Towards a RESTful Infrastructure for Digital Ecosystems

Alexandros Marinos
University of Surrey
Department of Computing
Guildford, Surrey, GU2 7XH, UK
+44 (0) 1483 68 2650
a.marinos@surrey.ac.uk

Sotiris Moschoyiannis
University of Surrey
Department of Computing
Guildford, Surrey, GU2 7XH, UK
+44 (0) 1483 68 2650
s.moschoyiannis@surrey.ac.uk

Paul Krause
University of Surrey
Department of Computing
Guildford, Surrey, GU2 7XH, UK
+44 (0) 1483 68
p.krause@surrey.ac.uk

ABSTRACT
In this paper, we describe key design aspects of digital ecosystems and how these can be realised in a web-like environment. In previous work we have discussed digital ecosystems in terms of digital infrastructures and the socio-economic context in which these are called to operate. We have framed the concept of a digital ecosystem around complex interactions between interdependent agents and have focused the discussion on important properties such as loose-coupling, no central point of control or failure, sustainability, resilience, and history. In this paper we describe an integrated set of design solutions for operationalising the key principles of digital ecosystems into a software infrastructure. The proposed reference architecture drives the construction of RESTful ecosystems that can support future internet applications, and do this in a way that is backwards compatible with the current web.

Categories and Subject Descriptors
H.5.4 Hypertext/Hypermedia Architectures, H.2.3 Query languages, C.1.4 Distributed architectures

Keywords
Digital Ecosystems, REST, Complex Systems, Architecture

1. Introduction
The concept of “ecosystem” is increasingly being used as a metaphor in business and systems thinking. In itself, this is a more inclusive view than the more traditional hierarchical, or value-chain models – it provides greater acknowledgement of the importance of a community as a whole in sustaining value creation. However, one can go much further in using the metaphor to drive the development of models that have real value in understanding and facilitating transformation in the Digital Economy. In doing so, we shall argue that, there is significant potential for opening up the future internet technologies.

Today’s e-commerce and collaborative infrastructure is present overwhelmingly on the web, with intermediaries such as travel portals composing services by other providers. In this sense it can be said to exhibit some of the properties of a digital ecosystem.

Currently, the web services space is being disrupted by the principles of Representational State Transfer (REST). While it is recently gaining adoption, REST was introduced by Roy Fielding [1], in 2000 to provide a formal description of the architectural style that had emerged in the World Wide Web. A RESTful web service is therefore a service that works in the context of the Web and applies its architectural principles, between them resource-orientation, statelessness, uniform interface, connectedness and resource manipulation through representations. Like many disruptive technologies, REST was not initially seen too poor to pose a threat to the WS-* protocol stack. However with the release of works such as [2] and the uptake by major vendors [3] [4], for human-to-machine and machine-to-machine scenarios is indicating that REST may be the approach with more potential for web-scale systems.

In talking about Digital Ecosystems (DEs) we need to be mindful that the term can mean different things to different people [5]. The definition proposed in [6] draws from the concept of an ecosystem in the purely biological sense. According to Boley and Chang, a digital ecosystem is “an open, loosely coupled, domain clustered, demand-driven, self-organising agent environment, where each agent of each species is proactive and responsive regarding its own benefit/profit (...) but is also responsible to its system.”

A key feature of this is that agents join freely and of their own volition. This is in contrast to a tightly coupled organisation in which the agents have pre-defined roles. The focus is very much on the autonomy of individual agents, and hence the global properties and institutions of such an ecosystem emerge (primarily) through self-organisation. This is a radical and potentially disruptive concept. However, we will argue later in this paper that current digital infrastructures, such as the Internet and its realisation through the Web, constrain the emergence of properties, and instead impose such from outside the (eco)system.

In previous work [7] we have discussed digital ecosystems along two dimensions: response diversity, which refers to the degree of resilience to shocks, and the need for a DE software infrastructure that allows SMEs to fully engage in the Digital Economy. Apart from the financial implications, SMEs are typically carriers of change, opportunity creation and innovation [8]. The discussion along both axes has been given in light of critical properties of an ecosystem such as resilience and sustainability of the socio-economic context and the supporting digital infrastructure.

In this paper we focus on providing a set of design solutions as an operationalisation of the key principles of digital ecosystems into a software infrastructure. In particular, we describe a reference architecture for designing the rules by which different
services are defined and composed in constructing a global digital environment. We discuss how this reference architecture for DEs makes use of existing Internet infrastructure while extending the architecture of the Web so that it can be used in a way that respects and supports the autonomy of SMEs. The proposed design solutions build on the REST architectural style [1] which underlies the web as we know it today. In particular, we show that REST over HTTP is most appropriate for operationalising the basic ecosystems principles in a reference architecture for digital ecosystems. In so doing, we also show RESTful ecosystems to be backwards compatible with the Web.

The remainder of this paper is structured as follows. Section 2 provides the background for framing the concept of a digital ecosystem, as a special case of the more well-studied complex adaptive systems, and draws the connection to aspects of the Web and its current architecture which are necessary, but sometimes found to be missing or be misinterpreted. In Section 3 we propose a suite of design solutions that evolve around the use of the REST architectural style for DEs over the web. Specifically, we cover RESTful services definition and their composition, a behavioural model for the analysis of complex interactions over resources, which require certain transactional guarantees, the issue of designing a fully distributed P2P network that can support RESTful interactions, and the design of a RESTful querying mechanism for returning service characteristics and their corresponding resources. These design solutions are then positioned into the bigger picture of a reference architecture for DEs on the Web in Section 4. The paper finishes in Section 5 with some concluding remarks and possible future extensions of this work.

2. Digital Ecosystems and the Web

New insights have been gained during the last ten years about the application of complexity science as a means for analysing systems comprising multiple actors that stand in layered, dynamic, interrelationships and are influenced by the environment (constraints) in which they operate. By and large, the two overarching features of complex adaptive systems [10] have to do with the dynamic interactions between agents that span different spatial and temporal scales and the emergence of ecosystem behaviours largely due to the global effects of local interactions between actors. A growing number of case studies have revealed the tight connection between resilience, diversity and sustainability of social, economic and ecological ecosystems (e.g. [11] or [7]).

Complex interactions between interdependent agents and the emergence of global properties or norms from local interactions feature prominently in digital ecosystems. In addition, as we have argued in [7], a digital ecosystem exhibits the adaptive capacity to evolve in ways that create opportunities while steering away from those that destabilise because of their potential for causing system dislocations. This adaptive capability in response to external shocks is thought to be improved by considering a set of rules and constraints in designing the digital infrastructure so that it satisfies certain underlying properties which can facilitate, or, at the very least not undermine, the resilience and sustainability of the corresponding social and economic activities.

One crucial property the infrastructure of a digital ecosystem should possess is the absence of central points of command and control, which also become single points of failure. A technological infrastructure that focuses around a small number of centralized hubs sets a fundamental limitation to the response diversity of digital ecosystems [7]. In addition, it creates a dependency on the organizations/agents that have ownership of the central hubs, and in so doing advocates monopolistic tactics as discussed at length in [12].

Another important aspect which in a certain important sense is imperative to lowering the barrier for adoption by SMEs has to with designing for loosely-coupled services. This means that a service is accessible and usable only through its public interface and other services can be composed with it without requiring access to the internal state of execution. It means that the local design and implementation data are kept transparent and service providers reveal only what they want to reveal.

There is strong indication that the behaviour of SMEs from an information systems and management perspective is driven by external uncertainty and independence [12]. On the one hand, small firms aim to remain independent and autonomous, and are prepared to avoid business activities which put their independence and autonomy at risk [13]. On the other hand, small firms are marked by external uncertainty that derives from their low capability to control the external socio-economic environment and conditions, being thus more likely to evolve and change over time than larger organisations [14]. It transpires that it is important to develop digital infrastructures that respect and support the local autonomy of SMEs; the long tail of diversity in any resilient digital ecosystem [7].

Another important feature of digital ecosystems that needs to be taken into account in designing the digital infrastructure is that of the dynamic interrelationships between its entities (agents, or more generally, entities) but also with the environment. By its very nature a digital ecosystem is dynamic environment where linkages between agents at multiple levels are continuously created and destroyed or lost. The fact that global or higher-level interactions are affected by lower-level or local ones gives an additional dimension to the dynamicity of the digital environment. Such interdependencies and the emergence of overall (eco)system behaviour should be reflected in the design of the digital infrastructure. We shall see in Section 3 how higher-level structures arise in the proposed architectural style from local interactions between agents organized in smaller and simpler network structures.

Considering the landscape of distributed computing today, the network that stands out is the Internet and particularly its incarnation as the Web. The browser as a user interaction paradigm has taken over even traditional desktop applications and users of varying technical ability are increasingly comfortable with the concept of URLs and links. At the infrastructure side, the internet and the web are supported by a mature and stable set of hardware and software. As it stands, the web exhibits resilience, scalability and loose coupling to a great extent.

However, the scale of adoption and the new demands and applications it has brought with it are starting to wear the fabric
of the web thin. The reliance on the centrally controlled and only partially distributed DNS system is the source of technical risks [15] and is also manipulated to yield control of information flow by governments [16] and potentially service providers.

Additionally, the barrier to entry as a content provider and the difficulty of scaling applications to the demands of the web has given rise to a number of trends that threaten the distributed nature of the web. One of them is ‘walled gardens’, services that enable users to share content and form connections, but only with other users that use the same service. A phenomenon often seen in walled gardens is the difficulty for a user to make their data fully portable as the service tightly controls access to it, even by the users themselves.

Another potential threat is the trend of cloud computing, which offers vast data centres that developers can deploy their services in and have the data centres deal with the scaling issues. The problem with this trend is that it creates vast and highly optimized monocolures, where a failure in one part of the network can affect large parts of the network itself. So the failure of a single component in a walled garden or cloud computing vendor can make all that part of the Web inaccessible, what is called a single point of control.

Another common threat by both these trends is the threat to user privacy and the exposure to single points of control. In a sense it seems that the web is falling victim to its own success, because while it scales very well as a distributed, scalable and loosely coupled communication medium, it does not deal with distributing control and avoiding the creation of hubs whose failure can affect the entire network. And this lack of design can be understandable given the ad-hoc and incremental way in which the internet has been developed and the age of the basic protocol specifications.

Regardless of the problems, the fact that the web has prevailed as a communications medium, has withstood the global scale of adoption, and is a ubiquitous interaction metaphor makes it valuable as a starting point for considering the realization of a digital ecosystem. Our efforts on providing a software infrastructure for DEs are aimed (at the very least) at supporting large-scale distributed applications that involve both knowledge and business services.

3. Design Solutions
The web is built on an architectural style called Representational State Transfer (REST) coined by Fielding in 1999 [1]. REST is an architectural style which is derived by progressively applying constraints to an initial unconstrained ‘null’ style. REST is an architectural style which is derived by progressively applying constraints to an initial unconstrained ‘null’ style. REST is optimized for simplicity (all constraints), scalability (statelessness) and loose coupling (uniform interface, hypermedia constraint, code on demand) and sacrifices a degree of efficiency to achieve this, which is partially mitigated by the cacheability provisions. This architectural style is a proven and, through the web, widely adopted foundation which is suitable if not sufficient as a digital infrastructure.

In this section we describe key design principles that inform the reference architecture for digital ecosystems. As mentioned before, we are targeting a web-like software infrastructure with specific characteristics so much of the design solutions evolve around effective use of the REST architectural style.

3.1 From SBVR to RESTful services
A RESTful service is the set of resources made available within the authority domain of a single service. This means that multiple resources with distinct URIs may make up a single service. Each resource exists to identify a single concept that is important to the service. These concepts can be as simple as a product or more abstract such as categories, payment methods etc. In a digital ecosystem, we need the ability to describe these resources such that their semantics can be understood by a consumer, whether human or automated, and this consumer can interact with these resources over the uniform interface of HTTP.

Currently the description effort is concentrated around the media type of each resource. However media types lack machine-readability and a standardized range of responsibilities. A media type can describe anything from a serialization format (in the case of application/xml) or be the entry point for an interaction protocol specified in detail, such as (application/atom+xml). As long as the consumer declares understanding of the media type, it is expected to be able to handle all resources of that media type. The registration of media types is managed by IANA and requires a complex procedure. After this is complete, clients need to be updated such that they understand the new media type, often a difficult task which involves multiple organizations and can take years to complete if at all. This process leaves out the long tail of resources that may subscribe to a basic media type that leaves their semantics undeclared as they may not fit one of the more specialized media types. The media type then only yields limited information about the resource, with the rest left up to a human to decipher. What is needed is a media type that allows enough expressivity such that resources within the digital ecosystem can adopt this type and use its expressivity to describe their more specialized semantics, within its framework.

There has been quite a bit of experience with the OMG standard Semantics of Business Vocabulary and Business Rules (SBVR [17]) within the Digital Ecosystems community [9], and this task plays to its strengths. As defined by [18], “SBVR provides a way to capture specifications in natural language and represent them in formal logic so they can be machine-processed”. Users are able to verify the specification directly by reading the structured natural language used by SBVR which can then be parsed and executed by a machine. SBVR, as a consequence of the Business Rules Approach [19] follows its mantra:

“Rules build on facts, and facts build on concepts as expressed by terms. Terms express business concepts; facts make assertions about these concepts; rules constrain and support these facts.”

While SBVR is a meta-model with models natively expressed as logical formulations, its most common serialization is SBVR Structured English. Terms (e.g. flight, seat, departure time),
Fact Types (e.g. passenger has booked seat on flight), and Rules (e.g. It is obligatory that each passenger has checked in at least 30 minutes before the departure time of the flight) are combined into models.

From a RESTful perspective, SBVR terms can be thought of as classes of resources. Also, fact types build on and connect terms and therefore can act as link declarations. A further benefit of SBVR is its Structured English serialization, which allows for descriptions to be readable by humans as well as machines. Since a centralized ontology is not compatible with a loosely coupled digital ecosystem, each resource owner can describe their resources in whichever way is appropriate to their application. Norms and standards that arise from common use should then be encouraged to become an emergent ontology with local variations rather than one imposed in a top-down fashion.

Resource descriptions can then be persisted in distributed storage in a highly replicated manner. This resource repository can then be searched, with the descriptions of the desired services being merged into a new vocabulary and additional rules being written by the consumer that constrain the potential service composition. This set of requirements can be converted to a transaction as shown in [19] a process which is dependent on the availability of a RESTful querying mechanism.

An additional feature of this model is that in case of error in querying or interacting with a service, the service can return the part of the model that causes the refusal of the request causing the consuming service to update its internal representation of the providing service but also change its own description in case it offers this service composition to other consumers as a provider itself. In this fashion, services interact not only on the message level, but also by affecting the models that each has of the other and propagate changes throughout the network when necessary, in what can be seen as higher-order interactions that emerge by and through the basic service interactions.

3.2 Behavioural Modelling and Analysis

As mentioned before, our aim is to provide an integrated set of design solutions that embody certain ecosystem principles and provide an open collaborative environment for interacting agents. In order to support a wide range of collaborative activities in a digital ecosystem, including both business services (e.g. B2B scenarios) and knowledge services (e.g. e-learning), we need to have a thorough understanding of the interactions involved. In distributed web applications we need to be able to assert that the required interactions have been designed properly and the application will always exhibit the desired behaviour.

In this section we outline the basics of a behaviour model for reasoning about distributed and concurrent interactions in the absence of a central point of command and control. The reasoning results in the elaboration of the behavioural scenarios which goes some way in increasing confidence in a successful outcome prior to execution.

When interactions over resources between different users involve several operations, these often need to be executed as a logical unit of work. This implies the concept of all or nothing execution which is central to transactions [20], whether these concern arranging travel for attending a conference or the collaborative editing of a document. It transpires that in order to ensure integrity of data and applications in a digital ecosystem we need to have some way of coordinating the execution of transactions, and do this in a way that satisfies the basic properties of a transactional setting without compromising the loose-coupling of the services involved. The formal model is used to provide the RESTful service interactions with transactional guarantees.

In previous work we have discussed a coordination model for distributed long-running transactions and described a formal semantics for capturing forward and compensating behaviour [21]. In [22] we have shown that interactions over a set of resources in our model, as discussed in Section 3.1, satisfy the constraints of both REST and atomic transactions. In what follows we briefly outline the use of a formal model in obtaining patterns of interactions that can be analysed prior to the deployment of a transaction scenario.

In the absence of a central point of control, the interactions between different agents need to be coordinated in a distributed fashion. This means that it is necessary to keep track (history) of the significant events, such as operations on resources, across the participants of the interaction. Therefore, instead of using a sequential process to describe the interaction from a single (controller component) viewpoint, as done in approaches based on process algebras like CSP [23], we use tuples of sequences to describe the behaviours of the interacting agents during a transaction on a set of resources. This draws upon Shields’ vector languages which provide a generic behavioural model for communicating systems and can express true-concurrency in that they give rise to asynchronous transition systems (ATSs) [24].

For instance, the transaction vector \((a_1, \Lambda, c_1)\) provides a snapshot of the interaction over a RESTful service that involves three resources. In particular, it describes that portion of the interaction in which user 1 has performed an operation \(a_1\) on the corresponding resource and user 2 has performed \(c_1\) on the third resource while no operation has been performed yet on the second resource (\(\Lambda\) denotes the empty sequence). The schemas for generating xml descriptions of the dependencies and the required orderings of the operations on resources can be found here [25].

In describing a given interaction scenario, we end up with a set of such vectors. This comes with a sound mathematical (order-
theoretic) structure that is expressive enough to distinguish between sequential, alternative and concurrent interactions. This means that we can reason about the resulting behaviours, along different paths of execution, without having to rely on centralised coordination. The resulting behavioural patterns (see Figure 1) can be analysed prior to deployment of the transaction as a means of preventing certain anomalies (such as race conditions) which could result in unexpected behaviour when the transaction actually takes place. Further details can be found in [21], [25].

In particular, we have shown in [25] how reasoning against order-theoretic properties of vector languages – namely, discreteness and local left-closure – can be used to identify missing behaviours that infer additional scenarios. These were simply unthought in the initial transaction design or indicate emergent behaviour, e.g. due to the subtle interplay between concurrency and nondeterminism in the interaction. Effectively, this is used to elaborate the initial scenarios of interaction to more comprehensive ones, which are gradually refined to include all desirable orderings of execution, and prohibit any pathological behaviours.

3.3 Distributed P2P network support

In this section we briefly discuss the main considerations behind designing a distributed network that can support the complex interactions taking place in a RESTful ecosystem. There is a consensus forming that peer-to-peer (P2P) solutions lend themselves naturally to digital ecosystem architectures [33]. More specifically, in our approach we are interested in providing a fully distributed P2P network that can support complex interactions between the networked participants.

A digital ecosystem is rather volatile, in terms of the characteristics of the participants (e.g. SMEs as opposed to large enterprises) and the interactions between them. This means that in addition to being fully distributed, the underlying P2P network should be highly dynamic in that it’s topology should continuously adapt to the actual usage made of the network. Moreover, a dynamic network topology is better suited to cope with various kinds of failure, which are likely in a highly volatile environment, especially when there is no central point of command and control.

The approach taken to the underlying network design is based on clusters of nodes, for dynamically forming permanent clusters rather than permanent nodes as is the case with the traditional static super peer solution. This network design is driven by the local interactions, and is described in detail in [26] while an implementation of the key ideas can be found on the Flypeer Open Source Project [32]. Here, we outline the main idea.

Whenever a transaction takes place between different resources available over a network (e.g the Web), a temporary overlay network is created by virtue of the linkages between the participants. The distributed coordination model for carrying out transactions over a set of resources, which was outlined in Section 3.2, feeds into these temporary networks, the so-called Virtual Private Transaction Networks (VPTNs). Typically, such temporary networks would disappear once the transaction completes its execution (irrespective of whether it terminated successfully or not). In contrast, in our design these become the main building block for the underlying P2P network that supports the complex interactions between participating entities.

The basic design feature of the P2P network has to do with the Virtual Super Peer (VSP) construct, shown in Figure 2. These clusters of stable nodes are used instead of the conventional super peer solution and allow for creating a connected network without generating dependency on (a few or) a single network infrastructure provider. A VSP is formed by selecting – to be more precise, by dynamically electing – the most stable node from each VPTN.

![Figure 2. Forming permanent clusters dynamically](image)

Connectivity in an unstructured network is of topmost concern and hence stability is a determining factor in forming VSPs to emulate permanent clusters of nodes in the network. Without going into further detail, stability is a measurement of expected availability time over actual availability time. Hence, it varies over time. The interested reader is referred to [26] for further details.

The VSPs are aggregates of most stable nodes but are formed dynamically and this means the VSP design solution also allows the network to reconfigure itself depending on the behaviour of (layers of) nodes. In so doing, it can withstand certain types of failure that typical scale-free networks find it difficult to recover from such as fragmentation or smart attacks.

This design of the network is an instance where higher-level structures in the supporting network topology of the digital infrastructure emerge from local interactions within smaller and simpler networks. This dynamic reorganization results in the emergence of a P2P network topology that continuously adapts to reflect the actual usage made of it by the participating agents or entities in the digital ecosystem.

A question that quickly comes about for fully distributed networks is that of addressing nodes. For this issue we are inspired by the approach of Git [27], the distributed version control system in which a node can generate a unique identifier...
for a new source code branch while being offline. The insight in this approach is that given a large enough identifier space, collisions are exceedingly unlikely. In our architecture we call this an Effectively Unique Identifier (EUI). Since an identifier can be claimed by anyone, there should be a mechanism whereby the true owner of an identifier can prove ownership. The mechanics of public-private key pairs allow us to guarantee this, having the user generate a public-private key pair. If the public key of the pair is used as part of a user’s identifier, then only that user can read messages encoded with that public key and only that user can produce messages that are decodable with that public key. This is of course based on the assumption that the private key never leaves the possession of the user.

3.4 RESTful SQL Querying

In Section 3.2 we discussed the need for a RESTful querying mechanism. This querying mechanism should allow service providers to be queried for absolute or relative service characteristics and return a result which contains the resources which fit the query. The language needs to be able to express queries to services while remaining within the architectural guidelines of the web. This means that languages such as SPARQL and YQL which use the POST operation for result retrieval cannot be considered as they forego the advantages of cacheability and linkability that proper use of HTTP can provide. Languages such as Microsoft’s .NET Data Services and Google’s GData come closer but fail to exhibit the flexibility of the relational model, lacking the ability to do JOIN and other operations common to SQL queries. Additionally, they do not have an open source reference implementation that can be used. While the work is currently in progress, the principles of the new querying language are that the queries should be expressible in the form of URIs and therefore can be exchanged and stored as bookmarks. The expressive power of the language should be comparable with SQL as much as possible although currently group functions have been left outside the scope. The basic format for the URL queries is:

```
http://host/database/{table(s)}/{filter(s)}?{parameters}
```

The `{table(s)}` variable can be replaced either with a single table or with multiple tables paired with join operators. The operator ‘+’ represents an inner join, ‘!+’ a left outer join, and ‘+!’ a right outer join.

The `{filter(s)}` are then applied to the single table or the table that has resulted from the join. These filters can be applied after other filters to express more fine-grained queries.

The `{parameters}` variable can contain the following elements:

- `!order` – the fields that the results are to be ordered by
- `!show` – the fields that are to be shown in the results
- `!page` – the page number that
- `!pagesize` – the size of the pages in number of records

As a result of making these features available, the prototypical querying language we have presented here can help make information available in the digital ecosystem and the web in a manner that is accessible by other machines. While we have presented a URI syntax for illustration reasons, the queries need not be tightly coupled to this syntax as a clients of a RESTful interface can discover resources by navigating hypermedia links without knowing the URI structure a priori.

This would mean that the results of a query would need to link to other relevant queries such that all possible queries form an interconnected graph, even if these links are produced with the use of forms. To enter the query graph, the entry URI could provide a list of links to each table which on clicking would lead to a simple query on the relevant table. From there the query can be iteratively refined.

4. Reference Architecture

The architecture that implements the ideas discussed in this paper is necessarily similar and compatible to the Web. A node with a browser should be able to participate in the RESTful Digital Ecosystem as it is able to do today in the Web. The only difference at this point is that the browser should be modified to resolve PRIs as well as URIs, which can be done by utilizing the extension capabilities of most modern browsers.

For more involved participation in the digital ecosystem, the same extension mechanism can be used to increase the capabilities of the browser to include a web server. This web server, combined with the identity infrastructure can allow the node to participate in the Digital Ecosystem as a first-class citizen, rendering the designations ‘server’ and ‘client’ mere roles to be assigned on a per-interaction basis. A node with HTTP client and server capabilities can then host its own resources and models of those resources, under the control of its owner and available to the ecosystem.
Beyond the custom resources, there is also the potential for collective capabilities to which many nodes can contribute such that they can appear to their user to be provided by the ecosystem as a whole rather than bespoke nodes. Basic instances of these capabilities are storage and computation services. Discussion of these patterns and the community currency incentive mechanism that can bring them to existence in a distributed system exists in the community cloud computing view of digital ecosystems [31]. These capabilities however, including the incentive mechanism need to be implemented as components in the node as well. Our reference architecture therefore, beyond some shared components, resembles a dual stack (see Figure 2). The one stack deals with local capabilities and data whereas the other is related to distributed capabilities available by the ecosystem as a whole. Between the two and interfacing with them at all levels is the service layer. This layer hosts services, which can be either entirely local or to various degrees distributed, based on the type of capabilities from both stacks that it utilises. This pattern is already in the stage of a prototype implementation [32] lead by Instituto de Pesquisas em Tecnologia e Inovação (IPTI) in the context of the OPAALS EU-FP6-NoE project, and the current roadmap includes further application of the principles of RESTful architecture on top of the in-browser transactional peer-to-peer network implementation it currently provides.

5. Conclusions and Future Work
Taking as a starting point the concept of an ‘ecosystem’ which has been widely used in social, economic and ecological studies, we discussed digital ecosystems in terms of a core set of generic ecosystem principles (e.g. resilience, emergence of global properties, sustainability) and then focused more on the properties of the corresponding software infrastructure (e.g. loose-coupling, no central point of control/failure or dependency). We also discussed the strengths and shortcomings of the Internet, and its incarnation as the Web, as an enabling digital infrastructure.

We described the key design principles behind a reference architecture for digital ecosystems that makes use of existing Internet infrastructure while extending the architecture of the Web (e.g. by introducing an identification mechanism that removes the dependency on the DNS system) so that it can be used in a way that respects and supports the autonomy of SMEs.

We have argued that the adaptive capability of a digital ecosystem in response to external shocks can be improved by considering a set of rules and constraints in designing the corresponding software infrastructure so that it facilitates, or at least does not undermine, the resilience and the emergence of global properties.

The proposed design solutions build on the REST architectural style [1] which underlies the Web as we know it today. In particular, we have argued that REST over HTTP is most appropriate for operationalising basic ecosystems principles in a reference architecture for digital ecosystems.

We have described a framework for formally describing RESTful services, a behaviour model for identifying the patterns of interaction for formal reasoning, a scheme for identity in a decentralised web-like system and a querying mechanism to be used for advanced interactions between RESTful services.

A key feature of the set of design solutions in the proposed reference architecture for RESTful Ecosystems is that it is straightforward to bridge the proposed RESTful ecosystem with the current web, and in so doing greatly ease the bootstrapping process.
6. ACKNOWLEDGMENTS
This work was supported by the EU through the FP6 project OPAALS, Contract No. 034824 FP6-NoE.

7. REFERENCES