System Modeling and Trust Evaluation of Distributed Systems

Nagham Alhadad, Patricia Serrano-Alvarado, Yann Busnel, Philippe Lamarre

System Modeling and Trust Evaluation of Distributed Systems

Nagham Alhadad[1], Patricia Serrano-Alvarado[1], Yann Busnel[1,2], Philippe Lamarre[3]

[1] LINA/Université de Nantes – France
[2] Crest (Ensai) Rennes – France
[3] LIRIS/INSA Lyon – France

Abstract. Nowadays, digital systems are connected through complex architectures. These systems involve persons, physical and digital resources such that we can consider that a system consists of elements from two worlds, the social world and the digital world, and their relations. Users perform activities like chatting, buying, sharing data, *etc.* Evaluating and choosing appropriate systems involve aspects like functionality, performance, QoS, ease of use, or price. Recently, trust appeared as another key factor for such an evaluation. In this context, we raise two issues, (i) how to formalize the entities that compose a system and their relations for a particular activity? and (ii) how to evaluate trust in a system for this activity? This work proposes answers to both questions. On the one hand, we propose SOCIOPATH, a metamodel based on first order logic, that allows to model a system considering entities of the social and digital worlds and their relations. On the other hand, we propose two approaches to evaluate trust in systems, namely, SOCIOTRUST and SUBJECTIVETRUST. The former is based on probability theory to evaluate users' trust in systems for a given activity. The latter is based on subjective logic to take into account uncertainty in trust values.

1 Introduction

In our daily life, we do social activities like chatting, buying, sending letters, working, visiting friends, *etc.* These activities are achieved in our society through physical, digital and human entities. For instance, if we want to *write and send a letter*, we might type it and print it, we might use a web application to indicate us the nearest mailbox, then we might consult another web application to provide us the schedule of the public transport that will allow us to reach the mailbox.

Each entity in this example plays a role enabling us to achieve this activity. Our PC and printer should enable us to write the letter and print it. The public transport must allow us to reach the mailbox. The web applications should let us retrieve the necessary information about our travel. The persons who work in the postal service should send the letter in a reliable way, *etc.*

Besides the explicit entities we identify, there are implicit ones playing an important role too. For instance, the installed applications on our PC should work properly. Our Internet connection should allow us to access the web applications. The information that we retrieve from the web applications should be reliable. The providers of the physical and digital resources in the postal service should provide reliable resources.

This simple example illustrates that *nowadays activities are achieved thanks to complex systems we* **rely on** *and we* **trust***, maybe unconsciously*.

Many activities are now purely digital (buying online, sharing data, blogging, chatting, and so on). They are supported by systems involving physical and digital resources, *e.g.,* servers, software components, networks, and PCs. However, the fact remains that these resources are provided and controlled by persons (individuals or legal entities) we depend on to execute these activities. The set of these entities and the different relations between them form a complex system for a specific activity. From this point of view, a digital system can be considered as a small society we rely on and we trust to perform our digital activities.

To perform a digital activity, users may face a lot of available options. Many criteria may guide them in their choice: functionality, ease of use, QoS, economical aspects, *etc*. Nowadays, trust is also a momentous aspect of choice.

Starting from these statements, two main issues arise:

1. How to formalize the entities of a system and the relationships between them for a particular activity?
2. How to evaluate trust in a system as a whole for an activity, knowing that a system composes several entities, which can be persons, digital and physical resources?

These points embody the main focus of this study. We argue that studying trust in the separate entities that compose a system does not give a picture of how trustworthy a system is as a whole. The trust in a system depends on its architecture, more precisely, on the way the implicit and explicit entities the users depend on to do their activities, are organized. Thus, the challenge in evaluating trust in a system is firstly, *to model the system architecture for a specific activity*. Secondly, *to define the appropriate metrics to evaluate the user's trust in a modeled system for an activity.*

This paper is organized as follows.[4] In Section 2, we propose an answer for the first question with SOCIOPATH, a metamodel that allows to model systems for a digital activity. This metamodel formalizes the entities in a system for an activity and the relations between them. To answer the second question, in Section 3, we propose SOCIOTRUST, an approach to evaluate trust in a system for an activity that uses probability theory. And in Section 4, we propose a second approach, SUBJECTIVETRUST, where we use subjective logic to take into account uncertainty to evaluate trust. Section 5 presents related works. Finally, we conclude in Section 6.

2 SOCIOPATH: Modeling a system

Nowadays, the most widespread architectures belong to the domain of distributed systems. Most of participants' activities on these systems concern their data (sharing and editing documents, publishing photos, purchasing online, *etc.*). As mentioned above, using these systems implies some implicit and explicit relationships, which may be partly unknown. Indeed, users achieve several activities without being aware of the used architecture. In our approach, we believe that users need to have a general representation of the used system including the social and digital entities. Based on this

[4] Shorter versions of some contributions of this paper have been published in [5,6,4,3].

representation, a lot of implicit relations can be deduced like the relations of the social dependence [12,29,10]. With SOCIOPATH [5], we aim to answer the following user's questions about her system:

Q1 Who are the persons that have a possibility to access a user's data? And what are the potential coalitions among persons that could allow undesired access to this data?
Q2 Who are the person(s)/resource(s) a user depends on to perform an activity?
Q3 Who are the persons that can prevent a user from performing an activity?
Q4 Who are the persons that a user is able to avoid to perform an activity?

These questions raise a core last one, *how much a user trusts a system for a specific activity?*

The analysis of systems is usually limited to technical aspects as latency, QoS, functional performance, failure management, *etc.* [9]. The aforementioned questions give some orthogonal but complementary criteria to the classical approach. Currently, people underestimate dependences generated by the systems they use and the resulting potential risks.

Thus, in this section, we propose SOCIOPATH that is based on notions coming from many fields, ranging from computer science to sociology. SOCIOPATH is a generic metamodel that is divided in two worlds: the social world and the digital world. SOCIOPATH allows to draw a representation (or model) of a system that identifies its hardware, software and persons as components, and the ways they are related (*cf.* Section 2.1). Enriched with deduction rules (*cf.* Section 2.2), SOCIOPATH analyzes the relations between the components and deduces some implicit relations. In SOCIOPATH, we propose some definitions that reveal main aspects about the used architecture for a user (*cf.* Section 2.3). An illustrating example shows how SOCIOPATH answers our motivating questions (*cf.* Section 2.4).

2.1 SOCIOPATH metamodel

The SOCIOPATH metamodel allows to describe the architecture of a system in terms of the components that enable people to access digital resources. It distinguishes two worlds; the *social world* and the *digital world*. In the social world, persons or organizations own any kind of physical resources and data. In the digital world, instances of data (including source codes) are stored and processes are running. Figure 1 shows the graphical representation of SOCIOPATH, that we analyze in the next.

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

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

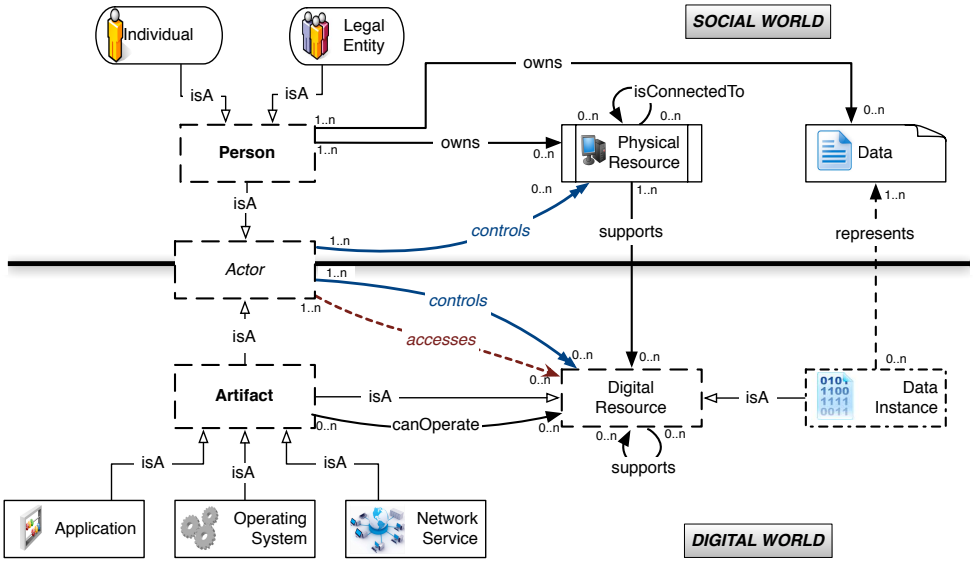


Fig. 1: Graphical view of SOCIOPATH as a UML class diagram.

The digital world has entities that are defined as follows:

- *Data Instance* is a digital representation of a *Data* that exists in the social world. For instance, a person has an address (*Data*) in the social world. *Data Instances* of her address can be present in different digital documents: letters ((*e.g.*, encoded using `.doc` format), contact applications, commercial databases, *etc*. Even if encoded using different formats, each data instance is a semantically equivalent instance of her address. Similarly, a source code is also a *Data Instance* implementing a software (text editor, mailer...) 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 process 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.

The relations proposed in SOCIOPATH are briefly described next. They have to allow to represent in a non naive way how a system is built. They should also help to highlight the links between the structure of a system and the confidence of a user within this system. We do not claim the proposed list to be exhaustive, and one can think about many other relations to describe a system. Providing a complete and minimal set of relations is an interesting question that is out of the scope of this article.

- *owns* is a relation of ownership between a *Person* and a *Physical Resource* (*owns(*P, PR*)*), or between a *Person* and some *Data* (*owns(*P,D*)*). This relation only exists in the social world.
- *isConnectedTo* is a relation of connection between two *Physical Resources* (*isConnectedTo(*PR_1, PR_2*)*). It means that two entities are physically connected, through a network for instance. This symmetric relation exists only in the social world.
- *canOperate* represents an *Artifact* that is able to process, communicate or interact correctly with a target *Digital Resource* (*canOperate(*F,DR*)*). This ability may be explicitly given, for instance, "Microsoft Word" *canOperate* the file letter.doc, or deduced from some general properties, for instance, "Microsoft Word" *canOperate* files of the form *.doc (as far as it can access them - see next relation).
- *accesses* represents an *Actor* that can access a *Digital Resource* (*accesses(*A, DR*)*). For instance, the operating system accesses the applications installed via this operating system; a person who owns a PC that supports an operating system accesses this operating system. The access relations we consider are: read, write, and execute.
- *controls* represents an *Actor* that can control a *Digital Resource* (*controls(*A, DR*)*). There should exist different kinds of control relations. For instance, a legal entity, 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* is a relation between two *Digital Resources* (*supports(*DR_1, DR_2*)*), or a *Physical Resource* and a *Digital Resource* (*supports(*PR, DR*)*). It means that the target entity could never exist without the source entity. We may say that the latter allows the former to exist. For instance, an operating system is supported by a given hardware, an application is supported by an operating system, or the code of an application supports this application.
- *represents* is a relation between *Data* in the social world and their *Instances* in the digital world (*represents(*D,DI*)*). For instance, 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.

For sake of simplicity, we consider that a person *provides* an artifact, if this person owns the data represented by the data instance which supports the artifact.

Applying SOCIOPATH makes possible non-trivial deductions about relations among entities. 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).

Use case 1 of a SOCIOPATH model: isolated PC. Figure 2 shows a simple SOCIOPATH model.[5] In the social world, a user John owns some Data and one PC. There are also legal entities as: Microsoft, provider of Windows, Microsoft Word (MSWord) and Microsoft Excel (MSExcel); Apple, provider of MacOS and Pages; and Oracle,

[5] In general, we consider that a model conforms to a metamodel.

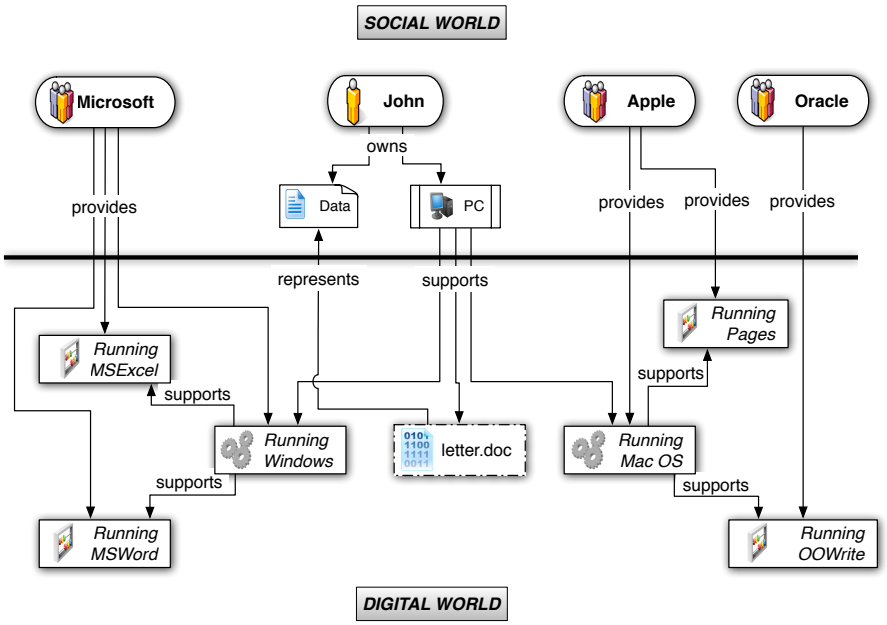


Fig. 2: Use case 1: isolated PC.

provider of Open Office Writer (OOWrite). 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 letter.doc.

We use this example to illustrate some deductions in Section 2.2. We deliberately propose a trivial example, in order to show clearly how SOCIOPATH can be applied and how some deductions and definitions are drawn. Table 1 summarizes the notations we use in the following.

2.2 Deduced access and control relations

The semantics of the components and the relations of a SOCIOPATH model allows to deduce more *control* and *access* relations. We use, a first order logic to describe the rules allowing such deductions.

The proposed deduction rules of SOCIOPATH are not exhaustive and by no means we pretend they capture the whole complexity of systems. They capture several aspects of a simplified vision of the systems that serves the purpose of building an understandable and expressive model. Table 2 shows these rules.

– Rule 1 states that 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 connected physical resources, then the artifact *accesses* the digital resource.

Concept	Notation	Set	Remark
Actor	A	$\mathbb{A}$	$A \in \mathbb{A}$
Artifact	F	$\mathbb{F}$	$F \in \mathbb{F}$
Digital resource	DR	$\mathbb{DR}$	$DR \in \mathbb{DR}$
Physical resource	PR	$\mathbb{PR}$	$PR \in \mathbb{PR}$
Data	D	$\mathbb{D}$	$D \in \mathbb{D}$
Data instance	DI	$\mathbb{DI}$	$DI \in \mathbb{DI}$
Operating system	OS	$\mathbb{OS}$	$OS \in \mathbb{OS}$
Path	σ	Υ	$\sigma \in \Upsilon$
Architecture or system	α	Λ	$\alpha \in \Lambda$
Activity	ω	$\mathbb{W}$	$\omega \in \mathbb{W}$
Activity path	σ^{ω}	Υ^{ω}	$\sigma^{\omega} \in \Upsilon^{\omega}$
Activity minimal path	$\widehat{\sigma^{\omega}}$	$\widehat{\Upsilon^{\omega}}$	$\widehat{\sigma^{\omega}} \in \widehat{\Upsilon^{\omega}}$
Set of activity restrictions	$\mathcal{S}$	$\mathbb{S}$	$\mathcal{S} \in \mathbb{S}$
Person or user	P	$\mathbb{P}$	$P \in \mathbb{P}$

Table 1: Glossary of notations (1).

- Rule 2 states that if a person *owns* a physical resource that *supports* an operating system, then the person *accesses* and *controls* this operating system.
- Rule 3 states that if an operating system *supports* and *can operate* an artifact, then it *controls* this artifact.
- Rule 4 states that if a person *owns* data represented in the digital world by a data instance which *supports* an artifact, then this person *controls* this artifact.
- Rule 5 states the transitivity of relation *accesses*.
- Rule 6 states the transitivity of relation *controls*.
- Rule 7 states that 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.

Starting from the use case 1, we apply the SOCIOPATH rules, and obtain the *accesses* and *controls* relations of Figure 3. For example, from Rule 2, we deduce that John accesses and controls the operating systems MacOS and Windows, and from Rule 4, we deduce that Microsoft controls the operating system Windows and Apple controls the operating system MacOS.

2.3 SOCIOPATH definitions

We next enrich SOCIOPATH with formal definitions to answer the motivating questions (Q1 to Q4) presented in the beginning of this section. Definitions concern activities, paths, and dependences. All of them can be automatically deduced from a SOCIOPATH model.

Definitions for activities and paths. A SOCIOPATH model expresses chains of access and control relations, *i.e.,* paths. A user follows a path to perform an activity in a system.

Rule	Formal definition
Rule 1	$\forall F \in \mathbb{F}, \forall DR \in \mathbb{DR}, \forall PR1, PR2 \in \mathbb{PR}: \bigwedge \begin{cases} canOperate(F, DR) \\ supports(PR1, F) \\ \bigvee \begin{cases} supports(PR1, DR) \\ \bigwedge \begin{cases} supports(PR2, DR) \\ isConnectedTo(PR1, PR2) \end{cases} \end{cases} \end{cases} \Rightarrow accesses(F, DR)$
Rule 2	$\forall P \in \mathbb{P}, \forall PR \in \mathbb{PR}, \forall OS \in \mathbb{OS}: \bigwedge \begin{cases} owns(P, PR) \\ supports(PR, OS) \end{cases} \Rightarrow \bigwedge \begin{cases} accesses(P, OS) \\ controls(P, OS) \end{cases}$
Rule 3	$\forall F \in \mathbb{F}, \forall OS \in \mathbb{OS}: \bigwedge \begin{cases} supports(OS, F) \\ canOperate(OS, F) \end{cases} \Rightarrow controls(OS, F)$
Rule 4	$\exists P \in \mathbb{P}, \exists D \in \mathbb{D}, \exists DI \in \mathbb{DI}, \exists F \in \mathbb{F}: \bigwedge \begin{cases} owns(P, D) \\ represents(DI, D) \\ supports(DI, F) \end{cases} \Rightarrow controls(P, F)$
Rule 5	$\forall A \in \mathbb{A}, \forall F \in \mathbb{F}, \forall DR \in \mathbb{DR}: \bigwedge \begin{cases} accesses(A, F) \\ accesses(F, DR) \end{cases} \Rightarrow accesses(A, DR)$
Rule 6	$\forall A \in \mathbb{A}, \forall F_1, F_2 \in \mathbb{F}: \bigwedge \begin{cases} controls(A, F_1) \\ controls(F_1, F_2) \end{cases} \Rightarrow controls(A, F_2)$
Rule 7	$\exists PR1, PR2 \in \mathbb{PR}, \exists OS1, OS2 \in \mathbb{OS}: \bigwedge \begin{cases} isConnectedTo(PR1, PR2) \\ supports(PR1, OS1) \\ supports(PR2, OS2) \end{cases} \Rightarrow accesses(OS1, OS2)$

Table 2: Deduced access and control relations.

In our analysis, we consider systems enabling users to perform a data-based activity. To do so, restrictions must be defined to impose the presence of particular elements in paths. 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). Another 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 ω).
We define an activity ω as a triple $(P, D, \mathcal{S})$, where P is a person, D is a datum and $\mathcal{S}$ is a set of ordered sets $\mathbb{F}$ in a model. So an activity ω is a subset of $\mathbb{P} \times \mathbb{D} \times \mathbb{S}$. The sets in the $\mathbb{S}$ component of an activity are alternative sets of artifacts to perform the activity, i.e., *each set allows the person to perform his activity. Thus, $\omega = (P, D, \mathcal{S}) \in \mathbb{P} \times \mathbb{D} \times \mathbb{S}$. For instance, the activity "John edits letter.doc", in use case 1, is defined as ω=(*John, Data, {{MSWord}, {Pages}, {OOWrite}}).

We call *paths* the lists of actors and digital resources describing the ways an actor may access a digital resource. A person may perform an activity in different ways and using different intermediate digital resources. Each possibility is described by a path.

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:

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

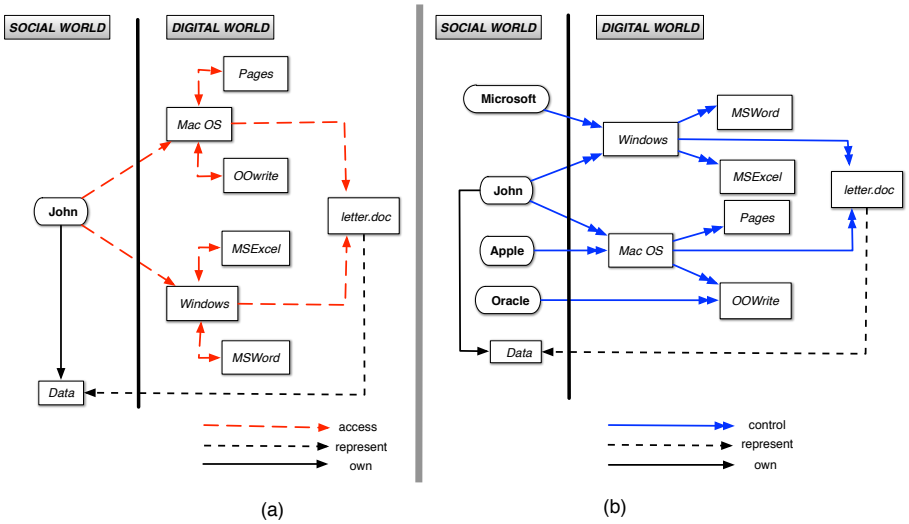


Fig. 3: The relations of *access* and *control* for use case 1 (isolated PC).

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 ω= $(P, D, \mathcal{S})$ is denoted Υ^{ω} and the set of all the paths in the model is denoted Υ.*

For example, if John wants to achieve the activity ω = "John edits letter.doc" using the architecture of use case 1, John uses Windows to work on the application MSWord, which uses Windows file system to access letter.doc so one of the ω-paths for this activity is:

{John, Windows, MSWord, Windows, MSExcel, Windows, letter.doc, Data}.

This path contains some unnecessary artifacts. For instance, MSExcel is an unnecessary artifact to edit letter.doc. It appears in the ω-path because there exists a relation *accesses* between it and the artifact Windows. We want to eliminate all the unnecessary elements from the ω-paths, so we define the activity minimal paths as follows.

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

- $\sigma^{\omega}[1] = \sigma'[1]$ *and ;* $\sigma^{\omega}[|\sigma^{\omega}|] = \sigma'[|\sigma'|]$;
- $\forall i \in [2 : |\sigma'|], \exists j \in [2 : |\sigma^{\omega}|], \sigma'[i] = \sigma^{\omega}[j]$;
- $\forall i \in [2 : |\sigma'| - 1], accesses(\sigma'[i-1], \sigma'[i])$.

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

For instance, for the activity $\omega =$ "John edits letter.doc", the path {John, Windows, MSWord, Windows, MSExcel, Windows, letter.doc, Data} has been eliminated because there is a path $\sigma' =$ {John, Windows, MSWord, Windows, letter.doc, Data} that satisfies the previous conditions. Thus the set of the ω-minimal paths for this activity are:

$$\alpha = \widehat{\Upsilon^{\omega}} = \left\{ \begin{array}{c} \{\text{John, Windows, MSWord, Windows, letter.doc, Data}\} \\ \{\text{John, MacOS, OOWrite, Windows, letter.doc, Data}\} \\ \{\text{John, MacOS, Pages, Windows, letter.doc, Data}\} \end{array} \right\}.$$

Definitions for dependences. Modeling systems with SOCIOPATH allows to underline and discover chains of *accesses* and *controls* relations. In the following, we introduce the definitions of digital dependences (Definitions 4 and 5) and social dependences (Definitions 6 to 9). Informally, the sets of digital dependences of a person are composed of the artifacts a user passes by to reach a particular element. The sets of social dependences are composed of the persons who control these artifacts and the physical resources that support them. We call digital dependences the sets of artifacts a user depends on, because artifacts belong to the digital world in SOCIOPATH. Similarly, we call social dependences the sets of persons and physical resources a user depends on, because they belong to the social world in SOCIOPATH. In the following, these concepts are defined formally and examples refer to use case 1.

Digital dependences. 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.

$$P \text{ depends } \textit{on } \mathcal{F} \textit{ for } \omega \text{ iff } \exists \mathcal{F} \subset \mathbb{F}, \forall F \in \mathcal{F}, \exists \sigma \in \widehat{\Upsilon^{\omega}} : F \in \sigma.$$

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

A person does not depend on all the sets of artifacts in the same way. Some sets may be avoidable because the activity can be executed without them. Some sets are unavoidable because 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 to all the ω-minimal paths.

Definition 5 (Degree of a person dependence on a set of artifacts).
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 ω-minimal 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 letter.doc" is equal to one, while the degree of dependence of John on the set {Pages, OOWrite} is equal to 2/3.

Social dependences. 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 in 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 $\mathcal{P}$ a set of persons.

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

For instance, one of the sets of persons John depends on for the activity "John edits letter.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 a person's dependence on a set of persons).
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 $\mathcal{P}$, denoted $d^{\omega}_{\mathcal{P}}$, 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 of John on the set {Oracle, Apple} for the activity "John edits letter.doc" is equal to 2/3. We recall that Oracle *controls* OOWrite and Apple *controls* MacOS.

We say a person depends on a set of physical resources for an activity if the elements of this set support the artifacts the person depends on.

Definition 8 (Person's dependence on a set of physical resources).
Let $\omega = (P, D, \mathcal{S})$ be an activity, and $\mathcal{PR}$ be a set of physical resources.

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

For instance, John depends on the set {PC} for the activity "John edits letter.doc".

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

Definition 9 (Degree of a person's dependence on a set of physical resources).
Let $\omega = (P, D, \mathcal{S})$ be an activity, let $\mathcal{PR}$ be a set of physical resources, let $\widehat{\Upsilon^{\omega}}$ be the ω-minimal paths. The degree of dependence of P on $\mathcal{PR}$, denoted $d^{\omega}_{\mathcal{PR}}$ is:

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

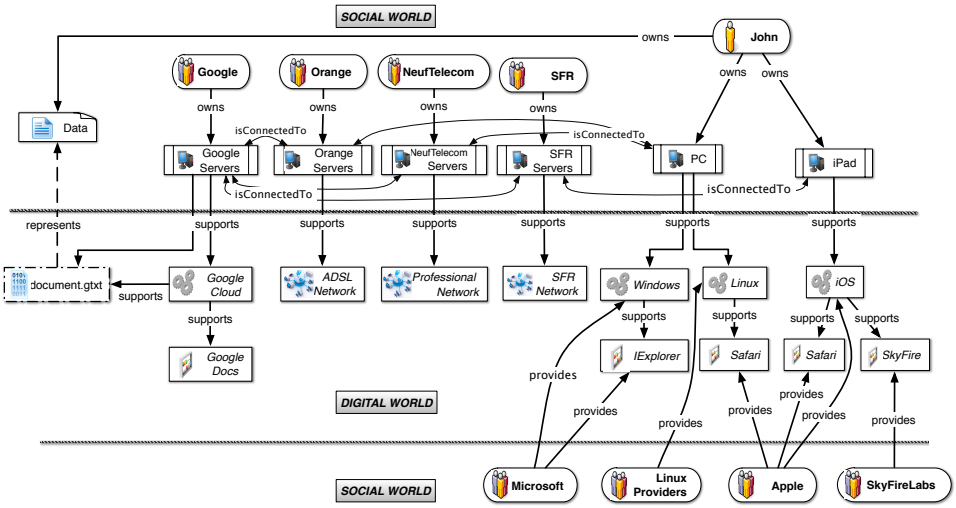


Fig. 4: Use case 2: GoogleDocs.

For instance, the degree of dependence of John on the set {PC} for the activity "John edits letter.doc" is equal to 1.

These definitions allow awareness of the user's dependences on the digital and social world. Another use case is presented in the next section to illustrate them.

2.4 Use case 2 of a SocioPath model: GoogleDocs

Figure 4 presents a SocioPath model corresponding to our use case 2 where John uses GoogleDocs for the activity "John reads document.gtxt". In the social world, John owns some Data, a PC and an iPad. We explicitly name only some legal entities who provide resources and artifacts: Microsoft for Windows and Internet Explorer (so called IExplorer), Google for GoogleDocs and Google Cloud services, SkyFireLabs for SkyFire, Apple, for the iOS operating system and the browser Safari and Linux Providers for Linux.NeufTelecom, Orange and SFR are telecom companies. John's iPad is connected to SFR Servers and John's PC is connected to NeufTelecom Servers and Orange Servers. In the digital world, the operating systems Windows and Linux are running on John's PC. Windows supports IExplorer 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 document.gtxt, which is supported by the physical resources owned by Google. We consider Google Cloud as the storage system used by Google Docs.

Analysis and results. Through this example we show that SocioPath provides answers to the motivating questions of this section.

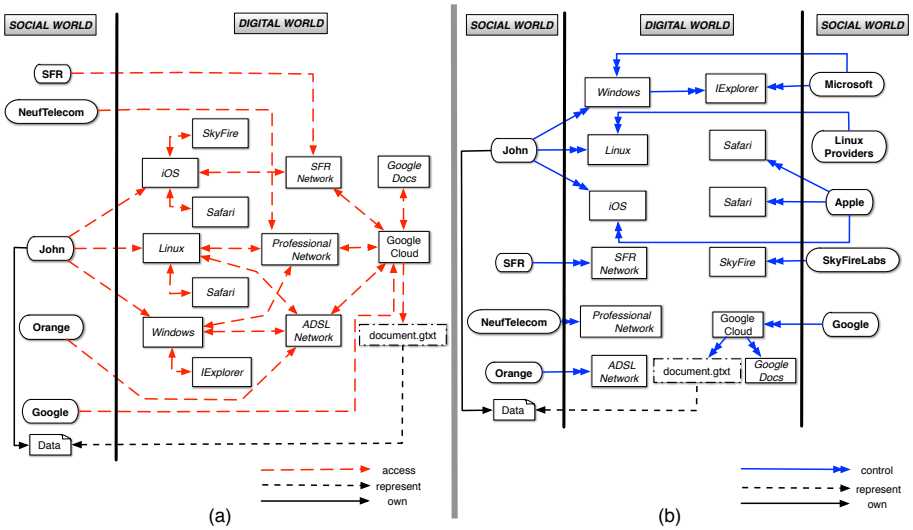


Fig. 5: Relations of access and control in the use case 2 (GoogleDocs).

Q1 *Who are the persons that have a possibility to access John's data? And what are the potential coalitions among persons that could allow undesired access to this data?*

By applying the deduction rules presented in Section 2.2, we deduce the relations of *access* and *control* that exist in this architecture. They are illustrated in Figure 5. By knowing the relations *accesses* in this model, *cf.* Figure 5 (a), John is able to know which persons have a possible path to his document. Thus, these persons can[6] access his data. In this example, they are: SFR, NeufTelecom, John, Orange, and Google.

Furthermore, by examining the persons who control the artifacts in the paths, *cf.* Figure 5 (b), it is possible to understand which coalitions may be done to access John's data. For example, Google can access document.gtxt directly because it controls all the artifacts of the path that enables it to reach it. Orange, instead, has a possible path to access John's data that passes through artifacts controlled by Google. So it must collude with Google to access John's data.

Q2 *Who are the person(s)/resource(s) John depends on to perform the activity "John reads document.gtxt"?*

If John wants to read document.gtxt, he needs a browser and GoogleDocs. So formally, we define this activity as ω=(John, Data, {{SkyFire, GoogleDocs}, {Safari, GoogleDocs}, {IExplorer, GoogleDocs}}). If we apply Definition 3, we find that John has six ω-minimal paths to read document.gtxt:

1. {John, Windows, IExplorer, Windows, ADSL Network, GoogleCloud, GoogleDocs, document.gtxt, Data};

[6] By *can*, we mean that a user may be able to perform an action, and not that she has the permissions to do it. In this work, we do not analyze access control and user permission constraints.

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 3: Sets of persons John depends on (use case 2 - GoogleDocs).

2. {John, Windows, IExplorer, Windows, Professional Network, GoogleCloud, GoogleDocs, document.gtxt, Data};
3. {John, Linux, Safari, Linux, ADSL Network, GoogleCloud, GoogleDocs, document.gtxt, Data};
4. {John, Linux, Safari, Linux, Professional Network, GoogleCloud, GoogleDocs, document.gtxt, Data};
5. {John, iOS, SkyFire, iOS, SFR Network, GoogleCloud, GoogleDocs, document.gtxt, Data};
6. {John, iOS, Safari, iOS, SFR Network, GoogleCloud, GoogleDocs, document.gtxt, Data}.

By applying the definitions of Sections 2.3, we obtain John's social and digital dependences, and the degree of these dependences for this activity. We show the results concerning some sets of persons John depends on in Table 3 and the degree of dependences on these sets in Figure 6. This information reveals how much John is autonomous from a specific person or a 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.

Q3 *Who are the persons that can prevent John from performing the activity "John reads document.gtxt"?*

Sets having a degree of dependence equal to 1, are the persons who can prevent John from "reading document.gtxt" because they cross all the ω-paths of this model. These sets are: G8, G9, G10, G12, and G19.

Q4 *Who are the persons that John is able to avoid to perform the activity "John reads document.gtxt"?*

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 G8 with a degree of 0.5), because this shows that there are other minimal ω-paths enabling John to read document.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.

SOCIOPATH is then useful in the evaluation process of a system with respect to trust requirements. This leads to the fifth question presented in the introduction of this section, namely *How much a user trusts a system for a specific activity?* We focus on answering this question in the following sections.

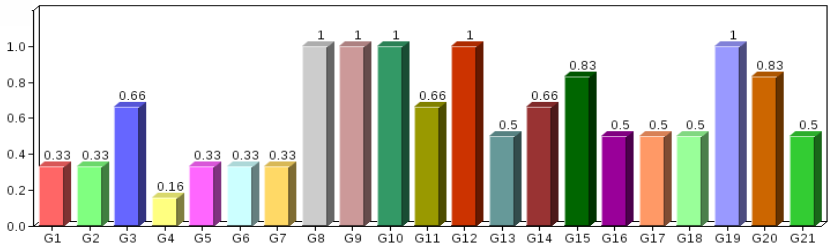


Fig. 6: Degree of dependence on persons' sets.

3 SOCIOTRUST: Evaluating trust in a system for an activity using probability theory

Trust has been widely studied in several aspects of daily life. In the trust management community [22,27,30,35,36,37], two main issues arise, *(i) how to define the trust in an entity, knowing that entities can be persons, digital and physical resources?* and *(ii) how to evaluate such a value of trust in a system under a particular context?* This second point embodies the main focus of this section.

We argue that studying trust in the separate entities that compose a system does not give a picture of how trustworthy a system is as a whole. Indeed, the trust in a system depends on its architecture, more precisely, on the way the implicit and explicit entities, which the users depend on to do their activities, are organized.

Inspired by this idea, we propose SOCIOTRUST [6], an approach to evaluate trust in a system for an activity. The system definition is based on SOCIOPATH models (*cf.* Section 2), which here are simplified to present the architecture of a system as a weighted directed acyclic graph (DAG). Levels of trust are then defined for each node in the graph according to the user who evaluates trust. By combining trust values using the theory of probability, we are able to estimate two different granularities of trust, namely, *trust in a path* and *trust in a system*, both for an activity to be performed by a person.

We begin this section introducing how to present a SOCIOPATH model as a directed acyclic graph in Section 3.1. Section 3.2 focuses on the main problem that faces trust evaluation that is the existence of *dependent paths*. We propose to solve this problem by using conditional probability. Section 3.4, evaluates our contribution with several experiments that analyze the impact of different characteristics of a system on the behavior of the obtained trust values. Experiments realized on both synthetic traces and real datasets validate our approach.

3.1 A SOCIOPATH model as a directed acyclic graph (DAG)

We simplify the representation of SOCIOPATH models by aggregating one artifact, the set of persons controlling it, and the set of physical resources supporting it, into only one component. The resulting set of components are the nodes of the DAG and the edges

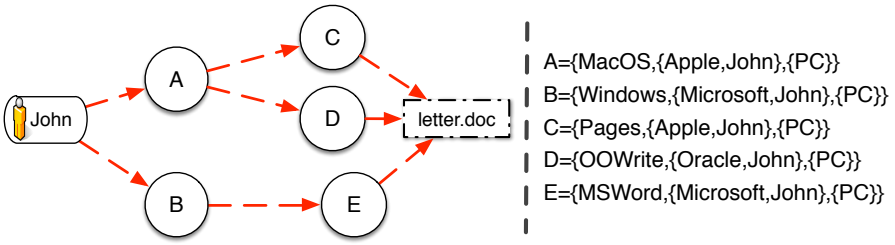


Fig. 7: The system for the activity "John edits letter.doc" as a DAG of use case 1.

are the *access* relations. A user performs an activity by browsing successive *access* relations through the graph, so-called through *activity minimal paths*.[7]

Definition 10 (A simplified system for an activity, α).
A simplified system that enables a user to achieve an activity, can be expressed as a tuple $\alpha =< \mathbb{N}_\omega, \mathbb{A}_\omega >$ *where:*

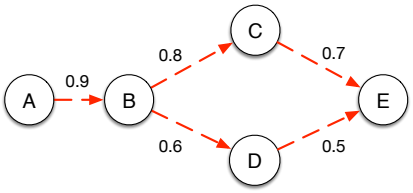
- ω *represents the activity the user wants to achieve as a triple* $(P, D, \mathcal{S})$ (cf. *Section 2.3).*
- $\mathbb{N}_\omega$ *represents the set of nodes* n *in a system for an activity such that* $\{P, D\} \subset \mathbb{N}_\omega$*, and each triple composed by one artifact, the persons who control it, and the physical resources that support it, are aggregated into one single node,* i.e., $n \in \mathbb{N}_\omega \setminus \{P, D\}$ *such that* $n \supseteq \{F, A, PR\}$ iff $controls(A, F) \wedge supports(PR, F)$.
- $\mathbb{A}_\omega \subseteq \mathbb{N}_\omega \times \mathbb{N}_\omega$ *represents the set of edges in a system. From the rules of* SOCIOPATH *and the aggregation we made for a node, our DAG exhibits only the relation* access.

Figure 7 illustrates a DAG obtained from the model of the use case 1 for the activity "John edits letter.doc", introduced in Figures 2 and 3 (*cf.* pages 6 and 9). In this example, all artifacts are supported by the physical resource (PC) owned by John. Here we consider only ω-minimal paths so the path containing the artifact MSExcel is not included. Considered artifacts are Windows, MacOs, MSWord, Pages, and OOWrite. For instance, the node A is a simplification of the artifact MacOS, along with the set of persons who control it {Apple, John} and the set of physical resource that supports it {PC}. Each edge of the DAG represents the relation *accesses*. The paths that enable John to edit letter.doc become: σ_1 ={A,C}; σ_2 ={A,D}; σ_3 ={B,E}.Notice that John, letter.doc, and Data are omitted in this paths' simplification. This type of graph will be used next as well as in Section 4.

3.2 The problem of dependent paths

Graph-based trust approaches [1,17,20,21,26,31], are especially used in social networks where the main idea of trust derivation is to propagate trust between two nodes in a

[7] If there is no ambiguity, we denote an activity minimal path (*i.e.,* ω-minimal path) through the DAG simply by a path σ and each path does not consider the source and the target nodes, *i.e.,* the person and the data instance and the data.



$\otimes$ average $\oplus$ maximum	$\otimes$ multiplication $\oplus$ comultiplication
Relation (1): $T_E^A = 0.825$	Relation (1): $T_E^A = 0.623$
Relation (2): $T_E^A = 0.8$	Relation (2): $T_E^A = 0.64$

Fig. 8: Results of Relations (1) and (2) applied to discrete and continuous metrics.

graph that represents the social network. In [1], authors propose a general approach where they divide the process of trust evaluation into two steps:

1. Trust combination through a path: the main idea is to combine the trust values among the intermediate edges of a path to obtain a trust value for this path. Several operators are employed ranging from basic operators like the minimum to new operators like *discounting* of subjective logic [19], *cf.* Section 4.1.
2. Trust combination through a graph: the main idea is to combine the trust values of all the paths that relate the source with the target, to obtain a single trust value for the graph. Several operators are employed, ranging from basic operators like the average to more recent ones like the *consensus* operator of subjective logic.

In [20,21], Jøsang *et al.* raised a problem of graph-based trust approaches if trust is evaluated through the previous two steps. They argue that some metrics do not give exact results when there are dependent paths, *i.e.*, paths that have common edges in the graph. To explain this problem, we give a simple example shown in Figure 8. We need to evaluate T_E^A, that is A's trust value in E. The paths between A and E are $path_1 = \{A, B, C, E\}$ and $path_2 = \{A, B, D, E\}$. There is a common edge between these two paths, which is $A \longrightarrow B$. Let $\otimes$ be the operator of trust combination through a path and $\oplus$ be the operator of trust combination through a graph. To evaluate T_E^A:

$$T_E^A = T_B^A \otimes ((T_C^B \otimes T_E^C) \oplus (T_D^B \otimes T_E^D)) \tag{1}$$

However, if we apply the previous two steps, T_E^A is computed as follows:

$$T_E^A = (T_B^A \otimes T_C^B \otimes T_E^C) \oplus (T_B^A \otimes T_D^B \otimes T_E^D) \tag{2}$$

Relations (1) and (2) consist of the same two paths, $path_1$ and $path_2$, but their combined structures are different (T_B^A appears twice in Relation (2)). In some metrics, these two equations produce different results. For instance, when implementing $\otimes$ as binary logic "AND", and $\oplus$ as binary logic "OR", the results would be equal. However, if $\oplus$ is the maximum function and $\otimes$ is the average function, the results are different (*cf.* Figure 8). It is also the case when $\otimes$ and $\oplus$ are implemented as probabilistic multiplication and comultiplication respectively.

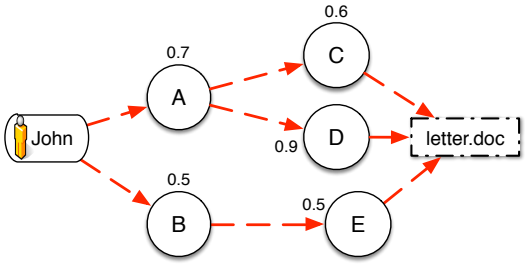


Fig. 9: The activity "John edits letter.doc" as a WDAG.

Concept	*Notation*	Remark
Trust in a node for an activity	$t(N)$	For a given activity ω achieved by a person P. The symbols ω and P are omitted in these notations for simplicity.
Trust in a path for an activity	$t(\sigma)$	
Trust in a system for an activity	$t(\alpha)$	
Event "N provides the expected services for an activity"	λ^N	
Event "P achieves an activity through the path σ"	λ^σ	
Event "P achieves an activity through the system"	λ^α	
Probability of an event	$\mathbb{P}(\lambda)$	

Table 4: Glossary of notations (2).

3.3 A probabilistic approach to infer system trust value

If a user needs to evaluate her trust in a system for an activity, she associates each node in the DAG with a trust value and the DAG becomes a weighted directed acyclic graph (WDAG). The notations used here are summarized in Table 4.

We define a function that associates each node with a trust value as $t : \mathbb{N} \rightarrow [0, 1]$ that assigns to each node a person's trust level within the interval $[0, 1]$, where 0 means not trustworthy at all and 1 means fully trustworthy. The values associated to nodes in Figure 9 are the levels of trust defined by John.

In this study, we adopt the definition of Jøsang *et al.* about trust [22]: *"trust is the probability by which an individual, A, expects that another individual, B, performs a given action on which its welfare depends"*.

According to this, we consider three notions (or granularities) of trust that are formalized in the next.

- **Trust in a node for an activity**: The trust value of a user P in a node N for an activity ω is the probability, by which P believes that N provides her the expected services for ω. Then, we have $t(N) = \mathbb{P}(\lambda^N)$.
- **Trust in a path for an activity**: The trust value of a user P in a path σ for an activity ω is the probability, by which P believes that σ enables her to achieve ω. Then, we have $t(\sigma) = \mathbb{P}(\lambda^\sigma)$.
- **Trust in a system for an activity**: The trust value of a user P in a system α for an activity ω is the probability, by which P believes that α enables her to achieve ω. Then, we have $t(\alpha) = \mathbb{P}(\lambda^\alpha)$.

Trust in a node for an activity. Trust in a node is evaluated from the point of view of the concerned user. There are several ways to construct this trust level. We can figure out different objective and subjective factors that impact this trust level, like the reputation of the persons who control the artifact, their skills, the performance of the physical resource that supports the artifact or the personal experience with this artifact. We thus have $t(N) = f(t_\omega^F, t_\omega^\mathcal{P}, t_\omega^{\mathcal{PR}})$, where t_ω^F, $t_\omega^\mathcal{P}$, $t_\omega^{\mathcal{PR}}$ are respectively the trust values assigned to an artifact F, the set of persons $\mathcal{P}$ who control F, and the set of physical resources $\mathcal{PR}$ that supports F for a given activity ω. The meaning of the resulting trust value in a node depends on the employed function f to compute this value [28]. For instance, if Bayesian inference is employed to evaluate it as is done in [24], the node trust value is considered as *"the probability by which a user believes that a node can perform an expected action for a given activity"* [13].
However, in this study, we do not address the issue of computing the trust value of a user in a node for an activity but we interpret it as the probability, by which a user P believes that a node N provides her the expected services for ω. Then, we have:

$$t(N) = \mathbb{P}(\lambda^N) \tag{3}$$

Trust in a path for an activity. A path in a system represents a way to achieve an activity. The trust level of a person P to achieve an activity through a particular path $\sigma = \{N_1, N_2, \ldots, N_n\}$ is the probability that all nodes $\{N_i\}_{i\in[1..n]}$ provide the expected services for the activity. Thus $\mathbb{P}(\lambda^\sigma)$ is computed as follows:

$$t(\sigma) = \mathbb{P}(\lambda^\sigma) = \mathbb{P}(\lambda^{N_1} \wedge \lambda^{N_2} \wedge \ldots \wedge \lambda^{N_n})$$

The event λ^{N_i} means that N_i provides the expected services for an activity. Since the graph is acyclic (only minimum activity paths are considered), then the nodes $N_1, \ldots, N_n$ are different in the path, thus each λ^{N_i} is independent from all others. Hence, we can rewrite the trust in a path as follows:

$$t(\sigma) = \mathbb{P}(\lambda^\sigma) = \mathbb{P}(\lambda^{N_1}) \times \mathbb{P}(\lambda^{N_2}) \times \ldots \times \mathbb{P}(\lambda^{N_n}) = \prod_{i=1}^{n} \mathbb{P}(\lambda^{N_i}) \tag{4}$$

Trust in a system for an activity. In general, a system is composed of several paths that represent the different ways a person has, to achieve an activity. The trust level of a person P in a system α to achieve an activity is the probability that she achieves her activity through at least one of the paths in the system. To evaluate the trust in a system for an activity, two cases have to be considered: *(i)* the paths are independent, *i.e.*, they do not have nodes in common[8] and *(ii)* the paths are dependent, *i.e.*, paths having nodes in common.

[8] The dependent paths in our graph are the paths that have common nodes (and not common edges) because the trust value is associated to a node, and not to an edge as in a social network.

Independent paths. Let $\{\sigma_i\}_{i\in[1..m]}$ be independent paths that enable a person P to achieve an activity. The probability of achieving the activity through a system, $\mathbb{P}(\lambda^{\alpha})$, is the probability of achieving the activity through at least one of the paths σ_i. Thus $\mathbb{P}(\lambda^{\alpha})$ is computed as follows:

$$t(\alpha) = \mathbb{P}(\lambda^{\alpha}) = \mathbb{P}(\lambda^{\sigma_1} \vee \lambda^{\sigma_2} \vee \ldots \vee \lambda^{\sigma_m})$$

Since the paths are independent then the equation can be rewritten as follows:

$$t(\alpha) = \mathbb{P}(\lambda^{\alpha}) = 1 - \prod_{i=1}^{m}(1 - \mathbb{P}(\lambda^{\sigma_i})) \tag{5}$$

For instance, if a person has two independent paths to achieve an activity then:

$$\begin{aligned} t(\alpha) = \mathbb{P}(\lambda^{\alpha}) &= \mathbb{P}(\lambda^{\sigma_1} \vee \lambda^{\sigma_2}) \\ &= 1 - (1 - \mathbb{P}(\lambda^{\sigma_1})) \times (1 - \mathbb{P}(\lambda^{\sigma_2})) \\ &= \mathbb{P}(\lambda^{\sigma_1}) + \mathbb{P}(\lambda^{\sigma_2}) - \mathbb{P}(\lambda^{\sigma_1}) \times \mathbb{P}(\lambda^{\sigma_2}) \end{aligned} \tag{6}$$

Dependent paths. When there are common nodes between paths, Relation (5) cannot be applied directly. To evaluate the trust through dependent paths, we begin with a simple case, where a system has two paths, before generalizing.

1. ***Two dependent paths with one common node.*** Let σ_1, σ_2, be two paths that enable a person P to achieve an activity. $\sigma_1 = \{N, N_{1,2}, \ldots, N_{1,n}\}$, $\sigma_2 = \{N, N_{2,2}, \ldots, N_{2,m}\}$. These two paths have a common node, which is N so they are dependent. Thus the probability that a person P achieves the activity ω through the system α is computed as follows:

   $$t(\alpha) = \mathbb{P}(\lambda^{\alpha}) = \mathbb{P}(\lambda^{\sigma_1} \vee \lambda^{\sigma_2}) = \mathbb{P}(\lambda^{\sigma_1}) + \mathbb{P}(\lambda^{\sigma_2}) - \mathbb{P}(\lambda^{\sigma_1} \wedge \lambda^{\sigma_2})$$

   The probability $\mathbb{P}(\lambda^{\sigma_1} \wedge \lambda^{\sigma_2})$ can be rewritten using conditional probability as the two paths are dependent.

   $$\begin{aligned} t(\alpha) = \mathbb{P}(\lambda^{\alpha}) &= \mathbb{P}(\lambda^{\sigma_1}) + \mathbb{P}(\lambda^{\sigma_2}) - \mathbb{P}(\lambda^{\sigma_2}) \times \mathbb{P}(\lambda^{\sigma_1}|\lambda^{\sigma_2}) \\ &= \mathbb{P}(\lambda^{\sigma_1}) + \mathbb{P}(\lambda^{\sigma_2}) \times (1 - \mathbb{P}(\lambda^{\sigma_1}|\lambda^{\sigma_2})) \end{aligned}$$

   We have to compute $\mathbb{P}(\lambda^{\sigma_1}|\lambda^{\sigma_2})$, which is the probability that P achieves the activity through σ_1 once it is already known that P achieves the activity through σ_2. Thus, it is the probability that N, $\{N_{1,i}\}_{i\in[2..n]}$ provides the expected services for this activity, once it is known that N, $\{N_{2,i}\}_{i\in[2..m]}$ provided the expected services. Thus, N has already provided the expected services. Hence, $\mathbb{P}(\lambda^{\sigma_1}|\lambda^{\sigma_2}) = \prod_{i=2}^{n}\mathbb{P}(\lambda^{N_{1,i}})$, where $\lambda^{N_{1,i}}$ is the event "$N_{1,i}$ provides the necessary services for the activity".

$$
\begin{aligned}
t(\alpha) &= \mathbb{P}(\lambda^{\alpha}) \\
&= \mathbb{P}(\lambda^{N}) \times \prod_{i=2}^{n} \mathbb{P}(\lambda^{N_{1,i}}) + \mathbb{P}(\lambda^{N}) \times \prod_{i=2}^{m} \mathbb{P}(\lambda^{N_{2,i}}) \times (1 - \prod_{i=2}^{n} \mathbb{P}(\lambda^{N_{1,i}})) \\
&= \mathbb{P}(\lambda^{N}) \times \left[\prod_{i=2}^{n} \mathbb{P}(\lambda^{N_{1,i}}) + \prod_{i=2}^{m} \mathbb{P}(\lambda^{N_{2,i}}) \times (1 - \prod_{i=2}^{n} \mathbb{P}(\lambda^{N_{1,i}})) \right] \\
&= \mathbb{P}(\lambda^{N}) \times \left[\prod_{i=2}^{n} \mathbb{P}(\lambda^{N_{1,i}}) + \prod_{i=2}^{m} \mathbb{P}(\lambda^{N_{2,i}}) - \prod_{i=2}^{m} \mathbb{P}(\lambda^{N_{2,i}}) \times \prod_{i=2}^{n} \mathbb{P}(\lambda^{N_{1,i}}) \right]
\end{aligned}
$$

From Relation (6) we can note that the term:

$$\prod_{i=2}^{n} \mathbb{P}(\lambda^{N_{1,i}}) + \prod_{i=2}^{m} \mathbb{P}(\lambda^{N_{2,i}}) - \prod_{i=2}^{m} \mathbb{P}(\lambda^{N_{2,i}}) \times \prod_{i=2}^{n} \mathbb{P}(\lambda^{N_{1,i}})$$

is the probability that P achieves the activity through $\sigma_1' = \{N_{1,2}, \ldots, N_{1,n}\}$ or $\sigma_2' = \{N_{2,2}, \ldots, N_{2,m}\}$, which are the paths after eliminating the common nodes. Thus the previous equation can be rewritten as follows:

$$t(\alpha) = \mathbb{P}(\lambda^{\alpha}) = \mathbb{P}(\lambda^{N}) \times \mathbb{P}(\lambda^{\sigma_1'} \vee \lambda^{\sigma_2'})$$

2. ***Two dependent paths with several common nodes.*** Let σ_1, σ_2, be two paths that enable a person P to achieve an activity. These two paths have several common nodes. By following the same logic as before, we compute the probability that a person P achieves activity ω through system α as follows:

$$t(\alpha) = \mathbb{P}(\lambda^{\alpha}) = \prod_{N \in \sigma_1 \cap \sigma_2} \mathbb{P}(\lambda^{N}) \times \mathbb{P}(\lambda^{\sigma_1'} \vee \lambda^{\sigma_2'})$$

where $\sigma_1' = \sigma_1 \setminus \sigma_2$, $\sigma_2' = \sigma_2 \setminus \sigma_1$.

3. ***Several dependent paths.*** A person may have several paths l with common nodes. Thus $\mathbb{P}(\lambda^{\alpha})$ is computed as follows:

$$t(\alpha) = \mathbb{P}(\lambda^{\alpha}) = \mathbb{P}(\lambda^{\sigma_1} \vee \lambda^{\sigma_2} \vee \ldots \vee \lambda^{\sigma_l}) =$$
$$\mathbb{P}(\lambda^{\sigma_1} \vee \lambda^{\sigma_2} \vee \ldots \vee \lambda^{\sigma_{l-1}}) + \mathbb{P}(\lambda^{\sigma_l}) - \mathbb{P}(\lambda^{\sigma_l}) \times \mathbb{P}(\lambda^{\sigma_1} \vee \lambda^{\sigma_2} \vee \ldots \vee \lambda^{\sigma_{l-1}} | \lambda^{\sigma_l}) \quad (7)$$

Let us discuss these terms one by one:

- The term $\mathbb{P}(\lambda^{\sigma_l})$ can be computed directly from Relation (4).
- The term $\mathbb{P}(\lambda^{\sigma_1} \vee \lambda^{\sigma_2} \vee \ldots \vee \lambda^{\sigma_{l-1}})$ can be computed recursively using Relation (7).
- The term $\mathbb{P}(\lambda^{\sigma_1} \vee \lambda^{\sigma_2} \vee \ldots \vee \lambda^{\sigma_{l-1}} | \lambda^{\sigma_l})$ needs first to be simplified. If we follow the same logic as before, the term $\mathbb{P}(\lambda^{\sigma_1} \vee \lambda^{\sigma_2} \vee \ldots \vee \lambda^{\sigma_{l-1}} | \lambda^{\sigma_l})$ can be replaced by the term $\mathbb{P}(\lambda^{\sigma_1'} \vee \lambda^{\sigma_2'} \vee \ldots \vee \lambda^{\sigma_{l-1}'})$ where we obtain each $\lambda^{\sigma_i'}$ by eliminating the nodes in common with σ_l.
- $\mathbb{P}(\lambda^{\sigma_1'} \vee \lambda^{\sigma_2'} \vee \ldots \vee \lambda^{\sigma_{l-1}'})$ can be computed recursively using Relation (7), and recursion is guaranteed to terminate while the number of paths is finite.

We are now able to evaluate the trust in a whole system α.

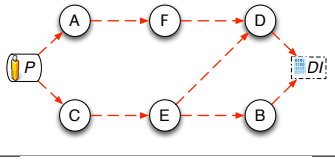
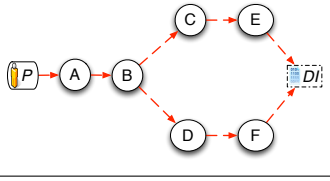
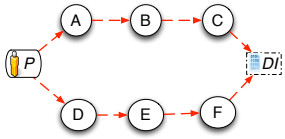
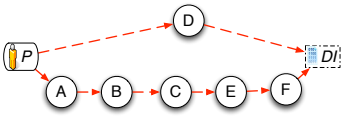
α	$t_\omega(\alpha)$	α	$t_\omega(\alpha)$
 α_1	0.4409	 α_2	0.0144
 α_3	0.507	 α_4	0.9003

Table 5: Different systems and their trust values.

3.4 Experimental evaluations

In this section, we present different experiments, their results, analysis, and interpretation. The main objectives are *(i)* to study the influence of the system organization on the computed trust values and *(ii)* to confront this approach with real users.

Influence of the system architecture on the trust value. This experiment studies the influence of the system organization on the computed trust value. We apply our equations on different systems that have the same number of nodes and the same values of trust assigned to each node, but assembled in different topologies as presented in Table 5. The values of trust associated to nodes A,B,C,D,E,F are $0.1, 0.2, 0.3, 0.9, 0.8, 0.7$ respectively.

We compute the trust value $t(\alpha)$ for each system. We obtain very divergent results varying from 0.0144 to 0.9003 as illustrated in Table 5. Thus, collecting the values of trust in each separated node in a system is not enough to determine if the system is trustworthy or not for an activity. One must also know how the system is organized. For example, in α_2, all the paths contain the nodes A and B and the trust values in these nodes are quite low, 0.1 and 0.2 respectively, so the system trust value is also low due to the strong dependency on these two nodes in this system.

Influence of the path length and the number of paths on the trust value. This experiment observes the evolution of the trust value for an activity according to some characteristics of the graph like path's length and number of paths. As a dataset, we consider random graphs composed of 20 to 100 nodes, and 1 to 15 paths. Each node in the graph is associated to a random value of trust from a predefined range.

First, the evolution of trust values according to the paths' lengths in a graph is evaluated. Each simulated graph is composed of 5 paths with lengths varying from 1 to 15 nodes. Different trust values were simulated in the ranges $[0.6, 0.9]$, $[0.1, 0.9]$ *etc.* Figure 10 illustrates the impact of the path length on the trust value. Note that, the

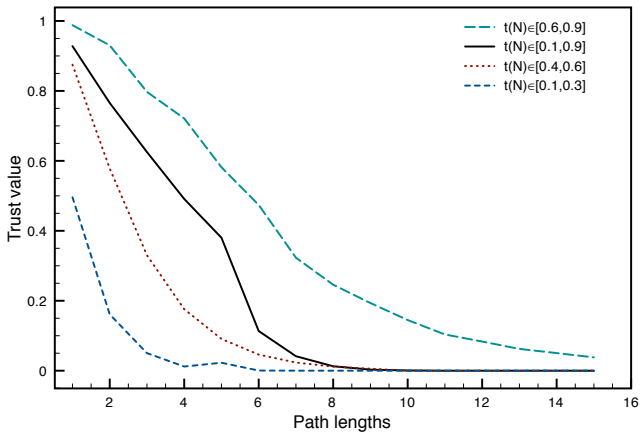


Fig. 10: System trust value according to the length of paths.

system trust value decreases when the length of paths increases. This reflects a natural intuition we had from the fact that trust values are multiplied.

Second, we set the path lengths to 5 nodes and we increased the number of paths from 1 up to 15 in order to observe the variation of the trust values. Again, different node trust values were simulated in the ranges $[0.7, 0.9]$, $[0.6, 0.9]$, *etc*. Figure 11 illustrates that the trust value increases as the number of paths increases. This reflects the intuition that the measure of trust in a system for an activity rises when the number of ways to achieve this activity increases.

Social evaluation (a real case). In order to evaluate our proposal in a real use case, we modeled part of the SVN system of LINA research laboratory[9] with SOCIOPATH. SVN (Subversion) is a client-server system to manage versions of files. The server allocates repositories of files and clients make copies of repositories. Copies of files contained in repositories can be modified at the client side, modification must be committed to generate new versions. Other clients must frequently update their copies. Persons on which SVN users depend on, are the LINA laboratory that owns the server and the software SVN, the engineer that controls the software at the server side of the SVN, the provider of the software SVN, the computer and the software at the client side, *etc*. We applied the rules of SOCIOPATH on this system for the activity "a user accesses a file on the SVN". Due to privacy issues, Figure 12 presents the DAG for this activity with anonymous nodes. For the sake of clarity, we simplify the underlying graph as much as possible.

Based on this context, we conducted an opinion survey among twenty members of LINA including, PhD students, professors and technicians about their level of trust in each node. For each person, we have computed the system trust value according

[9] https://www.lina.univ-nantes.fr/

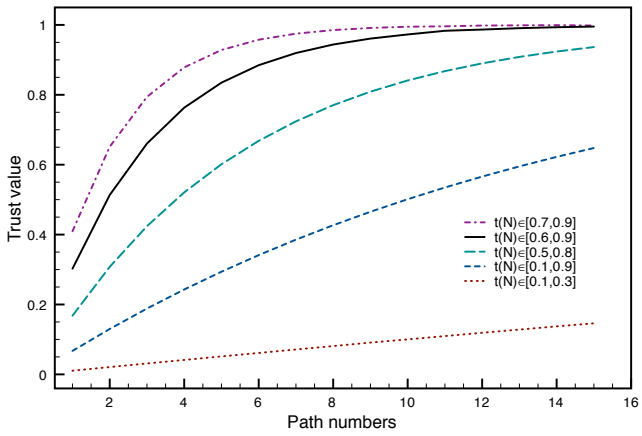


Fig. 11: System trust value according to the number of paths.

to the methodology presented in Section 3.3. Table 6 presents the data of the survey and the computed trust values. In a second phase, we asked each user if the SOCIOTRUSTproposal correctly reflects her trust towards the SVN system used in our lab. The possibilities of answer were simply `Yes` or `No`. The last column of Table 6 presents this feedback, where ✓ means that they are satisfied, and × means that they are not satisfied. 75% of the users are satisfied with the computation. Unsatisfied users argue that they expected a higher trust value. Some of the trust values associated to the nodes of the unsatisfied users, have relatively low values (around 0.5 or 0.6) compared to other users. These users explained that the lack of knowledge about some nodes leads them to define what they called *a neutral value* (*i.e.,* 0.5 or 0.6) that they considered neither trustworthy, nor untrustworthy. Clearly, such a behavior is not compatible with a probabilistic interpretation where 0.5 is like any other possible value between 0 and 1 and has nothing of neutral.

The explanations provided by users revealed an interesting point: even in a small environment and considering advanced users, no one is in possession of all the information necessary to construct an informed assessment. To conform to this reality and model this phenomenon, it is necesary to use a formalism allowing to express uncertainty related to incompleteness of available information. Extending our approach to use subjective logic [19], which can express uncertainty or ignorance, is the objective of the next section.

4 SUBJECTIVETRUST: Evaluating trust in a system for an activity using subjective logic

SOCIOTRUST is oriented to full-knowledge environments. However, in uncertain environments, users might not be in possession of all the information to provide a dogmatic opinion and traditional probability cannot express uncertainty. With subjective

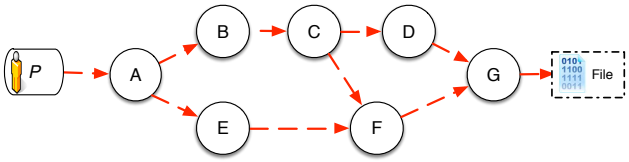


Fig. 12: LINA's WDAG for the activity "a user accesses a file on the SVN".

	A	B	C	D	E	F	G	System trust value	User's feedback
P_1	0.5	0.5	1	0.5	0.5	1	1	0.4375	✓
P_2	0.7	1	1	0.7	0.7	1	1	0.847	✓
P_3	0.5	0.5	1	0.7	0.5	1	1	0.4375	×
P_4	0.6	0.6	0.8	0.7	0.6	0.8	0.6	0.3072	×
P_5	0.8	0.8	1	0.8	0.8	1	0.9	0.8202	✓
P_6	0.9	0.9	1	0.9	0.9	0.9	0.9	0.9043	✓
P_7	0.6	0.6	0.7	0.6	0.6	0.6	0.7	0.2770	×
P_8	0.8	0.6	1	0.9	0.8	0.8	1	0.7416	✓
P_9	0.7	0.5	1	0.4	0.7	0.6	0.9	0.4407	✓
P_{10}	0.8	1	0.7	0.8	0.8	0.9	0.8	0.6975	✓
P_{11}	0.5	0.5	0.9	0.5	0.5	0.5	0.9	0.2473	×
P_{12}	0.95	0.95	0.8	0.8	0.95	0.95	0.8	0.8655	✓
P_{13}	0.8	0.9	0.8	0.7	0.95	0.8	0.7	0.6433	✓
P_{14}	0.8	0.7	0.9	0.7	0.9	0.8	0.8	0.6652	✓
P_{15}	0.9	0.8	0.8	0.9	0.9	0.9	0.8	0.7733	✓
P_{16}	0.7	0.6	0.6	0.6	0.8	0.7	0.6	0.337	✓
P_{17}	0.5	0.9	0.8	0.7	0.9	0.5	0.8	0.3807	×
P_{18}	0.7	0.7	1	0.7	0.6	0.7	1	0.6088	✓
P_{19}	0.8	0.8	1	1	1	0.8	1	0.8704	✓
P_{20}	0.9	0.9	0.8	0.9	0.9	0.9	0.8	0.7971	✓

Table 6: User's trust value in the system SVN in LINA.

logic [19], trust can be expressed as subjective opinions with degrees of uncertainty. In this section, we aim to take advantage of the benefits of subjective logic to evaluate trust.

The main contribution of this section is proposing a generic model named SUBJECTIVETRUST [4], for evaluating trust in a system for an activity taking into account uncertainty. By combining the user's opinion on a node, we are able to estimate two different granularities of trust, namely, *opinion on a path* and *opinion on a system*, both for an activity to be performed by a person. As we know, the main problem that faces trust evaluation based on a graph is the existence of *dependent paths*. To solve this problem, we propose two methods: **Copy** and **Split**.

Next section presents some preliminaries about subjective logic, then we present SUBJECTIVETRUST in Section 4.2 and finally some experiments in Section 4.3.

4.1 Preliminaries about subjective logic

In the terminology of subjective logic [19], an opinion held by an individual P about a proposition x is the ordered quadruple $O_x = (b_x, d_x, u_x, a_x)$ where:

- b_x (belief) is the belief that x is true.
- d_x (disbelief) is the belief that the x is false.
- u_x (uncertainty) is the amount of uncommitted belief.
- a_x is called the base rate, it is the a priori probability in the absence of evidence.

Note that $b_x, d_x, u_x, a_x \in [0,1]$ and $b_x + d_x + u_x = 1$. a_x is used for computing an opinion's probability expectation value that can be determined as $\mathbb{E}(O_x) = b_x + a_x u_x$. More precisely, a_x determines how uncertainty shall contribute to the probability expectation value $\mathbb{E}(O_x)$.

Subjective logic consists of a set of logical operations which are defined to combine opinions.

- Conjunction operator ($\wedge$) represents the opinion of a person on several propositions.
- Disjunction operator ($\vee$) represents the opinion of a person on one of the propositions or any union of them.
- Discounting operator ($\otimes$) represents the transitivity of the opinions.
- Consensus operator ($\oplus$) represents the consensus of opinions of different persons.

In our work, we rely on a graph to evaluate trust like in the social network domain, but our interpretation of the graph is different. For us, a graph represents *a system for a digital activity* and not *a social network*. This assumption plays an important role in the operations we apply for trust evaluation. That is why, in a social network, to evaluate trust through a path using subjective logic, the operator of discounting ($\otimes$) is used to compute the transitivity through a path, whereas, in our work, evaluating trust in a path is the trust in the *collection* of the nodes that form this path, *i.e.,* conjunction. In the same manner, to evaluate trust through a graph in a social network, the operator of consensus ($\oplus$) is used to evaluate the consensus of opinions of different persons through the different paths that form the graph, whereas, in our work, paths represent the ways one person disposes to achieve an activity, so evaluating trust in a graph is the trust in at least one of the paths or any union of them, *i.e.,* disjunction. In the following, we present the conjunction and disjunction operators that we use in SUBJECTIVETRUST.

- Conjunction represents the opinion of a person on several propositions. Let $O_x^P = (b_x^P, d_x^P, u_x^P, a_x^P)$ and $O_y^P = (b_y^P, d_y^P, u_y^P, a_y^P)$ be respectively P's opinion on x and y. $O_{x\wedge y}^P$ represents P's opinion on both x and y and can be calculated as follows:

$$O_x^P \wedge O_y^P = O_{x\wedge y}^P = \begin{cases} b_{x\wedge y}^P = b_x^P b_y^P \\ d_{x\wedge y}^P = d_x^P + d_y^P - d_x^P d_y^P \\ u_{x\wedge y}^P = b_x^P u_y^P + u_x^P b_y^P + u_x^P u_y^P \\ a_{x\wedge y}^P = \frac{b_x^P u_y^P a_y^P + b_y^P u_x^P a_x^P + u_x^P a_x^P u_y^P a_y^P}{b_x^P u_y^P + u_x^P b_y^P + u_x^P u_y^P} \end{cases} \tag{8}$$

$$\mathbb{E}(O_x^P \wedge O_y^P) = \mathbb{E}(O_{x\wedge y}^P) = \mathbb{E}(O_x^P)\mathbb{E}(O_y^P) \tag{9}$$

- Disjunction represents the opinion of a person on one of the propositions or any union of them. Let $O_x^P = (b_x^P, d_x^P, u_x^P, a_x^P)$ and $O_y^P = (b_y^P, d_y^P, u_y^P, a_y^P)$ be respectively P's opinion on x and y. $O_{x\vee y}^P$ represents P's opinion on x or y or both and can be calculated with the following relations:

$$O_x^P \vee O_y^P = O_{x \vee y}^P = \begin{cases} b_{x \vee y}^P = b_x^P + b_y^P - b_x^P b_y^P \\ d_{x \vee y}^P = d_x^P d_y^P \\ u_{x \vee y}^P = d_x^P u_y^P + u_x^P d_y^P + u_x^P u_y^P \\ a_{x \vee y}^P = \frac{u_x^P a_x^P + u_y^P a_y^P - b_x^P u_y^P a_y^P - b_y^P u_x^P a_x^P - u_x^P a_x^P u_y^P a_y^P}{u_x^P + u_y^P - b_x^P u_y^P - b_y^P u_x^P - u_x^P u_y^P} \end{cases} \tag{10}$$

$$\mathbb{E}(O_x^P \vee O_y^P) = \mathbb{E}(O_{x \vee y}^P) = \mathbb{E}(O_x^P) + \mathbb{E}(O_y^P) - \mathbb{E}(O_x^P)\mathbb{E}(O_y^P) \tag{11}$$

It is important to mention that conjunction and disjunction are commutative and associative.

$$O_x^P \wedge O_y^P = O_y^P \wedge O_x^P$$
$$O_x^P \vee O_y^P = O_y^P \vee O_x^P$$
$$(O_x^P \wedge O_y^P) \wedge O_z^P = O_x^P \wedge (O_y^P \wedge O_z^P)$$
$$(O_x^P \vee O_y^P) \vee O_z^P = O_x^P \vee (O_y^P \vee O_z^P)$$

However, the conjunction over the disjunction is not distributive. This is due to the fact that opinions must be assumed to be independent, whereas distribution always introduces an element of dependence.

$$O_x^P \wedge (O_y^P \vee O_z^P) \neq (O_x^P \wedge O_y^P) \vee (O_x^P \wedge O_z^P)$$

By using these operators, in the next section we combine the opinions on the nodes to estimate two different granularities of trust: opinion on a path and opinion on a system.

4.2 Inferring user's opinion on a system using subjective logic

This section presents SUBJECTIVETRUST, a graph-based trust approach to infer trust in a system for an activity using subjective logic.

The system definition is based on SOCIOPATH. To focus on trust in the system, the SOCIOPATH model is abstracted in a DAG as in SOCIOTRUST (*cf.* Section 3.1). In subjective logic, trust is expressed as an opinion, thus in this proposition, the DAG is weighted with opinions, *i.e.*, each node is associated with an opinion in the form (b, d, u, a). Figure 13 shows the WDAG of use case 1, where the values associated to nodes represent John's opinion on these nodes.

As in SOCIOTRUST, opinion on a node is evaluated from the point of view of the concerned user depending on her personal experience with this node. Several approaches have been proposed to obtain this opinion. In [19], authors translate the user's negative or positive observations to opinions. In [24,25], the opinion parameters are estimated by Bayesian inference. In this study, we do not address the issue of obtaining this opinion, we focus on combining the opinions associated on the nodes to obtain an opinion on a path and on a system for an activity.

Next sections show how an opinion on a path and an opinion on a system are evaluated by combining respectively the opinions on the nodes and the opinions on the paths, using the appropriate operators of subjective logic.

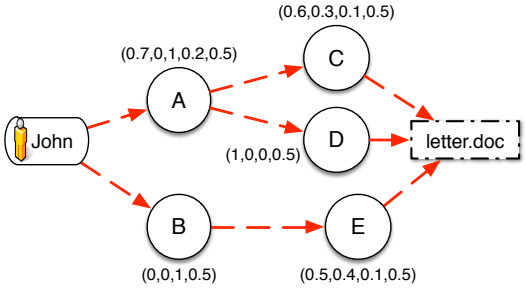


Fig. 13: The activity "John edits letter.doc" as a WDAG where weights are opinions (use case 1).

Opinion on a path for an activity. A path in a system represents a way to achieve an activity. An opinion on a path that contains several nodes can be computed by combining the opinions on the nodes that belong to it.

In trust propagation, the operator to build an opinion on a path is discounting because it allows to compute the transitivity of an opinion along a path [20,21]. However, if a person needs to achieve an activity through a path, she needs to pass through all the nodes composing this path. Hence, an opinion on a path is the opinion on all nodes composing this path.

The conjunction operator represents the opinion of a person on several propositions. Thus, it is appropriate to compute an opinion on a path from the opinions on the nodes.

Let $\sigma = \{N_1, N_2, \ldots, N_n\}$ be a path that enables a user P to achieve an activity. P's opinion on the nodes $\{N_i\}_{i\in[1..n]}$ for an activity are denoted by $O_{N_i} = (b_{N_i}, d_{N_i}, u_{N_i}, a_{N_i})$. P's opinion on the path σ for achieving an activity, denoted by $O_\sigma = (b_\sigma, d_\sigma, u_\sigma, a_\sigma)$, can be derived by the conjunction of P's opinions on $\{N_i\}_{i\in[1..n]}$. $O_{\sigma=\{N_1,\ldots,N_n\}} = \bigwedge\{O_{N_i}\}_{i\in[1..n]}$. Given Relation (8), we obtain the following generalization: $O_{\sigma=\{N_1,\ldots,N_n\}} =$

$$\begin{cases} b_{\sigma=\{N_1,\ldots,N_n\}} = b_{\bigwedge\{N_i\}_{i\in[1..n]}} = \prod_{i=1}^{n} b_{N_i} \\ d_{\sigma=\{N_1,\ldots,N_n\}} = d_{\bigwedge\{N_i\}_{i\in[1..n]}} = 1 - \prod_{i=1}^{n} (1 - d_{N_i}) \\ u_{\sigma=\{N_1,\ldots,N_n\}} = u_{\bigwedge\{N_i\}_{i\in[1..n]}} = \prod_{i=1}^{n}(b_{N_i} + u_{N_i}) - \prod_{i=1}^{n}(b_{N_i}) \\ a_{\sigma=\{N_1,\ldots,N_n\}} = a_{\bigwedge\{N_i\}_{i\in[1..n]}} = \frac{\prod_{i=1}^{n}(b_{N_i}+u_{N_i}a_{N_i}) - \prod_{i=1}^{n}(b_{N_i})}{\prod_{i=1}^{n}(b_{N_i}+u_{N_i}) - \prod_{i=1}^{n}(b_{N_i})} \end{cases} \tag{12}$$

Due to space constraint and as they are almost straightforward, the proofs of Relation (12) and the verifications of the correction (*i.e.*, $b_\sigma + d_\sigma + u_\sigma = 1$, $0 < b_\sigma < 1$, $0 < d_\sigma < 1$, $0 < u_\sigma < 1$ and $0 < a_\sigma < 1$) are presented in [3].

Opinion on a system for an activity. In trust propagation, to build an opinion on a target node in a graph, the consensus operator is used because it represents the consensus of the opinions of different persons through different paths [20,21]. In our work, an opinion on a system is the opinion of a person on one or several paths. Thus, the

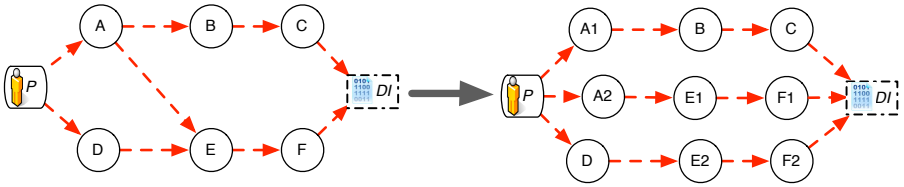


Fig. 14: Graph transformation.

disjunction operator is appropriate to evaluate an opinion on a system. In the following, we show how to build an opinion on a system when *(i)* the system has only independent paths and *(ii)* the system has dependent paths.

Independent paths. Let $\{\sigma_1, \sigma_2, \ldots, \sigma_m\}$ be the paths that enable a user P to achieve an activity. The user's opinion on the paths $\{\sigma_i\}_{i\in\{1..m\}}$ for an activity are denoted by $O_{\sigma_i} = (b_{\sigma_i}, d_{\sigma_i}, u_{\sigma_i}, a_{\sigma_i})$. The user opinion on the system α for achieving the activity, denoted by $O_\alpha = (b_\alpha, d_\alpha, u_\alpha, a_\alpha)$ can be derived by the disjunction of P's opinions in $\{\sigma_i\}_{i\in\{1..m\}}$. Thus, $O_\alpha = \bigvee\{O_{\sigma_i}\}_{i\in\{1..m\}}$. Given Relation (10), we obtain the following generalization: $O_{\alpha=\{\sigma_1,\ldots,\sigma_m\}} =$

$$\begin{cases} b_{\alpha=\{\sigma_1,\ldots,\sigma_m\}} = b_{\bigvee\{\sigma_i\}} = 1 - \prod_{i=1}^{m}(1 - b_{\sigma_i}) \\ d_{\alpha=\{\sigma_1,\ldots,\sigma_m\}} = d_{\bigvee\{\sigma_i\}} = \prod_{i=1}^{m} d_{\sigma_i} \\ u_{\alpha=\{\sigma_1,\ldots,\sigma_m\}} = u_{\bigvee\{\sigma_i\}} = \prod_{i=1}^{m}(d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^{m}(d_{\sigma_i}) \\ a_{\alpha=\{\sigma_1,\ldots,\sigma_m\}} = a_{\bigvee\{\sigma_i\}} = \frac{\prod_{i=1}^{m}(d_{\sigma_i}+u_{\sigma_i}) - \prod_{i=1}^{m}(d_{\sigma_i}+u_{\sigma_i}-u_{\sigma_i}a_{\sigma_i})}{\prod_{i=1}^{m}(d_{\sigma_i}+u_{\sigma_i}) - \prod_{i=1}^{m}(d_{\sigma_i})} \end{cases} \quad (13)$$

Again, the proofs of Relation (13) are available in [3].

Dependent paths. As we know, in subjective logic, as in probabilistic logic, the conjunction is not distributive over the disjunction. In SocioTrust, this problem has been resolved by using conditional probability. As there is not a similar formalism in subjective logic, for evaluating trust in a system we propose to transform a graph having dependent paths to a graph having independent paths. Figure 14 illustrates this transformation. The left side of this figure shows a graph that has three dependent paths. The dependent paths are[10]: $\sigma_1 = \{A, B, C\}$, $\sigma_2 = \{A, E, F\}$ and $\sigma_3 = \{D, E, F\}$. The common nodes are A, E and F. For instance, A is a common node between σ_1 and σ_2. In that transformation, A is duplicated in A_1 and A_2, such that in the new graph, $A_1 \in \sigma'_1 = \{A1, B, C\}$, and $A_2 \in \sigma'_2 = \{A2, E, F\}$, so is the case for the nodes E and F. The right part of Figure 14 shows the new graph after duplicating the common nodes. The new graph contains the paths $\sigma'_1 = \{A1, B, C\}$, $\sigma'_2 = \{A2, E1, F1\}$ and $\sigma'_3 = \{D, E2, F2\}$. Once this transformation is made, we can apply the Relations (12) and (13). To do so, we propose the following methods that are shown in Algorithms 1 and 2. Notice that lines 1-5 of these algorithms transform the graph.

[10] We recall that the person, the data instance, and the data are not considered in paths of the DAG.

```
Find all the paths σ_{i:i∈[1..n]} for an activity performed by a person
foreach σ_{i:i∈[1..n]} do
    foreach N_{j:j∈[1..length(σ_i)]} ∈ σ_i do
        foreach k ≠ i: N_j ∈ σ_k do
            Create a node N_ik
            O_{N_ik} ← O_{N_j}
            Replace N_j by N_ik in σ_k
        end
    end
end
```

Algorithm 1: Copy algorithm.

Copy. In this method, once the graph is transformed to obtain independent paths, we associate the opinion on the original node to the duplicated nodes. This method is based on the idea that the new produced path σ' maintains the same opinion of the original path σ. In this case $O_{\sigma_1} = O_{\sigma'_1}$ and $O_{\sigma_2} = O_{\sigma'_2}$.

Split: In this method, once the graph is transformed to obtain independent paths, in order to maintain the opinion on the global system, we split the opinion on the original dependent node into independent opinions, such that their disjunction produces the original opinion. Formally speaking, if node A is in common between σ_1 and σ_2, and the opinion on A is O_A, A is duplicated into $A_1 \in \sigma'_1$ and $A_2 \in \sigma'_2$ and the opinion O_A is split into O_{A_1} and O_{A_2}, where O_{A_1} and O_{A_2} satisfy the following relations: $O_{A_1} = O_{A_2}$ and $O_{A_1} \vee O_{A_2} = O_A$. The following is the relation of splitting an opinion into n independent opinions.

$$\bigwedge \begin{cases} O_{A_1} \vee O_{A_2} \vee \ldots \vee O_{A_n} = O_A \\ O_{A_1} = O_{A_2} = \ldots = O_{A_n} \end{cases} \Rightarrow$$
$$\begin{cases} b_{A1} = b_{A2} = \ldots = b_{A_n} = 1 - (1 - b_A)^{\frac{1}{n}} \\ d_{A1} = d_{A2} = \ldots = d_{A_n} = d_A^{\frac{1}{n}} \\ u_{A1} = u_{A2} = \ldots = u_{A_n} = (d_A + u_A)^{\frac{1}{n}} - d_A^{\frac{1}{n}} \\ a_{A1} = a_{A2} = \ldots = a_{A_n} = \frac{(1-b_A)^{\frac{1}{n}} - (1-b_A-a_A u_A)^{\frac{1}{n}}}{(d_A+u_A)^{\frac{1}{n}} - d_A{}^{\frac{1}{n}}} \end{cases} \quad (14)$$

Proofs of Relations (14) are provided in [3].

4.3 Experimental evaluation

In this section, we compare **Copy** and **Split** to a modified version of an approach of the literature named TNA-SL [21]. The latter approach is based on simplifying the graph by deleting the dependent paths that have high value of uncertainty, then, trust is propagated. In our work, trust is not propagated and a comparison to a propagation approach has no sense. Thus, we modify TNA-SL such that trust evaluation is made

```
Find all the paths σ_{i:i∈[1..n]} for an activity performed by a person
foreach σ_{i:i∈[1..n]} do
    foreach N_{j:j∈[1..length(σ_i)]} ∈ σ_i do
        foreach k ≠ i: N_j ∈ σ_k do
            Create a node N_ik
            O_{N_ik} ← opinion resulted from Relation (14)
            Replace N_j by N_ik in σ_k
        end
    end
end
```

Algorithm 2: Split algorithm.

by applying Relations (12) and (13) introduced in Section 4.2. We call this method "modified TNA-SL", denoted **mTNA** in the following.

We present different experiments, their results, analysis and interpretation. The main objectives are *(i)* to compare the proposed methods and evaluating their accuracy and *(ii)* to confront this approach with real users. The first two experiments are related to the first objective while the third experiment is devoted to the second objective. Next sections present the different experiments, their results, and analysis.

Comparing the proposed methods. To tackle the first objective, we experiment with a graph that contains only independent paths. The three methods, **mTNA**, **Copy** and **Split** give the same exact results as expected because the three of them follow the same computational model when graphs contain only independent paths. Then, we experiment on a graph that has relatively high rate of common nodes and dependent paths. 75% of the paths of the chosen graph are dependent paths and 60% of nodes are common nodes.

In our experiments, random opinions $O_N = (b_N, d_N, u_N, a_N)$ are associated to each node, and the opinion's probability expectation value of the graph, $\mathbb{E}(O_\alpha) = b_\alpha + a_\alpha u_\alpha$ is computed using the three methods, **mTNA**, **Copy** and **Split**. This experiment is repeated 50 times where each time represents random opinions of a person associated to the different nodes that compose the graph. We analyze the opinion's probability expectation values of the graph, $\mathbb{E}(O_\alpha) = b_\alpha + a_\alpha u_\alpha$ and not all the opinion parameters $O_\alpha = (b_\alpha, d_\alpha, u_\alpha, a_\alpha)$ for simplicity.

Figure 15 shows obtained results. We notice that the three methods almost have the same behavior, when the $\mathbb{E}(O_\alpha)$ increases in one method, it increases in the other methods, and vice versa. We also observe some differences among the three methods that are not always negligible like in experience 9 and 40 in Figure 15. This observation leads us to the question: *which of these methods give the most accurate results?* To evaluate the accuracy of **Split**, **Copy** and **mTNA**, we conduct the next experiments.

Studying the accuracy of the proposed methods. SOCIOTRUST that uses theory of probability to evaluate trust in a system, has the advantages that it has no approximations in case there are dependent paths thanks to conditional probability (*cf.* Section 3).

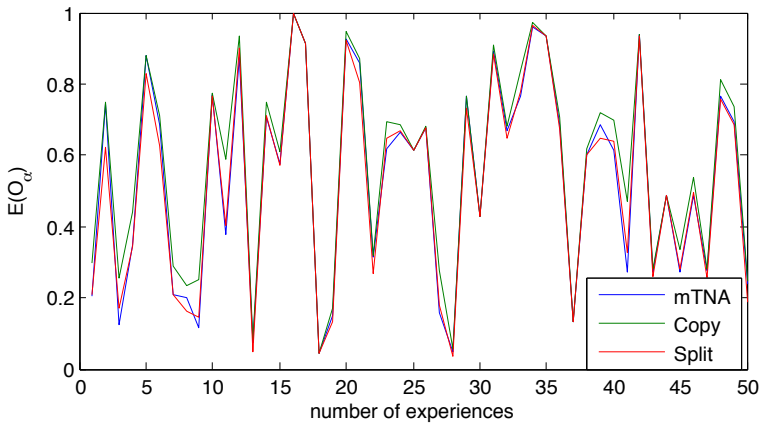


Fig. 15: Value of the probability expectation for 50 persons using the three methods **mTNA**, **Copy** and **Split**.

Thus it works perfectly if users are sure of their judgments of trust, *i.e.*, the values of uncertainty are equal to 0.

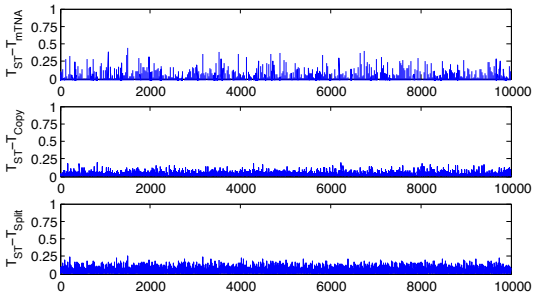
Subjective logic is equivalent to traditional probabilistic logic when $b + d = 1$ such that the value of uncertainty is equal to 0. When $u = 0$, the operations in subjective logic are directly compatible with the operations of the traditional probability. In this case the value of $\mathbb{E}(O) = b + au = b$ corresponds to the probability value.

Since SOCIOTRUST is based on probability theory, the obtained results by applying subjective logic if $u = 0$ should be equal to the ones using probability theory. We can evaluate the accuracy of the proposed methods by setting $u = 0$ and comparing the value of $b_\alpha = \mathbb{E}(O_\alpha)$ resulted from applying the three methods to the trust value obtained by applying SOCIOTRUST.

The experiments are conducted on the graph used in Figure 15. Random opinions $O_N = (b_N, d_N, 0, a_N)$ are associated to each node, and the probability expectation of the graph $\mathbb{E}(O_\alpha) = b_\alpha + a_\alpha u_\alpha = b_\alpha$ is computed. The notations T_{ST}, T_{mTNA}, T_{Copy}, T_{Split} respectively denote system's trust value resulting from applying SOCIOTRUST and system's opinion probability expectation resulting from applying **mTNA**, **Copy**, and **Split**.

To compare T_{ST} to T_{mTNA}, T_{Copy}, and T_{Split}, we simply compute the subtractions between them *i.e.*, $T_{ST} - T_{\text{mTNA}}$, $T_{ST} - T_{\text{Copy}}$, $T_{ST} - T_{\text{Split}}$. The average of each of the previous values are computed through 10,000 times to obtain a reliable value. The standard deviation (SD) is also computed to show how much variation from the average exists in the three cases. Figure 16 shows obtained results.

As we notice from Figure 16, **Copy** is the method that gives the closest results to SOCIOTRUST, the average of the difference of its result when $u = 0$ and the result of traditional probability over $10,000$ times is equal to 0.014, which is an indication that this method gives the nearest result to the exact result and its average error rate is



Method	Avg	SD
$\mid T_{ST} - T_{\text{mTNA}} \mid$	0.024	0.045
$\mid T_{ST} - T_{\text{Copy}} \mid$	0.014	0.020
$\mid T_{ST} - T_{\text{Split}} \mid$	0.032	0.037

Fig. 16: Difference between the opinion's probability expectation of a graph using **mTNA**, **Copy**, and **Split** when $u = 0$ and the trust value resulting from using SOCIOTRUST.

around 1.4%. **Copy** shows the most convincing result, with a standard deviation equals to 0.02.

The average error rate of **mTNA** (2.4%) is less than **Split** (3.2%), but the standard deviation of **mTNA** is 0.045 where in **Split**, it is 0.037. That means that in some cases, **mTNA** can give results that are farther than **Split** from the exact results. Thus, **Split** shows a more stable behavior than **mTNA**.

The objective of this experiment is not criticizing the proposed methods in the literature for the problem of dependent paths. These methods are proposed to deal with the problem of trust propagation through a graph, whereas, in our work we focus on evaluating trust towards the whole graph. The employed operators in our case are different from the employed operators in trust propagation. TNA-SL or any proposed method in the literature can work properly in their context.

In this experiment, we show that **Copy** is the method the more adaptable to be used with respect to the context of our work. Extensive simulations on different types of graphs are provided in [3] and follow the same behavior presented above.

Social evaluation (a real case). In this experiment we use the SVN system of the LINA research laboratory introduced in Section 3.4. Since subjective logic is not used yet in real applications, users are not used to build an opinion directly using this logic. We build these opinions ourselves from users' positive or negative observations as it is proposed in [19]. To do that, a survey is executed to collect the observations of LINA users about the nodes. The proposed questions collect information about the user's usage of a node and the quantity of using it and their observations. A local opinion on each entity is built for each user. The opinion and the opinion's probability expectation of the system are then computed using **Copy** for each user. The results are shown in Table 7.

We asked each user for a feedback about their opinion on the nodes and in the system. We were glad to notice that LINA users were satisfied of the obtained results, whereas when using SOCIOTRUST (*cf.* Section 3.4), 25% of users were not satisfied.

In the latter approach, when users do not have enough knowledge about a node, they assign the value 0.5, that they consider as neutral value. That leads to incorrect inputs that produce low trust values in a system. In SUBJECTIVETRUST, uncertainties are expressed in the opinions on the nodes and computing an opinion on a system is made considering these uncertainties. That shows that, in uncertain environments, it is more suitable to use subjective logic than probabilistic metrics for trust evaluations.

5 Related works

The state of the art of this work has two parts, the first one concerns the system modeling (Section 5.1) and the second one concerns the evaluation of trust (Section 5.2).

5.1 System modeling

At the beginning of this study, we searched for methodologies that could help us answer questions posed in Section 2 and we found interesting approaches in the domain of Enterprise Architecture (EA). In EA, the term "Enterprise" expresses the whole complex, socio-technical system [14], including people, information and technology. A widely known definition of an EA is: *"An Enterprise Architecture is a rigorous description of the structure of an enterprise, which comprises enterprise components, the properties of those components, and the relationships between them"*. The goal of this description is translating the business vision into models. To do that, analytical techniques are used to formalize an enterprise. This allows to produce models that describe the business processes, people organization, information resources, software applications and business capabilities within an enterprise. These models provide the keys that enable the enterprise evolution. Therefore, humans, technical resources, business information, enterprise goals, processes, the roles of each entity in an enterprise, and the organizational structures should be included in this description.

EA is very complicated and large [32]. To manage this complexity, EA Frameworks provide methods and tools that allow to produce enterprise models. Many frameworks have appeared and we studied two of the most used, The Open Group Architecture Framework (TOGAF) [18,23] and the OBASHI Business & IT methodology and framework [34]. More details about these frameworks are available in [3].

TOGAF follows the standard of modeling in four layers: 1) the technology layer, 2) the application layer, 3) the data layer, and 4) the business layer. It defines a metamodel that allows to formalize an enterprise and produce models (diagrams, catalogs and matrices) for the company stakeholders. The metamodel allows to define a formal structure of the components within an architecture like an actor, a role, a data entity, an application, and a business service. Besides the components, the metamodel defines the relationships between these components like an actor belongs to an organization unit or a role is assumed by an actor. TOGAF is a very rich and powerful framework. It produces a set of graphs represented by *diagrams* but none of them can be useful for our needs, *i.e.,* a representation of an activity achieved through a system for a given user. Besides that, obtaining TOGAF diagrams is a complex procedure that needs an expert person. This complexity and the high economical cost of TOGAF leads us to exclude this framework.

	O_A (b,d,u,a)	O_B (b,d,u,a)	O_C (b,d,u,a)	O_D (b,d,u,a)	O_E (b,d,u,a)	O_F (b,d,u,a)	O_G (b,d,u,a)	O_α (b,d,u,a)	$\mathbb{E}(O_\alpha)$
P_1	$(1,0,0,0.5)$	$(0,0,1,0.5)$	$(1,0,0,0.5)$	$(0.99,0.01,0,0.5)$	$(0.83,0,0.17,0.5)$	$(0.99,0.01,0,0.5)$	$(0.83,0,0.17,0.5)$	$(0.6820,0,0.3810,0.8389)$	0.9488
P_2	$(1,0,0,0.5)$	$(1,0,0,0.5)$	$(1,0,0,0.5)$	$(1,0,0,0.5)$	$(0.83,0,0.17,0.5)$	$(1,0,0,0.5)$	$(1,0,0,0.5)$	$(1,0,0,-)$	1
P_3	$(0.99,0.01,0,0.5)$	$(0,0,1,0.5)$	$(1,0,0,0.5)$	$(0.99,0.01,0,0.5)$	$(0,0,1,0.5)$	$(0.99,0.01,0,0.5)$	$(0.6,0,0.4,0.5)$	$(0,0,1,0.7753)$	0.7753
P_4	$(1,0,0,0.5)$	$(0,0,1,0.5)$	$(1,0,0,0.5)$	$(1,0,0,0.5)$	$(0.83,0,0.17,0.5)$	$(0.99,0.01,0,0.5)$	$(0.96,0,0.04,0.5)$	$(0.7888,0,0.2112,0.8604)$	0.9705
P_5	$(0.99,0.01,0,0.5)$	$(0,0,1,0.5)$	$(1,0,0,0.5)$	$(0.99,0.01,0,0.5)$	$(0,0,1,0.5)$	$(1,0,0,0.5)$	$(0.5,0,0.5,0.5)$	$(0,0,1,0.7500)$	0.75
P_6	$(0.99,0.01,0,0.5)$	$(0,0,1,0.5)$	$(1,0,0,0.5)$	$(0.9,0.1,0,0.5)$	$(0.5,0,0.5,0.5)$	$(1,0,0,0.5)$	$(0.6,0,0.4,0.5)$	$(0.2970,0,0.7030,0.7755)$	0.8422
P_7	$(0.99,0.01,0,0.5)$	$(0,0,1,0.5)$	$(1,0,0,0.5)$	$(0.9,0.1,0,0.5)$	$(0.83,0,0.17,0.5)$	$(1,0,0,0.5)$	$(1,0,0,0.5)$	$(0.8217,0,0.1783,0.8522)$	0.9736
P_8	$(1,0,0,0.5)$	$(0,0,1,0.5)$	$(1,0,0,0.5)$	$(0.9,0.1,0,0.5)$	$(0.5,0,0.5,0.5)$	$(1,0,0,0.5)$	$(1,0,0,0.5)$	$(0.5000,0,0.5000,0.8625)$	0.9313
P_9	$(1,0,0,0.5)$	$(1,0,0,0.5)$	$(1,0,0,0.5)$	$(0.95,0.05,0,0.5)$	$(0,0,1,0.5)$	$(0.99,0.01,0,0.5)$	$(0.96,0,0.04,0.5)$	$(0.9956,0,0.0044,0.7583)$	0.9989
P_{10}	$(0.99,0.01,0,0.5)$	$(0,0,1,0.5)$	$(1,0,0,0.5)$	$(0.9,0.1,0,0.5)$	$(0,0,1,0.5)$	$(0.8,0.2,0,0.5)$	$(0.98,0,0.02,0.5)$	$(0,0.0047,0.9953,0.7972)$	0.7934
P_{11}	$(0.99,0.01,0,0.5)$	$(0,0,1,0.5)$	$(1,0,0,0.5)$	$(0.95,0.05,0,0.5)$	$(0.72,0,0.28,0.5)$	$(0.99,0.01,0,0.5)$	$(0.96,0,0.04,0.5)$	$(0.6774,0.0001,0.3225,0.8489)$	0.9512
P_{12}	$(1,0,0,0.5)$	$(0,0,1,0.5)$	$(1,0,0,0.5)$	$(0.95,0.05,0,0.5)$	$(0.83,0,0.17,0.5)$	$(0.95,0.05,0,0.5)$	$(1,0,0,0.5)$	$(0.7885,0,0001,0.2114,0.8301)$	0.9640
P_{13}	$(1,0,0,0.5)$	$(0,0,1,0.5)$	$(1,0,0,0.5)$	$(0.95,0.05,0,0.5)$	$(0.83,0,0.17,0.5)$	$(0.95,0.05,0,0.5)$	$(0.83,0,0.17,0.5)$	$(0.6545,0.0001,0.3545,0.8110)$	0.9346
P_{14}	$(1,0,0,0.5)$	$(0,0,1,0.5)$	$(1,0,0,0.5)$	$(0.99,0.01,0,0.5)$	$(0.72,0,0.28,0.5)$	$(0.99,0.01,0,0.5)$	$(0.72,0,0.28,0.5)$	$(0.5132,0,0.4868,0.8186)$	0.9117
P_{15}	$(1,0,0,0.5)$	$(1,0,0,0.5)$	$(1,0,0,0.5)$	$(0.99,0.01,0,0.5)$	$(0.72,0,0.28,0.5)$	$(0.99,0.01,0,0.5)$	$(0.83,0,0.17,0.5)$	$(0.9870,0,0.0130,0.8492)$	0.9980
P_{16}	$(0.99,0.01,0,0.5)$	$(0,0,1,0.5)$	$(1,0,0,0.5)$	$(0.9,0.1,0,0.5)$	$(0.5,0,0.5,0.5)$	$(1,0,0,0.5)$	$(0.72,0,0.28,0.5)$	$(0.3564,0,0.6436,0.8011)$	0.8719
P_{17}	$(1,0,0,0.5)$	$(0,0,1,0.5)$	$(1,0,0,0.5)$	$(0.99,0.01,0,0.5)$	$(0.83,0,0.17,0.5)$	$(1,0,0,0.5)$	$(0.83,0,0.17,0.5)$	$(0.6889,0,0.3111,0.8447)$	0.9517
P_{18}	$(1,0,0,0.5)$	$(0,0,1,0.5)$	$(1,0,0,0.5)$	$(0.99,0.01,0,0.5)$	$(0.83,0,0.17,0.5)$	$(1,0,0,0.5)$	$(1,0,0,0.5)$	$(0.8300,0,0.1700,0.8737)$	0.9785
P_{19}	$(1,0,0,0.5)$	$(1,0,0,0.5)$	$(1,0,0,0.5)$	$(0.99,0.01,0,0.5)$	$(0,0,1,0.5)$	$(1,0,0,0.5)$	$(0.72,0,0.28,0.5)$	$(0.9196,0,0.0804,0.8525)$	0.9811
P_{20}	$(1,0,0,0.5)$	$(0,0,1,0.5)$	$(1,0,0,0.5)$	$(0.95,0.05,0,0.5)$	$(0,0,1,0.5)$	$(1,0,0,0.5)$	$(0.83,0,0.17,0.5)$	$(0,0,1,0.8836)$	0.8336

Table 7: Users' opinions in the system for the activity "a user access a file on the SVN" at LINA laboratory.

The OBASHI framework provides a tool for capturing, illustrating and modeling the relationships of dependency and the dataflows between business and IT environment in a business context. OBASHI does not have a specific metamodel to formalize the enterprise components, instead, it proposes a classification for the components which should be located in the layer that corresponds to their type. OBASHI has six layers: **O**wnership, **B**usiness processes, **A**pplications, **S**ystems, **H**ardware, and **I**nfrastructure. The OBASHI relationships describe the relations between the components, which follow the OBASHI rules. This model allows to create the business and IT diagrams (B&IT) and the dataflow, which are the main output of the OBASHI tool that helps the enterprise to develop its work and understand its needs. Despite its simplicity, OBASHI does not answer our needs. The B&IT Diagram and the data flow present a dependency graph that allows to find the sequences of the dependencies relations between the entities in an enterprise. In our work, the resulting model should represent an activity achieved through a system by a given user, more precisely, the model should contain the entities this user depends on to perform an activity and not the flow of dependencies between entities in a system.

In general, what mainly distinguishes SOCIOPATH from EA, is the social world that focuses on the persons who participate to the system. Instead of the social world, EA presents the business layer, which is mainly introduced by the component organization or organization unit. Hence, the analysis of the information in SOCIOPATH focuses on the needs of the person who uses a system including her social, digital and physical dependences. Whereas, in EA, the analysis of the information focuses on the needs of an enterprise including ameliorating its performance, choosing the best person for a particular task, *etc*.

The Business Process Modeling and Notation (BPMN) [11,33], is a standard to model business process mainly in the early phases of system development. To build diagrams, BPMN provides four categories of graphical elements. (1) Flow Objects, represent all the actions which can happen inside a business process determining its behavior. They consist of Events, Activities and Gateways. (2) Connecting Objects, provide three different ways of connecting various objects: Sequence Flow, Message Flow and Association. (3) Swimlanes, provides the capability of grouping modeling elements. Swimlanes have two elements through which modelers can group other elements: Pools and Lanes. And (4), Artifacts are used to provide additional information about Process that does not affect the flow. They are: Data Object, Group and Annotation. BPMN is a complete standard oriented to business users, business analysts, business staff and technical developers. For our specific needs, to define relations (of control, access, provides, *etc.*) among all the entities a (final) user depends on to achieve an activity, building our approach over BPMN is difficult because we have different focus and semantics. Mapping between this standard and our proposal may exist, but we have not investigated this direction.

5.2 Trust Evaluation

There are many approaches for evaluating trust in the literature [2,8,15,20,27] and several interesting surveys analyze them [7,16,22,37]. The approaches closest to our work

are those oriented to graphs [1,17,20,21,26,31]. They are especially used in social networks where the main idea of trust derivation is to propagate it between two nodes in a graph that represents the social network. A social network is a social structure composed of a set of persons (individuals or organizations) and a set of relations among these persons. It can be represented as a graph where the nodes are the persons and the edges are the relations between them. Trust between two persons in a social network can be evaluated based on this graph where the source node is the trustor, the target node is the trustee and the other nodes are the intermediate nodes between the trustor and the trustee. Values are associated to the edges to represent the trust value attributed by the edge source node towards the edge target node. To evaluate trust in a target node in a graph, in general, the following two steps are considered: *(1)* trust propagation through a path and *(2)* trust propagation through a graph employing different metrics and operators. Figure 8 (page 17) shows an example of trust relationships in a social network. For instance, B trusts C with the value 0.8.

Trust propagation focuses on finding a trust value from a person towards another given person through the multiple paths that relate them. For instance, in Figure 8, how much A trusts E, knowing that there are two paths that relate A with E, and that paths have nodes in common?

In graph-based trust approaches, this problem is either ignored [31], either simple solutions are proposed like choosing one path in a graph [26], or removing the paths that are considered unreliable [17,21]. In [21], Jøsang *et al.* propose a method based on graph simplification and trust derivation with subjective logic named, Trust Network Analysis with Subjective Logic (TNA-SL). They simplify a complex trust graph into a graph having independent paths by removing the dependent paths that have a high value of *uncertainty*. The problem of the previous solution is that removing paths from a graph could cause loss of information. To solve this problem, in another work [20] Jøsang *et al.*, propose to transform a graph that has dependent paths to a graph that has independent paths by duplicating the edges in common and splitting the associated opinions to them.

In this work we propose two approaches that deal with the problem of dependent paths. What differentiates our approaches from those of the literature is that we search to evaluate trust in a system as a whole for an activity and from the point of view of a person. In addition, we argue that the trust in a system depends on its architecture, more precisely, on the way the implicit and explicit entities, which the users depends on to do their activities, are organized.

Comparing trust approaches is hard, an approach is better if its produced trust values are lower (or higher) than another? Which is the reference to say what is a good trust value? That is why, in our experiments we make an effort to confront our proposed approaches to real users (*cf.* Section 3.4, page 23 and Section 4.3, page 33).

6 Conclusion and perspectives

Digital activities are achieved everyday by users through different systems. When users need to choose a system for a particular activity, they evaluate it considering many criteria like QoS, economical aspects, *etc.* This paper enlightens some aspects of digital

systems to improve users' expectations. The aspects we focused on are the user's digital and social dependences in a system for an activity, their degrees and the level of a user's trust towards the used system. To realize this approach, we fixed two main objectives:

1. Proposing a model that formalizes a system considering the different entities that compose it (physical, digital or social entities) and the relations between them.
2. Evaluating trust in a system for an activity based on this model.

In this paper, we proposed SOCIOPATH, a simple model that allows to formalize the entities in a system and the relations between them. In this contribution, we observed that the entities that compose a digital system can be digital, physical or human entities. We defined a model that formalizes all these entities and the relations between them. We provided this model with the rules that discover some implicit relations in a system and enriched it with definitions that illustrate some main concepts about the used system. SOCIOPATH allows to answer the user of some main questions that concern her used system.

Trust works in the literature focus on one granularity of trust; trusting a person, a product, a resource, *etc.* That reflects one entity in a used system. Trusting a system as a composition of a set of entities and relations between them has not been studied deeply.

By focusing on trust works existing in the literature, one direction drew our attention. This direction is trust propagation in social networks. This approach aims to propagate trust between two nodes in a graph that represents a social network. The propagated trust value results from combining trust values through this graph.

From SOCIOPATH models, we can obtain a directed acyclic graph (DAG) where nodes represent a set of entities that plays a role for achieving the users' activity and the set of edges represents the paths a user follows to achieve her activity.

Based on this DAG we proposed two approaches to evaluate trust in a system for an activity. The first one, SOCIOTRUST, is based on probability theory. It can be used in the field of full-knowledge environments. In presence of uncertainty, the second approach based on subjective logic, SUBJECTIVETRUST, is more suitable. The necessary relations and algorithms for combining the trust values towards the entities in the DAG have been provided and proved, and experiments have been conducted to validate these approaches.

All the evaluations of trust in a system we propose in this article are static. This is a limitation. To achieve a better comprehension of trust in a system and the parameters that can influence it, it will certainly be necessary to consider the evolution of trust over the time. We are convinced that such understanding is a a challenging issue. For this purpose, it is also necessary to compare synthetic trust and real trust of a user. Yet, to the best of our knowledge, there is no method to measure a distance or similarity between an assessment of confidence and the one felt by users. It is certainly possible to build on work already carried out in the fields of Information Retrieval or Social Sciences, but this is a problem we encountered without providing a complete answer. Indeed in our work, we collected users' impressions through a form and showed they feel closer to a proposal than the other. However, a general method of comparison and measurement between an assessment of the trust and the trust really felt remains to build.

It is also interesting to note that SOCIOPATH is not restricted to trust evaluation. Indeed, pointing out accesses and controls relations within an architecture is also related to privacy. Thus, as future work, it could be interesting to use SOCIOPATH to study the compliance of system with users privacy policies.

References

1. I. Agudo, C. Fernandez-Gago, and J. Lopez. A Model for Trust Metrics Analysis. In *Proceedings of the 5th International Conference on Trust, Privacy and Security in Digital Business (TrustBus)*, pages 28–37, 2008.
2. M. Al-Bakri, M. Atencia, and M.-C. Rousset. TrustMe, I Got What You Mean! - A Trust-Based Semantic P2P Bookmarking System. In *Proceedings of the 18th International Conference on Knowledge Engineering and Knowledge Management (EKAW)*, pages 442–445, 2012.
3. N. Alhadad. *Bridging the Gap between Social and Digital Worlds: System Modeling and Trust Evaluation*. PhD thesis, Université de Nantes, France, 2014.
4. N. Alhadad, Y. Busnel, P. Serrano-Alvarado, and P. Lamarre. Trust Evaluation of a System for an Activity with Subjective Logic. In *Proceedings of the 11th International Conference on Trust, Privacy, and Security in Digital Business (TrustBus)*, pages 48–59, Munich, Germany, 2014.
5. N. Alhadad, P. Lamarre, Y. Busnel, P. Serrano-Alvarado, M. Biazzini, and C. Sibertin-Blanc. SocioPath: Bridging the Gap between Digital and Social Worlds. In *Proceedings of the 23rd International Conference on Database and Expert Systems Applications (DEXA)*, pages 497–505, 2012.
6. N. Alhadad, P. Serrano-Alvarado, Y. Busnel, and P. Lamarre. Trust Evaluation of a System for an Activity. In *Proceedings of the 10th International Conference on Trust, Privacy & Security in Digital Business (TrustBus)*, pages 24–36, 2013.
7. D. Artz and Y. Gil. A Survey of Trust in Computer Science and the Semantic Web. *Web Semantic*, 5(2):58–71, 2007.
8. M. Atencia, M. Al-Bakri, and M.-C. Rousset. Trust in Networks of Ontologies and Alignments. *Knowledge and Information Systems*, 42(2):1–27, 2015.
9. C. Aurrecoechea, A. T. Campbell, and L. Hauw. A Survey of QoS Architectures. *Multimedia Systems*, 6(3):138–151, 1996.
10. P. Blau. *Exchange and Power in Social Life*. New York: John Wiley and Sons, 1964.
11. M. Chinosi and A. Trombetta. BPMN: An Introduction to the Standard. *Computer Standards & Interfaces*, 34(1):124–134, 2012.
12. R. M. Emerson. Power-Dependence Relations. *American Sociological Review*, 27(1):31–41, 1962.
13. D. Gambetta. *Trust: Making and breaking cooperative relations*, volume 13, chapter Can we Trust Trust, pages 213–237. Department of Sociology, University of Oxford, 2000.
14. R. E. Giachetti. *Design of Enterprise Systems: Theory, Architecture, and Methods*. CRC Press, Boca Raton Florida, 2010.
15. J. Golbeck. *Computing and Applying Trust in Web-based Social Networks*. PhD thesis, Department of Computer Science, University of Maryland, 2005.
16. J. Golbeck. Trust on the World Wide Web: a Survey. *Foundations and Trends in Web Science*, 1(2):131–197, 2006.
17. J. Golbeck and J. A. Hendler. Inferring Binary Trust Relationships in Web-Based Social Networks. *ACM Transactions on Internet Technology*, 6(4):497–529, 2006.
18. R. Harrison. *TOGAF Version 8.1*. Van Haren Publishing, New York, 2007.

19. A. Jøsang. A Logic for Uncertain Probabilities. *Uncertainty, Fuzziness and Knowledge-Based Systems*, 9(3):279–311, 2001.
20. A. Jøsang and T. Bhuiyan. Optimal Trust Network Analysis with Subjective Logic. In *Proceeding of the 2nd International Conference on Emerging Security Information, Systems and Technologies (SECURWARE)*, pages 179–184, 2008.
21. A. Jøsang, R. Hayward, and S. Pope. Trust Network Analysis with Subjective Logic. In *Proceedings of the 29th Australasian Computer Science Conference (ACSC)*, pages 85–94, 2006.
22. A. Jøsang, R. Ismail, and C. Boyd. A Survey of Trust and Reputation Systems for Online Service Provision. *Decision Support Systems*, 43(2):618–644, 2007.
23. A. Josey. *TOGAF Version 9: A Pocket Guide*. Van Haren Publishing, 2 edition, 2009.
24. L. Li and Y. Wang. Subjective Trust Inference in Composite Services. In *Proceedings of the 24th Conference on Artificial Intelligence (AAAI)*, 2010.
25. L. Li and Y. Wang. A Subjective Probability Based Deductive Approach to Global Trust Evaluation in Composite Services. In *Proceedings of the 9th IEEE International Conference on Web Services (ICWS)*, pages 604–611, 2011.
26. G. Liu, Y. Wang, M. Orgun, and E. Lim. Finding the Optimal Social Trust Path for the Selection of Trustworthy Service Providers in Complex Social Networks. *IEEE Transactions on Services Computing*, 6(2):152–167, 2011.
27. S. P. Marsh. *Formalising Trust as a Computational Concept*. PhD thesis, Department of Mathematics and Computer Science, University of Stirling, 1994.
28. D. H. Mcknight and N. L. Chervany. The Meanings of Trust. Technical report, University of Minnesota, Carlson School of Management, 1996.
29. L. Molm. *Structure, Action, and Outcomes: The Dynamics of Power in Social Exchange*. American Sociological Association Edition, 1990.
30. F. Moyano, M. C. Fernández-Gago, and J. Lopez. A Conceptual Framework for Trust Models. In *Proceedings of the 9th International Conference on Trust, Privacy and Security in Digital Business (TrustBus)*, pages 93–104, 2012.
31. M. Richardson, R. Agrawal, and P. Domingos. Trust Management for the Semantic Web. In *Proceedings of the 2nd International Semantic Web Conference (ISWC)*, pages 351–368, 2003.
32. M. Rohloff. Enterprise Architecture-Framework and Methodology for the Design of Architectures in the Large. In *Proceedings of the 13th European Conference on Information Systems (ECIS)*, pages 1659–1672, 2005.
33. The official BMPN Website. http://www.bpmn.org/, Last accessed May 2015.
34. The official OBASHI Website. http://www.obashi.co.uk/, Last accessed May 2015.
35. L. Viljanen. Towards an Ontology of Trust. In *Proceedings of the 2nd International Conference on Trust, Privacy and Security in Digital Business (TrustBus)*, 2005.
36. Z. Yan and S. Holtmanns. *Computer Security, Privacy and Politics: Current Issues, Challenges and Solutions*, chapter Trust Modeling and Management: from Social Trust to Digital Trust. IGI Global, 2007.
37. P. Zhang, A. Durresi, and L. Barolli. Survey of Trust Management on Various Networks. In *Proceedings of the 5th International Conference on Complex, Intelligent and Software Intensive Systems (CISIS)*, pages 219–226, 2011.