SocioPath: Bridging the Gap between Digital and Social Worlds

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

SOCIOPATH: Bridging the Gap Between Digital and Social Worlds

Nagham Alhadad[1] Philippe Lamarre[2] Yann Busnel[1]
Patricia Serrano-Alvarado[1] Marco Biazzini[1] Christophe Sibertin-Blanc[3]

[1] {*Name.LastName*}@univ-nantes.fr
[2] Philippe.Lamarre@liris.cnrs.fr
[3] Christophe.Sibertin-Blanc@univ-tlse1.fr

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

[2] Liris/Université de Lyon
37, Rue du Repos
69007 Lyon, France

[3] IRIT/Université de Toulouse 1
2, rue du Doyen-Gabriel-Marty
31042 Toulouse, France

Abstract. In everyday life, people use more and more digital resources (data, application systems, Internet, *etc.*) for all aspects of their life like administrative procedures, financial management, private exchanges, collaborative work, *etc*. This leads to non-negligible dependences on the digital distributed resources that reveal strong reliance at the social level, for instance on providers, physical or moral persons, of these resources.
Users are often not aware of their real autonomy regarding the management of their digital resources. Thus, currently, people underestimate social dependences generated by the system architecture they use and the resulting potential risks. We argue that it is necessary to be aware of some key aspects of system's architectures to be able to know dependences.
In this paper, we proposes SOCIOPATH, a generic meta-model to derive dependences generated by system's architecture. In particular, SOCIOPATH focuses on relations, like access, control, support, ownership, and so forth, among the different entities of the system (digital resources, hardware, persons, *etc.*). Enriched with deduction rules and definitions, SOCIOPATH allows to reveal the dependences of a person towards each entity in the system. SOCIOPATH could then be useful in the evaluation process of a system, as a modeling tool that bridges the gap between the digital and the social worlds.

1 Introduction

Nowadays, no more high level systems are still single station-based. In fact, most of systems are connected through complex architectures. The tremendous variety of applications using distributed systems [1] make them the norm of computer usage. On such system, most of the activities made by participants concern their data (sharing and editing documents, publishing photos, purchasing online, *etc.*) [2]. These usages inherently imply some relationships with numerous persons. Part of this group of people should be unknown, but one depends on it in several ways:

– Who are the person(s)/resource(s) a user depends on to perform an activity?

- Who are the persons who can prevent a user from performing an activity?
- Who are the persons whom a user is able to avoid when she performs an activity?
- Who are the persons that have a possibility to access a user's data? and what are the necessary coalitions between persons in order to access this data?

Some of these questions raise several issues as someone should be able to grab information about who I am, what I do, and so forth. That directly leads to privacy [3], trust [4] and security issues [5].

Historic analysis of such systems are usually limited to technical aspects as latency, functional performance, failure management [6], *etc.* The aforementioned collection of questions gives some orthogonal but complementary criteria of the classical approach. Currently, people underestimate social dependences [7] generated by the systems they use and the resulting potential risks, when they perform some activities. We argue that to be able to know dependences, it is necessary to be aware of some systems key aspects.

This paper proposes the SOCIOPATH meta-model. This approach is based on notions coming from many fields, ranging from computer science to sociology. SOCIOPATH is a generic meta-model that is divided in two worlds; the social world and the digital world. SOCIOPATH allows us to draw a representation (or model) of a system that identifies its hardware, software and persons as components, and the ways they are related. Enriched with deduction rules and definitions, SOCIOPATH analyzes the relations in the digital world, to deduce the relations of dependences in social worlds according to a specific activity concerning some data. SOCIOPATH could then be useful in the evaluation process of a system with respect to security, privacy and trust requirements, as a modeling tool that bridge the gap between the digital and the social world.

The paper is organized as follows. Section 2 introduces the SOCIOPATH meta-model and gives a simple example of its use. Deducted relations are presented in Sections 3, while Section 4 defines the way to compute the user's digital and social dependences. Section 5 presents a use case, in which SOCIOPATH is applied to a well-known scenario. Due to space constrains, we have not included the related works analysis, that is available in [8]. Finally, Section 6 concludes and points out our ongoing work [8].

2 SOCIOPATH meta-model

The SOCIOPATH meta-model allows to describe the architecture of a system in terms of the components that enable people to access digital resources. So that, the chains of dependences could be identified. It distinguishes on the first hand the *social world*, where human beings or organizations own any kind of physical resources and data, and the *digital world* on the second hand, where instances of data (including application's codes) are stored and processes are running.

In this study, we are interested in formalizing relations in the digital world, in order to derive dependences among persons in the social world. We only consider what the persons are able to do, in principle, rather than what they are permitted to do. Thus, imposed access or control restrictions are not considered in detail in this paper. Figure 1 shows the graphical representation of SOCIOPATH, that we analyze in the next sections.

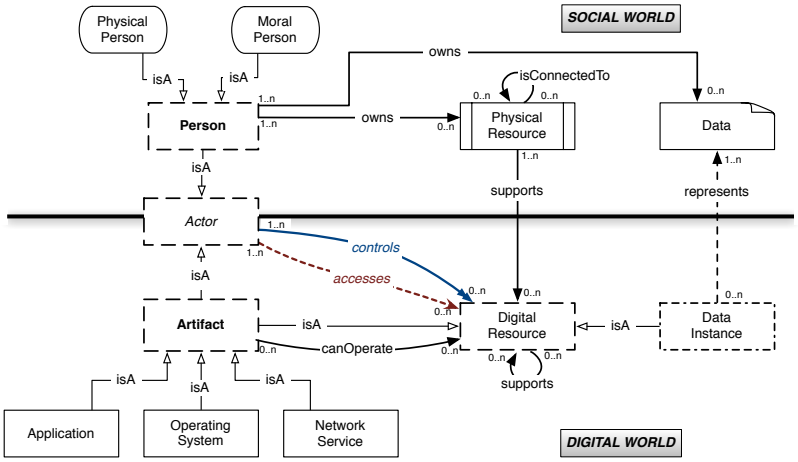

Fig. 1: Graphical view of SOCIOPATH as a UML class diagram.

2.1 The social world

The social world includes persons (users, enterprises, companies, *etc.*), physical resources, data and relations among them.

- *Data* represents an abstract notion that exists in the real life, and does not necessarily imply a physical instance (*e.g.*, address, age, software design, *etc.*);
- *Physical Resource* represents any hardware device (*e.g.*, PC, USB device, *etc.*);
- *Person* represents a generic notion that defines a Physical Persons like Alice or a Moral Persons like Microsoft.

2.2 The digital world

The digital world has nodes that are defined as follows:

- *Data Instance* represents a digital representation of *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*). In that way, the source code of a software is a representation of the software design in the digital world;
- *Artifact* represents an abstract notion that describes a "running software". This can be an *Application*, an *Operating System* or a *Network Service*. It may be a single processus or a group of processes that should be distributed on different locations, yet defining a single logically coherent entity;
- *Digital Resource* represents an *Artifact* or a *Data Instance*;
- *Actor* represents a *Person* in the social world or an *Artifact* in the digital world. This is the core concept of SOCIOPATH. Indeed, only *Actors* can access or control *Digital Resources* as presented below.

2.3 The relations in SOCIOPATH

Several relations are drawn in SOCIOPATH. In this section, we briefly describe them.

- *owns*: this means ownership. This relation only exists in the social world;
- *isConnectedTo*: this means that two nodes are physically connected, through a network for instance. This symmetric relation exists only in the social world;
- *canOperate*: this means that an artifact is able to process, communicate or interact with a target digital resource. This ability may be given as a part of the artifact specification (*e.g.*, "Microsoft Word" *canOperate* a `.doc` document) or deduced by some property of the system (*e.g.*, an operating system only *canOperate* files that are stored in a mounted partition);
- *accesses*: this means that *Actor* can access a *Digital Resource* (*e.g.*, the operating system accesses the applications installed via this OS; 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*: this means that an *Actor* can control a *Digital Resource*. There should exists different kinds of control relations. For instance, a moral person, who provides a resource, controls the functionalities of this resource. The persons who use this resource may have some kind of control on it as well. Each of these actors controls the resource in a different way;
- *supports*: this 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*: this is a relation between data in the social world and their instances in the digital world (*e.g.*, the source code of the operating system Windows is a representation in the digital world of the data known as "Microsoft Windows" in the social world).

Notice that most of the above relations are not symmetrical. The cardinality of the relations is given in Figure 1. Considering the relation *owns* as an example, a *Person* can own some *Physical Resources*, *i.e.* $[0..n]$, and every *Physical Resource* is owned by at least one *Person*, *i.e.* $[1..n]$.

Persons own some data in the social world. *Data* have a concrete existence in the digital world if they are represented by some *Data Instance* and supported by some *Physical Resource*. As an *Actor* in the digital world, a *Person* can access and control *Data Instances* representing her (and others') *Data*. This may be done through different resources, thus implying some dependences on other persons.

Moreover, we consider that a person *provides* an artifact (*cf.* the rightmost part of Figure 2) if this person owns data represented by a data instance which supports the artifact.

Applying SOCIOPATH makes possible 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 in different hosts).

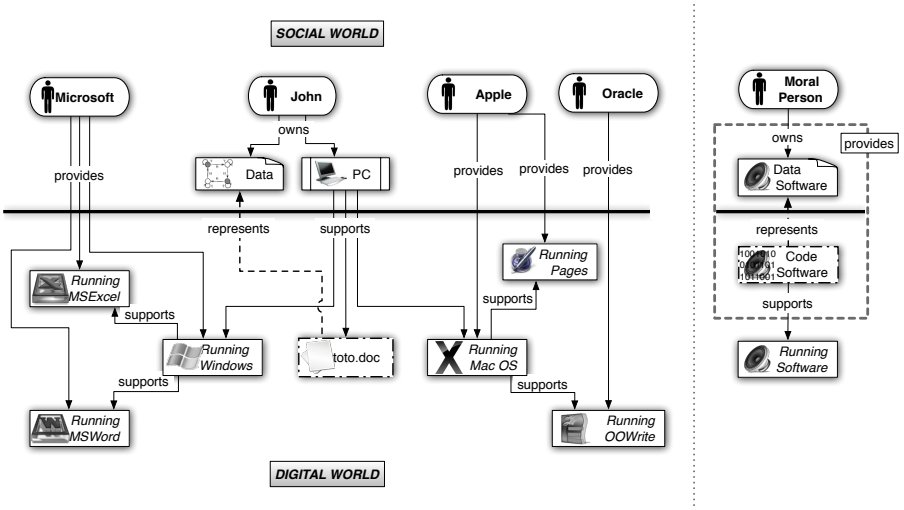

Fig. 2: Use case example: a document accessed by 2 different operating systems.

2.4 Example of SOCIOPATH model: single PC

Figure 2 shows a basic SOCIOPATH model of a use-case on a unique PC[1]. In the social world, a user `John` owns some `Data` and a `PC`. There are also moral persons as `Microsoft` (provider of `Windows`, `MSWord` – *aka* Microsoft Word – and `MSExcel` – *aka* Microsoft Excel), `Apple` (provider of `MacOS` and `Pages`) and `Oracle` (provider of `OOWrite` – *aka* Open Office Writer).

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`.

We use this example to illustrate some deductions in Section 3. We deliberately propose a trivial example, in order to show how SOCIOPATH can be applied and how some deductions and definitions are drawn.

Table 1 summarizes the notations we use.

3 Deduced access and control relations

The semantics of the constitutive elements and relations of a SOCIOPATH model allows to deduce more *controls* and *accesses* relations. We use ProLog, a First Order Logic (FOL) language to describe the rules allowing such deductions. Thus, in the next, the SOCIOPATH elements correspond to constants and the relations are described by binary predicates. For instance, $supports(OS, F)$ represents the *supports* relation between the operating system OS and the artifact F.

[1] We consider that a model conforms to a meta-model.

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	$\mathbb{DR}$	$\{DR : resource(DR)\}$	$\mathcal{DR}$	$\mathcal{DR} \subset \mathbb{DR}$	DR	$DR \in \mathbb{DR}$
Physical resource	$\mathbb{PR}$	$\{PR : phyresource(PR)\}$	$\mathcal{PR}$	$\mathcal{PR} \subset \mathbb{PR}$	PR	$PR \in \mathbb{PR}$
Data	$\mathbb{D}$	$\{D : data(D)\}$	$\mathcal{D}$	$\mathcal{D} \subset \mathbb{D}$	D	$D \in \mathbb{D}$
Data instance	$\mathbb{DI}$	$\{DI : dataInstance(DI)\}$	$\mathcal{DI}$	$\mathcal{DI} \subset \mathbb{DI}$	DI	$DI \in \mathbb{DI}$
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	$\sigma \in \Gamma$
Activity	$\mathbb{W}$	—	—	—	ω	$\omega \in \mathbb{W}$
Set of activity restrictions	$\mathbb{S}$	$\{\mathcal{S} = \mathscr{P}(\mathbb{F}^{\mathbb{N}})\}$	—	—	$\mathcal{S}$	$\mathcal{S} \in \mathbb{S}$

Table 1: Glossary of notations

The proposed deduction rules of SOCIOPATH are not exhaustive and by no means we pretend they capture the whole complexity of systems. For sake for simplicity, in the examples of this section, we will note the running applications (*e.g.,* `Running Windows`) directly with their name (*e.g.,* `Windows`).

- **Rule 1:** If an artifact can operate a digital resource and either the artifact and the digital resource are supported by the same physical resource or they are supported by physical resources connected to each other, then the artifact accesses the digital resource. $\forall F \in \mathbb{F}, \forall DR \in \mathbb{DR}, \forall PR1, PR2 \in \mathbb{PR}:$

$$\bigwedge \begin{cases} canOperate(F, DR) \\ supports(PR1, F) \\ \bigvee \begin{cases} supports(PR1, DR) \\ \bigwedge \begin{cases} supports(PR2, DR) \\ isConnectedTo(PR1, PR2) \end{cases} \end{cases} \end{cases} \Rightarrow accesses(F, DR) \quad (1)$$

  e.g., `Windows` accesses `MSWord`:
  $canOperate(\texttt{Windows}, \texttt{MSWord}) \wedge supports(\texttt{PC}, \texttt{Windows}) \wedge supports(\texttt{PC}, \texttt{MSWord}) \Rightarrow accesses(\texttt{Windows}, \texttt{MSWord})$.
- **Rule 2:** If a person owns a physical resource that supports an operating system, then the person accesses and controls this operating system.

$$\forall P \in \mathbb{P}, \forall PR \in \mathbb{PR}, \forall OS \in \mathbb{O}: \bigwedge \begin{cases} owns(P, PR) \\ supports(PR, OS) \end{cases} \Rightarrow \bigwedge \begin{cases} accesses(P, OS) \\ controls(P, OS) \end{cases} \quad (2)$$

  e.g., John accesses `Windows`:
  $owns(\texttt{John}, \texttt{PC}) \wedge supports(\texttt{PC}, \texttt{Windows}) \Rightarrow accesses(\texttt{John}, \texttt{Windows}) \wedge controls(\texttt{John}, \texttt{Windows})$.
- **Rule 3:** If an operating system supports and can operate an artifact, then it controls this artifact.

$$\forall F \in \mathbb{F}, \forall OS \in \mathbb{O}: \bigwedge \begin{cases} supports(OS, F) \\ canOperate(OS, F) \end{cases} \Rightarrow controls(OS, F) \quad (3)$$

  e.g., `Windows` controls `MSWord`:
  $supports(\texttt{Windows}, \texttt{MSWord}) \wedge canOperate(\texttt{Windows}, \texttt{MSWord}) \Rightarrow controls(\texttt{Windows}, \texttt{MSWord})$.
- **Rule 4:** If a person owns data represented in the digital world by a data instance which supports an artifact, then this person controls this artifact.

$$\exists P \in \mathbb{P}, \exists D \in \mathbb{D}, \exists DI \in \mathbb{DI}, \exists F \in \mathbb{F}: \bigwedge \begin{cases} owns(P, D) \\ represents(DI, D) \\ supports(DI, F) \end{cases} \Rightarrow controls(P, F) \quad (4)$$

e.g., Microsoft owns the software `Windows` which is represented in the digital world on John's PC by the data instance `CodeWindows` which supports the application `Windows`, so Microsoft controls `Windows`:

$owns(\texttt{Microsoft}, \texttt{Software-Windows}) \wedge represents(\texttt{CodeWindows}, \texttt{Software-Windows}) \wedge supports(\texttt{CodeWindows}, \texttt{Windows}) \Rightarrow controls(\texttt{Microsoft}, \texttt{Windows})$.

– **Rule 5:** The relation *accesses* is transitive.

$$\forall A \in \mathbb{A}, \forall F \in \mathbb{F}, \forall DR \in \mathbb{DR} : \bigwedge \begin{cases} accesses(A, F) \\ accesses(F, DR) \end{cases} \Rightarrow accesses(A, DR) \quad (5)$$

e.g., `MSWord` accesses `Windows` and `Windows` accesses `toto.doc`, so `MSWord` accesses `toto.doc`:

$accesses(\texttt{MSWord}, \texttt{Windows}) \wedge accesses(\texttt{Windows}, \texttt{toto.doc}) \Rightarrow accesses(\texttt{MSWord}, \texttt{toto.doc})$.

– **Rule 6:** The relation *controls* is transitive.

$$\forall A \in \mathbb{A}, \forall F_1, F_2 \in \mathbb{F} : \bigwedge \begin{cases} controls(A, F_1) \\ controls(F_1, F_2) \end{cases} \Rightarrow controls(A, F_2) \quad (6)$$

e.g., John controls `Windows` and `Windows` controls `toto.doc` so John controls `toto.doc`:

$controls(\texttt{John}, \texttt{Windows}) \wedge controls(\texttt{Windows}, \texttt{toto.doc}) \Rightarrow controls(\texttt{John}, \texttt{toto.doc})$.

– **Rule 7:** If two physical resources are connected to each other, and the first one supports an operating system and the second one supports another operating system, these two operating systems access to each other.

$$\exists PR1, PR2 \in \mathbb{PR}, \exists OS1, OS2 \in \mathbb{O} : \bigwedge \begin{cases} isConnectedTo(PR1, PR2) \\ supports(PR1, OS1) \\ supports(PR2, OS2) \end{cases} \Rightarrow accesses(OS1, OS2) \quad (7)$$

Starting from the example of Section 2.4, we apply the SOCIOPATH rules, and obtain the *accesses* and *controls* relations shown in Figure 3. Thus, from Rule 2, we deduce that John accesses the operating systems `Mac OS` and `Windows`, and from Rule 4, we deduce that Microsoft controls the operating system `Windows`.

4 Using SOCIOPATH to derive dependences for an activity

Modeling systems with SOCIOPATH allows to underline and discover chains of *accesses* and *controls* relations. This modeling is independent of the usage, and it can be used in different ways. In this work, we are interested in one of these usages: we want to use these relations to better understand the "social and digital dependences" among entities in the model. Thus, informally, the sets of digital dependences of a person are composed of the artifacts by which a user passed through to reach a particular element. The sets of social dependences are composed of the persons who control these artifacts.

Figure 4 shows an example of the drawn paths for a user Alice who wants to read the document `D`; `D` is a representation of Alice's data in the digital world. Alice should reach it via two available paths: {`A,B,C,D`} and {`A,B,E,F,D`}. There is an *accesses* relation between each consecutive elements of these paths. Each element of the path is controlled by one or more person in the social world (for instance, `E` is controlled by Persons `O` and `P`). The

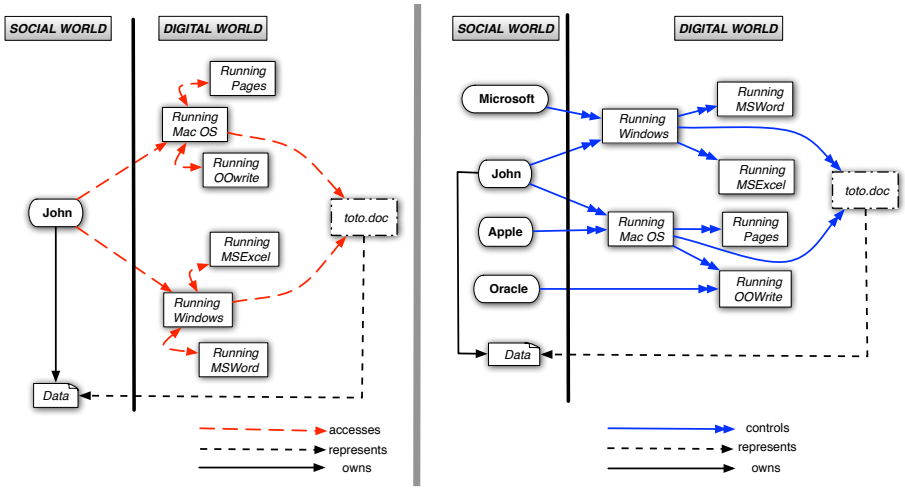

Fig. 3: The relation of access/control in the example: a document accessed by 2 different operating systems.

digital dependences are: $\{\{A\},\{B,E\},\{C,E\},\{B,F\},\{C,F\},\{D\}\}$, as the social ones are $\{\{M\},\{K\},\{Q,R\},\{P,R\},\{O,R\},\{L,Q\},\{L,O\},\{L,P\},\{N\}\}$. Moreover, we show the set of persons/artifacts that can prevent a user from performing her activity. For instance $\{B,F\}$ is a set of artifacts, and $\{Q,R\}$ is a set of persons that can collude to block Alice from reading the document D. The elements that appears separately like A or M are unavoidable elements. That means that if Alice wants to read D, she forcibly depends on these elements. In the following, all those concepts are defined formally. As above, examples in this section refer to Figure 2.

4.1 Activities and Paths

A user follows a path to perform an activity in a system. Here, we consider activities involving data (*e.g.*, copying a file, sharing a document, *etc.*). This means that some restrictions must be given to the ways the person might do their activity. We can do this by imposing the presence of particular elements in the path to do the activity. For instance, if a person wants to read a `.doc` document, she must use an artifact that can "understand" this type of document (*e.g.*, `MSWord` or `OOWrite`). Other example, if a person uses a SVN application, the artifacts "SVN client" and "SVN server" should be used and they should appear in the correct order within the path (usually, the SVN client should precede the SVN server).

Definition 1 (Activity).
We define an activity ω as a triple $(P, D, \mathcal{S})$, where P is a person, D is data and $\mathcal{S}$ is a set of ordered multisets of $\mathbb{F}$ in a model, so an activity ω is an element of $\mathbb{P} \times \mathbb{D} \times \mathbb{S}$. The sets in the $\mathbb{S}$ component of an activity are alternative sets of artifacts that are necessary for the person to perform her activity.

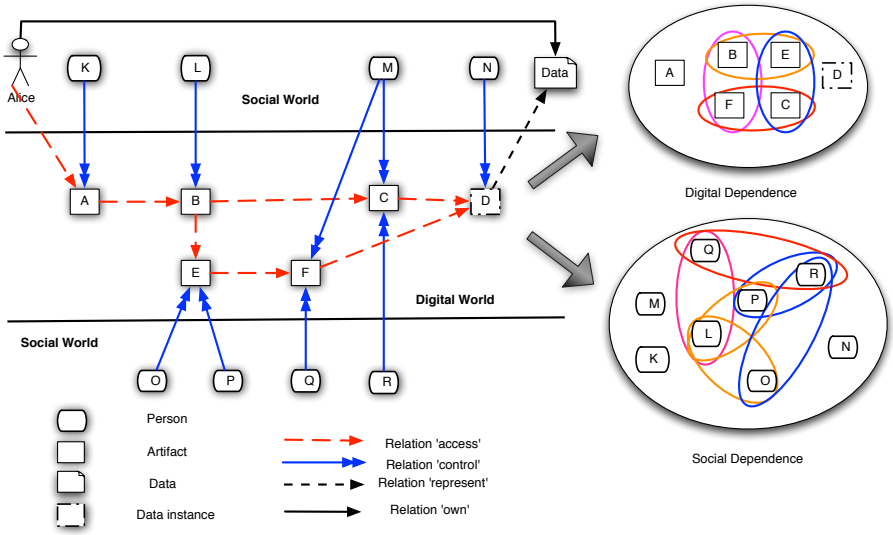

Fig. 4: Deriving dependences.

We call *paths* the lists of actors and digital resources that describe the ways an actor may access a resource. A person may perform an activity in different ways and using different intermediate digital resources. Each possibility can be described by a path. Given that we are interested in understanding the dependences related to an activity, we formally define a specific kind of paths related to this given activity.

Definition 2 (Activity path, or ω-path).
A path σ for an activity $\omega = (P, D, \mathcal{S}) \in \mathbb{P} \times \mathbb{D} \times \mathbb{S}$ is a list of actors and digital resources such that:

- $\sigma[1] = P$;
- $\sigma[|\sigma|] = D$;
- $represents(\sigma[|\sigma| - 1], \sigma[|\sigma|])$;
- $\forall i \in [2 : |\sigma| - 1], artifact(\sigma[i]) \wedge accesses(\sigma[i - 1], \sigma[i])$;
- $\exists s \in \mathcal{S}, s \subseteq \sigma$;

where $\sigma[i]$, denotes the i^{th} element of σ, and $|\sigma|$ the length of σ.

Notation: *Assuming that there is no ambiguity on the model under consideration, the set of ω-paths where $\omega = (P, D, \mathcal{S})$ is noted Υ^{ω} and the set of all the paths in the model is noted Υ.*

For examples, concerning the ω-paths for the activity ω = "John reads toto.doc" in Figure 2, we have:

$$\left\{ \begin{array}{l} \{\texttt{John, Windows, MSWord, Windows, MSExcel, Windows, toto.doc}\} \\ \{\texttt{John, MacOS, OOWrite, MacOS, toto.doc}\} \end{array} \right. .$$

In the first example of the ω-paths presented above: {`John, Windows, MSWord, Windows, MSExcel, Windows, toto.doc`}, The artifact `MSExcel` is an unnecessary element to read `toto.doc`. It appears in the ω-path as it exists the relation *accesses* between it and the artifact `Windows`, we need to eliminate all the unnecessary elements from the ω-path, so we define the minimal path as follows.

Definition 3 (Minimal path).
Let Υ^{ω} be a set of paths for an activity ω.
A path $\sigma \in \Upsilon^{\omega}$ is said to be minimal in Υ^{ω} iff *there exists no path σ' such that:*

- $\sigma[1] = \sigma'[1]$ *and ;* $\sigma[|\sigma|] = \sigma'[|\sigma'|]$*;*
- $\forall i \in [2 : |\sigma'|], \exists j \in [2 : |\sigma|], \sigma'[i] = \sigma[j]$.

Notation*: The set of minimal paths enabling an activity ω= $(P, D, \mathcal{S})$ is noted $\widehat{\Upsilon^{\omega}}$. For sake of simplicity, we name this set the ω-minimal paths.*

For instance, for the activity ω_1 = "John edits `toto.doc`", the set of the ω_1-minimal paths are:

$$\widehat{\Upsilon^{\omega_1}} = \left\{ \begin{array}{c} \{\texttt{John, Windows, MSWord, Windows, toto.doc}\} \\ \{\texttt{John, MacOS, OOWrite, MacOS, toto.doc}\} \\ \{\texttt{John, MacOS, Pages, MacOS, toto.doc}\} \end{array} \right\}.$$

Notation: Let say $F \in \sigma$ iff $\exists i$ such that $\sigma[i] = F$, and $s \subseteq \sigma$ iff $\forall F \in s, F \in \sigma$.

4.2 Dependence

The concepts of ω-path and ω-minimal path allow us to introduce the definitions of digital dependences (Definition 4 and 5) and social dependences (Definition 6 and 7). We say that a person depends on a set of artifacts for an activity ω if each element of this set belongs to one or more paths in the set of the ω-minimal paths.

Definition 4 (Person's dependence on a set of artifacts for an activity).
Let $\omega = (P, D, \mathcal{S})$ be an activity, $\mathcal{F}$ be a set of artifacts and $\widehat{\Upsilon^{\omega}}$ be the set of ω-minimal paths. P depends *on $\mathcal{F}$ for an activity ω* iff $\forall F \in \mathcal{F}, \exists \sigma \in \widehat{\Upsilon^{\omega}} : F \in \sigma$.

For instance, one of the sets on which John depends for the activity "John edits `toto.doc`" is {`MacOS, MSWord`}.

A person does not depend in the same way on all the sets of artifacts. Some sets may be avoidable *i.e.,* the activity can be executed without them. Some sets are unavoidable *i.e.,* the activity cannot be performed without them. To distinguish the way a person depends on artifacts, we define the degree of a person's dependence on a set of artifacts for an activity as the ratio of the ω-minimal paths that contain these artifacts.

Definition 5 (Degree of person dependence on a set of artifacts for an activity).
Let $\omega = (P, D, \mathcal{S})$ be an activity, $\mathcal{F}$ be a set of artifacts and $\widehat{\Upsilon^{\omega}}$ be the set of ω-minimal paths and $|\widehat{\Upsilon^{\omega}}|$ is the number of the minimal ω-paths. The degree of dependence of P on $\mathcal{F}$, denoted $d^{\omega}_{\mathcal{F}}$, is:

$$d^{\omega}_{\mathcal{F}} = \frac{|\{\sigma : \sigma \in \widehat{\Upsilon^{\omega}} \wedge \exists F \in \mathcal{F}, F \in \sigma\}|}{|\widehat{\Upsilon^{\omega}}|}$$

For instance, the degree of dependence of John on the set {MacOS, MSWord} for the activity "John edits toto.doc" is equal to one, where the degree of dependence of John on the set {Pages, OOWrite} is equal to 2/3.

Now, from the digital dependences we can deduce the social dependences as follows. A person depends on a set of persons for an activity if the persons of this set control some of the artifacts the person depends on.

Definition 6 (Person's dependence on a set of persons for an activity). *Let $\omega = (P, D, \mathcal{S})$ be an activity, and $\mathcal{P}$ be a set of persons.*

$$P \text{ depends on } \mathcal{P} \text{ for } \omega \text{ iff } \wedge \begin{cases} \exists \mathcal{F} \subset \mathbb{F} : P \text{ depends on } \mathcal{F} \text{ for } \omega \\ \forall F \in \mathcal{F}, \exists P' \in \mathcal{P} : controls(P', F) \end{cases}$$

For instance, one of the sets John depends on for the activity "John edits toto.doc" is {Oracle, Apple}.

The degree of a person's dependence on a set of persons for an activity is given by the ratio of the ω-minimal paths that contain artifacts controlled by this set of persons.

Definition 7 (Degree of person's dependence on a set of persons for an activity). *Let $\omega = (P, D, \mathcal{S})$ be an activity, $\mathcal{P}$ be a set of persons and $\widehat{\Upsilon^\omega}$ be the ω-minimal paths. The degree of dependence of P on $\mathcal{P}$, noted $d^\omega_\mathcal{P}$ is:*

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

For instance, the degree of dependence for John on the set {Oracle, Apple} for the activity "John edits the 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 digital and social world. We are then able to answer the questions presented in the introduction as we will see in the next section.

5 Use-case example: GoogleDocs

Context To illustrate the meta-model, Figure 5 presents a simple system where a person uses GoogleDocs, drawn by applying SOCIOPATH.

In the social world, the person John owns some Data, a PC and an iPad. We explicitly name only some moral persons who provide resources and artifacts: Microsoft (providing Windows and Internet Explorer so called IExplorer), Google (providing GoogleDocs and Google Cloud services), SkyFireLabs (providing the SkyFire application), Apple (providing the iOS operating system and the browser Safari) and Linux Providers. NeufTelecom, Orange and SFR are telecom companies whose servers and communication infrastructures are used by John's resources.

In the digital world, the operating systems Windows and Linux are running on John's PC. Windows supports Internet Explorer and Linux supports Safari. John's iPad supports the running iOS, which supports two applications, Safari and SkyFire. John's data are represented in the digital world by the document toto.gtxt which is supported by the physical resources owned by Google. We consider Google Cloud as the storage system used by the application GoogleDocs.

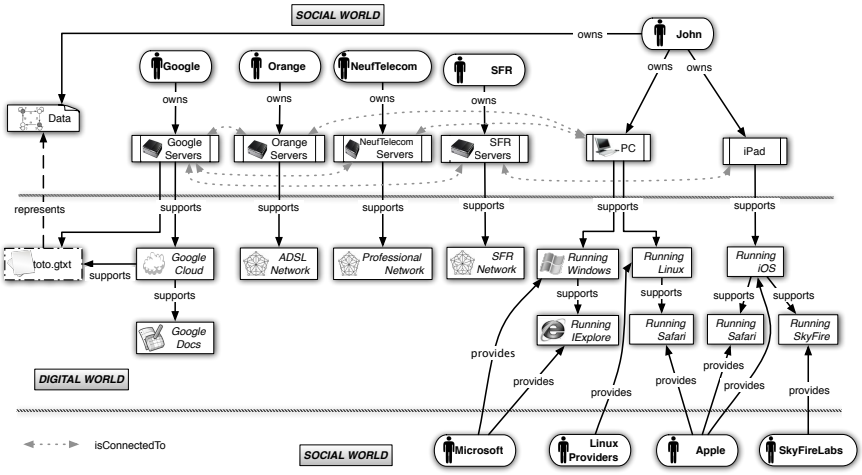

Fig. 5: GoogleDocs snapshot

Analysis and results We analyze this example using SOCIOPATHand answer some questions proposed in the introduction:

- *Who are the persons that have a possibility to access John's data? and what are the necessary coalitions between persons in order to access this data?*
  By applying the deduction rules presented in Section 3, we deduce the relations of "accesses" and "controls" that exist on this architecture. They are illustrated in Figure 6. John is able to know which persons can[2] access his data. Furthermore, by examining the persons who control the artifacts in the paths, it is possible to understand which coalitions may be done to access John's data. For instance, Google can access `toto.gtxt` directly, because it controls all the artifacts in the path from itself to `toto.gtxt` (Figure 7b). Marie (a user who has an access to `ADSL Network`), instead, has a possible path to access John's data that passes through artifacts controlled by Orange and Google (Figure 7c). So Marie must collude with Orange and Google in order to access John's data.
- *Who are the person(s)/artifact(s) John depends on to perform the activity "John reads `toto.gtxt`"?*
  If John wants to read the document `toto.gtxt`, he needs to use a browser and pass through the artifact `GoogleDocs`. So formally, we define the activity "John reads `toto.gtxt`" as ω=(`John`,`Data`,{{`SkyFire`,`GoogleDocs`}, {`Safari`,`GoogleDocs`}, {`IExplorer`,`GoogleDocs`}}). We apply the definitions of Section 4.1, the ω-paths are the following:

$$\forall \sigma \in \Upsilon^{\omega} : \bigwedge \begin{cases} \bigvee \begin{cases} artifact(SkyFire) \in \sigma \\ artifact(IExplorer) \in \sigma \\ artifact(Safari) \in \sigma \end{cases} \\ artifact(GoogleDocs) \in \sigma \end{cases} \tag{8}$$

[2] By *can*, we mean that a user may be able to perform an action, and not that she is actually permitted to. In this work, we do not analyze yet access control and user permission constraints.

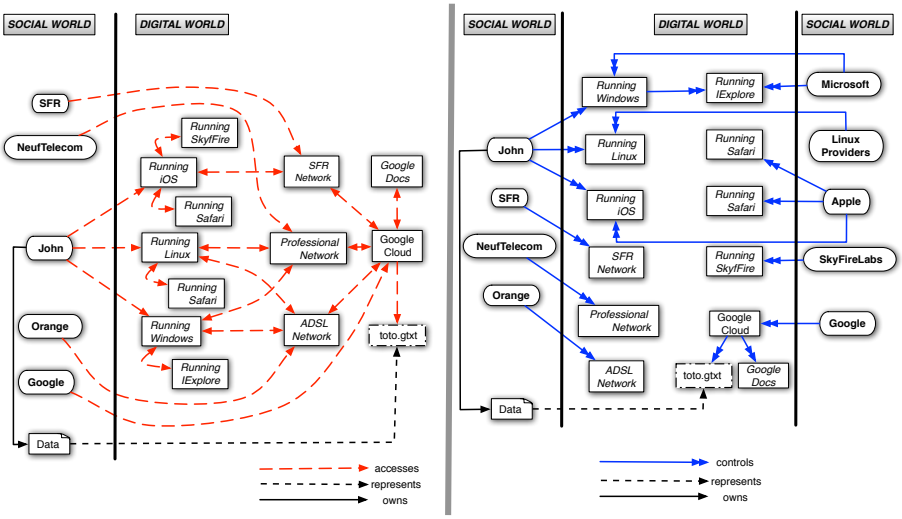

Fig. 6: Relations of access and control in the architecture GoogleDocs

Thus, John has six ω-minimal paths to read `toto.gtxt`:

- {*John, Windows, IExplorer, Windows, ADSL Network, GoogleCloud, GoogleDocs, GoogleCloud, toto.gtxt, Data*};
- {*John,Windows,IExplorer,Windows,Professional Network,GoogleCloud,GoogleDocs,GoogleCloud,toto.gtxt,Data*};
- {*John, Linux, Safari, Linux, ADSL Network, GoogleCloud, GoogleDocs, GoogleCloud, toto.gtxt, Data*};
- {*John, Linux, Safari, Linux, Professional Network, GoogleCloud, GoogleDocs, GoogleCloud, toto.gtxt, Data*};
- {*John, iOS, SkyFire, iOS, SFR Network, GoogleCloud, GoogleDocs, GoogleCloud, toto.gtxt, Data*};
- {*John, iOS, Safari, iOS, SFR Network, GoogleCloud, GoogleDocs, GoogleCloud, toto.gtxt, Data*}.

By applying the definitions of Sections 4.2, we obtain John's dependences on sets of artifacts/persons, and the degree of these dependences for the activity "John reads `toto.gtxt`". Due to space constraints, we show only the results of persons dependences in Table 2 and their degree in Figure 8. This information reveals how much John is autonomous from a specific person or set of persons. For instance, the degree of dependence on {`Microsoft`} is 0.33, and the degree of dependence on the set {`Apple, NeufTelecom`} is 0.83.

– *Who are the persons who can prevent John from reading his data?*
The list of sets on which the degree of dependence is equal to one are the persons who can prevent John from "reading `toto.gtxt`" because they cross all the ω-paths from John to his data. From Figure 8, these sets are: {`Google`}, {`Apple, Microsoft`}, {`NeufTelecom, Orange, SFR`}, {`NeufTelecom, Orange, Apple`}, {`Microsoft, SFR, Linux Providers`}.
– *Who are the persons whom John can avoid to read his data?*
John depends on the sets on which the degree of dependence is less than one, in a less dramatic way (*e.g.*, on the set {`SkyFireLabs, NeufTelecom`} with a degree of 0.5), because this shows that there are others minimal ω-paths enabling John to read `toto.gtxt` and the persons who belong to this set do not control any artifact in these paths. These sets enlighten the "combinations of persons" which John is able to avoid, at will.

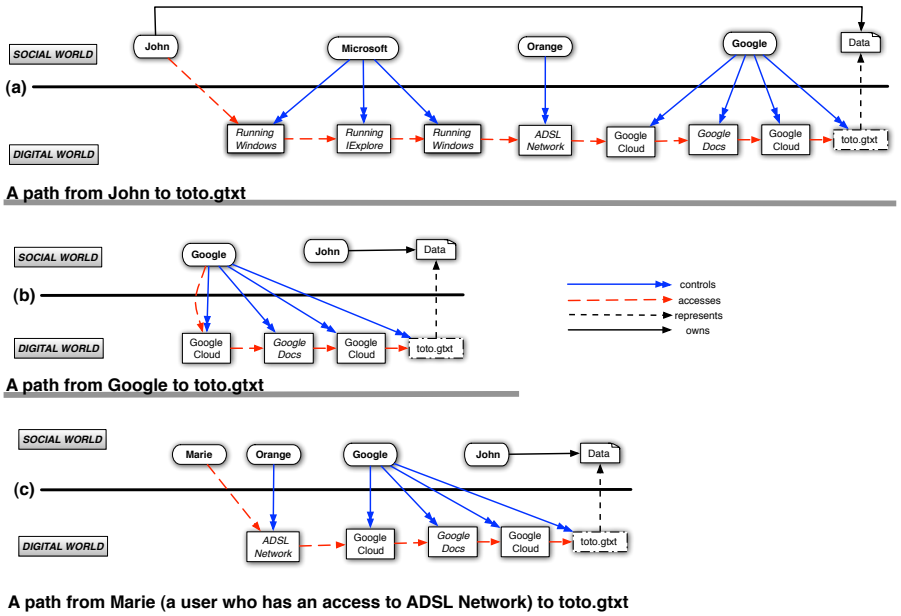

Fig. 7: Examples of paths from different persons to reach John's data

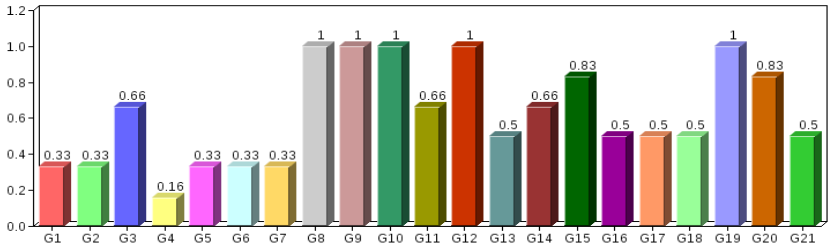

Fig. 8: Degree of dependence on persons' sets

This use case scenario shows how, by applying SOCIOPATH on an architecture, a user can evaluate the system by taking into account the dependence-related aspects. In spite of the simplistic example proposed, some of the outcomes go beyond the immediate understanding that an average user has of a system. Thus exposing system's characteristics that we believe are worth evaluating.

6 Ongoing work and Conclusion

SOCIOPATH, the meta-model proposed in this paper, allows to deduce the degree of dependences of a person on the entities of a system architecture for an activity like editing a document, sharing data, using a software, *etc*. With SOCIOPATH it is possible to know, for instance, whom can prevent a person to perform an activity or whom a

Group	Sets of persons John depends on	Group	Sets of persons John depends on
G1	{Microsoft}	G12	{Apple,Orange,NeufTelecom}
G2	{Linux Providers}	G13	{Microsoft,SkyFireLabs}
G3	{Apple}	G14	{Orange,SFR}
G4	{SkyFireLabs}	G15	{Apple,Orange}
G5	{SFR}	G16	{Microsoft,NeufTelecom}
G6	{NeufTelecom}	G17	{Microsoft,Orange}
G7	{Orange}	G18	{SkyFireLabs,NeufTelecom}
G8	{Google}	G19	{Microsoft,SFR,Linux Providers}
G9	{Microsoft,Apple}	G20	{Apple,NeufTelecom}
G10	{NeufTelecom,Orange,SFR}	G21	{Linux Providers,SkyFireLabs}
G11	{Linux Providers,SFR}		

Table 2: Sets of persons John depends on

person can avoid to perform an activity. This approach can be very valuable to evaluate risks when using a system architecture regarding autonomy, privacy or trust.

Our ongoing work concerns several aspects. First, in the current state of SOCIOPATH, we do not distinguish the different kinds of access and control of an actor to a digital resource, and accordingly we do not consider the intentions or expectations of a user regarding a digital resource. The SOCIOPATH meta-model can be completed with a typology of access and control, in order to define more precisely different types of dependences. Second, we do not distinguish between what persons can do and what they are allowed to do according to the law, the moral rules of persons or their public commitments. This leads to distinguish between dependences related to the system's architecture, and dependences related to social commitments. Third, the resulting dependences of using SOCIOPATH can be enriched with the notion of trust of a person toward the persons she depends on. Indeed, if a user does not trust some person, she will have to worry about the architecture-induced dependences on this person, whereas if she trusts a person, she will only be concerned about the commitments-related dependences. These problematics remains open and challenging for future works.

References

1. Coulouris, G., Dollimore, J., Kindberg: Distributed Systems: concepts and design. 2. print edn. Addison Wesley Publishing Company, Harlow, England (1994)
2. Weaver, A.C., Morrison, B.B.: Social networking. Computer **41** (2008) 97–100
3. Westin, A.: Privacy and freedom. 6. print edn. Atheneum, New York (1970)
4. Marti, S., Garcia-Molina, H.: Taxonomy of trust: categorizing P2P reputation systems. Computer Network Journal **50** (2006) 472–484
5. Lampson, B., Abadi, M., Burrows, M., Wobber, E.: Authentication in distributed systems: theory and practice. ACM Transactions on Computer Systems **10** (1992) 265–310
6. Aurrecoechea, C., Campbell, A.T., Hauw, L.: A survey of qos architectures. Multimedia Systems **6** (1996) 138–151
7. Emerson, R.M.: Power-dependence relations. Amer. Sociological Review **27** (1962) 31–41
8. Alhadad, N., Lamarre, P., Busnel, Y., Serrano-Alvarado, P., Biazzini, M., Sibertin-Blanc, C.: SOCIOPATH: In Whom You Trust? Technical report, LINA – CNRS : UMR6241 (2011)