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

Andrew Ponomarev, Vladimir Parfenov

St. Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences (SPIIRAS)
ITMO University, St. Petersburg, Russia
Motivation (1)

- TAIS is a tourist assistant application:
  - information about attractions taken from external sources;
  - recommendations;
  - route building;
  - etc…
The 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**.
Contribution

Ontology

- **What** to say
- **How** to say
- **When** to say
- Won’t they *quarrel* if it is said?

Interaction model

Formal model & verification technique
Ontology design goals

1. To provide a common vocabulary for all services involved in tourist information interchange.
2. To preserve the information about the provider of each “information piece” in the common knowledge space.
Existing tourist ontologies analyzed

- Tourist ontologies:
  - Harmonise ontology (HarmoNET)
  - cDOTT – modular ontology by R.Barta et al.
  - Task Model and Task Ontology for Intelligent Tourist Information Service by H.Park et al.
  - Travel Ontology by C.Choi et al.
  - ...

- Other useful ontologies:
  - Schema.org
  - GoodRelations
Ontology structure

**Generic layer**

Actions and intents of a tourist (examine, attend etc.)

Everything that might be attended: Places and Events, e.g. TouristAttraction

**Service layer**

Intangible concepts: Rating, Score, Address...

Tourist information

Mapping via isImplOf property

Namespace: http://spiiras.nw.ru/tio/gn/v1/

Namespace: http://example/service/gn/v1/
Interaction model. Pattern concept

- Interaction model is based on subgraph subscription feature of underlying smart space implementation (Smart-M3)
- *Graph pattern* is a graph, whose occurrences in large graph (smart space operational RDF ontology) are of interest
  - Node variables
    - RDF: Resources, literals
  - Arc variables
    - RDF: Predicates
Pattern example (1/2)

SPARQL subscription

```sparql
@PREFIX tiog: <...>
@PREFIX geo: <...>
SELECT ?user ?lat ?long
{
  ?user a tiog:Tourist .
  ?user tiog:hasGeoCoordinates [ a tiog:GeoCoordinates ;
        geo:lat ?lat ;
}
```

New:

<table>
<thead>
<tr>
<th>user</th>
<th>lat</th>
<th>long</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="mailto:example@gmail.com">mailto:example@gmail.com</a></td>
<td>59.954</td>
<td>30.32</td>
</tr>
</tbody>
</table>

Obsolete:
**Pattern example (2/2)**

```sparql
@PREFIX tiog: <...>
@PREFIX geo: <...>
SELECT ?user ?lat ?long
{
  ?user a tiog:Tourist .
  ?user tiog:hasGeoCoordinates [ a tiog:GeoCoordinates ;
    geo:lat ?lat ;
    geo:long ?long ; ] .
}
```

- **New:**
  - user | lat | long
  - mailto:example@gmail.com | 59.954 | 30.32

- **Obsolete:**
  - user | lat | long
  - mailto:example@gmail.com | 59.954 | 30.32
Each service wishing to join the information interchange must declare:

- An input ontology pattern specification.
- An action performed by the service.
- An output ontology pattern specification.
Services interaction example (1/3)
Services interaction example (2/3)
Services interaction example (3/3)

```
Tourist
  rdf:type
  hasGeoCoordinates
    mailto:example@gmail.com
    ex:nearBy
      _:node1
        hasUid
          uid:8378
          geo:lat
            30.32
          geo:long
            59.954
          hasGeoCoordinates
            _:node3
              geo:long
                30.32
              geo:lat
                59.954
              rdf:type
                GeoCoordinates
              hasName
                The Hermitage
            rdf:type
              TouristAttraction
            hasExpectedScore
              12.3
```
Services interaction model.
Metaphor

Photograph taken by Mark A. Wilson (Department of Geology, The College of Wooster).
Formal model. Elements

- **Goal**: to develop a technique that is able to reason about a set of services, to ensure that interaction of this set of services will not result in an abnormal behavior:
  - “Dead” service which is never activated
  - Infinite sequence of mutual activations and unbounded growth of the operational ontology in the smart space

- Generalize from smart space-based system to a pair of RDF graph $G$, representing some shared storage and a set of *actors* $Q$
  - Alternative representation of $G$: $G \subseteq LU \times LU \times (LU \cup LL)$, where $LU$ – a set of URIs, and $LL$ – a set of literals
  - Actors: $Q = A^+ \cup A^- \cup A$
    - *Initiative* – can make changes in graph in any moment (uncontrollable inputs). Producers $A^+ = \{a^+_i\}$, consumers $A^- = \{a^-_i\}$.
    - *Reactive* – can make changes only in response to other modifications ($A$)
Formal model. Actor type

- **A type** of an actor $T(a)$ specifies input and/or output graph patterns
- 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, $LU' = LU \cup \{\ast\} \cup B$, $LL' = LL \cup \{\ast\} \cup B$
- Output pattern $O$ is formed in a similar manner from a set of 3-tuples and a set of variables: $LU'' = LU \cup B$, $LL'' = LL \cup \{@\} \cup B$, $O = LU'' \cup LU'' \cup LL''$
- **Example:**

  $$T(a) = (\{?u, ?l, ?g, ?lt, ?lg, ?\},$$
  $$\{(?u, <Lat>, ?l), (?l <isa> ?g),$$
  $$\ \ (?l <lat> ?lt), (?l <Long> ?lg)\},$$
  $$\{(?u, <near>, ?\_), (?\_, <type>, <attraction>),$$
  $$\ \ (?\_, <name>, @)\})$$
Formal model. Evolution

- Evolution of a system \((G, Q)\) starts with an empty graph
- The graph may change as a result of some initiative actor’s action
- Reactive actors are then activated, according to their types
- **Chain:**
  - empty sequence of actor invocations is a chain;
  - \(a\) is a chain, iff \(a \in A^+\);
  - \(c \circ a\) is a chain, iff \(c\) is a chain and \(a \in A^+ \cup A^-\);
  - \(c \circ q \circ \{a_i\}\) is a chain, iff \(c\) is a chain, \(c \circ q\) 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 \(c \circ q\) matches input pattern of \(a\), whereas, graph, produced by \(c\) doesn’t match input pattern of \(a\).
- A chain is *simple* when it contains exactly one initiative actor invocation
Example 1

- Initiative actor $a^+$
  - $T(a^+) = (\{?u, ?o\}, \{(?u, \langle\text{near}\rangle, ?o)\})$

- Reactive actors:
  - $T(a1) = (\{?u, ?o\}, \{(?u, \langle\text{near}\rangle, ?o)\}, \{(?o, \langle\text{name}\rangle, @)\})$
  - $T(a2) = (\{?u, ?o\}, \{(?u, \langle\text{near}\rangle, ?o)\}, \{(?o, \langle\text{rating}\rangle, @)\})$
  - $T(a3) = (\{?n, ?o, ?r\},$
    - $\{(?o, \langle\text{name}\rangle, ?n), (?o, \langle\text{rating}\rangle, ?r)\},$
    - $\{(?u, \langle\text{rec}\rangle, @)\})$

- Possible simple chains:
  - $a^+ \bullet \{a_1, a_2\} \bullet a_3$

- Conclusions:
  - No “dead” actors
  - All simple chains are finite
Example 2

- Initiative actor $a^+$
  - $T(a^+) = (\{?u, ?o\}, \{(?u, \langle \text{near} \rangle, ?o)\})$

- Reactive actors:
  - $T(a_1) = (\{?u, ?o\}, \{(?u, \langle \text{near} \rangle, ?o)\}, \{(?u, \langle \text{near} \rangle, ?o)\})$
  - $T(a_2) = (\{?u, ?o\}, \{(?u, \langle \text{name} \rangle, ?o)\}, \{(?u, \langle \text{rec} \rangle, @)\})$

- Possible simple chains:
  - $a^+ \cdot \{a_1\} \cdot \{a_1\} \cdot \ldots$

- Conclusions:
  - $a_2$ is “dead”
  - Infinite chain exists
Verification idea

- Goal (revisited): ensure that every possible chain is finite and each actor is involved in at least one chain
- But, as initiative actor can be invoked in any part of the chain, there are potentially infinitely many chains, and they can not be examined
- So, a simplification:
- Ensure that every possible simple chain is finite and each actor is involved in at least one simple chain
- Method: enumerate all simple chains for a given set of actors
1) Maintains a set of actors and checks consistency on each new actor connection
2) Checks type conformance on each edit
Experiments

- Examine notification delays for SPARQL subscriptions for Smart-M3
- Parameters: pattern size (\textit{patsize}), number of subscriptions (\textit{n})
- Possible biases: KP machine workload, network throughput

![Diagram showing client machine and server machine connections]

SIB

RedSIB 0.9.0 (VMWare, 1500 MB RAM)

Intel® Core™ i7 3.9GHz 8GB RAM

![Graph showing delay (ms) vs. number of subscriptions]

Pattern size (\textit{patsize})
- 2
- 10
- 20

Number of subscriptions

Delay (ms)
Conclusion

- **Main objectives:**
  - ✓ Generic ontology for tourist information
  - ✓ Interaction model based on incremental growth of a shared ontology graph
  - ✓ Formal model for analysis and verification of systems, based on the proposed interaction model

- **Current research directions:**
  - Develop techniques to formally prove various properties of smart space systems, based on the proposed interaction model
  - Consider non-uniform actors
  - TAIS: Ontology elaboration to a more fine-grained level
Thank you!

Questions are welcome!