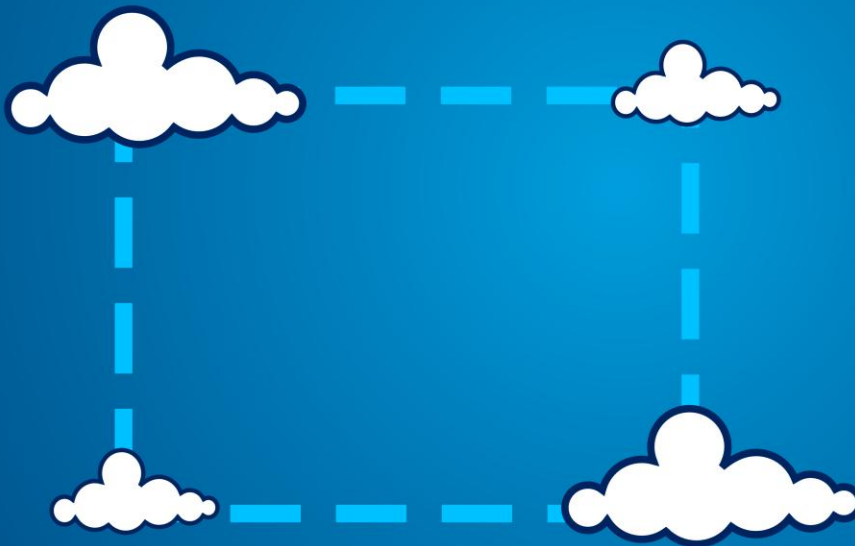




Agility™: Service Agreements for the Cloud



Agility: Service Agreements for the Cloud

Introduction

At Arjuna we believe that IT services will increasingly be delivered by an interconnected network of federated service providers. Providers will consume services from each other, organisations will cooperate together to share services to mutual advantage, service brokerage and trading will be commonplace.

While a lot of thought has been given to enabling the technical integration that is required in order to build such networks, we believe that insufficient effort has been directed towards issues of organisational integration. In particular we believe that much of the complexity in federated networks will come from the need to manage service agreements between organisations. The benefits of Cloud Computing can only be fully realized when the relationships between organisations, in the form of service agreements, can be “on-demand” in the same manner as the resources which make up the cloud.

Support for capturing service agreements and the means to modify service agreements are (almost) completely lacking in current cloud offerings. Without this support costly manual intervention is required when service requirements change. This represents a significant barrier to the uptake of the cloud by organisations who understand that change is an inevitable aspect of any business.

Arjuna Technologies have focused their efforts on this particular problem for a number of years and have developed ‘Agility’, a framework for the management of dynamic service agreements and of policy that responds to the creation, modification, maintenance and deletion of those agreements. Agility enables organisations to connect together (internally and/or externally) into a Federated Cloud - delivering [cooperation with accountability and control](#).

Agility is the glue for the Federated Cloud.

The Federated Cloud

Federation

Federation: an organisational structure where the parties concerned are autonomous but cooperate through agreement.

The Cloud Computing paradigm is, at its core, about improved sharing, where the means of sharing is through the consumption of services. With constantly improving network capacity IT users are now able, in many cases, to consume services sourced from third parties rather than managing their own applications. As a consequence services will increasingly be delivered by providers who can reduce service costs by taking advantage of economies of scale. This trend puts economic pressure on IT consumers to utilise shared services, and by implication the IT resources (hardware, software and staff) used to deliver those services. As prices fall, and shared services proliferate and mature, it is inevitable that we will see a gradual but consistent move from services delivered by dedicated IT to services delivered by shared IT.

However, for at least three reasons we are unlikely to move to a situation where all IT resources are in the hands of a few providers, irrespective of the economic benefits. Firstly, there will be many cases where sharing is unfeasible or undesirable – bespoke services may not deliver value to anyone but their creators, or may be of such competitive value that the creators will keep them to themselves; other services might have restrictions e.g. security or latency considerations, that prevent sharing from occurring. Secondly, the transition to using shared services will be very gradual (due to inherent conservatism and to sunk costs in existing dedicated IT) and therefore it will be many years before even all of those services which could be shared, will be. Thirdly, governments will not allow such strong economic controls to be concentrated in the hands of a few.

As a consequence, at Arjuna we believe that IT services will increasingly be delivered by an interconnected network of federated service providers. Providers will consume services from each other, organisations will cooperate together to share services to mutual advantage, service brokerage and trading will be commonplace. Given the variety and multitude of possible IT services we believe that the resultant network will be orders of magnitude more complex than other service delivery networks such as the electricity grid or telephone network. While a lot of effort has been aimed at enabling the technical integration that is required in order to build such networks, we believe that insufficient effort has been directed towards issues of organisational integration. In particular we believe that much of the complexity in federated networks will come from the need to manage service agreements between organisations.

Arjuna's Agility product delivers the service agreements that are the glue for the Federated Cloud.

Service agreements

Service agreement: an expression of functional and/or non-functional aspects of a service delivered from one party to another, along with any obligations upon either party.

When a service is both commissioned and delivered within a single organisation, the appropriate IT resources are designated and trialled until an effective service is deemed to have been provided. In this case both the consumers of the service and the suppliers belong to the same organisation, are ultimately

responsible to a single authority, and have common interest. Common interest results in a level of trust where formal contractual and detailed expressions of precisely what is required of the service can frequently be avoided. Knowledge of the specific dedicated resources being used and/or faith in the ability of the IT team within the organisation can reassure the IT consumer that the requisite quality of service will continue to be delivered.

However, today if you're commissioning an application delivering some service you'd ideally like to push it into the cloud and forget about it. You want the cloud to be opaque. You don't want to have to make resourcing decisions (and you particularly don't want to incur any expense associated with over-capacity). But if the cloud is opaque then you don't know what resources will be utilised, and as there isn't the same level of trust, how can you, or the consumers of your service, be confident that a satisfactory quality of service will be maintained?

Well, if the risks associated with the failure of the service to behave as expected are low enough so as to be ignored or effectively mitigated by the consumer then a general purpose, low-cost, cloud which offers simple 'best-effort' quality of service may well be 'good enough'. Many of the applications hosted on the cloud today do indeed fall into this category.

However, if the risks are perceived to be higher then a service may only be useable if backed by a service agreement which provides some specific contractual guarantees. Those guarantees might constrain the provider in some way e.g. by mandating the use of some specific form of replicated storage, the availability of skilled support staff, the legal jurisdiction within which data is to be contained; or might promise specific quality of service guarantees for reliability, availability, etc. By this means the consumer can be persuaded that the risk is lower than it might otherwise be. Additionally the provider might agree to compensate the consumer if quality of service is not maintained. This compensation could be through the provision of an effective alternative service or may even be financial. Service providers supporting this form of service agreement are effectively reducing the overall cost of using the service for individual consumers by mitigating their collective risk.

A service agreement could specify not only how the service can be operated and what it is expected to do, but also what might go wrong and what then happens, the legal, financial and support obligations, what reporting is to be delivered etc. In the real world service agreements (just like other legal contracts) can rarely be absolutely complete i.e. aspects remain open to interpretation, but in the cloud they certainly need to be much more explicit than they need be when services are utilised within a single organisation. The trust relationship which exists within an organisation needs to be replaced by a more formal relationship i.e. a contractual service agreement which clearly defines all relevant aspects of the service to be delivered.

Variable service agreements

Traditionally, a formal service agreement involves the creation of a legal contract with the associated costs. Those costs can be amortised across a service's consumers so long as the agreement is sufficiently generic to suit all. However, requiring consumers to sign up to (or tick box) a legal document has two distinct disadvantages. Firstly the terms of the contract are difficult to interpret at run-time and therefore it may be impossible for either consumer or provider to effectively monitor the service to ensure it is behaving as the agreement promised. Secondly, legal documents are time-consuming and expensive to vary. However, variation of service agreements is vital not only in order to provide consumers with choice but will also be required in order to allow for change over time.

As an example of the later requirement, let's imagine a provider offering a service agreement which is initially suitable for some consumer. However, given that change is inevitable, what happens when the required quality of service changes, perhaps because the importance of the service to the consumer increases (or declines), or the data security needs change, etc. The change may even concern some aspect of the service not previously considered in the original agreement. Does the consumer have to move to a different service provider (one able to satisfy the new required quality of service) or do they have to manually renegotiate a new quality of service from the provider? Doesn't this sound somewhat inflexible, undynamic and uncloud-like?

What is actually required is the ability to capture quality of service requirements and to modify those dynamically. The cloud can then respond to any changes e.g. by reassigning resources, as appropriate. Let's take a simple example. Imagine we have a new application that's providing a service to end-users and which has the potential to be extremely popular. The application is free when first launched and the application developer is cash-strapped. So, an initial quality of service requirement from the developer upon their cloud provider might be that the application can only consume \$100 worth of resources per day and additional load which would cause that to be exceeded should be shed in some controlled manner. This prevents an initial viral explosion in the service's popularity from bankrupting the developer. Later, once the success has been proven, the developer raises some cash and now wants the application to be capable of scaling to consume a maximum of \$1,000 per day.

Changing the quality of service requirements (and indeed setting the original quality of service) requires an agreement between the cloud user and the cloud provider. But, in this example and many others, the means of reaching that agreement could be fully automated. For example the cloud provider's response to an on-line request to increase the maximum cost might require a on-line cash deposit intended to underwrite the potential costs.

Note that our choice of 'cost' as a quality of service requirement might appear unusual. However, probably nothing is of greater importance to the cloud user. If the cloud user can't define cost limits who will? Service agreements need to capture the real business requirements of users not be couched in the language of the IT professional.

In our view, the inability to capture and then dynamically vary service agreements will significantly constrain enterprises' consumption of third party services in the cloud. Until this problem is resolved enterprises (and individuals) will be restricted to the consumption of flexible services with poorly defined quality of service, or of inflexible services backed by formal contracts.

Agility

Cooperation with Federation

Support for capturing service agreements and the means to modify service agreements are (almost) completely lacking in all current cloud offerings. Without this support costly manual intervention is required when service requirements change. This represents a significant barrier to the uptake of the cloud by organisations who understand that change is an inevitable aspect of any business.

Arjuna Technologies have focused their efforts on this particular problem for a number of years and have developed 'Agility', a framework for the management of dynamic service agreements and of policy that responds to the creation, modification, maintenance and deletion of those agreements. The framework does not attempt to define some standardised ontology but is instead open and extensible, leaving the semantics of a service agreement entirely in the hands of the parties concerned. *Agility* enables service agreements to be changed, as they inevitably will need to be, over the lifetime of a service and opens up opportunities for sophisticated brokering between clouds offering differing quality of service and of cloud federation which enables partners to share services in a controlled manner.

Agility is an overlay that is deployed as a stand-alone server run on each organisation which is to interact and provides a means to capture existing and future relationships with other organisations. (Note that an organisation might be an independent enterprise, a department or even an individual). In Figure 1 below, the organisations concerned have created agreements between one another thereby enabling them to share services to the benefit of all concerned. *Agility* enables organisations to be connected together into a 'Federation' in which cooperation occurs through mutual agreement but within which each party retains independence.

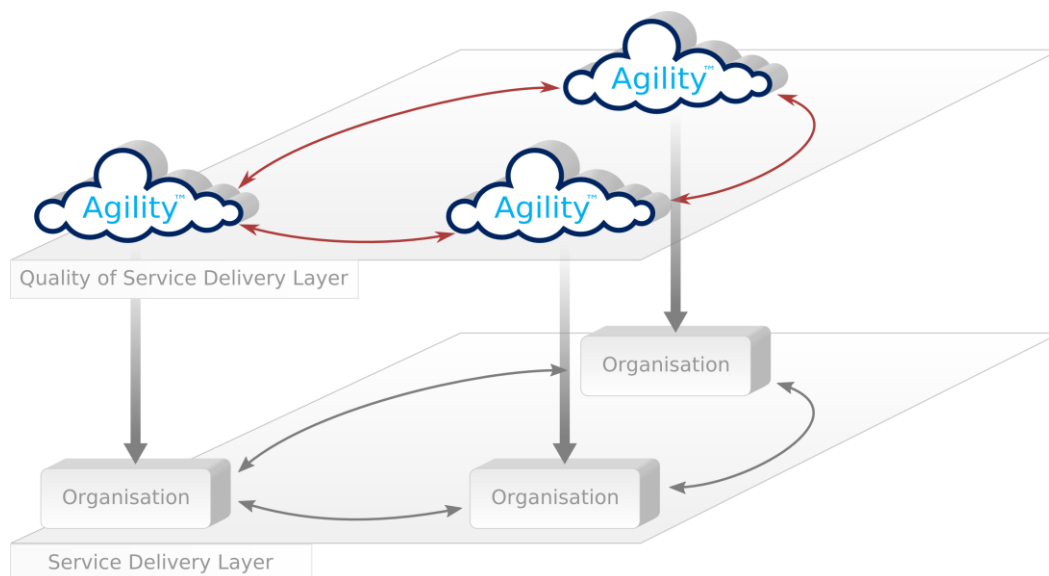


Figure 1: Cooperation with Federation

In addition to managing service agreements, each *Agility* server may interact with its organisation's underlying infrastructure. *Agility* might not interact at all, simply recording the relationships between the

organisations. However, the organisation can use *Agility* to monitor the infrastructure, in order to report on conformance, or to actively ensure conformance e.g. by driving resources.

Agility may enable cooperation between an organisation and its:

- Consumers i.e. end-users, or other organisations, consuming services delivered by the organisation.
- Peers e.g. a set of departments within an organisation.
- Suppliers i.e. organisations offering service to the organisation e.g. a public cloud or a supply chain partner.

Agility enables organisations to connect together into a Federated Cloud - delivering [cooperation with accountability and control](#).

Accountability with Service Agreements

Agility delivers accountability through its use of service agreements which record the responsibilities of the parties engaged in a relationship. With service agreements in place the organisation can monitor performance and can identify when quality of service is threatened in order to take remedial action. Any action taken may be independent of *Agility* or may be driven by automated Policy which has been registered with *Agility*, as we shall see in the next section.

A service agreement is an agreement between two parties and is used to define the obligations of each party typically, though not exclusively, with regard to service offered by one party to another.

Agility represents a Service Agreement as a set of Features, each consisting of a name/value pair. The semantics for the name and value of a Feature are entirely the concern of the two parties involved in the Service Agreement, and are transparent to *Agility*. Agreement is reached between two parties when both believe they understand the semantics of the Features contained within a Service Agreement, and agree that the agreement being proposed is both achievable and acceptable.

For example, a Service Agreement might contain the following Feature:

```
Feature: { "name": "Availability", "value": "Medium" }
```

Perhaps this Service Agreement defines the quality of service requirements for a particular set of services offered by one organisation to another, and this Feature defines the required availability of those services. The level required, 'Medium', is pre-defined and known to both parties and might mean, in this case, 99.99% availability during office hours, and 97.5% availability otherwise. Capturing this information as a Service Agreement delivers a clear audit trail and also allows the opportunity for later modification. For example when about to begin a business critical activity the consumer might temporarily propose that 'Availability' be set to 'High'.

The provider organisation will consider this proposal and, if it is achievable and acceptable - presumably involving a higher pre-agreed level of charging - will approve the modification. Once the business critical activity is complete the consumer can reset the level to 'Medium'.

Service Agreements may be created within the context of another Service Agreement. Such an agreement is termed a 'child' and the agreement providing the context is the 'parent'. Child agreements are ones for which the parent agreement applies, but which may impose further obligations upon the two parties. For example, a child Service Agreement pertaining to 'ServiceA' (one of the set of services supported),

created within the context of the Service Agreement described above, might contain the following Feature:

```
Feature: { "name": "DataConsistency", "value": "CorrectToToday" }
```

The meaning of this Service Agreement is that, service A must utilise information that is correct as of today. Perhaps this requires the provider to take some action to actively update the relevant data in order to ensure it is completely up to date. Once again the provider has to ascertain whether the request is achievable and acceptable before agreeing. In figure 2, the parent agreement is the original Service Agreement and the two children agreements shown might concern the quality of service requirements for two specific services.

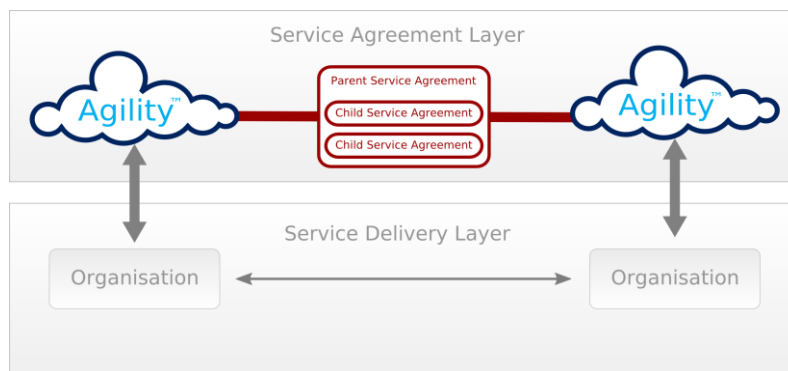
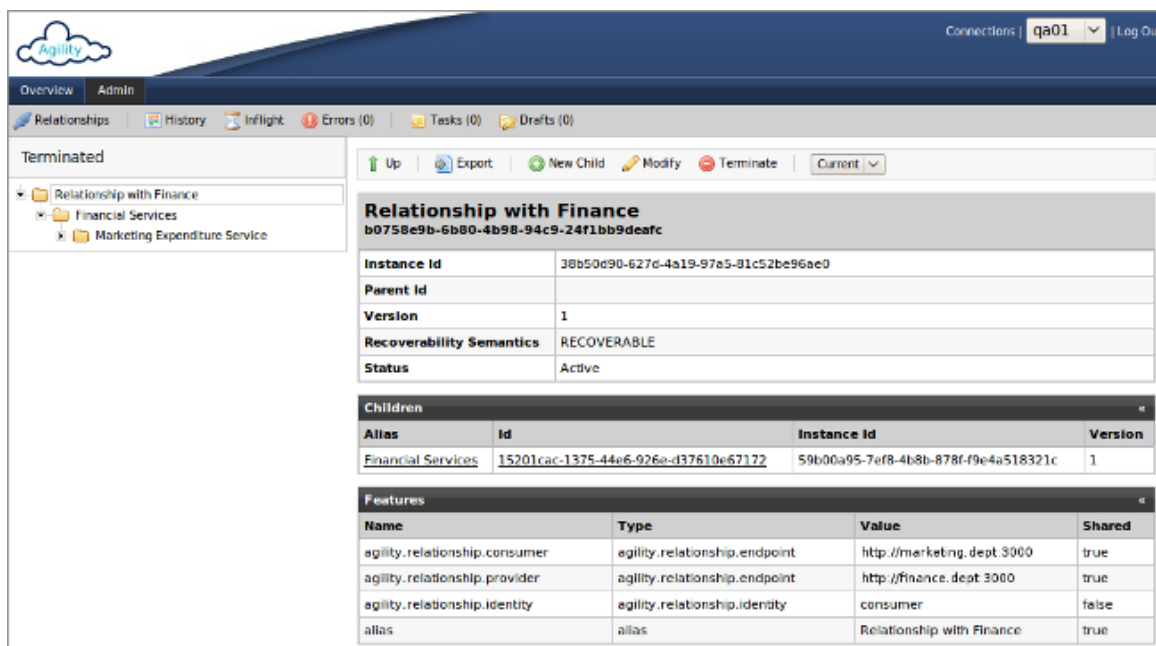


Figure 2: Accountability with Service Agreements

Agility comes with a portal that lets the administrator view all details of Service Agreements, as in Figure 3. As well as this real-time information Agility delivers a full audit trail which captures all changes to Service Agreements over time.



Children			
Alias	Id	Instance Id	Version
Financial Services	15201cac-1375-44e6-926e-d37610e62172	59b00a95-7ef8-4b8b-878f-f9e4a518321c	1

Features			
Name	Type	Value	Shared
agility.relationship.consumer	agility.relationship.endpoint	http://marketing.dept.3000	true
agility.relationship.provider	agility.relationship.endpoint	http://finance.dept.3000	true
agility.relationship.identity	agility.relationship.identity	consumer	false
alias	alias	Relationship with Finance	true

Figure 3: Agility Portal

Control with Policy

Policy: a set of rules or guidelines that direct an organisation's behaviour.

Agility puts control into the hands of an organisation's administrators through its support for pluggable policy. Within an *Agility* Server, Policy is structured as one or more Policy Modules. Multiple Modules can represent different aspects of an organisation's policy, or represent the interests of different actors within the organisation, or be concerned with particular events. For example one Policy Module might be concerned with legal aspects, another with financial auditing, and yet another with green issues.

Policy in *Agility* has two functions, the first of which is the management of Service Agreements.

Policy interacts with the *Agility* Server through a Policy API. By this means Policy Modules can control which Service Agreements are to be entered into, how they may be modified over time and under which conditions they may be terminated.

A change to a Service Agreement proceeds as follows:

1. A Policy module requests (or an individual manually requests), via the Policy API, the modification of a Service Agreement.
2. The *Agility* Server for the consumer organisation signals 'change proposed' to all Policy Modules within the consumer organisation and awaits permission from them to proceed, or a rejection of the proposed change.
3. If agreement to proceed is obtained a 'change proposed' is signalled to the *Agility* Server responsible for the provider organisation.
4. The *Agility* Server for the provider organisation signals 'change proposed' to all Policy Modules within the provider organisation and awaits permission from them to proceed, or a rejection of the proposed change.
5. If agreement to proceed is obtained it signals 'changed' to both the consumer and provider organisations and the modified Service Agreement is then in effect.

The Policy API requires that each Policy Module implements a listener which is capable of accepting the 'Proposed change' and 'Finalised change' events. All Policy Modules registered with an *Agility* Server receive those events and all may vote in the outcome of any 'Proposed change'. Modules must vote 'accept', 'reject' or 'ignore'. The proposed change will be accepted if at least one Module votes 'accept' and no modules vote 'reject'. By this means all Policy Modules within an organisation get the opportunity to have their say in any proposed change. That said, particular events will be of concern to particular Policy modules only and it is the default behaviour for a Policy Module to 'ignore' changes which are not of direct interest.

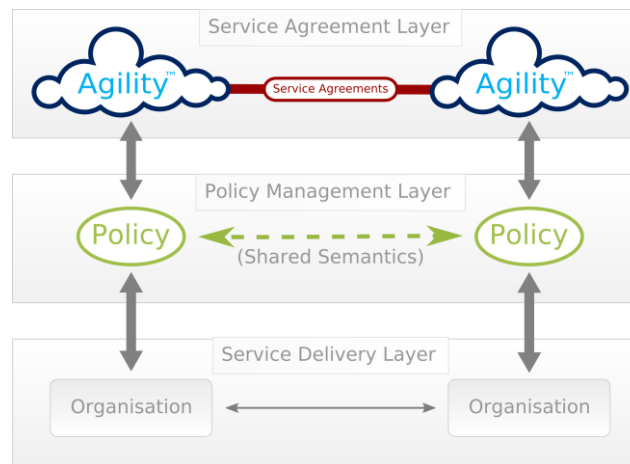


Figure 4: Control with Policy

Policy's second function is the management of the organisation with respect to Service Agreements.

Policy can interact with the organisation in order to report on the progress of the organisation with respect to its Service Agreements and/or to modify the behaviour of the organisation in order to ensure conformance with its Service Agreements. The implementation and behaviour of Policy Modules are not restricted by *Agility*, beyond the requirement that the Policy API be supported. Policy Modules can interact with other Policy Modules, processes, human users and data in whatever way they see fit. The overall behaviour of the organisation will however need to be directed by the organisation's requirements to meet their Service Agreement obligations and the understanding of the business, financial and legal implications of failing to do so. *Agility* does not in any sense 'enforce' Service Agreements or define how the organisation should behave. *Agility* simply records and reports on changes to Service Agreements as they occur.

Policy Modules may be added, replaced and removed at run-time. The addition or replacement of Policy Modules within two organisations can enable them to communicate with forms of agreement not envisaged when the system was first deployed. For example: new Policy Modules which are able to consider energy consumption could be added dynamically to reflect the organisation's responsibilities with regard to new carbon-reduction legislation. A new Feature 'Max Daily Energy' could then be added to the Service Agreement between two organisations and used to agree the maximum daily consumption of energy consumed by the provider with regard to that Service Agreement. This Feature would be ignored by previously registered Policy Modules but would only be considered by the new energy consumption aware Modules. The new modules would enable agreement to be reached on energy consumption and could implement monitoring capabilities to enable reporting of usage, and ultimately could even be used to influence the choice of resources utilised in order to reduce energy consumption e.g. by choosing modern, low-power servers in preference over older ones.

This ability to extend and modify the semantics of agreement is an essential feature of *Agility*, enabling the organisation to cope with inevitable change. For *Agility* it has been a deliberate design decision not to attempt to create a universal ontology for Service Agreements. Systems which attempt this approach tend to be hard to adapt and maintain, because there are an impractically large number of things on which agreement might need to be reached and even when a small set exists between a set of parties it is (almost) inevitable that changes to that set will be required over time. That said *Agility* does not prevent a group of organisations from utilising, or even enforcing, the use of their own specialised ontology.

Summary

Agility is the glue for the Federated Cloud.

Agility is an overlay that is deployed as a stand-alone server run on each organisation which is to interact and provides a means to capture existing and future relationships with other organisations. *Agility* enables organisations to be connected together (internally and/or externally) into a 'Federation' in which cooperation occurs through mutual agreement but within which each party retains independence - delivering **cooperation with accountability and control**.

Agility delivers **accountability** through its use of service agreements which record the responsibilities of the parties engaged in a relationship. With service agreements in place the organisation can monitor performance and can identify when quality of service is threatened in order to take remedial action.

Agility puts **control** into the hands of an organisation's administrators through its support for pluggable policy. Policy is a set of rules or guidelines that direct an organisation's behaviour and within *Agility* is structured as one or more Policy Modules each of which represents different aspects of an organisation's policy.

Agility's ability to capture and then dynamically vary service agreements will encourage enterprises to consume services in the Federated cloud. Without this form of support, enterprises (and individuals) will be restricted to the consumption of flexible services with poorly defined quality of service, or of inflexible services backed by formal contracts.



Arjuna Technologies is a world leading innovator in distributed computing and has an international reputation for delivering mission-critical products to global software vendors. Products developed by the company over the last decade are currently utilised by many thousands of businesses. Today Arjuna is applying its twenty years of research and development experience to enable the next major development in IT - the Federated Cloud.

Web www.arjuna.com
E-mail info@arjuna.com

Tel +44 191 243 0676
Fax +44 191 243 0677

Arjuna Technologies Ltd
Nanotechnology Centre, Herschel Building
Newcastle upon Tyne, NE1 7RU, UK