SocioPath: In Whom You Trust?

Nagham Alhadad Philippe Lamarre Yann Busnel
Patricia Serrano-Alvarado Marco Biazzini Christophe Sibertin-Blanc

[1] Firstname.Lastname@univ-nantes.fr
[2] Christophe.Sibertin-Blanc@univ-tlse1.fr

[1] LINA / Université de Nantes
2, rue de la Houssinière
BP 92208 – 44322 Nantes, France

[2] Université Toulouse 1 Capitole
2, rue du Doyen-Gabriel-Marty
31042 Toulouse, France

Research Report

September 9, 2011

Abstract

Distributed systems are getting more and more numerous, complex and used in a wide variety of applications. New solutions and new architectures arise (*e.g.*, clouds) that support new functionalities (*e.g.*, social networks). They pile up several software layers and, given that any software is directly dependent of the underlying layers, it can be unable to provide promised services whether any of these layers misbehaves. This evolution implies new non negligible *dependences* increasing with the number of actors involved in the system (*e.g.*, providers and users). Some dependences could be hidden by this layer stacking, implying a reduced transparency for users and a misunderstanding of her actual autonomy. We argue that users should be aware of the potential risks resulting from these dependences. To be able to deduce them, one should know the way the system works (architecture, involved resources, providers, participants, *etc.*). This would help to deduce the potential trust a user could or should have toward the system. We consider this of utmost importance, as professional efficiency and personal privacy could be compromised if untrusted actors control the access to others' resources. This work proposes SocioPath, a generic meta-model that allows to expose hidden or implied relationships among participants in the digital world, which also introduce dependences at the social level. The notions presented in this approach are basics of many fields, like security, privacy, trust, sociology, economy and so forth. SocioPath can be used in the evaluation process of a system as well as in its upstream design.

1 Problem definition

A large number of distributed systems arise nowadays that are more and more complex and used for a tremendous variety of applications. Actually, solutions proposed to users evolve toward new functionalities (*e.g.*, social networks), new architectures that support them (*e.g.*, clouds) and pile up several software layers. This evolution implies new non negligible *dependences* among providers and actors in the system.

When users need to choose a system, they are overwhelmed by the plethora of free and lucrative available options. To make a choice, they *evaluate* systems according to its perceived capability to satisfy their needs and the time they have to make their choice. Traditionally, the evaluation covers functional, technical and economical aspects. From the functional point of view, users analyze the quality of provided services and the user-friendliness of a system. Technical aspects orient the evaluation to operational criteria (*e.g.*, response time, reliability, availability, safety, security, *etc.*) but also to deployment and maintenance requirements. Moreover, economical aspects are considered, like the necessary investments to start up the system and to manage its long-term support.

From one user to another, those evaluation criteria have different weights and consequences and for the same applicative needs, different systems may be chosen. With a more technical approach, one may also consider that functionalities and performance of any software directly depend on underlying layers. If one of these layers misbehaves, the given software may be unable to provide the promised services. Thus the whole system architecture characteristics should be taken into account while choosing the single component.

In general users assume software developers and device manufacturers are competent and have the best intentions. Using a system entails the drawing of relationships of *trust* between users and providers and users are not always aware of those implicit relationships. A kind of "trust among participants" is *de facto* constructed, based on a mix of more or less conscious factors such as the quality of their exchanges [1, 2, 3, 4, 5] or the trust toward resources (data, programs, communications, *etc.*) and providers [6, 7, 8, 9]. In this work we consider that this necessary user's trust must be informed. We argue that users should be aware of the potential risks resulting from their dependences on the system, either by means of public information or by their own deductions. To this end, the way the system works (underlying architecture, involved resources, providers, participants, *etc.*) must be explicit. This would help users to deduce the potential trust they should have toward the system.

SocioPath, the meta-model we propose in this paper, allows to identify those relations among hardware and software components of a system that entail, in the social world, dependences among the actors that are involved in their availability, proper operating or use. The idea is that deduced relationships underline the potential repercussions of the trust users have toward the system in terms of security, privacy, social relationships, economy, *etc.* Thus, when assessing the suitability of a system, we should take into account functional, technical, economical but also dependence-related aspects.

This paper is organized as follows. Section 2 introduces SocioPath. Deduction rules and basic definitions are respectively presented in Sections 3 and 4. Section 5 illustrates different use cases of SocioPath. Finally, Section 6 presents a brief overview of related works and Section 7 gives some conclusions and points out our ongoing works.

2 SocioPath meta-model

SocioPath may be seen as a tool that helps representing the reality of two worlds, which come together and interact between each other: the *social world* and the *digital world*. SocioPath aims at providing a formalism to help the user to answer questions related to her relations in the social world (dependences on persons), as they emerge by the relations existing in the digital world (dependences on resources). Some interesting questions are:

1. *On whom the user depends to access her data?*
   If a user stores her data instances on a server, she depends at least on the provider of the server and on the person who owns the server.

2. *On which applications the user depends to access her data?*
   When data instances are stored outside the user's computer, she may access her data through FTP clients, web browsers, *etc.*

3. *Who can access user's data?*
   When a user stores her data instances on a server, the administrator of the server and the service provider can access her data.

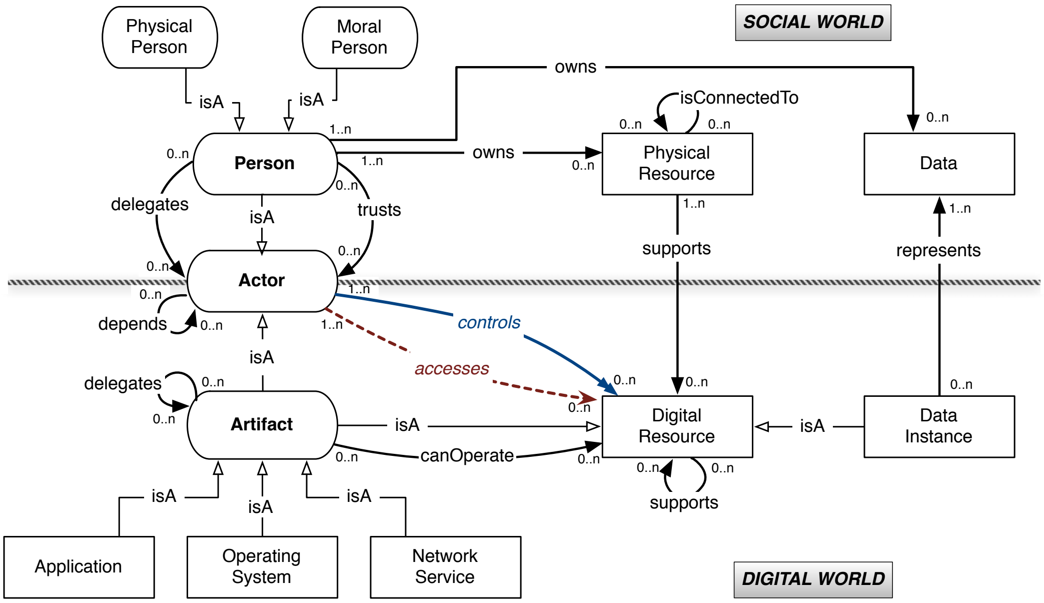

Figure 1: Graphical view of SOCIOPATH.

4. *Through which resources somebody else can access user's data?*
   Some of the resources a user depends on to access her data may be used by other users, to maliciously get the same data.

5. *What are the necessary coalition between persons to access a particular data instance?*
   Those persons on whom a user depends to access her data might be able to access her data, if they colluded.

Figure 1 shows the graphical representation of SOCIOPATH. We then give some of its deduction rules in Section 3 and some definitions in Section 4. Due to space constraints, only a subset of them is presented. All together, these tools are used to point out social dependences implied by the relations in the digital world.

2.1 The social world

The social world describes physical and moral persons (enterprises, companies, *etc.*), physical resources, data and the relations among them.

- *Person*: either a Physical or a Moral Person;
- *Data*: an abstract notion that does not necessarily imply a physical instance (*e.g.*, address, age, software, *etc.*);
- *Physical Resource*: hardware device (*e.g.*, PC, USB device, *etc.*).

2.2 The digital world

The digital world has nodes characterizing digital resources, artifacts, data instances, operating systems, networks services and applications.

- *Data Instance*: a digital representation of Data. It may be semantically equivalent to data that exist in the social world. For instance, a person has an address (Data) in the social world. Whenever she writes it in a file, she creates a semantically equivalent digital instance of her address in the digital world (Data Instance);
- *Artifact*: it may be an Application, an Operating System or a Network Service. We mean all of them to be "running software", thus considering them only in that they are being executed. By *Application*, we mean a whole running entity. It may be a single process or a group of processes that may be distributed in different locations, yet defining a single logically coherent entity;
- *Digital Resource*: an Artifact or a Data Instance;
- *Actor*: a Person in the social world or an Artifact in the digital world.

2.3 The relations in SocioPath

Several relations are drawn in SocioPath. We briefly describe them as follows.

- *owns*: it means ownership. This relation exists only in the social world;
- *isConnectedTo*: it means that two nodes are physically connected. This relation exists only in the social world and it is intrinsically symmetrical;
- *trusts*: relations of trust exist among persons and can be drawn from persons to artifacts. Assessing if and how a given architecture "deserves" the users trust toward the system is one of the future goal of our work;
- *delegates*: a Person can delegate another Actor to perform some kind of access or control on a resource. The same concept of delegation can be implemented among artifacts (*e.g.*, in large databases distributed transactions are often performed by means of chains of delegations);
- *canOperate*: it means that the artifact is able to process, communicate, interact with the target resource. This ability may be given as a part of the artifact specification (*e.g.*, MSWord canOperate the document toto.doc) or deduced by some contingent property of the system (*e.g.*, an operating system only canOperate those files that are stored in a mounted partition);
- *accesses*: an Actor can access a Digital Resource (*e.g.*, the operating system accesses the applications installed on it, or a person who owns a PC that supports an operating system accesses this operating system). The access relations we consider are: read, write, execute;
- *controls*: an Actor can control a Digital Resource. There may be different kinds of control relations. For instance, a moral person, who provides a resource to other persons, controls the functionality of this resource. The persons who use this resource have some control on it as well. Each of these actors controls the resource in a different way;
- *depends*: an Actor may depend on another Actor to perform an activity (*e.g.*, a person depends on Google when she accesses her data instances by using the GoogleDocs application);
- *supports*: it means that the target node could never exist without the source node. We may say that the latter allows the former to exist (*e.g.*, a running operating system exists only if it is hosted on a given hardware; an application is supported by the operating system that hosts it; the code of an application supports this application);
- *represents*: it is a relation that exists between data in the social world and their instances on the digital world (*e.g.*, the source code of the Windows operating system is a representation in the digital world of the data known as "Microsoft Windows ©" in the social world).

The cardinality of these relations is given in Figure 1. Notice that the relations are not generally symmetrical. An example of how to read our notation is the following: for the relation "*owns*", a Person owns $[0..n]$ resources, while every Resource is owned by at least one Person ($[1..n]$).

By applying SOCIOPATH, it is possible to make non-trivial deductions about relations among nodes. For instance, an actor may be able to access digital resources supported by different physical resources connected to each other (*e.g.*, a user can access processes running on different hosts).

Every person owns data in the social world. These data have a concrete existence in the digital world if they are represented by data instances and supported by physical resources. As an actor in the digital world, a person can access and control data instances representing her (and others') data. This may be possibly done through chains of delegations, or by accessing different resources, thus implying some dependence on other persons. In this work, we are particularly interested in formalizing the relations in the digital world, in order to derive the dependences among persons in the social world.

At the present stage of our research, we are focusing on what the users may be able to do, rather than on what they are permitted to do. Thus, imposed access and/or control restrictions are not considered here. A consequence is that, for instance, whenever we find that a person has an access to a resource, we do not imply that she is also granted the permission to actually access it. Similarly, we do not define the various kinds of control over resources; we rather consider the notion of control in its general meaning. Part of our future work will be devoted to specify the types of control and to study the integration of access control constraints in the meta-model.

2.4 Example of model: use case on a single PC

Figure 2 shows a basic model of a use case on one PC. In the social world, a user John owns some Data and a PC. There are also moral persons as Microsoft (provider of Windows, MSWord and MSExcel), Apple (provider of MacOS) and Oracle (provider of OOWrite). The rightmost part of Figure 2 clarify what we mean by "provider".

In the digital world, two operating systems exist on John's PC: Windows and MacOS. On Windows, two applications are available: MSWord and MSExcel. On MacOS are installed OOWrite and Pages. John's Data are represented in the digital world by the document toto.doc.

This example will be used int the next section to illustrate some deduction rules. We deliberately choose a simplified instance representation, in order to intelligibly show how SOCIOPATH can be applied and how its rules and deductions are drawn. We are aware that most of the conclusions here are somehow trivial. We are mainly interested in clarifying the application of our meta-model.

3 SOCIOPATH deduction rules

We use a language based on First Order Logic (FOL) to describe the model of a specific architecture. The edges between nodes are described by binary predicates, for instance, $supports(OS, F)$ means that the operating system OS supports the artifact F. Moreover, we propose some rules, based on this language, that formalize the relations in the architecture. Table 1 summarizes all the notations used in the following.

In the remainder of this section, we define and exemplify some deduction rules of SOCIOPATH concerning the relations access and control. These rules are not exhaustive and by no mean we pretend them to capture the whole complexity of a system. They capture several aspects of a simplified vision of the systems that serves the purpose of building an understandable and expressive model.

- An artifact accesses a digital resource, if the artifact can operate the digital resource and the artifact and the digital resource are supported by the same physical resource, or supported by different physical resources connected to each other.

$$\forall F \in \mathbb{F}, \forall R \in \mathbb{R}, \forall \overline{R}1, \overline{R}2 \in \overline{\mathbb{R}} : \bigwedge \begin{cases} canOperate(F, R) \\ supports(\overline{R}1, F) \\ \bigvee \begin{cases} supports(\overline{R}1, R) \\ \bigwedge \begin{cases} supports(\overline{R}2, R) \\ isConnectedTo(\overline{R}1, \overline{R}2) \end{cases} \end{cases} \end{cases} \Rightarrow accesses(F, R) \quad (1)$$

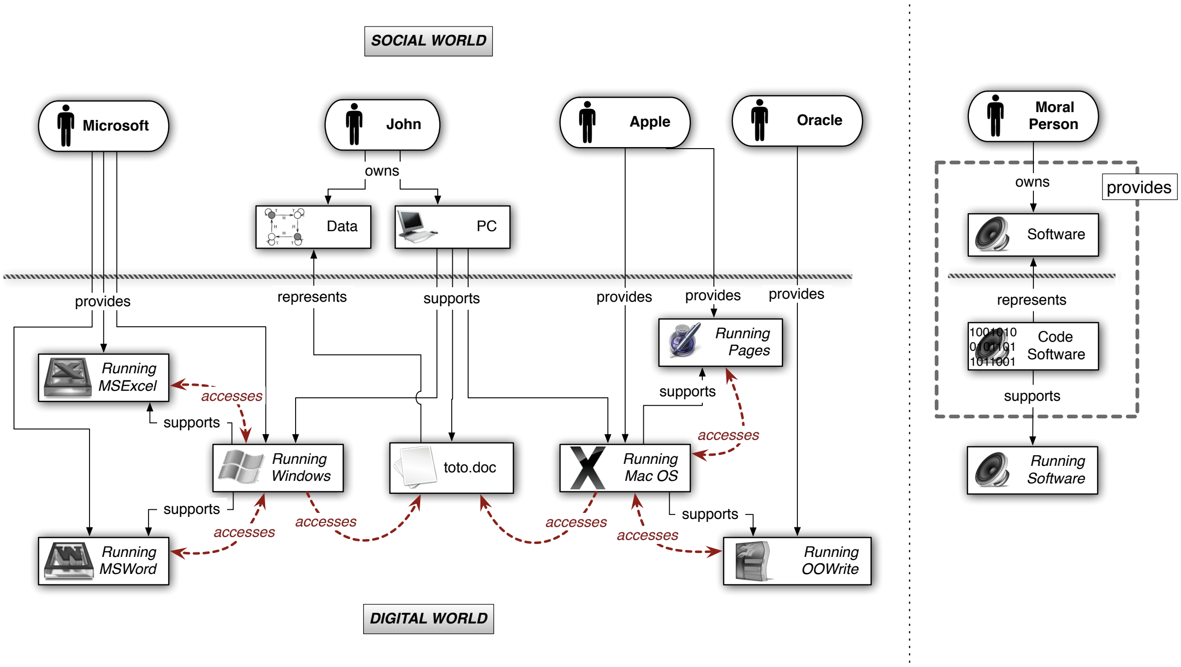

Figure 2: Use case example: a document accessed by 2 different operating systems.

e.g., Windows accesses MSWord:
$canOperate(\text{Windows}, \text{MSWord}) \wedge supports(\text{PC}, \text{Windows}) \wedge supports(\text{PC}, \text{MSWord}) \Rightarrow accesses(\text{Windows}, \text{MSWord})$.

- If an operating system supports an artifact and can operate this artifact, it controls this artifact.

$$\forall F \in \mathbb{F}, \forall OS \in \mathbb{O} : supports(OS, F) \wedge canOperate(OS, F) \Rightarrow controls(OS, F) \tag{2}$$

e.g., Windows controls MSWord, Windows controls MSExcel, MacOS controls OOWrite, MacOS controls Pages:
$supports(\text{Windows}, \text{MSWord}) \wedge canOperate(\text{Windows}, \text{MSWord}) \Rightarrow controls(\text{Windows}, \text{MSWord})$;
$supports(\text{Windows}, \text{MSExcel}) \wedge canOperate(\text{Windows}, \text{MSExcel}) \Rightarrow controls(\text{Windows}, \text{MSExcel})$;
$supports(\text{MacOS}, \text{OOWrite}) \wedge canOperate(\text{Windows}, \text{MSExcel}) \Rightarrow controls(\text{MacOS}, \text{OOWrite})$;
$supports(\text{MacOS}, \text{Pages}) \wedge canOperate(\text{Windows}, \text{MSExcel}) \Rightarrow controls(\text{MacOS}, \text{Pages})$.

- A person, who owns a physical resource that supports an operating system, accesses this operating system.

$$\forall P \in \mathbb{P}, \forall \overline{R} \in \overline{\mathbb{R}}, \forall OS \in \mathbb{O} : owns(P, \overline{R}) \wedge supports(\overline{R}, OS) \Rightarrow accesses(P, OS) \tag{3}$$

e.g., John accesses Windows:
$owns(\text{John}, \text{PC}) \wedge supports(\text{PC}, \text{Windows}) \Rightarrow accesses(\text{John}, \text{Windows})$.

- A person, who owns a physical resource that supports an operating system, controls this operating system.

$$\forall P \in \mathbb{P}, \forall \overline{R} \in \overline{\mathbb{R}}, \forall OS \in \mathbb{O} : owns(P, \overline{R}) \wedge supports(\overline{R}, OS) \Rightarrow controls(P, OS)) \tag{4}$$

e.g., John controls Windows:
$owns(\text{John}, \text{PC}) \wedge supports(\text{PC}, \text{Windows}) \Rightarrow controls(\text{John}, \text{Windows})$.

Basic type of instance	The set of all instances		A subset of instances		One instance	
	Notation	*Remark*	*Notation*	*Remark*	*Notation*	*Remark*
Person	$\mathbb{P}$	$\{P : person(P)\}$	$\mathcal{P}$	$\mathcal{P} \subset \mathbb{P}$	P	$P \in \mathbb{P}$
Actors	$\mathbb{A}$	$\{A : actor(A)\}$	$\mathcal{A}$	$\mathcal{A} \subset \mathbb{A}$	A	$A \in \mathbb{A}$
Artifact	$\mathbb{F}$	$\{F : artifact(F)\}$	$\mathcal{F}$	$\mathcal{F} \subset \mathbb{F}$	F	$F \in \mathbb{F}$
Digital resource	$\mathbb{R}$	$\{R : resource(R)\}$	$\mathcal{R}$	$\mathcal{R} \subset \mathbb{R}$	R	$R \in \mathbb{R}$
Physical resource	$\overline{\mathbb{R}}$	$\{\overline{R} : phyresource(\overline{R})\}$	$\overline{\mathcal{R}}$	$\overline{\mathcal{R}} \subset \overline{\mathbb{R}}$	$\overline{R}$	$\overline{R} \in \overline{\mathbb{R}}$
Data	$\mathbb{D}$	$\{D : data(D)\}$	$\mathcal{D}$	$\mathcal{D} \subset \mathbb{D}$	D	$D \in \mathbb{D}$
Data instance	$\overline{\mathbb{D}}$	$\{\overline{D} : dataInstance(\overline{D})\}$	$\overline{\mathcal{D}}$	$\overline{\mathcal{D}} \subset \overline{\mathbb{D}}$	$\overline{D}$	$\overline{D} \in \overline{\mathbb{D}}$
Operating System	$\mathbb{O}$	$\{OS : operatingSystem(OS)\}$	$\mathcal{O}$	$\mathcal{O} \subset \mathbb{O}$	OS	$OS \in \mathbb{O}$
Path	Γ	$\{\sigma : path(\sigma)\}$	Υ	$\Upsilon \subset \Gamma$	σ	$\sigma \in \Gamma$
Architecture	Λ	—	—	—	α	$\alpha \in \Lambda$
Criteria	$\mathbb{C}$	—	—	—	$\mathcal{C}$	$\mathcal{C} \in \mathbb{C}$
Activity	$\mathbb{W}$	—	—	—	ω	$\omega \in \mathbb{W}$

Table 1: Glossary of notations

- A person, who owns data represented in the digital world by a data instance which supports an artifact, controls this artifact.

$$\exists P \in \mathbb{P}, \exists D \in \mathbb{D}, \exists \overline{D} \in \overline{\mathbb{D}}, \exists F \in \mathbb{F} : \bigwedge \begin{cases} owns(P, D) \\ represents(\overline{D}, D) \\ supports(\overline{D}, F) \end{cases} \Rightarrow controls(P, F) \tag{5}$$

e.g., Microsoft controls Windows:
$owns(\mathsf{Microsoft}, \mathsf{Windows}) \wedge represents(\texttt{CodeWindows}, \mathsf{Windows}) \wedge supports(\texttt{CodeWindows}, \mathsf{Windows}) \Rightarrow controls(\mathsf{Microsoft}, \mathsf{Windows})$.

- The relation ‘accesses’ is transitive.

$$\forall A \in \mathbb{A}, \forall F \in \mathbb{F}, \forall R \in \mathbb{R} : accesses(A, F) \wedge accesses(F, R) \Rightarrow accesses(A, R) \tag{6}$$

e.g., MSWord accesses Windows, and Windows accesses `toto.doc`, so accesses `toto.doc`:
$accesses(\mathsf{MSWord}, \mathsf{Windows}) \wedge accesses(\mathsf{Windows}, \texttt{toto.doc}) \Rightarrow accesses(\mathsf{MSWord}, \texttt{toto.doc})$.

- The relation ‘controls’ is transitive.

$$\forall A \in \mathbb{A}, \forall F_1, F_2 \in \mathbb{F} : controls(A, F_1) \wedge controls(F_1, F_2) \Rightarrow controls(A, F_2) \tag{7}$$

e.g., John controls windows and windows controls `toto.doc` so John controls `toto.doc`:
$controls(\mathsf{John}, \mathsf{Windows}) \wedge controls(\mathsf{Windows}, \texttt{toto.doc}) \Rightarrow controls(\mathsf{John}, \texttt{toto.doc})$

4 SocioPath definitions

In this section we define some concepts we apply during the analysis of a system model. By means of these concepts we can deepen the understanding of the system and ultimately enlighten the actor’s dependences on resources and persons (thus addressing both the social world and the digital world) and the degree of these dependences. The examples in this section correspond to the system presented in Section 2.4.

4.1 Path

Definition 1 (Access path)**.**
A path $\sigma$is a list of actors and digital resources such that:

- $actor(\sigma[1])$*;*
- $\forall i \in [2 : |\sigma|], artifact(\sigma[i]) \wedge accesses(\sigma[i-1], \sigma[i])$*;*
- $resource(\sigma[|\sigma|])$*;*

where $\sigma[i]$, respectively $|\sigma|$, denotes the i^{th} element of σ, respectively the length of σ.
The access paths in the architecture α is noted Υ_α or, simply Υ where there is no ambiguity for α.

Examples of the access paths in the architecture presented in the Section 2.4 are the following:

```
[John, Windows, MSWord, Windows, toto.doc],
[Windows, toto.doc],
[Windows, MSExcel, Windows, toto.doc],
[MacOS, OOWrite, MacOS, toto.doc],
[John, MacOS, OOWrite, MacOS, toto.doc],
[John, Windows, MSExcel, Windows, MSWord, Windows, toto.doc],...
```

When a path σ_2 uses additional resources with respect to σ_1, it is quite natural to say that σ_1 is longer than σ_2. This leads to a notion of order.

Definition 2 (Order over paths)**.**
Let α be an architecture. Let σ_1 and σ_2 be two paths within α: $\sigma_1, \sigma_2 \in \Upsilon_\alpha$
$\sigma_1 \leq \sigma_2$ iff[1] *there exists a function f such that:*

- $\left\{ \begin{array}{cccc} f: & \sigma_1 & \mapsto & \sigma_2 \\ & \sigma_1[i] & \rightarrow & \sigma_2[j] \end{array} \right.$
- $\forall i \in [1 : |\sigma_1|] : f(\sigma_1[i]) = \sigma_2[i]$
  i.e.*, The function f maps elements of σ_1 to identical elements in σ_2.*
- $\forall i \in [1 : |\sigma_1| - 1] : \xi(f(\sigma_1[i]), \sigma_2) \leq \xi(f(\sigma_1[i+1]), \sigma_2)$*, where $\xi(x, \sigma)$ gives the rank of x in the path σ.*
  i.e.*, The elements are in the same order within the two paths*

Notation: Consider the following notations:

$$\sigma_1 = \sigma_2 \iff \sigma_1 \leq \sigma_2 \text{ and } \sigma_2 \leq \sigma_1$$
$$\sigma_1 < \sigma_2 \iff \sigma_1 \leq \sigma_2 \text{ and } \sigma_2 \nleq \sigma_1$$

According to the aforementioned example (see Section 2.4), we can notice that $\sigma_1 < \sigma_2$:

$$\left\{ \begin{array}{l} \sigma_1 = \texttt{[John, Windows, MSWord, Windows, toto.doc]}, \\ \sigma_2 = \texttt{[John, Windows, MSWord, Windows, MSWord, Windows, toto.doc]}. \end{array} \right.$$

Definition 3 (Minimal path)**.**
Let α be an architecture. Let Υ be a set of paths within an architecture α.
A path σ of Υ is said to be minimal in Υ iff

$$\nexists \sigma' \in \Upsilon : \sigma' <_f \sigma.$$

Notation: The set of minimal paths is noted $\widehat{\Upsilon}$.

[1] *iff* denotes “if and only if”.

In our example, `[John, Windows, toto.doc]` is a minimal path from a user John to `toto.doc`.

Different paths may be used to for the same activity. To be able to focus on subsets of these paths, we define some criteria.

Definition 4 (Criteria over paths)**.**
Let Υ be a set of paths. A criterion is a function that:

$$\mathcal{C} : \Upsilon \mapsto \{true, false\}$$

Notation: $\Upsilon^{|\mathcal{C}} = \{\sigma : \sigma \in \Upsilon \wedge \mathcal{C}(\sigma) = true\}$ *is the subset of* Υ *restricted to the correct paths with respect to some criteria* $\mathcal{C}$*.*

Criteria may be used to select some paths, for instance:

- To select which paths enable a person to access a data instance.
- To select which paths enable a person to understand a data instance (*e.g.*, to understand a `.doc` document, a person needs to use an application like MSWord or OOWrite, *etc.*).
- To select which paths make natural use of resources (*e.g.*, when a user use a SVN service, the path should contain the SVN client and SVN server to be able to access the document, and the SVN client should precede the SVN server).

We can define *functionally equivalent* access paths, which are the set of paths that comply with specific criteria. Consider two paths σ_1 and σ_2 such that

$$\sigma_1[1] = \sigma_2[1] \quad \wedge \quad \sigma_1[|\sigma_1|] = \sigma_2[|\sigma_2|].$$

The set of equivalent paths for the actor A to access resource R is noted $\Upsilon^{A,R}$.

It is also worth considering the minimal functionally equivalent paths, noted $\widehat{\Upsilon^{A,R}}$. The formal definition of the criteria over paths can be made in different ways, like, for instance, by using regular expressions.

4.2 Activities

Intuitively, an activity is related to an actor who wants to do something concerning a resource (*e.g.*, to access a directory, to copy a file, to edit a document, *etc.*). We assume that the criteria allow to obtain the complete set of paths corresponding to an activity.

Consider the following example of activity: "John accesses `toto.doc`". The corresponding criterion $\mathcal{C}$ should be:

$$\sigma_1[1] = \textsf{John} \quad \wedge \quad \sigma_1[|\sigma_1|] = \texttt{toto.doc}.$$

4.3 Dependence

We are now able to define the concepts of actor's dependence on a set of artifacts and on a set of persons, along with the degree of these dependences.
Let ω be an activity related to an actor A and concerning a resource R, and $\mathcal{C}$ be its criteria.

Definition 5 (Artifact classification for an activity)**.**

Usable artifact

The usable artifacts are the artifacts that appear in one of the access path.

$$F \textit{ is a } \text{usable artifact} \textit{ for } \omega \text{ iff } \exists \sigma \in \Upsilon^{|\mathcal{C}}, F \in \sigma.$$

e.g.*, MSExcel is a usable artifact for the activity: "John accesses* `toto.doc`*".*

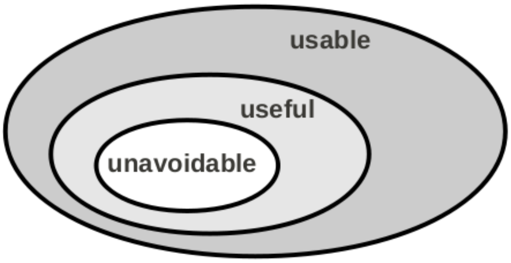

Figure 3: Artifact classification with respect to an activity.

Useful artifacts

The useful artifacts are artifacts that appear in one of the minimal path.

$$F \textit{ is a } \text{useful artifact} \textit{ for } \omega \text{ iff } \exists \sigma \in \widehat{\Upsilon^{|\mathcal{C}}}, F \in \sigma.$$

e.g., *Windows is a useful artifact for the activity: "John accesses* `toto.doc`*".*

Unavoidable artifact

the unavoidable artifacts are the artifacts that appear in every minimal path.

$$F \textit{ is an } \text{unavoidable artifact} \textit{ for } \omega \text{ iff } \forall \sigma \in \widehat{\Upsilon^{|\mathcal{C}}}, F \in \sigma.$$

e.g., *If the user John has only the Windows operating system, then Windows is an unavoidable artifact for the ω: "John accesses* `toto.doc`*".*

Figure 3, illustrates the classification of the artifacts with respect to an activity.

Definition 6 (Actor's dependence on a set of artifacts for an activity)**.**

Let $\mathcal{F}$ be a set of artifacts.

$$A \text{ depends } \textit{on } \mathcal{F} \text{ iff } \forall \sigma \in \Upsilon^{|\mathcal{C}}, \exists F \in \mathcal{F} : F \in \sigma$$

For instance, one of the sets John depends for the activity "John accesses `toto.doc`" is: `[Windows, MacOS, MSWord, OOWrite, Pages]`.

Definition 7 (Actor's strict-dependence on a set of artifacts for an activity)**.**
Let $\mathcal{F}$ be a set of artifacts.

$$A \textit{ strictly-depends on } \mathcal{F} \text{ iff } \forall \mathcal{F}' \subsetneq \mathcal{F}, ((A \textit{ depends on } \mathcal{F}) \wedge \neg(A \textit{ depends on } \mathcal{F}')).$$

For instance, John strictly-depends on the set `[Windows, MacOS]` for the activity: "John accesses `toto.doc`".

Definition 8 (Degree of actor dependence on a set of artifacts for an activity)**.**
Let $\mathcal{F}$ be a set of artifacts.

The degree of dependence of A on $\mathcal{F}$, noted $d^A_{\mathcal{F}}$, is based on frequency of presence of $\mathcal{F}$ elements in the paths related to the activity ω:

$$d^A_{\mathcal{F}} = \frac{|\{\sigma : \sigma \in \widehat{\Upsilon^{|\mathcal{C}}} \wedge \exists F \in \mathcal{F}, F \in \sigma\}|}{|\widehat{\Upsilon^{|\mathcal{C}}}|}.$$

There are three minimal paths for the activity “John edits toto.doc“:

- `[John, Windows, MSWord, Windows, toto.doc];`
- `[John, MacOS, OOWrite, MacOS, toto.doc];`
- `[John, MacOS, Pages, MacOS, toto.doc].`

The degree of dependence of John on MSWord for the activity “John edits the document toto.doc” is equal to 1/3.
The degree of dependence of John on MacOS for the activity “John edits the document toto.doc” is equal to 2/3.
The degree of dependence of John on the set `[MacOS, MSWord]` for the activity “John edits the document toto.doc” is equal to 1.
The degree of dependence of John on the set `[Pages, MSWord]` for the activity “John edits the document toto.doc” is equal to 2/3.
The degree of dependence of John on the set `[Windows, MSWord]` for the activity “John edits the document toto.doc” is equal to 1/3.
The degree of dependence of John on the set `[Pages, OOWrite]` for this activity is equal to 2/3.

Definition 9 (A set of persons controls a set of resources)**.**
Let $\mathcal{R}$ be a set of resources, and $\mathcal{P}$ be a set of persons,

$$\mathcal{P} \textit{ controls } \mathcal{R} \text{ iff } \wedge \begin{cases} \forall R \in \mathcal{R}, \exists P \in \mathbb{P} : controls(P,R) \Rightarrow P \in \mathcal{P} \\ \forall P \in \mathcal{P}, \exists R \in \mathcal{R} : controls(P,R). \end{cases}$$

For instance, the set `[Microsoft, Apple, John]` controls `[Windows, MacOS].`

Definition 10 (Actor’s dependence on a set of persons for an activity)**.**
Let $\mathcal{P}$ be a set of persons.

$$A \textit{ depends on } \mathcal{P} \textit{ for } \omega \text{ iff } \wedge \begin{cases} \exists \mathcal{F} \subset \mathbb{F} : A \textit{ depends on } \mathcal{F} \textit{ for } \omega \\ \mathcal{P} \textit{ controls } \mathcal{F}. \end{cases}$$

For instance, John depends on `[John, Microsoft, Apple]` for the activity: “John accesses `toto.doc`”.

Definition 11 (Degree of actor’s dependence on a set of persons for an activity)**.**
Let $\mathcal{P}$ be a set of persons.

The degree of dependence of A on $\mathcal{P}$, noted $d^A_{\mathcal{P}}$ is based on frequency of presence of $\mathcal{P}$ persons who control F in the paths Υ related to the activity ω:

$$d^A_{\mathcal{P}} = \frac{|\{\sigma : \sigma \in \widehat{\Upsilon^{|\mathcal{C}}} \wedge \exists P \in \mathcal{P}, \exists F \in \sigma, controls(P,F)\}|}{|\widehat{\Upsilon^{|\mathcal{C}}}|}$$

For instance, the degree of dependence of John on Apple for the activity “John edits the document toto.doc” is equal to 2/3.
The degree of dependence of John on Oracle for the activity “John edits the document toto.doc” is equal to 1/3.
The degree of dependence of John on Microsoft for the activity “John edits the document toto.doc” is equal to 1/3.
The degree of dependence of John on the set `[Microsoft, Apple]` for the activity “John edits the document toto.doc” is equal to 1.
The degree of dependence of John on the set `[Microsoft, Oracle]` for the activity “John edits the document toto.doc” is equal to 2/3.
The degree of dependence of John on the set `[Oracle, Apple]` for the activity “John edits the document toto.doc” is equal to 2/3.

By means of these definitions, we can finally be explicitly aware of the user’s dependences in the social world. This may be useful to assess the potential trust she should have toward the various participants in the system.

5 Examples

5.1 GoogleDocs

To illustrate the meta-model, Figure 4 presents a model drawn by applying SocioPath. This model represents a system where a user uses GoogleDocs.

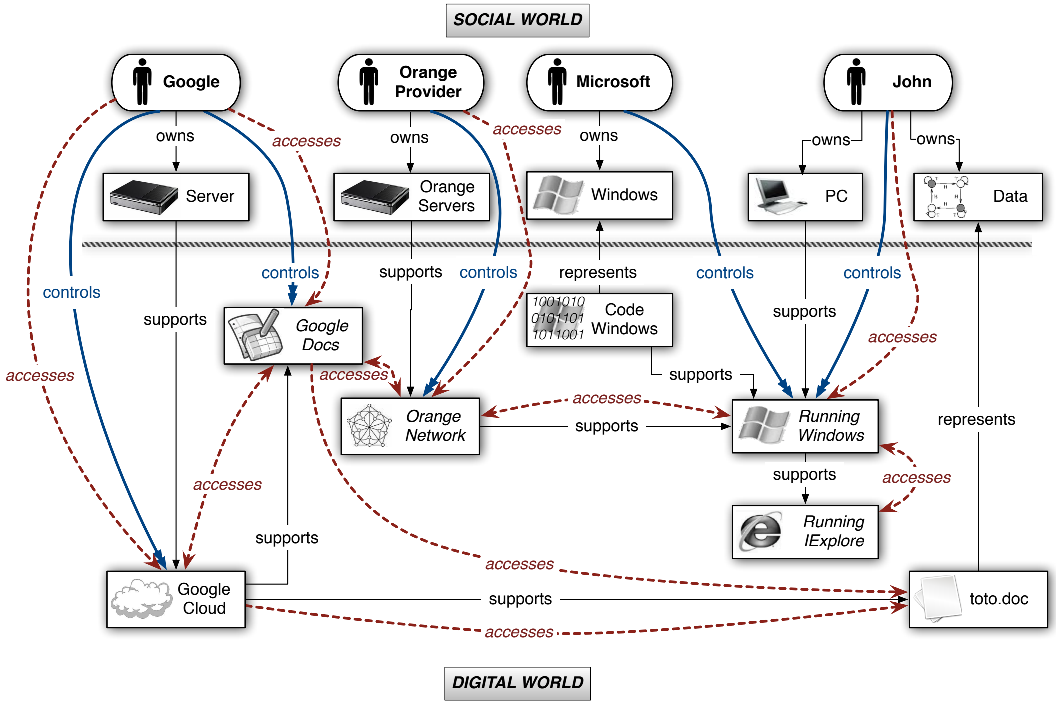

Figure 4: GoogleDocs snapshot

In the social world, the person 'John' owns some data and a PC. There are also Microsoft (provider of Windows and Internet Explorer) and Google (provider of GoogleDocs service and Google Cloud).

In the digital world, the Windows operating system is running on John's PC and it supports Internet Explorer. John's data are represented in the digital world by the document `toto.doc` which is supported by the physical resources owned by Google. We consider Google Cloud as the storage system used by the application GoogleDocs.

According to SocioPath and its rules, John accesses Windows, which is supported by his PC. As Windows supports Internet Explorer, they can access each other and Windows controls Internet Explorer.

The physical resource that supports the network is connected to John's PC and to the physical resources owned by Google, so the network services, GoogleDocs and Windows may access each other. The Google Cloud supports GoogleDocs that can access the file `toto.doc`.

Thus, if John wants to access his document, he passes through Windows, then Internet Explorer, then Windows again. Then by accessing the network, he gets to GoogleDocs, then to Google Cloud, and finally to his document. According to Section 2, 3 and 4, we can answer the questions introduced in Section2.

- **Considering the activity "John accesses `toto.doc`"**:

  1. On whom John depends to access his data?

On Microsoft, Network providers and Google.

2. On which applications John depends to access his data?

   On Windows, Network, GoogleDocs and Google Cloud.

- **Considering the activity "Other persons access `toto.doc`"**:

3. Who can access John's data?

   Microsoft, Network providers and Google.

4. Through which resources they can access the John's data?

   For Microsoft, through Windows, the network, Google Docs and Google Cloud.

   For the network provider, through the network, Google Docs and Google Cloud.

   For Google only through Google Cloud.

5. What are the necessary coalition between persons to access `toto.doc`?

   A coalition is needed by all the persons listed in the answer to question 4, who do not own all the resources they must use. This makes evident that Google doesn't need to collude with anyone.

- **Considering the activity "John accesses and understands `toto.doc`"**:

6. On whom John depends to understand his data?

   On Microsoft, Network providers and Google.

7. On which applications John depends to understand his data?

   On Windows, Internet Explorer, Network, GoogleDocs and Google Cloud.

5.2 SVN

In the following use case example, we apply SocioPath to a slightly more complex system and deduce some dependences implied by its architecture. Again, what is interesting for us is to show an intelligible way to apply our meta model to a real scenario, rather than presenting surprising outcomes. Figure 5 shows the architecture of a system where the user 'Philippe' uses the application 'SVN' to reach his own document `toto.doc`. The latter is supported by a PC owned by an administrator, and shared with the user 'Patricia'.

Formalizing such a system allows to illustrate the dependences of a user who performs an activity, *e.g.*, to check out the document `toto.doc`. As we said, in this work we do not deal with the formal definition of the criteria that characterize an activity. We just notice here that the activity "Philippe checks out `toto.doc`" implies several non-trivial aspects. For instance, Philippe must use an SVN client connected to an SVN server to access the given document; other possible paths that do not include this constraint are not acceptable, in this specific case.

In the given scenario, Philippe owns a PC that supports MacOS, which supports an SVN client. In order to keep the figure as readable as possible, not all the actors involved in the activities are represented (*e.g.*, the persons who provide the aforementioned software are not shown). Philippe's PC can be connected to different networks: Orange (`OrangeNW`) and SFR (`SFRNW`), which are connected to a specific artifact representing the global Internet network (`InternetNW`). Then, the set of local networks also includes the national french university's network (`RenaterNW`) and the one of the University of Nantes specifically (`UnivNantesNW`). The administrator's PC is connected to `UnivNantesNW`. All the providers of these networks appear in our architecture in the social world (except the ones of University of Nantes, for sake of simplicity). Some nodes do not require a fine-grained detailed description (*e.g.* the networks) and are considered as "black boxes", controlled by their providers.
In the following, whenever necessary to avoid ambiguity, we write the name of the resource owner right before the resource name itself (*e.g.*, the SVN Client owned by Philippe may be noted as `PhilippeSVNclient`).

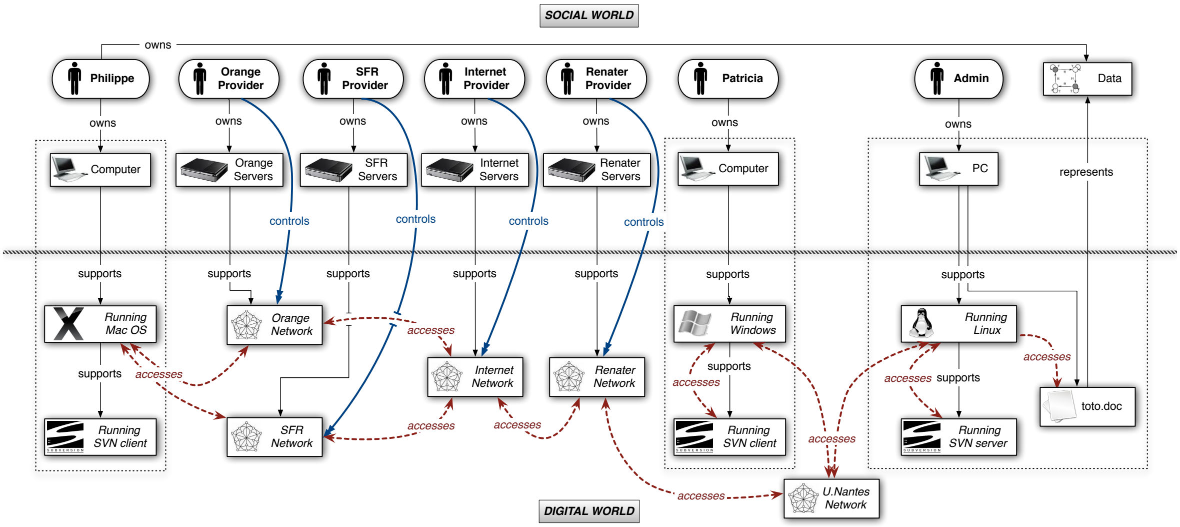

Figure 5: SVN

- **Considering the activity "Philippe checks out `toto.doc`"**:

  Philippe depends on the following sets of artifacts: `[SFRNW, OrangeNW]`, `[PhilippeMacOS]`, `[AdministratorSVNserver]`, `[AdministratorLinux]`, `[InternetNW]`, `[RenaterNW]`, `[UnivNantesNW]`, `[PhilippeSVNclient]`.

  Philippe is forced to pass either through SFRNW or through OrangeNW and not through both of them, so he depends on this set, but not on each of the two artifacts. For he is forced to pass through all the other artifacts, he thus depends on each of them.

  Philippe depends on the following sets of persons: `[SFRNW Provider, OrangeNW Provider]`, `[Philippe]`, `[Apple]`, `[SVN Provider]`, `[InternetNW Provider]`, `[RenaterNW Provider]`, `[UnivNantesNW Provider]`, `[Linux Provider]`, `[Administrator]`.

  Philippe depends on the set `[SFRNW Provider, OrangeNW Provider]` because he is forced to pass through an artifact controlled by either one of them, and his degree of dependence on each of them is equal to 1/2. Beside this case, he totally depends on each of the other persons, because he is forced to pass through the artifacts they control.

- **Considering the activity "Philippe shares `toto.doc` with Patricia"**:

  Let us analyze the paths by means of which Patricia can access `toto.doc`. In this case Patricia has one minimal path: `[Patricia, PatriciaWindows, PatriciaSVNclient, PatriciaWindows, UnivNantesNW, Linux, SVNserver, Linux, toto.doc]`.

  So Philippe depends on the following sets of persons for the activity "Philippe shares `toto.doc` with Patricia": `[SFRNW Provider, OrangeNW Provider]`, `[Philippe]`, `[Patricia]`, `[Apple]`, `[SVN Provider]`, `[InternetNW Provider]`, `[RenaterNW Provider]`, `[UnivNantesNW Provider]`, `[Linux Provider]`, `[Administrator]`.

  If Patricia does not want to (or is not able to) disclose the details of her architecture to Philippe, they will be encapsulated in a black box, controlled by her.

By analyzing the paths that other persons could have to access `toto.doc`, we conclude that each of the following persons has a possible path to access `toto.doc`:
`[SFRNW Provider], [OrangeNW Provider], [Philippe], [Patricia], [SVN Provider], [InternetNW Provider],[RenaterNW Provider],[UnivNantesNW Provider],[Linux Provider],[Administrator].`
This makes evident that the administrator can autonomously access `toto.doc`, whereas all the other persons need to collude at least with the administrator, who controls the access to `toto.doc`.

6 Related work

Frameworks and tools to create models of IT (Information Technology) systems are widely used in the context of EAM (Enterprise Architecture Management). EAM aims at giving a structured description of large IT systems in terms of their business, application, information and technical layers, with the additional goal of understanding how existing architectures and/or applications should be changed to improve business or strategic goals.

EAM frameworks are mainly developed by governmental institutions (FEAF[2], FDIC[3]), defence industries (DODAF[4], NAF[5], AGATE[6]) or large IT consortia (TOGAF[7], GERAM[8], RM-ODP[9]). Their main concern is to assess the interoperability of different systems by clearly defining compatibility standards. There are few examples of proprietary (IAF[10], OBASHI[11], PROMIS[12]) and open source frameworks (TRAK[13], MEGAF[14]) as well, though, which are more business-centric. A key benefit they provide is the ability to support decision making in changing businesses, by bringing together business models (*e.g.*, process models, organizational charts, *etc.*) and technical models (*e.g.*, systems architectures, data models, state diagrams, *etc.*).

The motivation of SOCIOPATH rather focuses on dependence and trust relationships, but it converges with such frameworks in some aspects. In the following we give a brief overview of some of those frameworks.

RM-ODP[15] (Reference Model of Open Distributed Processing), also known as ITU-T Rec.X.901-X.904 and ISO/IEC 10746, is a reference model in computer science, which provides a framework for the standardization of open distributed processing. It supports distribution, interworking, platform and technology independence and portability, together with an enterprise architecture framework for the specification of ODP systems. Besides being mostly concerned with enterprise integration and business-related aspects, RM-ODP is a generic set of standards and tools to create and manage aspect-oriented models of systems. RM-ODP analyzes and decomposes the systems in great details, according to five different viewpoints, each of them being very specific and mainly focusing on standard compliance. Aiming at different goals, SOCIOPATH gives a simpler overview of a system that is meant to inform the users about the relations that are implied by the system architecture, without exposing technical details.

TRAK[16] (The Rail Architecture Framework) is a general systems-oriented architecture framework that can be used to describe both hard and soft systems. It conforms to the standard for architecture description ISO/IEC 42010. TRAK has five architecture perspectives, each of which contains a number of related views (22 in total). The TRAK meta-model specifies the allowed object types that can be used when modelling and the relationships

[2] http://www.whitehouse.gov/omb/assets/fea_docs/FEA_CRM_v23_Final_Oct_2007_Revised.pdf
[3] http://www.fdic.gov/index.html
[4] http://cio-nii.defense.gov/sites/dodaf20/DM2.html
[5] http://www.nhqc3s.nato.int/ARCHITECTURE/_docs/NAF_v3/ANNEX1.pdf
[6] http://www.achats.defense.gouv.fr/article33349
[7] http://pubs.opengroup.org/architecture/togaf9-doc/arch/
[8] http://www.mel.nist.gov/workshop/iceimt97/ice-gera.htm
[9] http://www.rm-odp.net/
[10] http://www.capgemini.com/services-and-solutions/technology/soa/soa-solutions/ent_architecture/iaf/
[11] http://www.obashi.co.uk/
[12] http://pro-mis.com/framework.html
[13] http://trak.sourceforge.net/
[14] http://megaf.di.univaq.it/
[15] http://www.rm-odp.net/
[16] http://trak.sourceforge.net/

between them. Each TRAK view is specified by an architectural viewpoint, which states which stakeholder concerns it addresses, it determines what is shown and how it is presented and how the view must be consistent with other views. The strong goal-oriented approach of TRAK is somehow similar to our focus on users' activities, which are an important concept in SOCIOPATH. Anyway, all TRAK's perspectives and views are meant to evaluate the enterprise architecture with respect to the ability to fulfil a given purpose, while SOCIOPATH aims at revealing hidden relations and dependences among persons and well identified digital actors of the system.

TOGAF[17] (The Open Group Architecture Framework) is a framework for enterprise architecture which provides an approach for designing, planning, implementation, and governance of an enterprise information architecture. It is modelled at four levels: Business, Application, Data, and Technology. SOCIOPATH converges with TOGAF in some concepts of the technology level, where the technical architecture of the enterprise is modelled. Unlike SOCIOPATH, the technical architecture of TOGAF focuses on several aspects of the software engineering process (*e.g.*, system requirement, objectives, maintenance, evolution, reuse, integration) while describing the hardware, software and network infrastructure needed to support the deployment of applications.

The OBASHI methodology provides a framework and method for capturing, illustrating and modelling the relationships, dependencies and data flows between business and IT assets and resources in a business context. The six layers that modelize OBASHI are : Ownership, Business Process, Application, System, Hardware and Infrastructure. OBASHI gives a big picture that helps to design, optimise and monitor a business by analyzing the relations of Connection, Dependence, Spatial, Set, Layer and Sequential). SOCIOPATH converges with OBASHI in analyzing the relations in the level of application and system to deduce the relations between persons using these applications. OBASHI aims at providing a tool for the directors of the enterprise to monitor their business.

None of these works considers trust relations among users. SOCIOPATH aims at improving the users satisfaction and awareness, by making evident the relations of dependence (thus indicating the necessity of trust) among the social and the digital actors of the given services.

7 Ongoing works and conclusion

This paper introduces SOCIOPATH, a meta-model that formalizes systems in order to reveal the relations of dependence and trust among participants. A formalism of SOCIOPATH is given by several definitions, upon which we have defined some rules, based on first order logic. These rules only captures those aspects of the model that are needed to build the relations among the system's components.

Rules and definitions have been implemented in ProLog, in order to develop a tool, based on SOCIOPATH that infers dependences automatically. Such a tool may be very valuable in all the situations that require a person to evaluate the degree of inter-dependence of the various components of a given architecture. For instance, it may help a manager in understanding all the implications entailed by decisions such as: switching from a corporate licensed software to an open source alternative, choosing a network service provider over a competitor one, validating a risk assessment plan for a given logistic architecture, *etc.* Moreover, by applying SOCIOPATH to build a model of her own system, a user may easily evaluate the "cost" of replacing something, in terms of side-effects and dependence shifts. One may also be able to evaluate the system's exposition to risks of misbehavior or failures of components the system itself depends on.

SOCIOPATH can be used to point out accesses, controls and relations within an architecture. This is particularly useful to check whether the system respects the trust and the privacy expected by its users. Being able to test an architecture compliance with respect to users' privacy policies and trust models is one of our future goals.

We are currently investigating the implication posed by different kinds of control the users may have on system components, with the goal of including access control and restrictions, typical of most common scenarios. A further line of research is devoted to investigate the amount of information about the system, that is needed to derive hidden relations by applying SOCIOPATH. The service level agreements of the system's components, rather than inner design and implementation details (that may not be available or disclosed), should be enough to draw

[17] `http://pubs.opengroup.org/architecture/togaf9-doc/arch/`

meaningful conclusions.

We believe that the use of our meta-model is not limited to the few possible ways presented above. SocioPath may be used by a standard user, to better analyze and develop useful insights about the digital world, upon which everyone relies more and more.

References

[1] Cornelli, F., Damiani, E., di Vimercati, S.D.C., Paraboschi, S., Samarati, P.: Choosing Reputable Servents in a P2P Network. In: Int. Conference on World Wide Web (WWW). (2002) 376–386

[2] Damiani, E., di Vimercati, D.C., Paraboschi, S., Samarati, P., Violante, F.: A Reputation-Based Approach for Choosing Reliable Resources in Peer-to-Peer Networks. In: Int. Conference on Computer and Communications Security (CCS). (2002) 207–216

[3] Fahrenholtz, D., Lamersdorf, W.: Transactional Security for a Distributed Reputation Management System. In: Int. Conference on E-Commerce and Web Technologies (EC-WEB). (2002) 214–223

[4] Kamvar, S.D., Schlosser, M.T., Garcia-Molina, H.: The Eigentrust Algorithm for Reputation Management in P2P Networks. In: Int. Conference World Wide Web Conference (WWW). (2003) 640–651

[5] Gupta, M., Judge, P., Ammar, M.: A Reputation System for Peer-to-Peer Networks. In: Int. Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV). (2003) 144–152

[6] Carchiolo, V., Longheu, A., Malgeri, M.: Reliable Peers and Useful Resources: Searching for the Best Personalised Learning Path in a Trust and Recommendation-Aware Environment. Information Sciences **180** (2010) 1893–1907

[7] Wang, L., Hill, R.: Trust Model for Open Resource Control Architecture. In: Int. Conference on Computer and Information Technology (CIT). (2010) 817–823

[8] Yoon, J.P., Chen, Z.: Service Trustiness and Resource Legitimacy in Cloud Computing. In: Int. Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC). (2010) 250–257

[9] Varalakshmi, P., Nandini, M., Krithika, K., Aarthi, R.: An Optimal Trust Based Resource Allocation Mechanism for Cross Domain Grid. In: Int. Conference on Recent Trends in Business Administration and Information Processing (BAIP). (2010) 342–348