Open and Interoperable Socio-technical Networks

Andrei Ciortea
École Nationale Supérieure des Mines
FAYOL-ENSMSE, LSTI, F-42023
Saint-Étienne, France
University Politehnica of Bucharest
313 Splaiul Independenţei
Bucharest, Romania
Email: andrei.ciortea@emse.fr

Olivier Boissier
and Antoine Zimmermann
École Nationale Supérieure des Mines
FAYOL-ENSMSE, LSTI, F-42023
Saint-Étienne, France
Email: olivier.boissier@emse.fr
Email: antoine.zimmermann@emse.fr

Adina Magda Florea
University Politehnica of Bucharest
313 Splaiul Independenţei
Bucharest, Romania
Email: adina.florea@cs.pub.ro

***Abstract*—Developing applications across the physical-digital space requires the homogeneous interconnection of people, physical devices, services and various data sources as first-class entities of complex socio-technical systems. In this paper, we describe socio-technical networks (STNs) as the building blocks of a semantic, open and distributed Social Web of Things. We address the problem of enabling autonomous non-human agents as participants in an open set of STNs. Our approach is to provide agents with machine-readable descriptions of STNs, of operations required for participating in such systems, and of supported implementations for those operations. Towards this aim, we present the STN ontology and we illustrate its applicability. Even though the STN ontology is a work in progress, using the core concepts and properties described in this paper we are able to create concrete specifications of STN platforms. We discuss the positioning of this ontology with several well-known and related vocabularies.**

I. Introduction

Interconnecting people, heterogeneous physical devices, services and various data sources is one of the main challenges that need to be addressed in order to realize the full potential of the Internet of Things (IoT) vision. The Web of Things (WoT) paradigm tackles the problem of application-level interoperability by employing the World Wide Web as an application architecture [1]. Thus, reusing widely accepted Web standards and protocols, the WoT aims at creating a widely distributed platform in which RESTful physical devices provide services that can be easily composed and integrated in Web applications [2]. The WoT thus lowers the entry barrier for developing IoT applications, enabling Web developers to build applications that extend to the physical world, across a large set of heterogeneous physical devices able to perceive and possibly act upon their environment.

At this point, however, the problem is half-solved. Composing and searching for services in a Web-scale ecosystem is a challenging task. An equal challenge imposed to the everyday user is managing and interacting with a large number of heterogeneous interconnected physical devices. In a previous publication, we have presented our vision for a Social Web of Things [3], in which autonomous and proactive things are enabled as first-class entities of socio-technical networks (STNs). An STN encodes connections among people and things, and it evolves over time through the actions of its autonomous citizens. The goal is to use the information embedded in its topology for improving the discovery of services, the dissemination or retrieval of information. Such networks have also been identified as effective uniform interfaces for managing a large number of heterogeneous interconnected products and services[1].

Socio-technical networks, as defined in our vision, are not an entirely new concept. Even though they may or may not be explicitly modeled as networks, similar systems already appear in commercial products and services offered by early stage IoT startups[2][3]. In fact, some of the existing social networking platforms, such as Twitter, even though not originally designed for non-human users, are also not limited to human users[4]. Nevertheless, most of these systems are not open, nor interoperable. In order to achieve a seamless integration of the physical and digital worlds, and to be able to fully exploit the potential of the IoT, it seems crucial for application developers not to be limited to one STN or the other, but to be able to develop applications on a Web-scale infrastructure.

The motivation of our work is to build a semantic, open and distributed Social Web of Things (SWoT). Our approach is to provide smart things with machine-readable descriptions of STNs, of the operations they may perform in order to participate in such networks, and of the supported implementations for these operations. In this paper, we present the STN ontology, which provides the core concepts and properties required to describe open and interoperable STNs. Using this ontology, developers can build specifications of STN platforms, making them accessible to autonomous non-human entities.

The STN ontology is a work in progress. Modules that are currently defined address the challenge of enabling smart things to form and function across multiple STNs. Other challenges remain to be addressed, many of which are related to the normative dimension of an STN and enabling control over the autonomy of smart things. Some of these aspects are emphasized in the discussion of the illustration scenario presented in Section IV.

This paper is structured as follows. Section II describes in further detail STNs and a layered model for the SWoT. Section III presents the STN ontology, which is the foundation

[1] http://www.ericsson.com/uxblog/2012/04/a-social-web-of-things/
[2] http://www.thingworx.com
[3] http://www.smartthings.com
[4] https://dev.twitter.com/docs/platform-objects/users

of our approach. Section IV illustrates the core elements defined in this ontology through a concrete specification of an STN platform. Before concluding, section V discusses the alignment of the STN ontology with several well-known related vocabularies.

II. Socio-technical Networks

An STN encodes the different relations among people and smart things, such as friendship, ownership, provenance or colocation. For instance, all appliances in a house, together with its inhabitants, may form an STN. Similarly, a world-wide network of environmental sensors may also form a widely distributed STN. An STN spans across the physical-digital space. It mirrors the physical world by creating digital counterparts of physical and abstract entities and relating them to one another. In the same time, applications running in the digital world may reflect back into the physical world (e.g. through actuators).

Situating an STN in the physical world requires positioning the STN in the associated spacetime continuum. The structure of an STN is given by a labeled directed multigraph, which we call the socio-technical graph (STG). An STG is an instance of an STN at a given point in time, and the evolution of an STN may be represented through a succession of STGs. In the same time, an STG embeds the spatial concepts and relationships (if any), such as places and localization, required for building the spatial dimension of its STN.

An STN may not exist outside of the minds of its members, and therefore each member may have his own representation of the STG. However, in most cases, the STG is represented explicitly in the digital world and the STN is reified within an STN platform, i.e. a system that provides a collection of features enabling users to participate in an STN. Common features include creating and using user accounts, publishing social media content or interacting with other platform users. A platform may be owned and controlled by a single hosting party, such as Facebook or Twitter, or it may be owned and controlled locally by several hosting parties. This notion of STN platform follows the definition for social platforms given by the W3C Social Web Incubator Group [4].

A. A layered model for the Social Web of Things (SWoT)

Applying the principle of separation of concerns, we use the layered model depicted in Figure 1 for exploring in further detail STNs and our architecture for the SWoT. Our approach is built on top of the WoT, which we illustrate through an abstraction layer that offers a RESTful API for the WoT. This is the entry point in our model.

1) Agency Layer: The first layer abstracts from individual persons and smart things by modeling all autonomous entities in an STN as agents. An STN is agnostic to the internal model or state of its agents. The agentification process should abide to the REST architectural style, and thus preserve the properties of the overall system, such as scalability of interactions, generality of interfaces and evolvability of independent components. An agent may be part of several STNs, therefore another concern addressed at this layer is designing identifiable agents, for instance by using a universal identification mechanism, such as WebID [5] or OpenID Connect [6].

Fig. 1. A layered model for the SWoT.

2) Social Layer: At this layer, individual agents become connected in social structures. Such structures may be modeled explicitly as groups, or they may emerge from individual connections created among agents. In addition to modeling social structures in STNs, another concern addressed at this layer is decoupling agents from the STN platforms they use by providing a uniform interface that allows them to function across an open set of STN platforms.

3) Normative Layer: As we arrive to the third layer, we now deal with a set of autonomous agents, possibly connected in different social structures. The main concern addressed at this layer is enabling control over the behavior of agents, such as participation in social structures, dissemination of information or execution of actions in the physical world.

4) Application Layer: The Application layer is concerned with the business logic of applications, but also with providing mechanisms and protocols for managing services, content and other resources.

The work presented in this paper is mostly focused around the Social layer. We have started our investigation by looking at existing social networking platforms with high user adoption, widely accepted standards, recommendations and ontologies, to the aim of extracting the commonality that would allow us to build a generic model for STNs.

III. The STN Ontology

Based on the aforementioned commonality, the STN ontology[5] provides the foundation for building semantic, open and distributed STNs. Our aim is to provide autonomous non-human agents with machine-readable descriptions of STNs, which they may use to reason upon, and to facilitate their participation in such systems. Developers may use the STN ontology both for describing existing STNs and for guiding the design of new STNs.

The design principles that guided the development of the STN ontology are generality, simplicity and separation of concerns. The STN ontology has a modular design and is intended to be extensible. The ontology currently defines 3 modules: *STN-Core*, which provides the core concepts and properties required for describing STNs, *STN-Operations*, which describes the core operations that may be performed by an agent in an STN, and *STN-Operations-HTTP*, which

[5]The STN ontology is a work in progress.

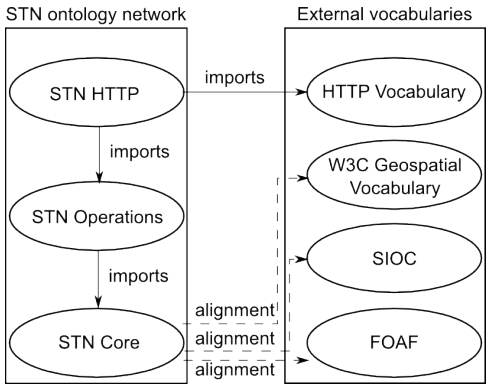

Fig. 2. The STN Ontology.

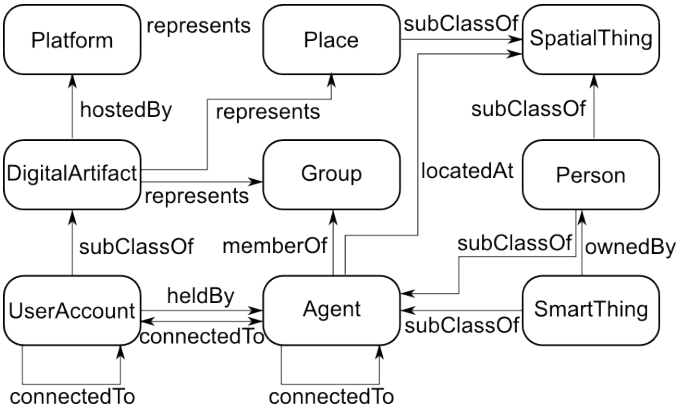

Fig. 3. An overview of STN-Core.

provides the concepts and properties required for describing HTTP-based implementations of STN operations.

The STN ontology is aligned with several widely accepted vocabularies, as show in Figure 2. The alignments are further discussed in Section V.

We have chosen to formalize the STN ontology in OWL 2. In addition to being a W3C recommendation, OWL 2 is built on top of RDF, which is a format suitable for representing widely distributed graphs.

It is worth mentioning that even though the work presented in this paper is focused around building STNs for the WoT, the STN ontology has a modular design that may be extended to model STNs outside of the Web.

Throughout this paper, we are to use the prefix bindings presented in Table I.[6]

TABLE I. PREFIX BINDINGS.

Prefix	URI
stn	http://purl.org/stn/core#
stn-ops	htttp://purl.org/stn/core/operations#
stn-http	htttp://purl.org/stn/core/operations/http#

A. STN-Core

STN-Core provides the core concepts and properties required for describing an STN. A simplified overview of this module is given in Figure 3.

1) Agents, digital artifacts and user accounts: The class `stn:Agent` includes all autonomous entities that interact in the physical-digital space, most commonly persons, organizations, physical or virtual objects. We distinguish between two disjoint sub-classes of agents: `stn:Person`, which represents people, and `stn:SmartThing`. A smart thing is an agent that most commonly embeds the logics of a given physical or virtual object, although it may also represent a composite application, e.g. one built on top of several other smart things.

STN-Core follows the Agents & Artifacts meta-model [7], and defines `stn:DigitalArtifact` as the class of virtual tools used by agents to participate in STNs. A digital artifact may be a digital counterpart of an abstract or physical entity[7], such as a digital counterpart of a physical place, or it may exist independently, such as an `stn:Message`. Digital artifacts exist in the digital world and are hosted on STN platforms.

An agent may participate in an STN either directly, or through the use of a user account. An `stn:UserAccount` is a digital artifact that represents the digital counterpart of an agent. A user account is related to an agent through the `stn:heldBy` property, and the entity acting through that user account is assumed to have been delegated by and acting for the agent. An agent may hold multiple, possibly anonymous, user accounts on different STN platforms. Through introducing user accounts, we keep a separation between an agent and its projection within the scope of an STN. This distinction is essential when an agent does not have undisputed control over its participation in an STN. For instance, in an STN hosted by a third party, an agent's actions may be under the control of the agent, the hosting party and any party that might be able to impersonate the agent.

2) The social structure: The building blocks of an STN's social dimension are *groups* and *connections*.

The class `stn:Group` is the class of groups, defined as collections of agents with an arbitrary number of members. A group is an abstract entity that may be associated with one or more digital artifacts. For instance, the AI-MAS research team of University "Politehnica" of Bucharest may be associated with a Facebook group and with a LinkedIn group, which are its digital counterparts. Groups may conform to a specification given at the Normative layer (cf. Section II-A) and may be governed by a set of norms, however this is out of the scope of this paper.

The `stn:connectedTo` property represents a unidirectional connection that relates agents, user accounts and groups to one another. A connection gives the smallest possible social construct in an STN (i.e. a dyad). As connections evolve, they form networks, which may embed useful information in their emergent topologies. For instance, agents may be incentivized to build connections with similar others: smart things owned by the same person may be incentivized to connect with one

[6]The OWL 2 specifications of the three modules are available online at the URIs given in Table I.

[7]An entity may have several digital counterparts, and a digital counterpart may have several digital representations.

another towards the aim of enhancing service discovery in home automation scenarios.

The stn:ownedBy property is a sub-property of stn:connectedTo and relates a smart thing to its owner (e.g. an agent, a group). Given that ownership is a broad concept, no formal restrictions are added on the range of this property. Being able to associate an autonomous smart thing with an accountable owner seems sensible, and any smart thing should ultimately trace back to a natural or juridical entity, such as a person or an organization. For instance, limiting the number of registered smart things per accountable owner might be used for a controlled dynamic registration of smart things.

3) The spatial dimension: STN-Core provides the minimal set of elements required for anchoring an STN in the physical space.

The class stn:SpatialThing is the class of entities with spatial characteristics. stn:Person is a kind of spatial thing, as it is the case, for instance, when we say that a smart wristband is on someone's wrist. stn:Place is another kind of spatial thing, such as geographical regions or cities.

Anything may be related to an stn:SpatialThing through the stn:locatedAt property, which represents a generic localization relation. This relation is intended to be extended through other vocabularies with more versatile spatial relations.

STN-Core separates a spatial thing from the geometric representation of its spatial characteristics (e.g. points, lines, polygons), therefore aligning itself with previous work done in this area [8]. However, STN-Core is limited to describing and positioning spatial things with respect to other entities in an STN. Extending the spatial dimension with geometric representations and spatial relations, either through reusing a specialized vocabulary or through introducing a new module, is left as future work. We discuss this further in Section V.

B. STN-Operations

It is desirable to enable smart things to access and participate in an open set of STNs. STN-Operations addresses this problem by defining a set of generic operations and parameters that allow agents to participate in any STN compatible with STN-Core, therefore decoupling agents from the STN platforms they use.

The class stn-ops:Operation is the class of all operations, such as actions or queries, that may be performed by an agent within the scope of an STN platform. An STN platform may support a subset of the operations defined in this vocabulary.

Each operation may be related to several parameters through an stn-ops:hasParam property. A parameter may be required or optional. stn-ops:Parameter is the class of all parameters. For instance, stn-ops:DisplayedName is a kind of parameter which denotes the name displayed for an agent in an STN. Each STN platform might have a different representation or a different name for a parameter in this class, however the semantics remain the same.

An stn-ops:Action is an operation that affects the STG and thus defines a transition from one state of an STN to another. For instance, creating a connection to another agent would have the effect of adding an edge to the STG. This action requires, in addition to the performer of the action, an stn-ops:UserID as another parameter.

An stn-ops:Query is an operation that only retrieves information from the STG. For instance, getting the connections of a user retrieves a list of all the agents, user accounts or groups to which that queried user is connected. Running this query would also require an stn-ops:UserID parameter (i.e. the targeted user).

C. STN-Operations-HTTP

Our objective is to build open and interoperable STNs. STN-Core and STN-Operations deal with interoperability at a conceptual level by providing a least common denominator for STNs. STN-Operations-HTTP deals with interoperability at the implementation level, and provides the concepts and properties required for describing HTTP-based implementations of STN operations.

To achieve our objective, autonomous smart things must be able to function across the Web APIs exposed by an open set of STN platforms. If an STN platform is Web-compliant and follows a resource-oriented architecture, enabling interoperability at this level implies agreement on the format of resource identifiers, the format of resource representations, the semantics of resource operations (e.g. GET, PUT, POST, DELETE), which requests support a payload and in what format, and what is the format of the returned responses. There are two approaches for tackling this challenge: either through defining a standard API that STN platforms may then implement, or through providing translations from a core model to the API offered by each STN platform. The former approach requires STN platforms to implement the standard API, and thus it requires strong community support[8]. In this paper, we take the latter approach.

The STN-Operations-HTTP module defines the concepts and properties required to translate STN operations to HTTP requests. Using this vocabulary, developers can create a specification that maps a set of supported operations to the API of a given platform. This approach thus provides an instruction manual for non-human agents and does not impact an existing API in any way. This module imports the HTTP vocabulary [10] and defines stn-http:STNRequest as a kind of http:Request[9]. In addition to the expressivity already offered by the HTTP vocabulary, it also adds support for specifying a security protocol for the authentication and authorization of HTTP requests.

IV. Illustration

John wants to connect his Web-enabled smart TV to his favorite STN platform, ThingsNet[10]. The smart TV retrieves ThingsNet's STN description document, which, sim-

[8]For instance, OpenSocial [9] follows a similar approach to build applications that run on multiple social networking platforms.

[9]The prefix http is used to denote the namespace http://www.w3.org/2011/http#

[10]The name ThingsNet, as well as any URIs used in this section, are fictitious. Any similarities with existing platforms or services is merely coincidental.

ilar to the Robots Exclusion Protocol[11], is located in an /stnspec.ttl file. The description uses the STN ontology to specify the main properties of the platform, such as the base URL of its API, the supported authentication protocol, and the implemented STN operations:

```
<http://www.thingsnet.com/#>
   a stn:Platform ;
   stn-http:baseURL
         <https://api.thingsnet.com/0.1/> ;
   stn-http:supportsAuth stn-http:WebID ;
   stn-ops:supports <#createUserAccount> ,
            <#createConnection> ,
            (...) ,
            <#getUserConnections> .
```

ThingsNet is an open platform, which means that some of its operations might be available to unregistered agents, who may be hosted on different STN platforms. Such operations include, for instance, retrieving public data or creating cross-platform connections. ThingsNet uses WebID for uniquely identifying agents.

In order to have access to the full set of operations, an agent must dynamically register on ThingsNet by performing a CreateUserAccount action:

```
<#createUserAccount>
   a stn-ops:CreateUserAccount ;
   stn-ops:implementedAs
    [ a stn-http:STNRequest ;
      http:methodName "POST" ;
      http:requestURI </users/create> ;
      http:headers
         ( [ a http:RequestHeader
         http:fieldName "Content-Type"
         http:fieldValue "application/json"
         ]
      ) ;
      stn-ops:hasInput
         [ a stn-ops:OwnerID ;
         stn-ops:paramName "owner_id";
         stn-ops:required true ;
         ] ;
      stn-ops:hasInput
         [ a stn-ops:MyDisplayedName ;
         stn-ops:paramName "displayedName" ;
         stn-ops:required true ;
         ] ;
      stn-ops:hasInput
         [ a stn-ops:MyDescription ;
         stn-ops:paramName "description" ;
         ] ;
      ] .
```

Following the given operation specification, the smart TV is able to register on ThingsNet by sending a POST request to https://api.thingsnet.com/0.1/users/create. The JSON payload of this request includes the two required parameters: owner_id, which is John's WebID, and displayed_name, a preconfigured name to be displayed for this user (e.g. "John's smart TV"). The WebID of the smart TV itself, i.e. the performer of the action, is automatically extracted from the certificate used for authentication [5]. The platform returns a 201 Created response and the URI of the newly created user account hostedBy ThingsNet. An ownedBy relation is added between the smart TV and John. After registration, the smart TV has access to all the operations supported by the ThingsNet platform.

One of the supported queries returns the connections of a given user. Implementing an operation specification similar to the one presented above, the smart TV sends a GET request to <baseURL>/connections/ids with the required user_id parameter set to John's WebID. The platform returns a 200 OK response and a list of John's connections.

A behavior programmed in the smart TV is to connect to all smart things having the same owner. The list of connections returned for John contains several other smart things he owns. The smart TV connects to each of them by implementing the <#createConnection> action through the appropriate HTTP request given in the specification.

Some of John's connections are pointing to user accounts hosted on different STN platforms. In order to lookup information about or interact with those agents, John's smart TV will have to retrieve and implement the operation specifications of the hosting platform. The required steps are similar with the ones described at the beginning of this section, the difference being that the smart TV should probably not register on any other platform without John's permission.

John's smart TV can record usage information, such as what movie is currently running or the recent movie history (with ratings, if any are available). Crawling open and interoperable STN platforms, the smart TV can search for other TVs that are sharing similar information and are stn:ownedBy John's friends. Using this aggregated information, the smart TV can augment its movie recommendations with results based on what movies are John's friends currently watching, what movies they have watched recently or what movies are trending among his friends.

The smart TV can function autonomously across multiple STNs: the STN ontology provides usable descriptions of STGs, a specification of the semantics of STN operations and parameters (e.g. CreateUserAccount and DisplayedName), and a mechanism for implementing these operations as HTTP requests. However, this illustration scenario leaves important challenges to be addressed.

For instance, John can configure his smart TV such that it will not register on any platforms other than ThingsNet or that it will share information about his movies only with a selected number of close friends. However, if John owns a large number of heterogeneous smart things, it would be cumbersome, if not impossible, to keep track with all of them, especially as they crawl from STN to STN. It would be desirable to have a mechanism through which users can express preferences and policies to be taken into account by the concerned agents.

Similarly, STN platforms would need to make their terms of service available and machine-readable. Common rules would include API rate limits, however other limitations may

[11] http://www.robotstxt.org

also be made explicit in an STN, such as a maximum number of friends or a maximum storage quota. One could easily imagine how autonomous non-human entities may end up consuming a lot of resources rather quickly.

Lastly, it would be desirable to have coordination mechanisms that allow developers to build complex applications in the physical-digital space.

These challenges, among others, emphasis the need for enabling control over the autonomy of agents and are to be addressed at the Normative layer in future work.

V. Related Work

The STN ontology is aligned with several well-known vocabularies, which we discuss in this section.

A. FOAF

FOAF [11] is a widely accepted vocabulary for describing characteristics of people, social groups and relations among them. FOAF defines, among others, the concepts of agent, person and group.

In FOAF, agents are defined as "things that do stuff". STN-Core defines agents as "autonomous entities that interact in the physical-digital space". We consider `stn:Agent` and `foaf:Agent`[12] to be equivalent classes. Both vocabularies define persons, i.e. the class of all people, as agents and spatial things (cf. Section V-B). Thus, we consider classes `stn:Person` and `foaf:Person` to be equivalent as well. Similarly, both vocabularies define groups as collections of individual agents. However, in FOAF, `foaf:Group` is a sub-class of `foaf:Agent`, while in STN-Core not all groups are agents. Therefore, the notion of group in STN-Core is more general than the one defined in FOAF.

STN-Core defines the `stn:connectedTo` property as a generic connection, without requiring any reciprocity. FOAF defines the `foaf:knows` property, which "relates a `foaf:Person` to another `foaf:Person` that he or she knows" and requires "some form of reciprocated interaction". Having the `foaf:knows` property implies being a `foaf:Person`. We consider `foaf:knows` to be a specific type of connection among agents, and can thus be used in STNs alongside `stn:connectedTo` or other types of connections.

The `foaf:based_near` property[13] relates a `geo:SpatialThing`[14] to another `geo:SpatialThing`. The `stn:locatedAt` property relates anything to an `stn:SpatialThing`. We consider `stn:SpatialThing` to be equivalent to `geo:SpatialThing` (cf. Section V-B), and thus `stn:locatedAt` is more general than `foaf:based_near`.

Given this alignment between STN-Core and FOAF, all social networks of people described in FOAF can be easily translated to STNs described in the STN ontology.

[12]The prefix `foaf` is used to denote the namespace `http://xmlns.com/foaf/0.1/`

[13]At the moment of writing this paper, the status of `foaf:based_near` is "testing".

[14]The prefix `geo` is used to denote the namespace `http://www.w3.org/2003/01/geo/wgs84_pos#`

B. W3C Geospatial Vocabulary

The W3C 2003 Geo vocabulary provides a namespace for representing geospatial information about things. This vocabulary defines a spatial thing as "anything with spatial extent, i.e. size, shape, or position". STN-Core defines spatial things as "entities with spatial characteristics". We consider `stn:SpatialThing` and `geo:SpatialThing` to be equivalent classes. The 2003 Geo vocabulary defines a property `geo:location` which relates anything to a `geo:SpatialThing`. We consider `stn:locatedAt` equivalent to this property.

The W3C 2007 Geospatial Vocabulary is a complete update of the 2003 Geo vocabulary and supersedes it, maintaining backward compatibility [8]. The 2007 vocabulary introduces a model for basic feature properties of Web resources. Any appropriate content is characterized as a geographic feature which may have an associated geometric representation (e.g. point, line, polygon).

As discussed in Section III-A, STN-Core is limited to characterizing entities as spatially-located things, and may be used in conjunction with geospatial vocabularies, such as the ones described in this section.

C. SIOC Core Ontology

The SIOC (Semantically-Interlinked Online Communities) Core Ontology describes online communities [12]. Many terms in SIOC are defined with reference to FOAF. For instance, a `foaf:Agent` may hold a `sioc:UserAccount`[15], which may be used to create resources, such as a `sioc:Post` (e.g. articles, messages). All posts are items which may be part of a container stored in a space. For instance, a `sioc:Forum` is a `sioc:Container` hosted on a `sioc:Site`, which is a `sioc:Space`.

In STN-Core, a digital artifact is any non-autonomous entity that exists in the digital world and resides on a platform. Therefore, the notion of digital artifact is more general than the notions of item and container, as defined in SIOC, and a `sioc:Item` or a `sioc:Container` could be a kind of `stn:DigitalArtifact`. We consider an `stn:Platform` to be a kind of `sioc:Space`. Therefore, the property `stn:hostedBy`, which relates digital artifacts to platforms, is more specific than `sioc:has_space`, which relates anything to a space.

A sub-class of `stn:DigitalArtifact` is `stn:UserAccount`. STN-Core defines a user account as "the digital counterpart of an agent within the scope of an STN platform". SIOC defines a user account as "an online account of a member of an online community". Therefore, we consider an `stn:UserAccount` to be a kind of `sioc:UserAccount`, but which is also an `stn:DigitalArtifact`, i.e. it exists in the digital world and is resides on an `stn:Platform`.

The only digital content item created by agents that is currently described in STN-Core is `stn:Message`, a kind of

[15]The prefix `sioc` is used to denote the namespace `http://rdfs.org/sioc/ns#`

`stn:DigitalArtifact`. An additional module for modeling social media content is foreseen to be introduced in the STN ontology.

D. OpenSocial

The vocabularies presented so far describe people and networks of people, geospatial characteristics of Web resources and information in online communities. While these vocabularies are more closely related to STN-Core and describing STNs, OpenSocial is more closely related to the STN-Operations and STN-Operations-HTTP modules, and enabling smart things to function in different STNs.

OpenSocial provides a common set of APIs that websites can implement, allowing developers standard access to social information [13]. The initiative was launched by Google, together with MySpace and other social networking platforms, and has a strong industry support.

Defining a standard API for building social applications on the Web requires a significant community effort. We take a different approach. The STN-Operations-HTTP vocabulary describes HTTP-based implementations of the core STN operations. Using this vocabulary, developers can translate STN operations to the API of a given platform. Therefore, this approach does not necessarily require platforms to buy into the STN ontology and has no impact on an existing API. The drawback, however, is that the expressivity of the STN ontology might be limited for the requirements of platforms that are completely agnostic to this ontology. Given that the STN ontology is a work in progress, at the moment of writing this paper it is premature to make an objective evaluation of such aspects.

VI. Conclusion and Future Work

We have addressed the challenge of enabling autonomous non-human agents to function across an open set of STNs. Our approach is to create machine-readable descriptions of STNs and of the operations that may be performed for participating in such systems. Towards this aim, we have presented the STN ontology. While still a work in progress, the modules presented in this paper provide the concepts and properties required for describing (i) a core instance of an STN, (ii) what are the core operations that can be performed in an STN, and (iii) how can these operations be implemented on the Web. Therefore, the core elements necessary for designing and implementing open and interoperable STNs are defined in these modules. We have illustrated their applicability by describing a concrete specification of an STN platform, and we have discussed some of the important challenges that remain to be addressed.

We have presented the alignment of the STN ontology with a few widely accepted vocabularies, such as FOAF. This alignment enables smart things to access a wider spectrum of STNs. It remains as future work to enlarge this spectrum by creating specifications for widely accepted common APIs or social networking platforms, such as OpenSocial or Twitter.

Enriching the concepts and operations that may be described through this ontology remains as future work. Several additional modules are currently foreseen to be introduced towards this end, such as STN-Media and STN-Media-Operations, for modeling and acting on social media items, and STN-Norms and STN-Norms-Operations, for modeling the normative dimension of STNs.

Acknowledgment

The work has been funded by the Sectoral Operational Programme Human Resources Development 2007-2013 of the Ministry of European Funds through the Financial Agreement POSDRU/159/1.5/S/132397.

References

[1] D. Guinard, "A web of things application architecture: Integrating the real-world into the web," Ph.D. dissertation, ETH Zurich, 2011.

[2] D. Guinard, I. Ion, and S. Mayer, "In search of an internet of things service architecture: Rest or ws-*? a developers perspective," in *Mobile and Ubiquitous Systems: Computing, Networking, and Services*. Springer, 2012, pp. 326–337.

[3] A. Ciortea, O. Boissier, A. Zimmermann, and A. M. Florea, "Reconsidering the social web of things: position paper," in *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication*. ACM, 2013, pp. 1535–1544.

[4] H. Halpin and M. Tuffield, "A Standards-based, Open and Privacy-Aware Social Web," http://www.w3.org/2005/Incubator/socialweb/XGR-socialweb-20101206/, December 2010, accessed: 2014-03-24.

[5] A. Sambra and S. Corlosquet, "WebID 1.0, Web Identity and Discovery," https://dvcs.w3.org/hg/WebID/raw-file/tip/spec/identity-respec.html, accessed: 2014-03-24.

[6] N. Sakimura, J. Bradley, M. Jones, B. de Medeiros, and C. Mortimore, "OpenID Connect Core 1.0," http://www.openid.net/specs/openid-connect-core-1_0.html, February 2014, accessed: 2014-03-24.

[7] A. Omicini, A. Ricci, and M. Viroli, "Artifacts in the a&a meta-model for multi-agent systems," *Autonomous agents and multi-agent systems*, vol. 17, no. 3, pp. 432–456, 2008.

[8] J. Lieberman, R. Singh, and C. Goad, "W3C Geospatial Vocabulary," http://www.w3.org/2005/Incubator/geo/XGR-geo/, October 2007, accessed: 2014-03-24.

[9] The OpenSocial Foundation, "OpenSocial Core API Server Specification 2.5.1," http://opensocial.github.io/spec/2.5.1/Core-API-Server.xml, August 2013, accessed: 2014-03-24.

[10] J. Koch, C. A Velasco, and P. Ackermann, "HTTP Vocabulary in RDF 1.0," http://www.w3.org/TR/HTTP-in-RDF10/, May 2011, accessed: 2014-03-24.

[11] D. Brickley and L. Miller, "FOAF vocabulary specification 0.99," http://xmlns.com/foaf/spec/, January 2014, accessed: 2014-03-24.

[12] U. Bojars and J. G. Breslin, "SIOC Core Ontology Specification," http://rdfs.org/sioc/spec/, March 2010, accessed: 2014-03-24.

[13] The OpenSocial Foundation, "OpenSocial Specification 2.5.1," http://opensocial.github.io/spec/2.5.1/OpenSocial-Specification.xml, August 2013, accessed: 2014-03-24.