily in data mining large populations, while others provide specific services to users in exchange for access to data or monetary payment.

D. Agents

Dataware agents operate on behalf of the user, providing them with functionality requiring direct intervention in the control paths between catalog, sources and processors. Implemented as XMPP bots, they support features from Privacy Butlers [4] to Biometric Daemons [5].

E. Applications and the Runtime

Processors introduce applications by negotiation with the user via the catalog. This negotiation involves the processor first using metadata from the catalog to determine if the user has suitable sources to run the application. If so, then the processor asks the user to sign the application so that the sources will execute it. Each application is represented as a list of components, each of which must be signed, consisting of:

- A source identifier, specifying the source and account to execute this computation;
- A computation, i.e., a piece of code;
- A destination, to which the result of the computation will be sent; and
- A schedule, according to which the computation will be executed.

If the user is willing to permit the application to access their data, the catalog signs each application component using the secret shared with the relevant source. This signature prevents the processor subsequently modifying any part of the component: the user permits exactly that computation to execute against their data according to the specified schedule, storing the results at the given location.

Once signed, the processor presents the components of the application to the designated sources, which verify the presented signature and if valid, execute the computation according to its schedule, sending the results to the given destination. As the computation executes, the runtime logs data accesses, enabling a complete audit log of data use to be presented to the user. If subsequent JOIN-style operations are required, the destination can be specified as a trusted third-party source, which the user is happy to permit to hold, potentially highly detailed, intermediate results. For a given user, such sources have similar security requirements to the user's catalog so co-locating such a source with the catalog provides at least one such rendezvous point.

F. Interfaces

Dataware components provide a range of APIs enabling consumers to interact with users' data, as outlined below.

- *list-sources(user) = source-metadata*

  This enables data processors to obtain metadata from the catalog about the dataware sources enabled in that particular user's personal container.

- *install(application) = application-id*

  This is called by the data processor on the catalog. It asks the catalog to sign the presented application so that the data processor can have it executed by the relevant data sources. The catalog essentially acts as mediator between the data processor and the user, either applying a pre-defined policy (e.g., "accept all applications originating from tax authority") on behalf of the user, or contacting the user for their direct involvement. The resulting application-id serves both as proof that the user is happy for the application to be run according to the specified schedule, and as an identifier for the catalog to track which applications have been enabled.

- *uninstall(application-id)*

  This allows any suitably authenticated user interface to remove permission from an installed application. When it is invoked, the catalog will instruct each source on which the application has been installed to remove the application. Once this is done, the data processor that installed the application is notified that it has been uninstalled.

- *result-ready(signature, sub-id, timestamp)*

  In addition to storing the result as directed by the application, when a dataware source completes a particular execution of an application it notifies the relevant processor that the result is ready. The invocation is made by the relevant source, and the parameters then uniquely identify the result: the signature specifies the application, the sub-id allows an application to contain more than one piece of code for a given source, and the timestamp is the start time of the execution as recorded by the source.

In addition to these programmatic interfaces, dataware components provide a number of user interfaces. Among the first user-perceived benefits of this system is likely to be simply the ability for the user to manage their online accounts via a single site, their catalog. Dataware sources also provide simple interfaces for the user to manage the connection of the source to their catalog. In the case of native dataware sources, this presentation is under the complete control of the source implementer. For legacy dataware sources, the underlying service will constrain the form that the interface can take.

G. Progress to Date

We are in the process of implementing the Personal Container within the Dataware architecture. Current status is that a prototype catalog has been built, along with a shim that exposes Facebook as a legacy dataware source. Construction of the first native dataware service, and the first simple data processors and applications, using manual signing, is underway.

VI. Living with Dataware

In the UK, the Power of Information Task Force [6] reported on the economic gains that can be achieved from better use of government data. Likewise the Open Government Directive [7] in the USA aims to provide greater transparency in government through availability of government-collected information to citizens. This has lead to initiatives such as data.gov.uk in the UK, data.gov in the USA, and similar initiatives in other countries that seek to deliver on the vision presented.

Another important source of data underpinning the future digital economy is the wealth of personal information that exists but is currently inaccessible in a form suitable for personal applications and services to use. An important role for government in this area is to guide industry to achieve open standards to foster competition in the market, both to avoid the creation of data monopolies and open the market for innovative applications. In fact in a global economy this requires an international standardization effort, and herein lies a major challenge.

In particular, all the example applications presented exploit our existing "digital footprints". However, on the horizon are new sources of personal data that will simultaneously enable interesting new applications but also provide even greater privacy challenges, e.g., personal biosensors, smart meters, etc. All have the possibility to be used for nefarious purposes, and within the digital economy research programme we aim to understand and address the societal issues raised by this growing, and increasingly detailed, digital record to appropriately guide future technical development of dataware.

References

[1] R. Mortier, *et al.* "The Personal Container, or Your Life in Bits." Proceedings of Digital Futures 2010, October 2010. Nottingham, UK.

[2] C. Mulligan and R. Mortier, "Competitive Co-Creation of Value." Proceedings of Digital Futures 2010, October 2010. Nottingham, UK.

[3] P. St-Andre (ed.), "Extensible Messaging and Presence Protocol (XMPP): Core." IETF Network Working Group, RFC3920. October 2004.

[4] R. Wishart, D. Corapi, A. Madhavapeddy and M. Sloman. "Privacy Butler: A Personal Privacy Rights Manager for Online Presence." IEEE PerCom Workshop on Smart Environments (SmartE '10), March 2010.

[5] P. Briggs and P. Olivier, "Biometric Daemons: Authentication via Electronic Pets." Proceedings of CHI 2008, Florence Italy, April 5-10. ACM Press.

[6] Power of Information Task Force, February 2009, Cabinet Office, UK. Now available through online National Archive http://webarchive.nationalarchives.gov.uk/

[7] Open Government Directive, December 2009, http://www.whitehouse.gov/sites/default/files/microsites/ogi-directive.pdf