Verification-Enabling Interaction Model for Services in Smart Space: a TAIS Case

Andrew Ponomarev[*][†], Vladimir Parfenov[†]
[*]SPIIRAS, St.Petersburg, Russia
[†]ITMO University, St.Petersburg, Russia
ponomarev@iias.spb.su, parfenov@mail.ifmo.ru

***Abstract*—Smart spaces are used to build semantically enriched communication media for integration of different services. Generally, the services functioning in one particular smart space can be developed by different teams and without some general perspective. This may result in instability of different kinds and improper functioning of smart space-based system overall. In this paper, an interaction model is proposed for Tourist assistant – TAIS, a smart space-based service-oriented mobile application that provides a tourist information about attractions around based on his/her preferences and current situation in location region. The proposed interaction model is backed by a two-layered ontology of tourism domain and a formal model that can be used to ensure stability of the overall application.**

I. Introduction

There are currently many mobile systems and prototypes that offer tourist services such as restaurants, museums, architectural attractions. In many cases, however, these services are treated independently and are not integrated with each other. It means that a typical service that aims to provide an information support for a tourist has its own database of tourist attractions (and/or hotels, events, restaurants) and a specialized mobile application that interacts with a user and displays the information from the service database. Incompleteness of each particular database gives rise to a variety of integrative approaches where information from several service databases is merged and reconciled. A thorough review of different approaches to tourist information support can be found, for example, in [1].

In this paper a goal- and data- centric approach is applied to tourist information support. Informally, that means a user does not associate received value (information) with some particular service but enjoys the result of the work of co-existing (and possibly collaborating) services in the smart space.

This paper continues the work on the Tourist Assistant – TAIS mobile application described in [1], [2]. The goal of this paper is to address some particularity of TAIS design that may hamper the development of third-party services for tourist information support communicating with available TAIS services. Specifically, structured description of tourist attractions and context used during services' interchange involves two kinds of representation: RDF syntax enforced by Smart-M3 smart space platform, which is used as communication media between TAIS services, and XML syntax that is, according to current TAIS implementation, embedded in some RDF triples [2].

In this paper, an RDFS ontology for tourist information is proposed, that enables pure RDF exchange between tourist information services and allows using only RDF tools to process and browse any tourist information circulating in smart space. The proposed ontology is supplemented by an interaction model describing how exactly services should interact in the smart space using this ontology. TAIS is based on Smart-M3 smart space platform, therefore blackboard interaction model and several kinds of limitations (discussed in detail in the respective section) imposed by this platform have severely influenced the proposed design.

Another contribution of this paper is a formal model that compliments the proposed interaction model. Intent of this formal model is to build and develop a method of verification of ensemble of services to make sure that this particular ensemble plays together well. The proposed formal model introduces a notion of type for a service, and some formal rules that help to analyze possible activation sequences of services, communicating via an RDF-based smart space, based on their types. This opens way to implement more predictable and stable smart space-based system, though sacrificing a versatility of service behavior (as all actions performed by service should conform to a predefined type). In some ways, this is similar to how some kind of errors are avoided during compile-type in statically typed programming languages.

In a wider perspective, a goal of this paper is to contribute to a development of a smart space-based information bus for tourist information that would be a convenient communication media for tourist information services of various vendors and would enable the creation of integrated tourist information support systems based on Semantic Web and smart space technologies.

Rest of the paper structured as following. Section II briefly describes related work on tourist support systems and relevant ontologies. Section III introduces TAIS and contains some basic information about smart space implementation – Smart-M3, TAIS is based on. Section IV introduces the tourist information ontology design. Section V presents the interaction model. Sections VI contains an example of one service definition to follow the proposed interaction model. Section VII introduces the formal model and Section VIII contains some experimental data about SPARQL subscriptions performance on Smart-M3.

II. Related work

One of the contributions of this paper is the ontology for communication between various tourist information services. This ontology should be expressive enough to describe all types of tourist information that are processed by TAIS and be able to describe other types of tourist information that may be useful in the future. For better understanding of potential types of information, ontology should be able to describe, available ontologies in the domain of tourism were analyzed. Another reason for a thorough analysis of currently available ontologies in the domain of tourism is that the number of structured information sources is increasing and the ontology being developed should be easily mapped to other common ontologies in the field as chances are that a potential provider of tourist information is already using one of these ontologies internally.

The difficulty of creating ontology of tourism domain is discussed in detail, for example, in [4]. The idea is that during a tourist trip a person can do almost anything he/she can do in an everyday life: go shopping, eat out, go sightseeing etc. Therefore, an ontology of tourism can include almost entire ontology of customer services coupled with cultural one. On the other hand, this ontology should somehow deal with dates, time intervals, geography and other common concepts that are best dealt with by some upper-level ontology.

One of the most elaborate ontologies in the domain of tourism is Harmonise ontology originally proposed in [3], now being the central element of HarmoNET (Harmonization Network for Exchange of Travel and Tourism Information) that aims to create a framework for data exchange in the tourism industry. The focus of this ontology is on events and accommodation.

A modular ontology is proposed in [4]. The authors of that paper use some ideas of Harmonise and propose an ontology that is centered around concepts Entity (which, in its turn, may be Spatial, SpatialTemporal, or Temporal), Service, Activity and TouristType. These concepts structure the ontology and for more specific activities (Accomodation, Gastronomy) or more general concepts (Weather, Time) other ontologies are used. Tourist types are taken from [5].

The [6] considers generic tasks and task ontology based on travelers' perspectives, and intelligent tourist information services using them. Therefore, they propose 1) a task model of travelers' perspective based on their needs and activities, 2) a task ontology using the generic tasks, their activities, relations, and properties, and 3) an intelligent tourist information system using task ontology based on various tasks and activities of travelers. The system consists of Tourist Contents Service (TCS) and Task-Orient Menu Service (TMS) parts, and can provide various intelligent tourist information services through task-oriented menus. Tourism domain ontology is centered around following concepts: Accommodation, Attraction, Entertainment, Festival/Event, Food, History/Culture, Location, Weather, Shopping, Transportation.

In [7] intelligent recommendation system based on Jeju travel ontology is proposed. The system can recommend the tourist more intelligent information using properties, relationships of travel ontology. It is also responsible for finding personalized attractions and plotting location of traveler.

The authors of [8] propose the Ontology-based Intelligent Ubiquitous Tourist Information System (OiUTIS) for an interactive tourist information service tailored to both tour services and travelers in ubiquitous environments.

The e-tourism ontology in [9] is built mostly as an example of Semantic Web technology stack and to the best of authors' knowledge is not mature enough to be used in a production system. It is based on three main questions that can be asked when developing tourism applications: a) What can a tourist see, visit and what can he do while staying at a tourism destination? b) Where are the interesting places to see and visit located? c) When can the tourist visit a particular place?

There are also several ontologies that are not exactly deal with tourist information, but may be useful, as they describe either broader part of world than tourism or cover in detail some subset of information usually relevant for tourists.

Shema.org project [10] provides a collection of schemas that webmasters can use to markup HTML pages in ways recognized by major search providers, and that can also be used for structured data interoperability (e.g. in JSON). Search engines including Bing, Google, Yahoo! and Yandex rely on this markup to improve the display of search results, making it easier for people to find the right Web pages. There are several concepts and relations relevant to tourism domain. Upper level structure (of the relevant subset) includes: Action, CreativeWork, Event, Intangible (e.g., Rating, StructuredValue, such as Contact or GeoCoordinates), Organization, Person, Place (including TouristAttraction).

GoodRelations [11] is a vocabulary for publishing all of the details of products and services in a way friendly to search engines, mobile applications, and browser extensions. It is by design more tailored to electronic commerce and is centered around BusinessEntity, Offering, ProductOrService and Location.

III. Tourist Assistant - TAIS

Tourist assistant – TAIS is an intelligent mobile tourist guide that allows tourists to get information about attractions around the current geographic location based on tourist context and ratings assigned by other tourists. Information about attractions is extracted from different internet services (like Wikipedia, Wikivoyage, Wikitravel, Panoramio, Flickr) "on the fly" that allows to use the guide in any region of the world and get actual at the moment information. Tourist assistant – TAIS provides information about public transport and car sharing possibilities for the tourist with drivers nearby for comfortable reaching preferred attractions.

Tourist assistant - TAIS includes several components [2, 12, 13, 14], this paper focuses on interaction of the following subset of them:

- client application installed to the user mobile device that shares tourist context with the smart space and provides the tourist results of guide application operation;
- attraction information service that implements retrieving and caching the information about attractions;
- recommendation service that evaluates attraction/image/description scores based on ratings that have been saved to internal database earlier.

Main tasks of client application are: share information about tourist context, profile, and actions; communication with smart space; provide results to the tourist; and share tourist ratings of attended attractions, browsed descriptions and images with the smart space.

The attraction information service is responsible for providing information about attractions with description and photos around location.

The recommendation service implements ranking attractions, images, and descriptions for providing the tourist the best attractions to see and the best images and description of chosen attraction for acquaintance.

Interaction between components of TAIS is performed by means of smart space platform Smart-M3, described in [15].

The Smart-M3 platform consists of two main parts: information agents and kernel. The kernel, in its turn, consists of two elements: Semantic Information Broker (SIB) and data storage. Information agents are software entities installed on the mobile devices of the smart space users or on some server machines providing services to all users of the smart space. For example, TAIS client application as well as TAIS services (attraction information service, recommendation service and others) are all information agents in terms of Smart-M3 platform.

Information agents interact with SIB through the Smart space Access Protocol (SSAP). The SIB is the access point for receiving the information to be stored, or retrieving the stored information. All this information is kept in the data storage as a graph that conforms to the rules of the Resource Description Framework (RDF). According to these rules, all information is represented by "Subject - Predicate - Object" triples. Most important operations supported by SSAP are inserting, removing, updating, querying RDF triples and subscribing to an insertion or deletion of a given RDF graph pattern.

The subscription to RDF graph patterns is the only way to provide a control flow for information agents interaction via Smart-M3 platform. Subscription can be defined either in the form of a simple triple pattern or in the more complex form of graph pattern described by a SPARQL query.

IV. Ontology design

According to the FIPA specification, ontology includes a vocabulary (i.e. a list of logical constants and predicate symbols) for referring to the subject area, and a set of logical statements expressing the constraints existing in the domain and restricting the interpretation of the vocabulary. Ontologies, therefore, provide a vocabulary for representing and communicating knowledge about some topic and a set of relationships and properties that hold for the entities denoted by that vocabulary [16].

A. Ontology design goals, constraints and scope

Goals of ontology design in this research effort: a) to provide a common vocabulary for all services participated in tourist information interchange; b) to preserve the information about the provider of each «information piece» in the common knowledge space.

Formal specification of common vocabulary along with the enumeration of allowed concept combinations (i.e., the first goal completion) would enable third-party services to «understand» graph patterns contained in smart space and be able to augment them in a semantically correct way.

The second goal is to give a tourist an opportunity to determine the supplier of different pieces of information in common space. It may give users an opportunity to prioritize different information suppliers in manual or automatic way. For example, there may be several attraction recommendation services each assigning expected utility for attractions, but a user may think that one of that services better predicts his/her interest as others. This is very close to the trust layer in classical Semantic Web stack.

From the technological point of view, there are several ways to approach this goal. First, different ways of reification – an RDF instrument that allows one to describe RDF triples with another triples. The problem with reification is that is requires either encoding each triple as four triples in the triple store or it requires some native support from RDF tools. Moreover, reification is an overkill w.r.t. the goal analyzed, as there is no need to add some *unique* information to each triple. Instead, the entire RDF graph should be «colored» by the limited number of service signatures. Another way to approach this problem is named graphs model, an extension of RDF model [17]. Albeit, at best of authors' knowledge Smart-M3 Platform does not have a built-in support of named graphs.

To meet the second design goal, a two-level ontology is introduced. The first layer (also called generic layer) defines cornerstone concepts of the tourist information domain and should be accepted and understood by any party that is interested in tourist information exchange. Examples of the concepts defined in the generic layer are Attraction, Rating etc. The second layer (service layer) defines service-specific terms for concepts of generic layer and maps them to the respective concepts via equivalence relation. So, in an example concerning multiple recommendation services, there is a property in a generic layer of the ontology corresponding to expected utility, but those multiple recommendation services do not use this property; instead, each service defines its own property for expected value in its own service ontology and maps it to generic property with a special ontology instrument.

It also must be noted, that the presented version of the ontology contains a high-level conceptual structure of tourist domain and a subset that covers information demands during sightseeing only, setting aside other tourist activities as accommodation, transportation, and so on.

B. Generic layer ontology

The proposed ontology is described in RDFS. Generic layer ontology is based on [4] and [10] and is built around four core classes: Tourist, Entity, Action, and Virtual. Tourist class instance corresponds to one tourist and its properties describe tourist's inclinations, preferences, and current state. Entity class' instances correspond to places (subclass Place) and events (subclass Event). Action class instances represent various actions and intents of the tourist (examine surroundings, examine place in detail etc.). Finally, Virtual class' subclasses define intangible concepts like user rating, score, address, geographic coordinates, etc.

All the concepts URIs used by the generic layer ontology are prefixed with the "http://spiiras.nw.ru/tio/gn/v1/" which will be omitted in the following text or replaced with a prefix "tiog" where appropriate.

Tourist class is used to state the fact that a subject is a person who is performing a vacation trip and is ready to receive various information about interesting surroundings and ways to spend time in a jovial way.

Statement (hereinafter, Turtle [18] syntax is used to write RDF triples)

```
<mailto:u@gmail.com> a tiog:Tourist .
```

is used to declare an ontology node with URI mailto:u@gmail.com as an instance of Tourist class.

Tourist instance is an allowed domain for the following properties (all the listed properties are defined in the namespace tiog):

- hasKeyword – literal-valued property corresponding to one area of interest of the tourist expressed as keyword. May have multiple values.
- hasGeoCoordinates:lat – tourist's coordinates. GeoCoordinates class instance, which is a pair of geo[1]:lat and geo:long.
- nearBy – points to an Attraction that is near to the tourist. May have multiple values.

TouristAttraction class (a subclass of Place, which is a subclass of Entity) denotes some physical place that a tourist might like to attend. Properties of the TouristAttraction are the following (all listed properties are defined in the namespace tiog):

- hasAddress – postal address of the attraction. Address class instance.
- hasGeoCoordinates – geographic coordinates of the attraction. GeoCoordinates class instance.
- hasMap – URL to a map of the place.
- hasUid – unique attraction identifier (literal).
- hasImageUrl – URL to an image of the place.
- hasName – the name of the TouristAttraction.
- hasAlternateName – an alias for the item (literal).
- hasDescription – a short description of the TouristAttraction.
- hasRating – a rating that was assigned to a Attraction by one user, if any. Rating class instance.
- hasExpectedScore – an expected score of how this attraction would be interesting to the user. Score class instance.

C. Service layer ontology

Generic layer ontology defines the set of concepts and properties that are used to describe tourists and attractions. However, the direct use of these terms would violate the second goal of the ontology – the resulting network of ontological definitions in common smart space would not contain an inkling on the originator of some fact. To resolve this issue the service layer ontology is introduced. The service layer ontology is a set of properties (and only properties) mapped to the generic ontology with special property tiog:isImplOf.

[1] W3C Geo: http://www.w3.org/2003/01/geo/wgs84_pos#

That is, if, for example, some service holds a huge collection of photographs it may define a property http://photoba.nk/tio/implementation/imageUrl (later referred as pb:imageUrl) and declare in smart space that its pb:imageUrl is an implementation of the tiog:imageUrl:

```
pb:imageUrl
tiog:isImplOf tiog:imageUrl.
```

After that photo service annotates nodes of the common ontology with pb:imageUrl and other services would be able to infer that, first, pb:imageUrl holds a URL of an image of the attraction, second, that that image was provided by http://photoba.nk.

For convenience, every property of the generic layer ontology is also mapped into itself with tiog:isImplOf.

V. Service Interaction Model

Service interaction model is based on the ontology subgraph monitoring and subscription capability of the underlying smart space implementation. Each application that is willing to take part in the tourist information exchange should start with identifying the ontology subgraph patterns that provide the required input. These patterns and events associated with them (such as creating and dissipation) serve also as signals to perform some actions. It is paramount that mutual dependencies between applications be avoided. Interaction takes the form of augmentation of the ontology graph.

When a new ontology subgraph matching the specified pattern is detected, the service action is triggered. Depending on the purpose of the service, its action may include adding new facts into the common ontology graph. When triggering specialization disappears, service must response to it by removing all the ontology graph data that was added into the smart space in response to creation of that pattern, if any. The rationale here is that in each moment of time ontology graph representing current situation must be consistent. Parts of the graph added by the service on detection of trigger pattern depend on parts that form that trigger pattern and should be removed upon dissipation of the latter to maintain ontology graph consistency. There is also a technological reason for that, namely any arc of the ontology graph must be controlled by exactly one service (so that not a single graph node or edge can become an orphan, at least in the process of normal functioning of all the services).

So, each service design must declare:

- an input ontology graph pattern specification;
- an action that is performed by the service;
- an output ontology graph pattern specification.

Ontology graph pattern specification is a set of arcs and nodes with ontology labels (or unbounded) that can be looked for and matched to some parts of shared ontology in smart space.

It is an open question whether a service action can use any information contained in the common ontology but not in the input pattern specification. In frequently updated ontologies in would easily result in race conditions and inconsistencies of different kind. Therefore, the rule of thumb in service design is to include all information that is needed for service operation in the input pattern.

Smart-M3 platform provides the required capabilities of ontology graph monitoring through the mechanism of SPARQL subscriptions. Knowledge processor can initiate a subscription operation by sending a SPARQL request to the SIB. SIB evaluates this request in the current smart space contents, returns the result of the query to the knowledge processor, and takes the responsibility to inform the knowledge processor each time the result of this SPARQL expression changes.

Therefore, input ontology graph pattern specification should be encoded as a SPARQL subscription query in such a way that each row of the result corresponds to one instance of that pattern. Then, upon each modification of the ontology graph the service will receive all the pattern instances that appeared in the ontology graph and all pattern instances that disappeared. As it was described earlier, these events are followed by the service performing target actions or removing output subgraph respectively.

This mechanism is clarified by the following example. Let a service must react on a user movement and update the list of nearby attractions. User movement is reflected in the ontology as a mutation of one or two user coordinates. User type must also be analyzed, because nearby attractions should be shown only to users that currently characterize themselves as tourists (during vacations, for example). Input pattern for the listed service requirements is shown in Fig. 1.

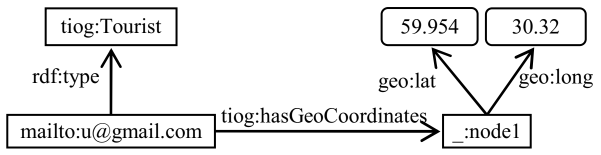

Fig. 1. Example input pattern

This pattern can be described with the following SPARQL query:

```
SELECT ?user ?lat ?long
{
  ?user a tiog:Tourist .
  ?user tiog:hasGeoCoordinates [
    geo:lat ?lat ;
    geo:long ?long ] .
}
```

TABLE I. EXAMPLE RESULT OF A QUERY

user	lat	long
<mailto:u@gmail.com>	59.954	30.32
<mailto:z@gmail.com>	57.15	65.533

Each row of this query corresponds to one instance of the input pattern. As SPARQL subscription operation of the Smart-M3 returns the result of query evaluation in the current ontology, the result of this subscription query will contain all users that are typed as tiog:Tourist with known coordinates (Table I).

The set of the received input patterns should be processed immediately resulting in augmenting the smart space ontology with the list of nearby attractions information.

For each row of this table the service generates a pack of triples describing the nearby attractions:

```
<mailto:u@gmail.com> tiog:nearBy [
    a tiog:TouristAttraction ;
    tiog:hasUid "uid:8378" ;
    tiog:hasName "Palace Square"
].
```

Afterwards, the SIB notifies the service each time this table is altered. For example, if the user mailto:u@gmail.com moves and his/her mobile device detects this and reflects the new location in smart space SIB sends to the service a couple of data tables – deleted items and added items. So, if mailto:u@gmail.com moves to (60, 30.33) then notification of the service looks like shown in Table II.

TABLE II. SPARQL SUBSCRIPTION CHANGE NOTIFICATION

Deleted		
mailto:u@gmail.com	59.954	30.32
Inserted		
mailto:u@gmail.com	60	30.33

The obsolete (deleted) row is interpreted as a sign of dissipation of the input pattern and results in cleaning smart space from the triples that were previously added by the service for user1_node. After that, new (inserted) row is interpreted as a new input pattern and a new triples are added into smart space:

```
<mailto:u@gmail.com> tiog:nearBy [
    a tiog:TouristAttraction ;
    tiog:hasUid "uid:9580" ;
    tiog:hasName "Tauride Gardens"
].
```

VI. EXAMPLE SERVICE INTERACTION SPECIFICATION

In the following section, a complete example of service taking part in the proposed interaction model is explained. It is a TAIS recommendation service. It listens to the list of attractions that are near the user (that list is provided by some other smart space service) and rate that attractions based on tacit and explicit user preferences. The key point concerning interaction specification is that input pattern (activating this service) and output pattern should be provided.

Service ontology includes properties for rating and score which are declared to be implementations of the respective properties of the generic layer ontology tiog:hasRating and tiog:hasExpectedScore. Prefix URI for service ontology can be http://example/tiog/rec/. Then, upon service initialization following triples are added into smart space:

```
@prefix rec:
  <http://example/tiog/rec/> .
@prefix tiog:
  <http://spiiras.nw.ru/tio/gn/v1/> .
rec:hasUserRating
    tiog:isImplOf
    tiog:hasUserRating .
rec:hasExpectedScore
    tiog:isImplOf
    tiog:hasExpectedScore .
```

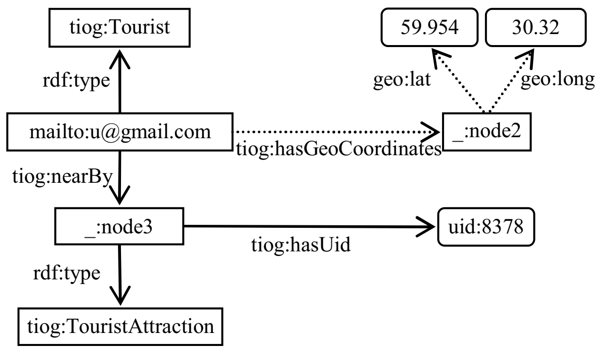

Fig. 2. Input ontology graph pattern of the recommendation service

The input for this service is an attraction list (attraction ratings, certainly, are also input, but it is not discussed here in detail). More specifically, input data should include user identifier (to retrieve user preferences from the preferences database) and attraction identifier (to ratings of that attraction from the preferences database). To adapt this kind of service to the proposed interaction model this input information must be described as an ontology pattern. Having in mind the fact, that tourist surroundings are represented in the ontology in the way, depicted in the Fig. 2 (input pattern arcs are highlighted by solid lines), SPARQL description of the input pattern can be written as follows:

```
SELECT ?user_id ?loc_node ?loc_uid
{
  ?user_id a tiog:Tourist.
  ?user_id ?nearprop ?loc_node.
  ?nearprop tiog:isImplOf
    tiog:nearBy.
  ?loc_node a tiog:TouristAttraction.
  ?loc_node tiog:hasUid ?loc_uid.
}
```

This query returns one row for each attraction that is said to be in the user surroundings and the respective user identifier.

For each new row – returned on subscription or in notification (as inserted row) the service inserts into smart space rating of the given attraction assigned by the user user_id and expected score of the attraction for that user (Fig. 3). In the Fig. 3 inserted triples are shown by dash lines, fet:nearBy is defined by the service that added attraction description as an implementation of tiog:nearBy.

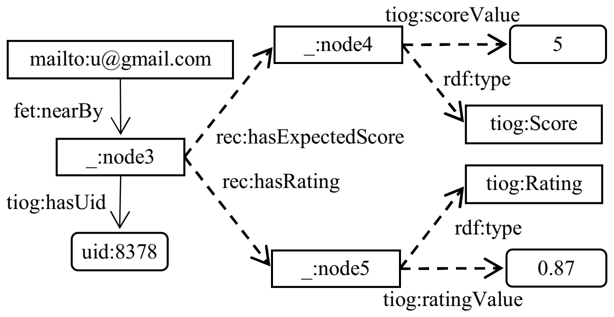

Fig. 3. Output ontology pattern

For each deleted row of the subscription query results recommendation service removes triples with userRating and expected score for ?loc_node – exactly the same triples that were inserted during the new row processing.

VII. Formal model

To analyze a multi-agent system, based on smart space interactions as it was described in the previous sections, a formal model is introduced. The analysis is needed to ensure that a particular system constructed is stable and viable. For example, to ensure that functioning of some set of services won't result in an endless "loop" of mutual service activations and an unbounded growth of operational ontology stored in a smart space. In this section, the proposed formal model and formal technique are described.

Let a system consist of a shared RDF graph G and a set of actors Q. Another representation of G is a subset of $LU \times LU \times (LU \cup LL)$, where LU is a set of URIs, LL – a set of literals.

Let the system consist of a shared directed graph G build according to RDF rules. and a set of actors Q. The nodes of graph $V = \{v_i\}$ and arcs $D = \{d_i\}$ have labels from the set L associated with them, i.e. l: $V \cup D \rightarrow L$, $l(v_i)$ will denote the label associated with the node v_i. This graph represents a common RDF-based ontology that is accessed by multiple actors. It actually greatly simplifies (broadens) RDF model, that distinguishes between literal nodes and URI nodes and imposes several restrictions on how nodes can be connected, however, it is not important from the point of view of this formal model.

Actors can do two things: to listen to changes in the common graph and to change common graph in some way. They can be of two types: initiative and reactive. The difference is that initiative actors can make changes in the common graph G in an arbitrary moment, they represent uncontrollable inputs to the system, and reactive actors perform graph G modifications only in response to some other modifications in graph, therefore, they represent logics of the system itself. Moreover, among initiative actors producers and consumers are distinguished. The former can only add some graph to G, and the latter can only remove some graph from G. Let's denote producers by adding «+» sign to an actor, like $A^+ = \{a_i^+\}$, consumers by adding "-" sign, like $A^- = \{a_i^-\}$, and reactive actors simply by $A = \{a_i\}$, hence $Q = A^+ \cup A^- \cup A$.

A type of an agent specifies input and output graph patterns. Input pattern is a graph structure the actor is processing, so the agent activates each time a new instance of input pattern is detected in G. Output pattern is a subgraph structure that is inserted by the actor in response to each insertion of an input pattern instance and is removed in response to removal of each input pattern instance.

Input pattern P is formed from a set of 3-tuples of extended sets LU and LL, and a set of variables. Formally, let B be a set of variables, $LU' = LU \cup \{*\} \cup B$, $LL' = LL \cup \{*\} \cup B$, here «*» is a special element, whose role will be clarified later. Then $P \subset LU' \times LU' \times LL'$. Output pattern O is formed in a similar manner from a set of 3-tuples and a set of variables: $LU'' = LU \cup B$, $LL'' = LL \cup \{@\} \cup B$, $O = LU'' \times LU'' \times LL''$.

Actor's type will be denoted as $T(a)$. Producer's type consists of a set of variables and an output pattern, consumer's type consists of a set of variables and an input pattern, whereas reactor's type consists of a set of variables, input and output patterns. For clarity, elements of variables set will be prefixed with a question mark (e.g., $?x$), elements of LU set will be surrounded by angular brackets (e.g., $<lat>$), and elements of LL will be enclosed in quotes (e.g., "river"). Example of some reactor agent follows:

$T(a) = (\{?u, ?l, ?g, ?lt, ?lg, ?\_\},$

$\{(?u, <lat>, ?l), (?l <isa> ?g),$

$(?l <lat> ?lt), (?l <long> ?lg)\},$

$\{(?u, <near>, ?\_), (?\_, <type>, <attr>),$

$(?\_, <name>, @)\})$

The evolution of a system consisting of a shared graph G and a set of actors Q starts with an empty graph ($G_0 = \varnothing$). The graph may change only as a result of execution of some initiative actor. Let the executed actor is a_s^+. Upon execution, it adds some subgraph to G, according to its type, producing a new version of the shared graph G_1. After that, each actor is tested whether its input pattern matches some subgraph of G_1.

Graph G matches input pattern $P = \{p_i\}$, iff there is such a binding of variables (and "*"), used in input pattern, to a set $LU \cup LL$, that for every element of P (a, b, c) there is a triple (a', b', c') in RDF graph G, such that $a' = a$, $b' = b$, $c' = c$. If there are several possible bindings, then each of them is processed by actor independently. Special element "*" can be bound to any node or arc. As a result of execution, each actor adds a subgraph, conforming to the output pattern to a common graph. Special element "@" is used to describe any literal value.

A process of graph evolution (and its state in the specified moment) can be described by a chain of actors' activations. Let's formally define in the following way:

1) empty sequence of invocations is a chain;

2) a is a chain, iff $a \in A^+$ (i.e., a single producer invocation forms a valid chain);

3) $c \bullet a$ is a chain, iff c is a chain and $a \in A^+ \cup A^-$ (i.e., initiator can be appended to any chain forming a new valid chain);

4) $c \bullet q \bullet \{a_i\}$ is a chain, iff c is a chain, $c \bullet q$ is a chain, $a_i \in A$, for every i, a_i is active, and there is no active $s \in A$, $s \notin \{a_i\}$. Actor a is active, iff a graph that is produced as a result of $c \bullet q$ matches input pattern of a, whereas, graph, produced by c doesn't match input pattern of a.

A chain is called simple if it contains exactly one initiative actor invocation (starts with it), other chain elements (if any) are reactive actors' invocations.

For example, let's consider a system consisting of one initiative actor a^+, $T(a^+) = (\{?u, ?o\}, \{(?u, <near>, ?o)\})$ and three reactive actors a_1, a_2, and a_3 whose types are defined as follows:

$T(a_1) = (\{?u, ?o\},$

$\{(?u, <near>, ?o)\},$

$\{(?o, <name>, @)\})$

$T(a_2) = (\{?u, ?o\},$

$\{(?u, <near>, ?o)\},$

$\{(?o, <rating>, @)\})$

$T(a_3) = (\{?n, ?o, ?r\},$

$\{(?o, <name>, ?n), (?o, <rating>, ?r)\},$

$\{(?u, <rec>, @)\})$

In such a system there may exist only one simple chain (as there is only one initiative actor), and it is the following: $a^+ \bullet \{a_1, a_2\} \bullet a_3$.

Let's consider another example system, consisting of one initiative actor a^+, $T(a^+) = (\{?u, ?o\}, \{(?u, <near>, ?o)\})$ and two reactive actors a_1, a_2 with the following types:

$T(a_1) = (\{?u, ?o\},$

$\{(?u, <near>, ?o)\},$

$\{(?u, <near>, ?o)\})$

$T(a_2) = (\{?u, ?o\},$

$\{(?u, <name>, ?o)\},$

$\{(?u, <rec>, @)\})$

In a system like that, the only simple chain is following: $a^+ \bullet \{a_1\} \bullet \{a_1\} \bullet \ldots$. In other words, we may notice that: a) this system has an infinite chain, b) that some actors are not involved in that chain.

The first task to be solved with the help of this formal representation is to ensure that the provided set of actors coordinate well, i.e. doesn't bear a risk of eternal loops. Formal interpretation of this is to prove that each possible chain is finite. As the number of possible chains (allowing initiative execution at any point of time) may be large. The first step is to show that each simple chain in this system is finite, making an assumption that chain executions are unlikely to interleave.

A proposed technique to do so is to emulate building of all possible chains using type definitions of actors.

VIII. Experiments

As the proposed approach is based on SPARQL subscription feature, it is important to know how well the chosen smart space implementation (i.e., Smart-M3) handles this kind of subscriptions. To find this out, an experiment was made. The setup included two computers: a virtual machine (VMWare, 1500 MB RAM) on an Intel® Xeon® 2.4GHz server running SIB (RedSIB 0.9.0), and a workstation (Intel® Core™ i7 3.9GHz 8GB RAM) running multiple KPs. The goal of the experiment was to evaluate a delay between making a change in smart space and receiving a notification about that change through SPARQL subscription. The influence of two parameters on this delay was analyzed: number of SPARQL subscriptions that should be notified upon change and the number of triples in subscription pattern. To accomplish the goal several experiments were performed. Each experiment included creation on the workstation *n* KPs each with one SPARQL subscription (all subscriptions were the same and included *patsize* triples), making a change in the smart space from workstation that fired the subscription

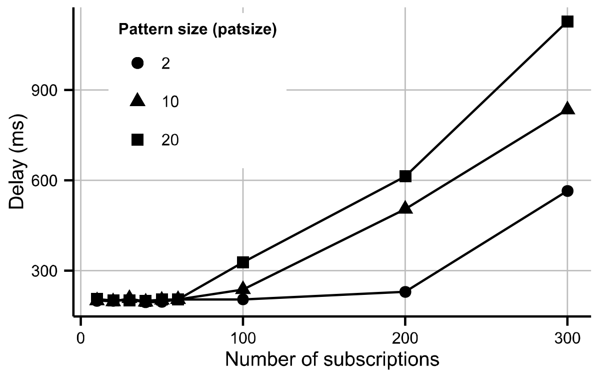

Fig. 4. Median notification delay from the number of SPARQL subscriptions with different subscription pattern sizes

notification process and logging notification delays from the moment of making the change. For each combination of *n* and *patsize* 10 changes were made to obtain average values.

The results are summarized in Fig. 4. It shows median notification delay (in milliseconds) for *n* subscriptions and different pattern sizes. It can be seen that in simplest cases delay was about 200 ms, which is probably "the cost" of parsing, network communication etc. For small patterns, consisting of 2 triples, delay grows rather slow for small number of subscriptions, for patterns as large as 20 triples (which are unlikely to be used widely) median delay exceeds 1 sec with less than 300 subscriptions.

It also important to note, that with the number of subscriptions more than 500 RedSIB failures were not uncommon.

IX. CONCLUSION

In this paper an interaction model and an ontology are proposed for smart space services providing tourist information support. The principle that lies behind the interaction model is incremental growth of an ontology graph as a result of contribution of different services. Service design guidelines are discussed and analyzed.

A formal computational model based on common graph mutation was also proposed. This model can potentially describe a wide range of applications based on blackboard exchange. It is evident, that formal verification technique can be used only if each actor actively provides its type. However, in current implementation of Smart-M3 any KP can join and add/remove any triple and define any subscription without sharing information about its possible future actions. Of course, it gives versatility to this framework and systems based on this, but severely limits possibilities for any stability checking. Therefore, for the systems, based on the proposed interaction model it is reasonable to implement an additional layer between KPs and Smart-M3 shadowing the current Smart-M3 interface, but offering a type-based one instead. The type-based interface layer would require from any KP joining to smart space to provide its type (in the sense discussed in this paper) and limit modifications initiated by that KP to those that conform to previously defined type.

Another direction of future work is to develop techniques to prove formally a broader set of properties of smart space-based systems using this interaction model.

Finally, the proposed interaction model and the respective formal model are centered around uniform chains, i.e. when adding of some subgraph to the shared graph may cause only adding another subgraph, but may not cause removal. This is not true for many applications and interactions patterns (e.g. ordinary message passing), which limits practical impact of this work. Therefore, a more general case need to be considered.

ACKNOWLEDGMENT

The presented results are a part of the research carried out within the project funded by grants #13-07-00271, #14-07-00345 of the Russian Foundation for Basic Research. This work was also partially financially supported by Government of Russian Federation, Grant 074-U01.

REFERENCES

[1] A. Smirnov, A. Kashevnik, N. Shilov, N. Teslya, A. Shabaev "Mobile Application for Guiding Tourist Activities: Tourist Assistant – TAIS", *in Proceedings of the 16th Conference of Open Innovations Association FRUCT*, 2014, pp. 95-100.

[2] A. Smirnov, A. Kashevnik, A. Ponomarev, N. Shilov, M. Shchekotov, N. Teslya, "Smart Space-Based Intelligent Mobile Tourist Guide: Service-Based Implementation", *in Proc. of the 15th Conference of Open Innovations Association FRUCT*, 2014, Saint-Petersburg, Russia, pp. 126-134.

[3] O. Fodor, H. Werthner "Harmonise: A Step toward an Interoperable E-Tourism Merketplace", *International Journal of Electronic Commerce*, Vol. 9, No. 2, Winter 2004/2005, pp. 11-39.

[4] R. Barta, C. Feilmayr, B. Pröll, C. Grün, H. Werthner "Covering the Semantic Space of Tourism: An Approach Based on Modularized Ontologies", *Proceedings of the 1st Workshop on Context, Information and Ontologies*, 2009, article 1.

[5] H. Gibson, A. Yiannakis, "Tourist Roles, Needs and the Life Course". *Annals of Tourism Research* 29, 2002, pp. 358–383.

[6] H. Park, A. Yoon, H.-C. Kwon "Task Model and Task Ontology for Intelligent Tourist Information Service", *International Journal of u- and e- Service, Science and Technology,* vol. 5, no. 2, 2012, pp. 43-57.

[7] C. Choi, M. Cho, J. Choi, M. Hwang et al. "Travel ontology for Intelligent Recommendation System", *in Proc. of the 3rd Asi International Conference on Modelling and Simulation,* 2009, pp. 637-642.

[8] H. Park, S. Kwon, H.-C. Kwon "Ontology-based Approach to Intelligent Ubiquitous Tourist Information System", *in Proc. of the 4th International Conference on Ubiquitous Information Technologies and Applications*, 2009, pp. 1-6.

[9] J. Cardoso, "Developing an OWL Ontology for E-Tourism"*, in Semantic Web Services, Processes and Applications*, Springer, 2006.

[10] Schema.org official site. Web: schema.org.

[11] GoodRelations official site. Web: http://www.heppnetz.de/projects/ goodrelations/.

[12] A. Smirnov, A. Kashevnik, A. Ponomarev, N. Shilov, M. Shchekotov, N. Teslya, "Recommendation System for Tourist Attraction Information Service", *Proceedings of the 14th Conference of Open Innovations Association FRUCT*, State University of Aerospace Instrumentation, 2013, pp. 148-155.

[13] A. Smirnov, N. Shilov, A. Kashevnik, N. Teslya, S. Laizane, "Smart Space-based Ridesharing Service in e-Tourism Application for Karelia Region Accessibility. Ontology-based Approach and Implementation", in proc. 8th Int. Joint Conference on Software Technologies, July 29-31, 2013, Reykjavik, Iceland, pp. 591-598.

[14] A. Smirnov, A. Kashevnik, S. Balandin, S. Laizane, "Intelligent Mobile Tourist Guide: Context-Based Approach and Implementation", Internet of Things, Smart Spaces, and Next Generation Networking, Lecture Notes in Computer Science, vol. 8121, Aug. 2013, pp 94-106.

[15] J. Honkola, H. Laine, R. Brown, O. Tyrkko, "Smart-M3 Information Sharing Platform", in *Proceedings IEEE Symp. Computers and Communications (ISCC'10)*. IEEE Comp. Soc.; Jun. 2010, pp. 1041-1046.

[16] FIPA 98 Specification. Part 12 - Ontology Service. Geneva, Switzerland, Foundation for Intelligent Physical Agents (FIPA), 1998. Version 1.0. Web: http://www.fipa.org.

[17] J. Carroll, C. Bizer, P. Hayes, P. Stickler "Named graphs, provenance and trust", *Proc. of the 14th International Conference on World Wide Web*, 2005, pp. 613-622.

[18] RDF 1.1 Turtle. Terse RDF Triple Language. Web: http://www.w3.org/TR/turtle.