Rethinking sustainability: a vision for DCIs

One of the fundamental aspects of our renewed SIENA vision is the focus on long-term sustainability, both of the structures themselves and the assets that they create (like software, knowledge, experience). Sustainability is the property of a project of being able to maintain itself, both through external contributions and internal activities. In this sense, it can be seen as an inherent property that provides the guarantee for external adopters that the project does have the means to survive – and thus can be trusted upon as a platform.

Sustainability is a problem that all the initiatives face; in fact, the EGI/NGI survey analysis states that "half of the NGIs funding is guaranteed only on a year-by-year basis or every two years, while for some of them the funding situation is unclear. In the most critical situations, some of the NGIs do not have a fixed budget and are only being funded on-demand for specific activities, or they have applied for national funding but have yet to receive confirmation. It seems that in most cases, a sustainability funding scheme is still not defined yet funding beyond 2014 is critical."[1] Further pushing the need for sustainability, Public Administrations need a clear long-term vision before committing to the use of a project or infrastructure (taking into consideration the fact that most PAs must provide services for longer time-frames than commercial entities – sometimes 10 years or more). A project that is critically dependent on a single funding source, or for which a clear roadmap is not apparent, may not be qualified to be used as a basis for a long-term IT project by its Public users.

This section is not meant to suggest to individual projects a specific sustainability strategy; merely to provide hints at some potential models that may be analysed and decided upon according to the individual DCI strategy – this taking into account the specific legal limitations that some organisations may have in place, limiting the possible offerings to non-government groups or the kind of service offering possible.

The basis for a sustainability analysis starts with the identification of the "assets", that is the aspects (tangible or intangible) that provide a differentiating value; those assets, in the DCI case, are:

- infrastructure
- software/source code
- knowledge
- non-material assets like "brand name" or privileged access to people and contacts

These assets should be considered as potential tools to be leveraged to obtain external contributions (monetary or not) that contribute to maintain a long-term viability of a platform, or a reduction in the current costs of maintaining it as a way to guarantee continued support even in lieu of a reduction of public funding. Each aspect should be separately analysed, taking in consideration the fact that an organisation may decide to employ a mixed model, that is, an offering directly or indirectly monetised based on the combination of more than one asset (as an example, a PaaS offering that integrates software, support and infrastructure).

Infrastructure: Considering infrastructure as a collection of physical resources that support computational tasks, its potential monetization model is in the form of consumption or fixed lease, where one or all of the properties of the physical infrastructure itself (storage, bandwidth, CPU, memory) can be turned into tradeable commodity units. The most famous example is Amazon EC2,

[1] EGI / NGI Policy Session -- Survey Analysis, retrieved at *go.**egi**.eu/NGI-SAR1 7/6/2011*

where such units are packaged into computational instances, while bandwidth and additional storage are separately priced. Other properties that may be leveraged are non-physical or aggregate properties, like reliability, latency, quality of service, scalability, geographical location (fundamental for processing or storing of privacy-sensitive data) or access to special-purpose hardware (like encryption or video processing cores).

Property	Type	Description
CPU	Pure	Individual computational units
Memory	Pure	Transient, high-speed storage
Storage	Pure	Persistent traditional storage
Connectivity	Pure	Both infra-network and external (internet)
Access to specialized HW	Pure	Specialized computing units, storage (like SSD), networking (dedicated networks, secure interconnects, high-speed infra-center connections)
Geographical location	Pure	Spatial localization – both related to physical properties, and set relation (inclusion, exclusion) to political/legal boundaries
Reliability	Aggregate	Inherent capability of maintaining the initial properties of an asset over a defined period of time
QoS	Aggregate	Capability of respecting a predefined set of properties (eg. Latency, minimum available bandwidth)
Scalability	Aggregate	Availability of on-demand additional asset units in a predefined period of time

Each of these aspects of physical assets may be priced individually or in aggregate form, to create flexible units that can be offered through a fixed list or using a market mechanism.
To be able to offer such a service (even indirectly, through a reselling party) the infrastructure proprietor must be able to provide the necessary insulation properties, scalability and billing (the last aspect usually missing). Since many DCIs lack this part, or may found difficult to engage with commercial entities while at the same time continue their current research efforts, a possible solution may be the creation of a separate spin-off company, that acts as a single intermediary for the acquisition of resources that are then offered to Public Administrations or other actors. This separation of roles guarantees that on the side of the DCIs the administrative burdens are limited, while at the same time provide the necessary support and commercial activities that are necessary for a traditional monetary based exploitation.

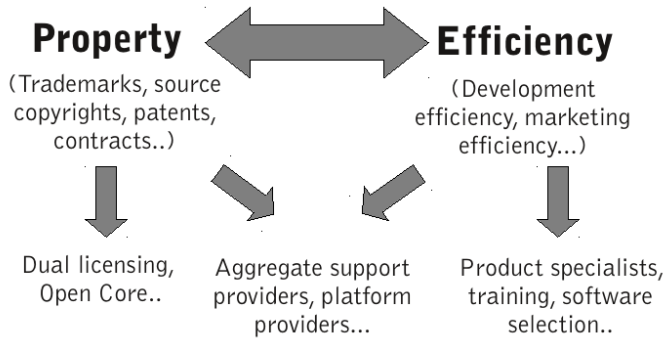
Software (and other non-physical assets): the vast majority of source code produced by DCIs and ancillary projects is composed of code licensed as open source, introducing some additional possibilities and constraints in comparison to the more traditional proprietary licensing. A potential advantage of an open source code base is the opportunity for collaboration with other entities on code development and maintenance, thus substantially reducing the on-going costs for preserving the code and guaranteeing its evolution. Along with knowledge on the code itself, and its potential applications in real world problems, these shared development models may help in turning a project into an ecosystem of entities, with the DCI at the centre to provide the necessary roadmap, and a set of external commercial entities that co-operate on the market, pushing back the changes to improve the original project. This cooperation may reduce the ongoing production costs enough to limit the need of an

external monetization, even though this usually require a long enough time to create an external community of contributors.
There are several examples of such a cooperation in action; for example, the Eclipse project (a development IDE acquired by IBM, and later released in part as open source) demonstrated the power of such a collaboration by creating an ecosystem that contributed more than 1.7B$ in value out of an initial investment of 40M$. It is also important to properly assess the non-code contributions that may provide an equally important value to the project; an example is the OpenCascade kernel: "In the year 2000, fifty outside contributors to Open Cascade provided various kinds of assistance: transferring software to other systems (IRIX 64 bits, Alpha OSF), correcting defects (memory leaks…) and translating the tutorial into Spanish, etc. Currently, there are seventy active contributors and the objective is to reach one hundred. These outside contributions are significant. Open Cascade estimates that they represent about 20% of the value of the software."[2]
As for physical aspects, we can classify potential models that are based on open source licensing using a model that is based on whether the held asset is extrinsic (a property like re-licensing rights, non-open components or trademark) or intrinsic (like a specific knowledge of the source code). These two aspects are weaved into a set of possible business models:

Model	Type	Description
Dual licensing	extrinsic	Two licenses on the same code, one open and one proprietary. Users that want to integrate the code in non-open applications need to pay a licensing fee. Example of this model is Oracle's BerkeleyDB
Open Core	extrinsic	Part of the code is licensed under an exclusive proprietary model; usually a part that provides additional flexibility, convenience or efficiency.
Indirect revenues	intrinsic	Receives an indirect monetization flow from a set of external actors
Product Specialist	intrinsic	Provides consulting, support, training and advice on the use and extension of the code.
Platform Provider	intrinsic	Packages a set of components (all licensed as OSS) under a single structure that facilitates management and reduces maintenance costs. It may optionally provide legal protection like indemnification or ancillary services like training, software selection or provide aggregate support.
R&D cost sharing	intrinsic	Effort for code development and maintenance is shared among a group of independent actors, with an optional central coordination entity.

It is important to recognize the fact that extrinsic models tend to limit substantially the amount of external contributions (due to the additional requirements for code assignments), while some actors may be barred from contributing code under specific licenses. In general, intrinsic models are more efficient for code that already exists, while extrinsic models are usually preferred by external investors like Venture Capital groups, given the similarity with the traditional software market. A simple way to look at the different models it to place them under a continuum between property (something that is sold) and efficiency (an intrinsic property):

[2] RNTL report "New economic models, new software industry economy", 2006



In this continuum models based on property have the least capability to create an external ecosystem and the highest barrier against potential competitors; on the other hand, efficiency-based models have little or no barrier (any third party may enter the market offering services on the same OSS code) but the probability of external contributions is significantly higher.

Combined models: DCIs are in the peculiar situation of being able to offer at the same time both kind of assets- physical and immaterial. It is thus advisable to at least explore the possibility of designing models that take advantage of both aspects, like offering tuned software designed explicitly for the hardware that is offered as an execution platform, or creating a combined hardware-software platform to be offered as a higher-order package compared to current assets providers like Amazon. These Platform-as-a-Service (PaaS) offerings are becoming more and more relevant especially for commercial entities that are just starting to face the problems of manipulating, analyzing and moving big data sets or using specialized algorithms that were up to a few years ago relegated to the scientific and research environments. Examples of such an evolution are Amazon Elastic MapReduce, that provides an on-demand Hadoop service, but opportunities for more complex environments like scientific workflows or language execution environments like R exist and may provide a significant differentiating element compared to existing non-DCI providers.

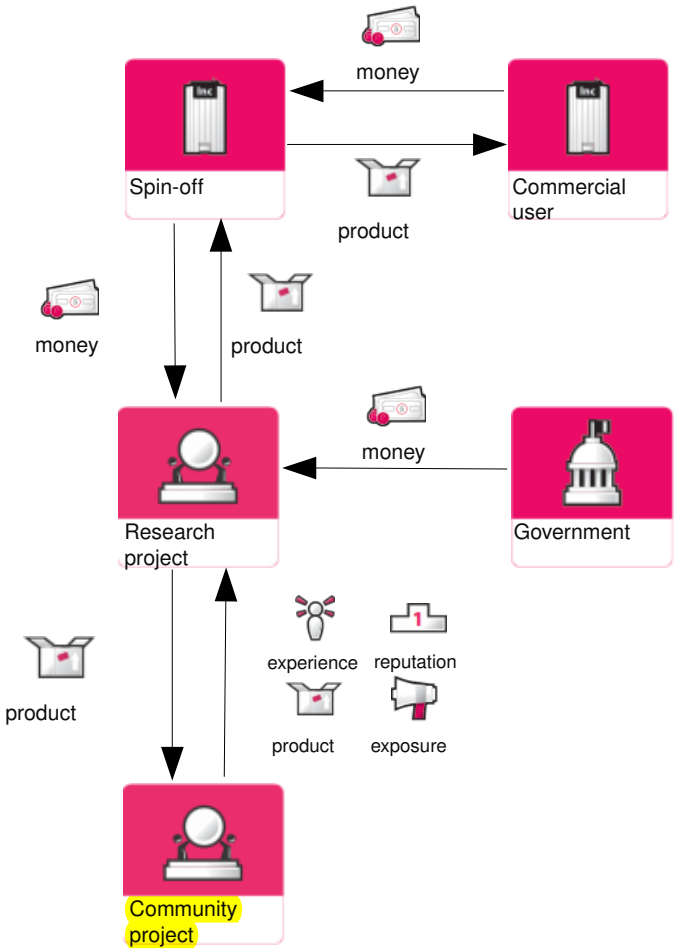
Examples of sustainability models for DCI projects

Our intent is to demonstrate that by selecting individual, relevant "building blocks" from the tables it is possible to create a potential, sustainable business model for selected projects. We will describe, in the follow-up, one such model designed around a software project. It is important to recognize that this should be considered a pure example, and not direct suggestion of whatever model we believe is more valid or structurally better; we believe that each DCI should decide what assets should be leveraged and to what degree.

We consider a research project that maintains two external assets: a community project (that reflects the results of involvement of non-research financed actors) and a commercial spin-off that participates in the commercialization of the results of the research project. The other independent actor is the government/public entity, that provides external founding to proactively promote the development of the research project.

The actors give and take something, depending on their role: for example, the spinoff takes the results of the research ("the product") and invest additional effort in making it suitable for commercial use. This use is then promoted to the end-users, that pay for the enhanced product, and in exchange the spinoff funds the development of the research project on which their work is based. Similarly, the community project is the focus of development that is not based on a monetary exchange; the end users

take advantage of the product in its rough form, and give back both contributions (for example patches, documentation, other material) that collectively becomes part of the research project if valuable, but other immaterial aspects like visibility, experience and reputation, that contribute to the overall value of the research project – and of those derived from the research one, like the spinoff and the community project. This positive feedback can provide substantial value, reducing the cost of maintaining and growing the source code base in a significant way; projects that are organized with this structure can reduce their ongoing costs by more than 50%, enabling not only a better use of the available resources, but increasing visibility and reputation in ways not possible by pure research efforts.
A schematic of such a model can be:



This model can provide not only the necessary independence from external influences (guaranteeing the value of the research project) but an easier flow of funding from additional sources while at the same time providing an additional dissemination channel for the results of the project itself. Actual examples of this model can be found in the OpenNebula project (with several commercial spin-offs), the R statistical language and many others.