Enforcing scientific data sharing agreements

Michael Wilson, Shirley Crompton, Brian Matthews
STFC e-Science Centre,
Rutherford Appleton Laboratory
Oxon, OX11 0QX UK
{michael.wilson, shirley.crompton, brian.matthews}@stfc.ac.uk

Alexey Orlov
European Microsoft Innovation Centre
Ritterstrasse 23
Aachen, Germany
alexyo@microsoft.com

Abstract—There are currently strong drivers from funding bodies for the scientific community to both preserve data and make it available more widely, whilst ensuring that privacy and confidentiality are maintained. Institutional data sharing policies will be required to encapsulate the response of individual institutions to these pressures. When organisations share data they adopt these policies into data sharing agreements between institutions which must then be enforced. The approach to enforcing scientific data sharing agreements presented here is based upon a combination of trust, leading-edge technology, licensing and legal frameworks. The scientific community must decide what balance between these components it wants to adopt.

Keywords-controlled natural language; scientific data management policies; policy based management; formal model checking; policy usability; formal methods usability.

I. INTRODUCTION

In recent years public research funding councils have been trying to maximize the return on their investment in science. To this end they have taken measures to ensure that scientific data is preserved for uses other than those of the investigator who created it [1] so that it can be: used by research peers to validate published scientific results; to be available for meta-studies and secondary studies which may identify phenomena which were too small to be significant in the original study alone; be available to set parameters and to test new models and simulations; to be available for new science not considered when the original data was collected. Once it is preserved for these purposes, research organizations need research data management policies and data sharing policies to establish the rights and responsibilities of the various users of the data.

Data sharing agreements (DSA) have become common in recent years between enterprises that want to share data. DSAs state who can use some data, for what purpose, when, where, how the data should be managed while it is being used, and how it should be disposed of at the end of the agreed period. DSA are legally binding contractual agreements between organizations which usually apply the data management policy of the enterprise contributing the data on the enterprise who wants to have access to it. They are an established tool used when one public service needs access to personal information held by another body about an individual to perform their function. They have also become common for environmental data between the responsible organizations in different jurisdictions, or between those who collect data and those who wish to analyze it. Sometimes the data sharing agreements refer to published data policies, while on other occasions they incorporate their text within the agreement – for simplicity in this paper they will be assumed to be the latter.

The Science and Technology Facilities Council (STFC) operate large scientific facilities for the UK including the ISIS neutron facility used to examine the structure of materials. Consequently, STFC stores Petabytes of scientific data that are collected on those facilities, which is made available to the research team who collected it, research peers, and others in accordance with DSAs between STFC and the other parties[1].

Many public sector DSAs cover data held on paper files, and are enforced through physical security on the files and the locations in which they can be used. Scientific data is stored digitally, can be transmitted to the user over the Internet, and can be analyzed using many programs which produce derived data. At STFC the data is cataloged using the iCAT data management system [2] which supports the discovery of data and its access. This paper addresses the problem of how scientific data sharing agreements can be electronically enforced through scientific data management systems such as iCAT. It presents one test bed of the Consequence project[2] which developed a general framework for using policy based access and usage control to enforce data sharing agreements. This paper addresses those aspects which are most significant to scientific data management leaving the details of the enforcement technologies to references to previously published papers in the specialized security literature.

The next section describes the content of scientific DSAs and the requirements they put on the expressiveness of a language to represent them. Section 3 describes how DSAs are created and enforced, and outlines the different actors involved, and their requirements on a usable system. Section 4 then addresses policy languages required to support the proposed approach. Section 5 describes an architecture proposed to enforce these requirements on the iCAT data management system, while Section 6 describes an example set of policies from a DSA and how the implemented system performed at

[1] e.g. the ISIS data policy can be found at http://www.isis.stfc.ac.uk/user-office/data-policy11204.html
[2] Consequence project: http://www.consequence-project.eu/

The research reported in this paper was partially funded by the EU FP7-ICT-2007-1 programme through grant 214859 to the Consequence consortium.

enforcing them. Finally, in conclusion, future work is considered.

II. SCIENTIFIC DATA SHARING AGREEMENTS

A study of scientific DSAs [7] showed that they usually contain clauses addressing legal and administrative issues, the purpose of the agreement and justification for usage, penalties, and constraints and obligations on data usage. We consider here how these clause types can be enforced.

A. Administrative Clauses

The clauses addressing legal and administrative issues define terms used in the agreement, the parties involved, the duration of the agreement, the process of reviewing and updating the agreement, and the applicable law of the agreement as well as signatures of authorized representatives of the parties. These clauses are mainly written in legal natural language based on contractual conventions and cannot be enforced electronically beyond identifying the parties and duration.

The clauses addressing the purpose of the agreement and justification for usage describe the purpose of the required data sharing in layman's language, and details of why general data sharing or data confidentiality principles (depending on the culture of the organizations involved) are being breached by this agreement. These clauses are written in natural language which to provide a context to a court resolution and cannot be enforced electronically.

The clauses which define penalties for breach of service level agreements (SLA) are regularly enforced automatically in cloud systems by adjusting accounting credits [11], but for DSAs only the containment action of preventing further access or usage to data can be automatically enforced as other penalties require legal actions.

B. Data Access and Usage Clauses

The clauses which define constraints and obligations on data access and usage, constitute most of the clauses in DSAs, and provide a fruitful subject for automatic enforcement. In presenting a framework for DSA enforcement, [9] have treated all DSA clauses as obligations, but the clauses in a sample of scientific DSA reviewed in this study contained clauses using all three deontic modalities of authorization and prohibition as well as obligation which will be considered here in more detail.

Authorization clauses include those familiar in conventional role based access control stating who can access what data when, but also where, based on the evaluation of the value of attributes of the subject, resource or environment. Scientific data is available to members of the research team that generated it immediately it is produced, but there is usually an embargo period before other groups can access and use it. To express these clauses in an enforceable way requires a policy language which captures the action permitted with attributes for the role of the actor which can be resolved to an identity, a time attribute and temporal operators to define times *before* and *after* which actions are authorized, attributes of the data over which the access is authorized, and location attributes over which geographical constraints apply.

The authorization clauses reviewed do not only authorize access but also usage actions [12]. In many cases, they only authorize transformations on the data to be undertaken using identified programs, where only a subset of the actions which those programs can perform may be authorized. One reason for restricting usage is to control derived data. The storage of derived data in iCAT has been described already [10], and an applicable method for securing derived data has also been proposed in the present study [8] which will not be detailed here. The second main reason for usage control is to prevent the re-identification of anonymised individuals in sensitive data by combining data sets. A common solution to this problem is to permit remote access only to a data enclave [6] where users can only perform operations on the data in the physical enclave – even though they may have remote access to it. A requirement in the present study is to allow users to download files on their own computing resources since the data volumes are so large that an enclave would require prohibitively large centralized computing resources. The price of supporting data download with usage control is that the data would have to be encrypted and the applications would have to include code to enforce the usage policies and decrypt the data, so standard analysis tools would not be supported. For data from the ISIS neutron facility available through iCAT maybe analyzed using the Mantid framework[3] which could be modified to adopt the proposed security solution.

Prohibition clauses are used to explicitly set the baseline policy for sharing data that no access is permitted without explicit authorization, although this can be set as the default in the enforcement implementation and does not need to be explicit as a policy. Prohibition clauses are also used to refine authorization clauses by providing exceptions to them. For example, authorization may be permitted to all registered users after the embargo period, except in stated countries, where the exception is stated as a prohibition on access from those locations.

Obligation clauses address issues such as confidentiality, and dissemination of data to third parties, which are resolved through the combination of data encryption and identity based authorization policies for usage, and obligations to take actions on events. Obligations to take actions can fall on the human user or on the system they use. The obligations on the human user cannot be enforced automatically, but obligations on the system to take actions can be enforced on either access events, or on arbitrary events. For example, the system can inform the principle investigator who created a dataset each time it is accessed by somebody who was not in the original project team. The common obligation to dispose of the data at the end of the agreement or after a set period is effectively performed when the access and usage authorizations expire, so this does not have to be explicitly performed as an obligation.

C. Limits of Enforcement

One limit on a solution based on encrypted data and usage control policies lies in the trustworthiness constraints on the infrastructure. Unless users operate trusted computing platforms, any threat to the approach through the operating system, code interpreters or other parts of the

[3] http://www.mantidproject.org/Main_Page

software/hardware stack below the evaluation executable would be open to attack. Although the current study has demonstrated the solution described below using a trusted computing platform, it is unreasonable to assume that users will adopt them. There are two arguments around this inability to technically enforce security at this level: firstly, that the data do not have sufficient value to justify the effort required to attack the solution in this way, and secondly that if such attacks were mounted, then the DSA includes legally enforceable penalties which should provide a further deterrent to doing so. Neither of these arguments are acceptable in the security community, but they are accepted by science managers using a risk-management based approach, who are learning to co-operate without guaranteed trust by using contracts as trust substitutes [4].

A potential limitation on enforcement by access and usage policies is that, since the variables in a policy must be evaluated when the action it applies to is made, the user's system must either be on-line or a cache must be provided to store values when off-line. This solution to off-line access is a development of the license based approach commonly used in digital rights management (DRM) to protect music and other commercial media [13].

D. *Requirements for Policy Enforcement*

From this analysis of scientific DSAs several requirements have been derived on the expressiveness of a language to specify policies, and on the architecture which will be required to enforce them. The languages need to be able to express: the modalities of authorization, prohibition and obligation; actions including access and sub-types of usage where the subject can be the user or the system, with attributes for user role which can be mapped to user identities; data types organized in hierarchies; current date/time and specified date/times; temporal relations of *before* and *after*; current spatial location, specified spatial locations and spatial relations of *inside* or *outside*; conditionals using *if* (for event evaluation) and *when* (for continuous evaluation) as well as conjunctions and disjunctions. In the next section the lifecycle of the DSA will be considered to identify further requirements on the management of scientific DSAs.

III. THE DSA LIFECYCLE

Management of enterprise policies is a major and important element of governance. Enterprise governance, risk and compliance programs (GRC) usually manage the policy life cycle based on a formal process defined for management of the actual policy document. This policy management lifecycle is a refinement of the more general document or record management life cycle which normally includes stages of create, approve, use, and then archive or disposal. The use of policies by automated enforcement requires these steps, but also automated steps for deployment, and finally enforcement as shown in figure 1.

Policies are drafted in order to achieve the enterprise objectives given the constraints of legislation and regulation, in the light of current best practice among competitors, and the risks as analyzed in the enterprise. Policies are drafted by senior management with advice from their corporate lawyers, so whatever tool support is provided for this stage of the process must provide a user experience appropriate for such users.

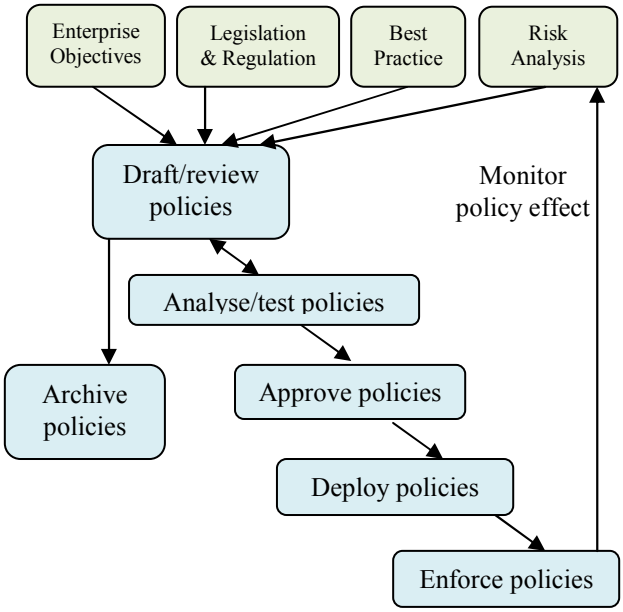

Figure 1. The policy management lifecycle.

Complete natural language (NL) understanding for policies is not realistic today, but a controlled natural language (CNL) syntax can be used for policy rules, to be parsed into an internal logical format. CNLs are a subset of NL that imposes restrictions on the interpretation of NL expressions. As a subset of NL, CNLs are ideally suited to address the usability gap: they capitalize on all the comprehension and productivity benefits of NL, without necessarily undermining the potential for machine-based processing of logical content. A CNL approach should ease the adoption of a policy language expressive enough to meet the requirements developed in the previous section.

Once drafted, policies in a human operated system are often analyzed for their language and consistency with other policies. In an automated system a broader range of formal analyses for different properties [18] can be modeled and those models formally checked.

From the lifecycle, requirements have been derived for a CNL interface for managers to author policies and for a formal analysis of the policies before approval. Since the later stages of the lifecycle are self explanatory, it is now possible to consider the details of the policy languages to be used and then an architecture which can meet the requirements identified.

IV. DSA POLICY LANGUAGES

The requirements show the need for three policy languages: a CNL, an analyzable language and an enforceable language for policies. Each of these is now considered in turn.

A. Controlled Natural Language

In order to meet the requirements including usability and any expectation of adoption by the scientific data management community, the CNL design needs to adopt existing representations and tools to provide a generic language which can be specialized with a vocabulary for each domain. The CNL needs to meet three requirements: to be sufficiently expressive to represent the real DSA policies; the restrictions it imposes on the interpretation of NL expressions cannot be too great to obtain the benefits of transferring the comprehension and productivity benefits of NL; it must be customizable in practice by the user community without demanding specialized skills and tools which are unavailable.

The proposed CNL [20] uses an ontology to specify the semantics of vocabulary terms represented in the W3C standard language OWL. Normally ontologies are not the same as lexicons, so to allow the ontology to serve both roles, one of the core distinctions is between *displayed* and *non-displayed* terms. *Displayed* terms are lexical items in CNL and *non-displayed* terms are only used to define the semantics. For example, the concept of *ExperimentalStation* may be used as a term to define experiments by reference to the facility where they are performed and the facility provider organization which is a party to a DSA, but the term *ExperimentalStation* itself may not be used in the DSA policies since it is defined as a *non-displayed* term which is not in the vocabulary of the CNL. In addition to the vocabulary, the CNL has a syntax which is defined as a simple state transition engine. The syntax of the CNL can be generic across domains, while the vocabulary would be specific to specific domains such as scientific data management.

B. The Analyzable Language

The analyzable language needs to describe models for which model checking software is already available which could analysis the DSA policies for significant properties and provide results which can be understood by scientific managers to guide their policy design. Model checking is a formal technique for the automated analysis of system models against required system properties. Once a suitable model and property have been specified, no further interaction by the analyst is required. However, this does not make the method straightforward since the checker must be provided with appropriate and complex input data. Furthermore, counter-examples generated by the system are often difficult to interpret. Because of these complexities, model checking is not commonly used, and, despite its obvious benefits, exhaustive exploration of system models based on finite state descriptions is not exploited within industrial systems design [19]. To determine the language that would best take advantage of model checking techniques, we represented scientific DSAs in three formal policy languages: a policy language for which a CNL editor is already available to save the effort of implementing our own: SecPAL [5]; the process algebra based policy language POLPA [7], and the state based language event B [3]. All three languages were capable of representing the scientific DSA policies, and of performing model checking .

However, for all three representations it was necessary to define the significant properties to be checked against the model of the DSA policies in a formal notation. Formal methods experts usually consider properties such as mutual exclusion and liveness which had no meaning to the scientific managers who are the planned users. All three formal notations were regarded by users as being difficult to comprehend due to unfamiliar symbols and underlying rules of interpretation that are not apparent. To overcome this usability barrier it was necessary to provide a user interface which hid the formal language, automatically creating it from the CNL and only presenting the results of the analysis to the users. POLPA was chosen because the MAUDE tool used for model checking with it was the easiest to modify with a user friendly interface. The entry of significant properties using the POLPA language was still supported for expert users, but for scientific managers it was decided to perform model checks on two properties by default with understandable output of the results. Firstly, the ontology was used to provide the hierarchy or roles and of data types as input to check which roles could perform which actions named in the policies against which data types with the results presented as a table. Secondly, the set of policies was checked for any conflicts on the actions; that is was any action both authorised and prohibited, or both obliged and prohibited - even by default.

C. The Enforceable Language

For the enforceable policy language standard languages such as XACML are not sufficiently expressive to meet the requirements set. A new policy language EPL was designed and implemented which met the requirements.

Having established the three policy languages required it is necessary how to translate from one to another. To ensure that the semantics of the input language (CNL) are retained, it should be refined into POLPA which in turn should be refined into EPL. However, insufficient effort was available to implement refinement between the languages, and transformations incorporating expansion of terms based on the ontological hierarchies, and the mapping of terms in the CNL to those used in the application were used from CNL to both POLPA and CNL. The use of transformations rather than refinement introduces risks that the analysis does not represent the exact meaning of the CNL or the exact enforced EPL which need to be quantified, and explained to the user in order to understand their impact on the CNL semantics.

V. THE ARCHITECTURE

The architecture will be considered in two stages, firstly for the generation of enforceable policies by a DSA Subsystem and then the enforcement of those policies by the policy enforcement Subsystem.

The generation of enforceable policies follows the policy lifecycle shown in Figure 1 driven by a lifecycle manager to progress the DSA through the lifecycle stages using a workflow engine which presents feedback to the user through a user interface. The implementation used the MS-SharePoint workflow engine for the DSA lifecycle manager. Figure 2 shows the DSA lifecycle manager at the heart of the architecture of the DSA subsystem.

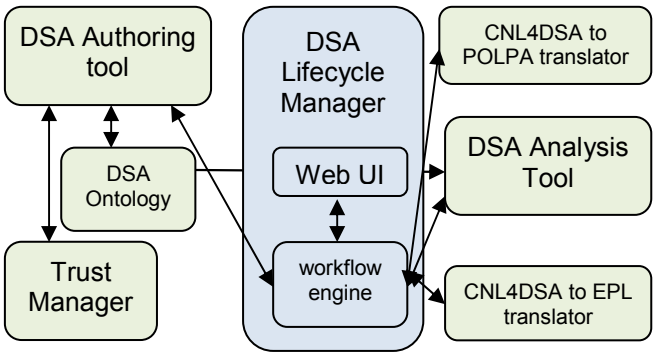

Figure 2. The architecture of the DSA Subsystem.

The process of managers and lawyers drafting and reviewing policies in a DSA needs to be supported by an editor for the CNL for the DSA. The DSA editor will draw upon the controlled vocabulary for a DSA in a particular domain which is represented as on ontology in the W3C standard language OWL, and which can be created using existing supported tools such as Protégé.

In order to provide the enforceable policy language with sufficient detail to operate, the CNL4DSA version of the DSA needs to include details of trusted services, encryption keys, and security domains which the DSA policy authors cannot be expected to know. These must be entered into the Trust Manager through a simple web form interface by a technical expert. The DSA authoring tool accesses the Trust Manager to retrieve these details, but they are also available for tool which will map the CNL4DSA language produced by the DSA authoring tool into the enforceable policy language EPL.

Once the DSA has been drafted in the CNL4DSA language the DSA lifecycle manager will call a translator to produce the POLPA version of the policies which it passes to the DSA Analysis Tool [7].

After the DSA has been analyzed, and the analysis results reviewed by the managers, the DSA can be edited in the authoring tool to correct any errors identified (e.g. conflicting policies). Once the authors have completed the DSA the workflow engine will send it to the named signatories for digital signature. Once returned it has been approved, and the workflow engine will deploy it to the policy server in the DSA enforcement system at the date and time when it comes into effect.

Current policy enforcement architectures contain a PEP (Policy Enforcement Point), PDP (Policy Decision Point), and PIP (Policy Information Point) with other optional components. The PEP intercepts the action requests from users and sends the requests to the PDP. The PDP makes action decisions according to the security policy or policy set and, using attributes of the subjects, the resource, and the environment obtained by querying the PIP. The access decision given by the PDP is sent to the PEP. The PEP fulfills the obligations and either permits or denies the action request according to the decision of PDP. The attribute values provided by the PIP need to be obtained from one or more STS (Secure Token Service) for location, identity etc. For each obligation action which is performed the PEP needs to call an obligation handler to carry out the operation, e.g. send an e-mail to a scientist. Together these conventional components constitute a core enforcement server around which the rest of the policy enforcement subsystem will be built.

In addition to the modules to manage the policy enforcement, modules are also required to manage the encryption and decryption of the data, the key management to support this, and the management of license publishing when data is accessed to support subsequent off-line usage. The server end point details needed to support the key management, and STS will have been entered into the Trust Manager. For the Consequence architecture, the encryption management software and the policy enforcement software are wrapped together behind an interface called the Protected Data Object Application Programmer Interface (DPOAPI).

A. Architecture in the Science Data Scenario

In the STFC science data scenario, the purpose of the architecture [21] is to allow scientists in remote locations to download the data that they want from the File Store catalogued in iCAT and analyze it within the constraints of the agreed data management policies. The key architectural decisions regard the trustworthiness of the respected domains. It has been conceded that scientists cannot be expected to operate trusted computing platforms for the client applications, which has already opened a significant security risk. The secure solution to the architecture would be to operate a single policy server in the domain of the application server (i.e. the data provider) so both the client and server application programs always use that to test actions against the policies. However, this would not permit off-line access when the policy server is not available, which is a requirement on the system. In order to support this requirement it is necessary to operate a policy server on the application client which can acquire the current values of attributes through the PIP so that the PDP can evaluate policies off-line. Both these policy servers are hidden from the application programmer behind the DPOAPI, as is the testing of off-line status and the management of which policy server to use. However, this solution is only sensible if the application client is operating in a trustworthy domain. That is to say, the solution is only secure when scientists who use the system can be trusted not to modify the policies held in the client PDP.

The solution proposed involves a combination of the technical enforcement of policies with the supporting legal redress for breach of the agreement between the data provider and the data user. The solution is not a purely technical one. This combined solution to data management security would not be acceptable in a military or even some commercial situations where the risks are unacceptable. The question is open to debate as to whether the risks are acceptable in the scientific domain. As shown in Figure 1, the policy management life cycle includes a risk analysis component where these risks need to be addressed. However, since the implementation of secure remote data management solutions is novel in the scientific community, we need more debate, and data to know how to assess these risks.

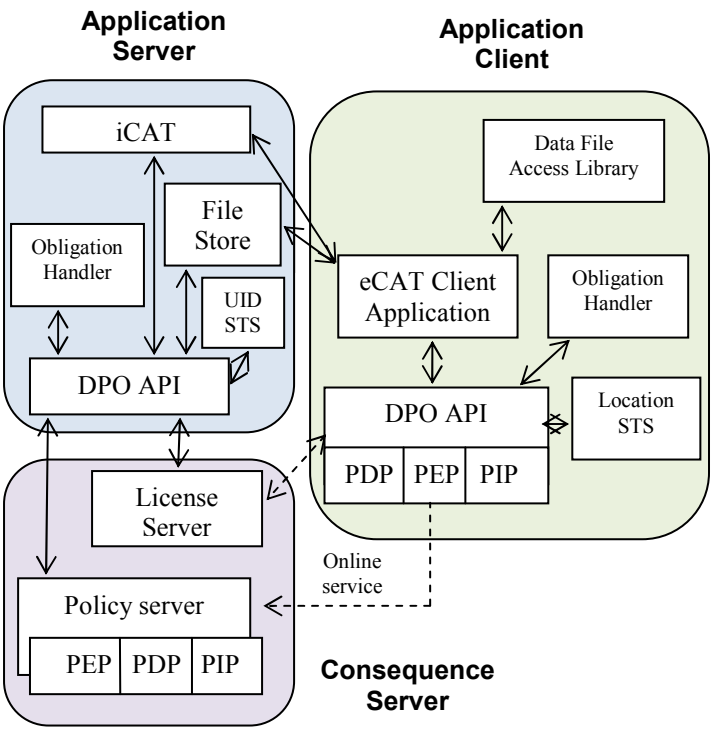

Figure 3. A simplified Consequence enforcement architecture.

Figure 3 shows a simplified version of the Consequence enforcement architecture for the iCAT scientific data cataloging system. When deployed it requires three virtual machines: the user's client machine to which they will download data discovered in the iCAT catalogue and downloaded from the appropriate File Store; the application server which runs the iCAT catalogue, with the File Store containing the data files themselves, the obligation handler required to perform any obligation actions from the application server and an STS to authenticate user identity; and the Consequence server which runs the policy server and the publication license server. In practice more than three physical machines would be used to deploy this architecture efficiently.

At runtime, the client application must ask the DPOAPI if a requested action is permitted before that action is performed. Actions include access such as downloads, local read and write operations (to any device including file stores, screen, printer etc…), as well as transformation operations on data. Since the data is held in encrypted form, access requests which do not obtain authorization from the policy service will not be able to decrypt the data. This is a secure solution and one which is not onerous to the application programmer.

On the server side, the application must encrypt data in accordance with the agreed data policies so that they will only be accessible through the DPOAPI. In practice the application server programmer should make no assumptions about the policies and ensure that all data are encrypted so that when the policies change, the application does not require any changes but simply leverages the flexibility built-in to the policy based approach. For example, if the policies refer to data, and the ontology defines *metadata* and *experimental data* as subtypes of data then the policy applies to both the metadata held in the catalogue and to the data held on the File Store. However, if the initial policies are incomplete, and only refer to *experimental data*, then the application should still ensure that *metadata* is encrypted under the guiding principle that the scope of policies could be expanded. The application of this principle is not always self evident, for example, it was not immediately obvious to the application programmer in this study that the log data on usage of iCAT and the File Store should be encrypted since it was not addressed in any data policies. Once a secure system such as that described here is adopted it not only requires the technical and legal solutions, but also requires changes to the assumptions of application developers.

VI. An example of DSA enforcement

A test bed of the Consequence architecture was implemented to demonstrate a scientific data management scenario. This involves a principle investigator (PI), with two co-investigators from two different universities, visiting the ISIS neutron source to use an instrument on one of the beam lines, to obtain experimental data on the structure of a crystal. They would then access the data from their university locations, in order to analyze it using a web or dedicated client..

When using the instrument on the beam line they worked with a local STFC beam line scientist who also wanted access to some of the data to calibrate the instrument. The experimental data consists of a combination of images and numerical data from the instrument which are stored in a file in the NeXus data format. The PI later attended a conference in country X which is not considered as trusted where she considered presenting the data in contravention of the DSA. After a three year embargo period the PI published a paper on her findings derived from the data, with the result that other scientists wanted access to the data, to validate the analysis.

The scenario was governed by a subset of the ISIS data management policy included in DSA between STFC and the universities. The seven DSA policies used in the test scenario to assess the coverage of the DSA requirements within the scenario were:

Authorisation Policies

1. Before the end of the three year embargo period, access to the experimental data is restricted to the principal investigator and co-investigators.
2. After the embargo period, the experimental data may be accessed by all users.
3. Beam line scientists can access image data produced on their experimental station.

Prohibition Policy - access

4. Access to numerical data should be denied to users in either country X or Y.

Prohibition Policy - usage

5. If a user does_not_use a STFC Client then that user cannot visualize experimental data.

Obligation Policies

6. After a public user downloads data then the system must notify the principal investigator.
7. After a public user reads any data then the system must log the event.

The policies were entered into the DSA Authoring Tool by several managerial scientists. Anecdotal support was provided for the result of [15], in that the user group talked positively about the comprehensibility of the CNL interface in preference to alternative policy language expressions of the same policies, but there was only minor improvement in their performance to construct CNL policies over alternative policy languages. For example, the CNL version of the NL obligation policy 7 above is shown below:

```
IF a User hasRole a PublicUser
AND a NexusFile has as data category
experimental data
THEN AFTER that User Read that
NexusFile
        THEN a System MUST Log a Event
```

In this example, words in all capitals are CNL keywords, while capitalised words are taken from the domain ontology for ISIS data. The policy is reasonably understandable without specialist knowledge, but even an editor which only offered the next possible terms to enter was not enough to enable manager's to write in this CNL.

One way in which this CNL was evaluated was by issuing a questionnaire to ten senior research managers and ten system administrators from European scientific data centres, presenting the policies in NL, CNL and EPL, and asking if the CNL was more like the NL than the EPL. As expected, the judgement was that CNL was significantly more like NL than EPL (z = 3.75, confidence > 95%). More interestingly, there was also a highly significant difference (t =-9.74, confidence > 99.9%) between the answers given by the two groups of subjects. The managers found the CNL a greater improvement in usability over the enforceable policy language than did the system administrators, who were more familiar with using computer languages. The limitations of CNL in this situation are well documented [16], and while both the examples used here, and in other recent studies [14] show advantages over enforceable query languages, there is no clear path to improve the usability of CNL.

Following the editing of the policies the scientific managers analysed them, and used the feedback from the model checker to guide their editing. Users made no attempt to define properties in the specification language since, as expected [17] it was considered opaque and its processing pre-conceptions were unapproachable. However, the two tests undertaken on the DSA with improved user interfaces (conflicting policies and the table of access rights) were appreciated in the reports of users. As planned, this output was used to edit DSA policies by managers.

Once the DSA lifecycle shown in Figure 1 had been pursued this far, the next steps required little user interaction to approve and deploy policies. The DPOAPI was implemented in .Net using C#, a Java interface was provided to it using the JNBridge interoperability tools which were used in the iCAT test bed; the C# interface was used in another test bed for Crisis Management in this study. The iCAT test bed also used both the MS-ADFS STS and the Java Metro STS to demonstrate generality of STS interfaces.

The location service feeding the Location STS for the iCAT test bed was implemented using a GPS tracker local to the client machine, while in the Crisis Management test bed, the AeroScout WiFi RFID location system was used to give 30cm location accuracy. The GPS tracker's unique identifier was checked to ensure that the location of the correct signal was being used, although since the GPS was not an integral part of the client machine there was a risk that the location of the tracker was being checked and not the location of the client machine.

The enforcement of the scenario policies was tested by running the system using user identities for each role, and testing the temporal, spatial and data type variations in the policies expressed, and distinctions in the underlying ontology not expressed in policies. In all cases the policy enforcement server operated in accordance with the policies.

VII. CONCLUSION

The approach to enforcing scientific data sharing agreements presented here is based upon a combination of trust and risk management, leading-edge technology, licensing and legal frameworks. Scientific communities need to decide what balance between these components they want to adopt for their data management.

Policies need to be approved by senior managers, but should they delegate the authority to draft then to junior technical experts whom they trust and approve policies they cannot understand, or should technology provide policy editing tools which will allow them to draft and review policies themselves? The current study confirms the usability problems previously identified in both policy authoring and formal analysis through model checking, while also demonstrating the potential benefits of these techniques. To move from the position where the policy management risks are mitigated through trust relationships to one where they are mitigated through direct understanding will require considerable further research to improve the usability of CNL for policies and of formal model checking.

The requirements collected for scientific data sharing agreements for policy content were beyond the expressive power of standard policy languages such as XACML, so the project had to develop its own language EPL. As policy languages are more widely adopted, there will be growing pressure to increase the expressive power of such standardised languages. The scientific data management community must ensure that its voice is well represented so that the needs of the scientific community are supported by this evolving technology.

The scientific community cannot afford to maintain unique ICT infrastructures. Although e-science operates at the bleeding edge of ICT innovation, it relies upon technologies it

has adopted being assimilated into mainstream product lines within a manageable timescale in order to sustain their use over the long term. To this end, the Consequence framework presented here for a scientific data test bed has also been demonstrated for test beds for the emergency services and for Microsoft Office with the expectation of parts of it being adopted into mainstream commercial products.

Finally, the scenario test policies used in the example above, and the vast majority of the policies encountered in the study of scientific data policies refer to data derived from publically funded research. In practice, large facilities, as well as many universities rely upon a balance of publically and commercially funded research. The enforcement infrastructure presented here may appear overly secure for the requirements of publically funded research, but is it able to meet the requirements of commercially funded research as well? No university or facility will want to run parallel data management infrastructures for these two funding streams so the need for a single flexible infrastructure that can meet the needs of both communities would clearly be preferable, but does the approach proposed here meet that demand?

ACKNOWLEDGMENT

The authors acknowledge the considerable contribution to the Consequence project by the consortium partners: David Golby of BAE Systems; Ilaria Matteucci, Marinella Petrocchi, and Fabio Martinelli of CNR IIT; Giovanni Russello of Create-Net; Marco Luca Sbodio of Hewlett-Packard Innovation Centre, and Enrico Scalavino, Vaibhav Gowadia, and Emil Lupu of Imperial College London.

REFERENCES

[1] J. Kaye, C. Heeney, N. Hawkins, J. de Vries, and P. Boddington, “Data sharing in genomics - re-shaping scientific practice”, Nature Reviews Genetics, 10, 331-335, 2009. doi:10.1038/nrg2573

[2] D. Flannery, B. Matthews, T. Griffin, J. Bicarregui, M. Gleaves, et al, "ICAT: integrating data infrastructure for facilities based science", in Proc of the 5th IEEE Int. Conf. on e-Science, pp.201-207, 2009. doi:10.1109/e-Science.2009.36

[3] A. Arenas, B. Aziz, J. Bicarregui and M. Wilson, “An event-B approach to data sharing agreements”, in Proc. of the 8th international conference on integrated formal methods (IFM'10), Dominique Maury and Stephan Merz (Eds.). Berlin, Heidelberg: Springer-Verlag, pp 28-42, 2010.

[4] A. Arenas and M. Wilson “Contracts as trust substitutes in collaborative business”, IEEE Computer, 41(7) pp 80-83, 2008.

[5] B. Aziz, A. Arenas, and M. Wilson “SecPAL4DSA : a policy language for specifying data sharing agreements”, in Secure and trust computing, data management, and applications, Proc. 8th FTRA int. conf. (STA 2011), J.J. Parc et al. Eds., CCIS 186, Berlin, Heidelberg: Springer-Verlag, 2011.

[6] J. Lane, and S. Shipp, “Using a remote access data enclave for data dissemination”, Int. Journal of Digital Curation, 2(1), pp 128-134, 2007.

[7] I. Matteucci, M. Petrocchi, and M. Sbodio “CNL4DSA: a controlled natural language for data sharing agreements”, in Proc. of the 2010 ACM Symposium on Applied Computing (SAC '10). ACM, New York, 616-620, 2010. doi:10.1145/1774088.1774218

[8] E. Scalavino, V. Gowadia, and E. Lupu. “A labelling system for derived data control”, in Proc. 24th annual IFIP WG 11.3 working conf. on data and applications security and privacy (DBSec'10), Sara Foresti and Sushil Jajodia (Eds.). Berlin, Heidelberg: Springer-Verlag, pp 65-80, 2010.

[9] V. Swarup, L. Seligman, and A. Rosenthal, “A Data Sharing Agreement Framework”, in Information Systems Security, Bagchi, A. and Atluri, V. (Eds.), LNCS 4332, Berlin, Heidelberg: Springer-Verlag, pp 22-36, 2006. doi: 10.1007/11961635_2

[10] E. Yang, B. Matthews, and M. Wilson, "Enhancing the Core Scientific Metadata Model to Incorporate Derived Data", in Proc. IEEE 6th Sixth Int. Conf. on e-Science, pp.145-152, 2010.

[11] D. Golby, M. Wilson, L. Schubert, and C. Geuer-Pollmann, “An assured environment for collaborative engineering using web services”, in Proc. of the conf. on Leading the Web in Concurrent Engineering: Next Generation Concurrent Engineering, Parisa Ghodous, Rose Dieng-Kuntz, and Geilson Loureiro (Eds.). Amsterdam: IOS Press, 111-119, 2006.

[12] L. Krautsevich, A. Lazouski, F. Martinelli, P. Mori, and A. Yautsiukhin, “Usage control, risk and trust”, in Proc. 7th int. conf. on Trust, privacy and security in digital business (TrustBus'10), Sokratis Katsikas, Javier Lopez, and Miguel Soriano (Eds.). Berlin, Heidelberg: Springer-Verlag, 1-12, 2010.

[13] Q. Liu, R. Safavi-Naini, and N. Sheppard, “Digital rights management for content distribution”, in Proc. of the Australasian information security workshop conference on ACSW frontiers (ACSW Frontiers '03), Chris Johnson, Paul Montague, and Chris Steketee (Eds.), Vol. 21. Australian Computer Society, Inc., Darlinghurst, Australia, 49-58, 2003.

[14] L. Shi and D. Chadwick, “A controlled natural language interface for authoring access control policies”, in Proc. of the 2011 ACM Symposium on Applied Computing (SAC '11). New York: ACM, 1524-1530, 2011. doi 10.1145/1982185.1982510

[15] K. Vaniea, C-M. Karat, J. B. Gross, J. Karat, and C. Brodie, “Evaluating assistance of natural language policy authoring”, in Proc. of the 4th symposium on Usable privacy and security (SOUPS '08). New York: ACM, 65-73, 2008.doi 10.1145/1408664.1408674

[16] P. Inglesant, M. A. Sasse, D. Chadwick, and L. Lei Shi, “Expressions of expertness: the virtuous circle of natural language for access control policy specification”, in Proc. of the 4th symposium on usable privacy and security (SOUPS '08). New York: ACM, 77-88, 2005. doi:10.1145/1408664.1408675

[17] D. Carew, C. Exton, and J. Buckley, "An empirical investigation of the comprehensibility of requirements specifications", in Proc. Int. Symposium on Empirical Software Engineering, 17-18, 2005. doi: 10.1109/ISESE.2005.1541834

[18] A.K. Singh, and A.K. Bandyopadhyay, "Verifying mutual exclusion and liveness properties with split preconditions", Journal of Computer Science and Technology, Vol.19, Iss.6, pp.795, 2004,

[19] K. Loer and M. Harrison, “Towards Usable and Relevant Model Checking Techniques for the Analysis of Dependable Interactive Systems”, in Proc. of the 17th IEEE int. conf. on Automated software engineering (ASE '02). Washington, DC, USA: IEEE Computer Society, 223, 2002.

[20] M. Sbodio and M. Casassa Mont. “A System Enabling the Lifecycle Management of Electronic Data Sharing Agreements”, in Proc. of the Hewlett Packard Technical Conference, Orlando, Florida, USA, April 3 - 6, 2011.

[21] V. Gowadia, E. Scalavino, E. C. Lupu, D. Starostin, and A. Orlov. “Secure Cross-domain Data Sharing Architecture for Crisis Management”, in Proc. ACM CCS DRM workshop, Chicago, 2010.