A Semi-Automatic Approach for Semantic IoT Service Composition

Grigoris Tzortzis and Evaggelos Spyrou
Institute of Informatics and Telecommunications
NCSR “Demokritos”, Greece
Problem Definition (1)

Internet of Things (IoT)
- Integration of everyday physical objects with the world wide web
- IoT adopts a service-oriented architecture (SoA), where all “things” are exposed as (semantic) web services

Service Composition in IoT
- Combine the available services in an IoT ecosystem to construct a new, composite service that fulfils some desired functionality
- Discover appropriate services and interconnect them
- Ensure that all services are invokable
Problem Definition (2)

Service composition is **primarily about matching service outputs** (and effects) to service **inputs** (and preconditions)

**Approaches to service composition**

- Manual vs **Semi-automatic** vs Automatic
- Syntactic vs **Semantic**
Necessary Tools

- **Ontology**
  - Semantic annotations for services
  - **We propose a smart meeting room ontology**

- **IoT-ready platform**
  - Interconnection and coordination of vast number of heterogeneous devices
  - Devices exposed as services
  - Ontology support and reasoning over ontologies
  - We use the SYNAISTHISI platform\(^1\) developed at IIT, NCSR “Demokritos”

---

\(^1\)G. Pierris et al., SYNAISTHISI: An Enabling Platform for the Current Internet of Things Ecosystem, PCI, 2015
The SYNAISTHISI platform (1)

- Available services are registered into a service registry, implemented by an RDF triplestore.

- They follow the IoT paradigm and are divided into:
  - S-type services corresponding to sensors that sense the physical world.
  - P-type services corresponding to processors (algorithms) that process the measurements of the S-type services and/or the processed results of other P-type services.
  - A-type services corresponding to actuators that are used for the actuation of devices/signals based on the acquired results.
The SYNAISTHISI platform (2)

- Services exchange information with messages via the MoM
- Information is shared by “publishing” through specific topics
- Services that need to use information, “subscribe” to the appropriate topics
Smart Meeting Room Ontology

Models the SPA services of smart meeting rooms
- Domain-specific
- High-level and low-level concepts
- Enhance service discovery and composition
- Reuses existing ontologies — IoT-A, SSN, QU, QUDT
- Integrated into the SYNAISTHISI platform

Statistics
- ~200 classes
- ~50 Datatype properties
- ~50 Object properties

1Preliminary version: C. Akasiadis et al., *Developing Complex Services in an IoT Ecosystem*, WF-IoT, 2015
Resource Model (excerpt)

Describes the characteristics of the **device** hidden behind a service
Service Model (excerpt)

Describes the characteristics of the **service** exposing the device
Service Composition

We propose a **semi-automatic approach** for SPA service composition as part of the SYNAISTHISI platform

Main features

- Utilizes **semantics** of the smart meeting room ontology
- Minimum human intervention
  - The **platform guides the developer** in building a composite service
  - Service discovery and interconnection is the responsibility of the platform
- Based on **matching** services’ outputs to inputs
  - Preconditions are ignored, effects are treated as special type of output of A-type services
Service Composition Algorithm

**STEP 1:** matching list of $S(X)$

- $X$ is an approximate match of $X$

**STEP 2:** developer chooses $s_2$

- $s_2$ is an appropriate service

**STEP 3:** generated queries: $S(A)$, $S(B)$

- let us assume that $s_2$ has inputs “$A$”, “$B$”

**repeat process**

until: S-type services are selected

RESULT: composed service...

...senses the physical world and produces output “$X$”
The service request contains the desired outputs of the composite service

- Outputs are declared using a suitable concept from an ontology
The service request contains the desired outputs of the composite service

- Outputs are declared using a suitable concept from an ontology

A matching list contains services whose output matches a particular input of another service

- An independent service discovery is launched to populate the matching list
- Utilization of semantics in finding matches (explained later)
- The developer must choose a service from the matching list presented to him
The service request contains the desired outputs of the composite service

- Outputs are declared using a suitable concept from an ontology

A matching list contains services whose output matches a particular input of another service

- An independent service discovery is launched to populate the matching list
- Utilization of semantics in finding matches (explained later)
- The developer must choose a service from the matching list presented to him

If non-empty matching lists are found for all inputs of a service, the service is invokable

- S-type services can be readily invoked when selected → Service discovery is unnecessary

**Service Composition Algorithm**
Semantic Relaxation

Exploit **semantic hierarchical relationships** to decide if output concept \( A \) matches input concept \( B \)

- \( \text{exact}(A, B) \rightarrow \text{Same URI or OWL equivalent} \)
- \( \text{plugin}(A, B) \rightarrow A \text{ is subsumed by } B \)
- \( \text{subsume}(A, B) \rightarrow A \text{ subsumes } B \)

\( \text{exact} < \text{plugin} < \text{subsume} \) in terms of semantic relaxation degree
Advantages of the Service Composition Approach

- Guarantees that the composite service:
  - Satisfies the service request
  - All its services can be invoked

- Semantic relaxation
  - Avoid syntactic barriers
  - Permit approximate solutions when exact ones do not exist

- Service discovery and interconnection is the responsibility of the platform

- The developer only defines a service request and selects services from platform-generated matching lists
  - Even experienced users can perform service composition
Use Case: Creating a People Counting Service (1)

- One of the pilots of the SYNAISTHISI project was a smart meeting room.
- Among the goals was the minimization of user discomfort, environmental impact and monetary costs.
- To achieve these goals, an estimation of the number of people present within the room was necessary.
- Since cameras were installed, a computer vision approach was followed.
- Several services were developed to support the functionalities of the smart meeting room.
Use Case: Creating a People Counting Service (2)

The complex people counting service may be composed using simpler services.

The developer should have a basic knowledge in the field of computer vision.
Use Case: Creating a People Counting Service (3)

The resulting composite service:

More details and evaluation of this algorithm may be found in:
Open Issues and Future Work (1)

- Use lightweight standards to annotate services and their IOPEs
  - SAWSDL, hRest
  - More native to services

- Pursue semantically-aware automatic service composition
  - Graph-based
  - AI planning-based
  - Appropriate for end-users that are not developers
Open Issues and Future Work (2)

- Service composition should consider:
  - Functional requirements
  - Non-functional requirements e.g. location, reputation, QoS
  - User preferences

- Service marketplace that supports the full cycle of producing, delivering and trading a service
  - Exploit service composition to deliver complex applications to end-users
Thank you!

Questions?