Towards Cloud Services Marketplace: Interactive Service Retrieval Methodology and System

Yexi Jiang
Florida International University

Outline

- CSM Background and Motivation
- CSM System Overview
- Conversational Service Retrieval
  - Service Knowledge Base
  - Simultaneous Service Filtering & Configuration
  - Semantic Query Engine
- Future Work
- Conclusion

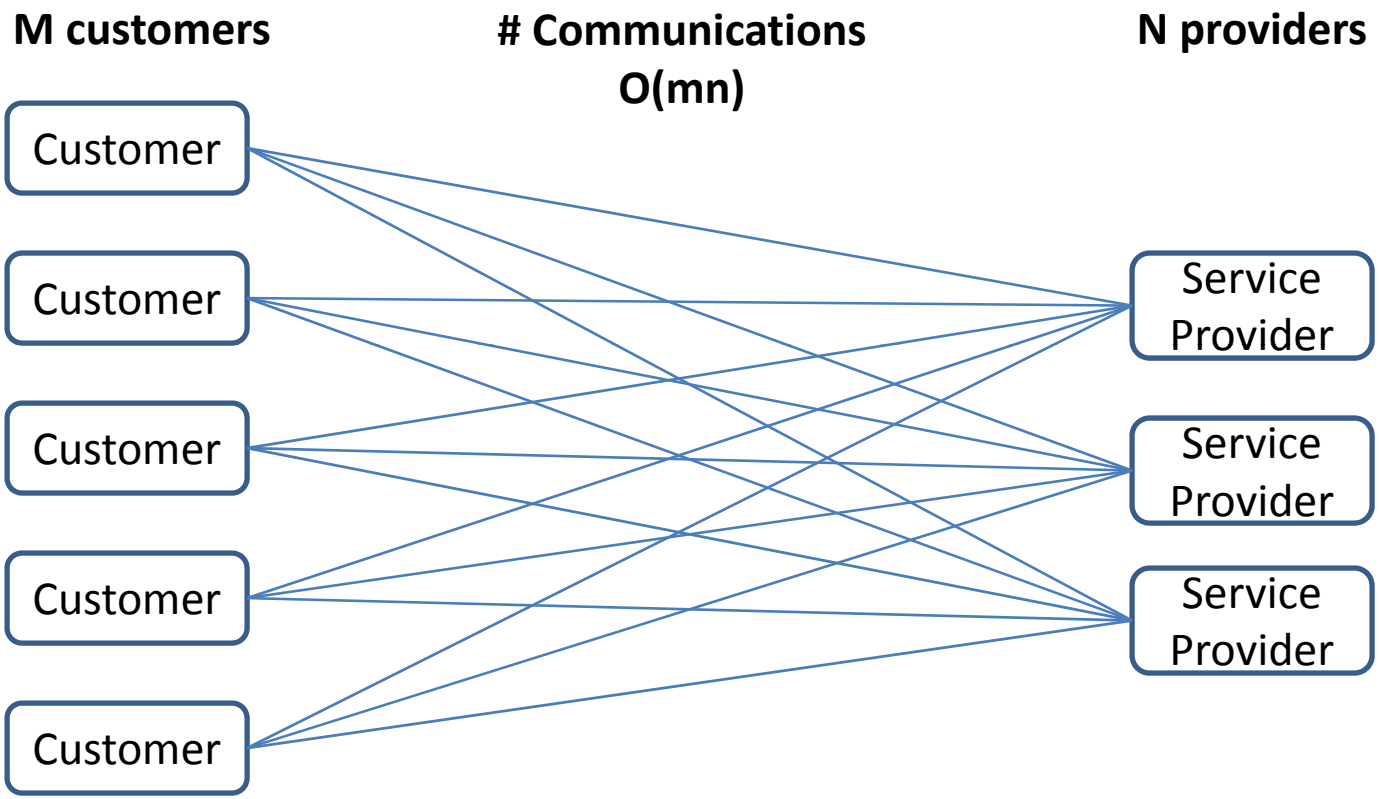
CSM Background

Customer Challenge: How to find a suitable service in cloud?

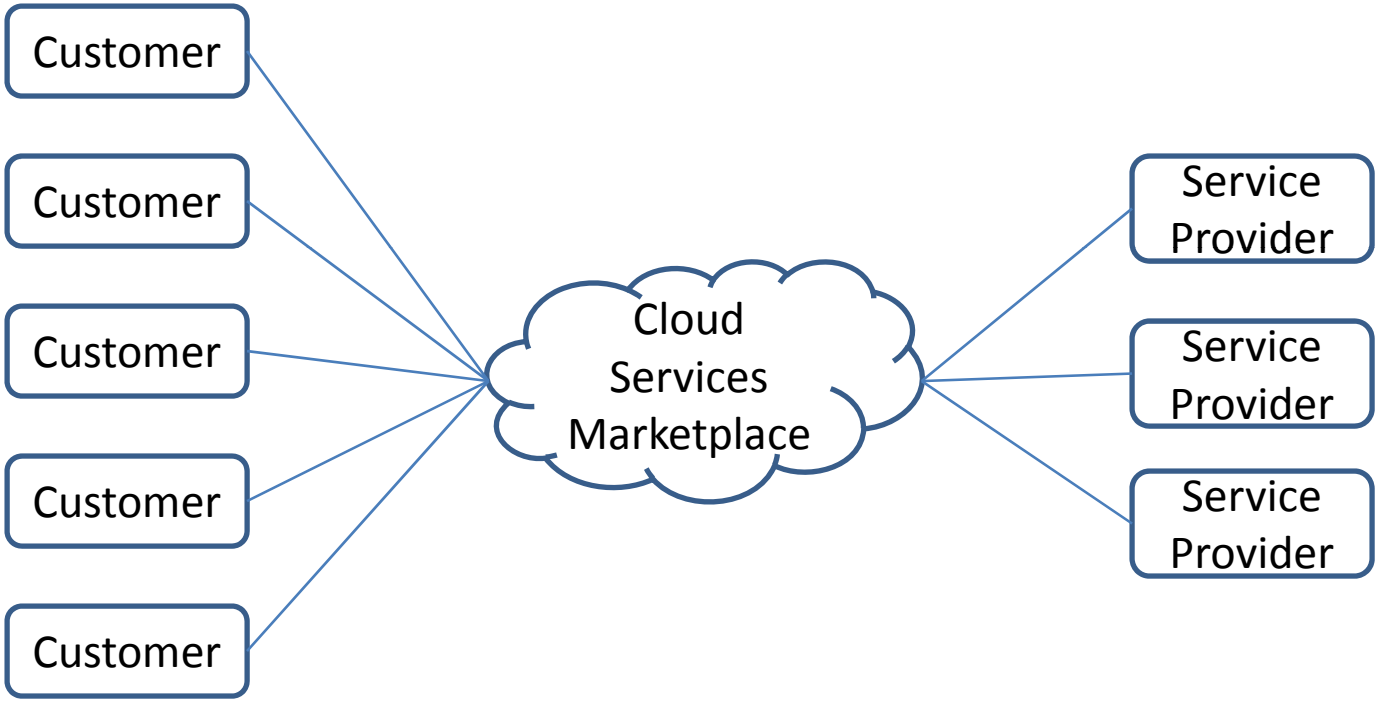
Current Product Retrieval Solutions and Limitations

- Current strategy: From providers' web site, Service agent
  - No systematic way to compare services
  - Time consuming: Cost up to days to find proper services
- Amazon, ebay
  - Keyword and faceted based search is not enough to find complex services
  - No support for service configuration and composition

Service Ecosystem



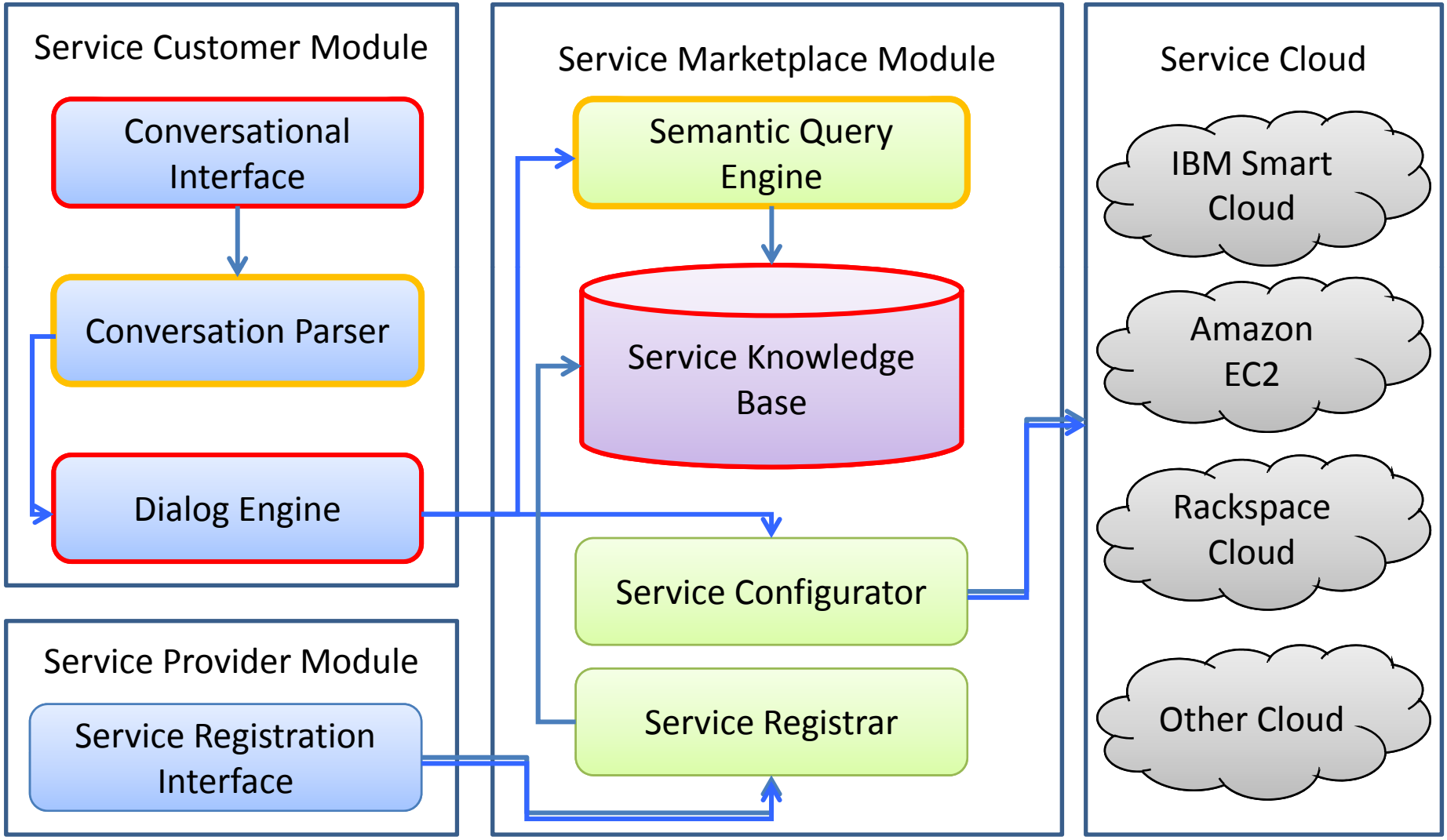
Service Ecosystem



Cloud Services Marketplace: **Features**

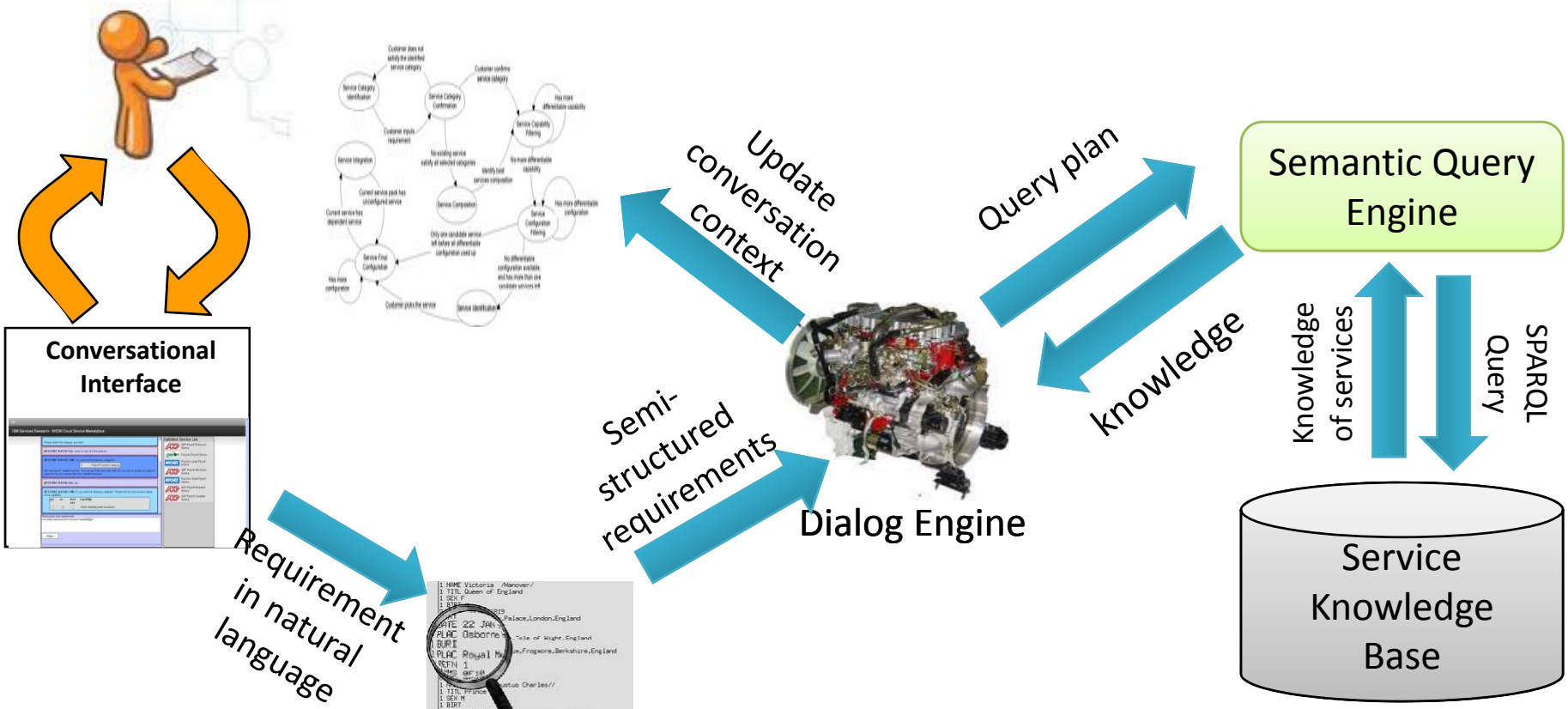
- Owns deep knowledge about services, can understand customer's intention
- Supports customer friendly conversational service acquisition
- Conducts filtering and configuring services simultaneously for enhanced service filtering

CSM System Overview



Dialog Engine Prototype

The centerpiece that guides the conversation, and coordinates user input processing and backend information retrieval

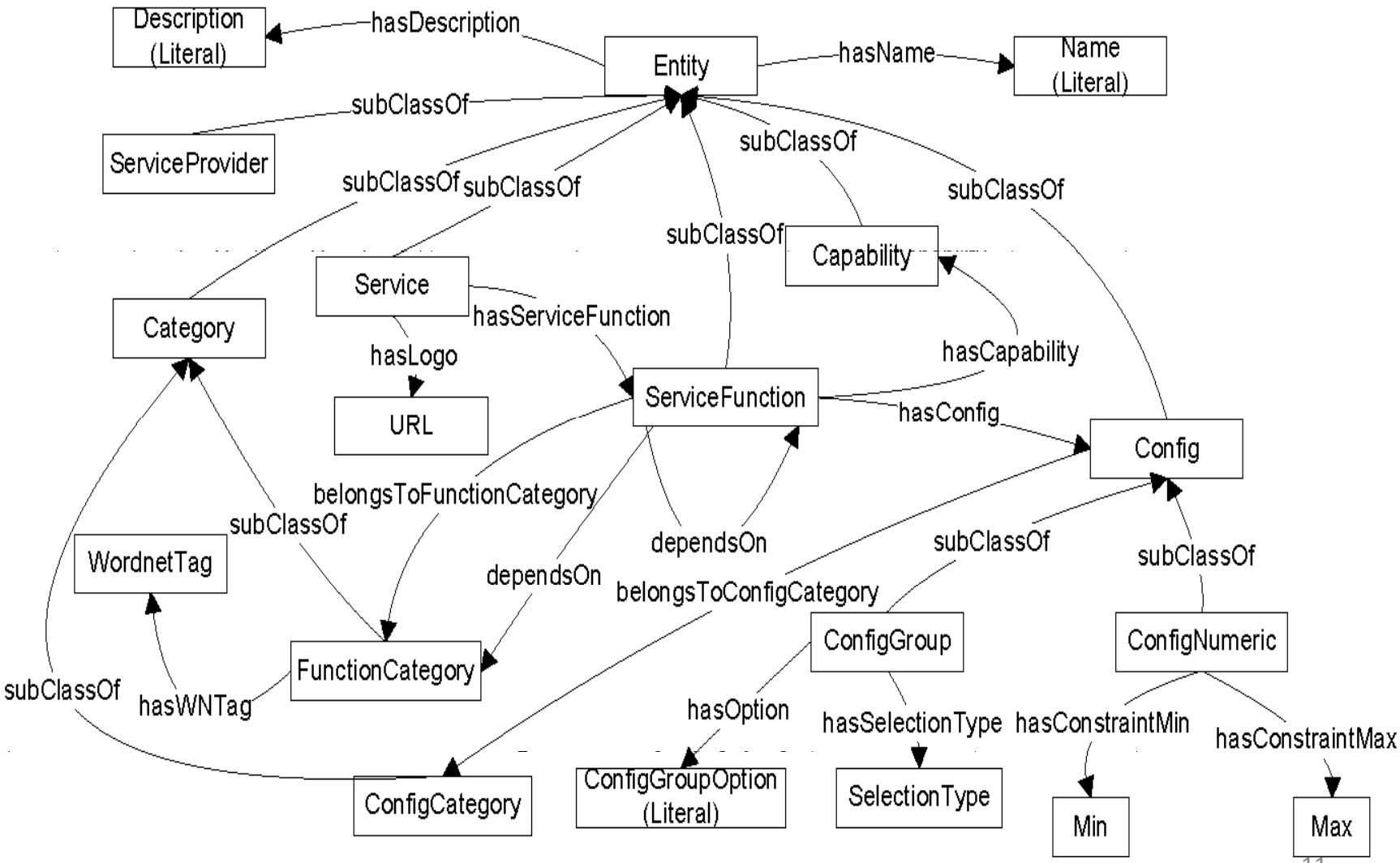


Service Knowledge Base Prototype

- Concepts in Service definition ontology
  - Descriptions -- name, description, provider, semantic tag
  - Functionalities -- capability, configuration
  - Relationships -- functionality category, dependency

- Current Content: 50+ service categories, 2000+ service providers, 2600+ services
- E.g. categories: ‘Storage’, ‘Payroll’, ‘Virtual Infrastructure’, ‘Advertising’, ‘Search’, ‘Email’, ‘Fax’ …

Service Definition Ontology



Conversational Interface Prototype

IBM Services Research - WOW Cloud Service Marketplace

(08/06/2012 21:36:10) CSM: Welcome to CSM, please tell us want kind of service you want?

(08/06/2012 21:37:1) You: I need to pay my employees

(08/06/2012 21:37:9) CSM: 2 categories were found. Here is the list:

(No.1)	Payroll
(No.2)	Payment

Please select a category.

(08/06/2012 21:37:11) You: 1

(08/06/2012 21:37:12) CSM: You selected the following categories:

Please enter your requirement:

Please enter your requirement...

Enter

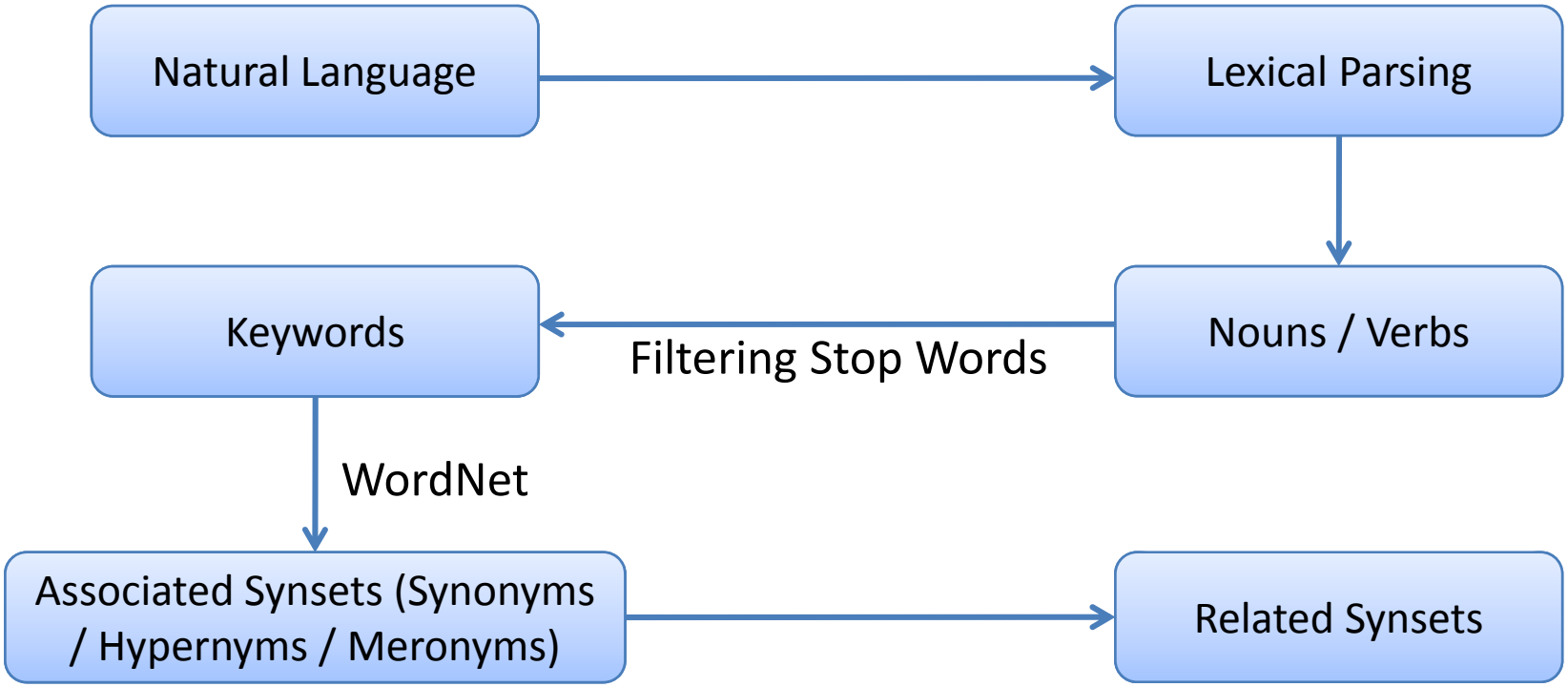
Matched Service List

ADP Payroll Mid Sized Service

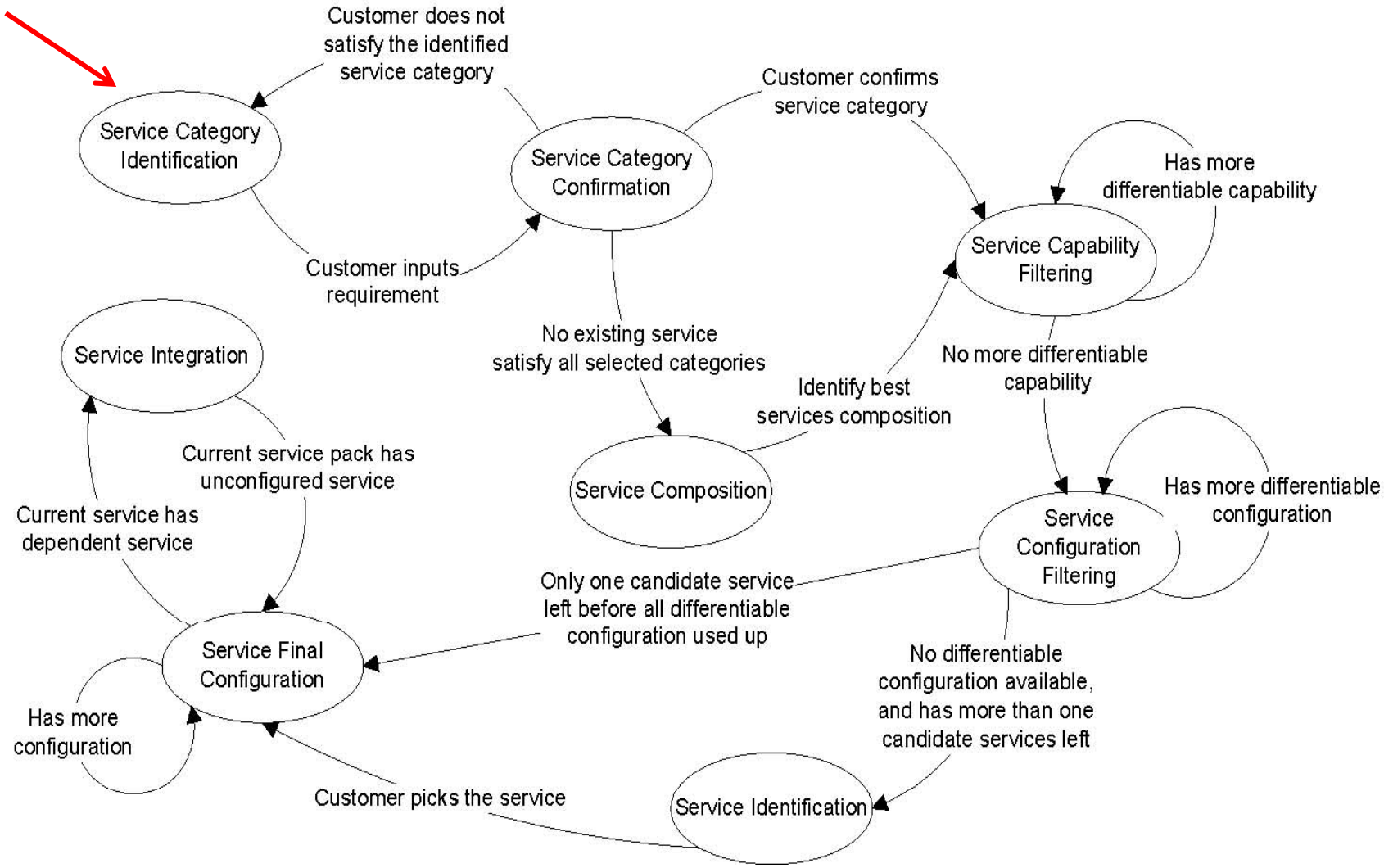
ADP Payroll Complete Service

ADP Payroll Enhanced Service

Requirement Parser



Logic Flow for Service Acquisition



Candidate Service Filtering

- Utilize common capability and configuration as filtering condition to narrow down candidate services search space.
- Capability: Describe what the services can do
- Configuration: Describe the selectable choices of the services
- Build inverse index of the capabilities/configurations, and pick the most effective filtering condition each time

Candidate Service Filtering Example

- Utilize common capability and configuration as filtering condition to narrow down candidate services search space.
- E.g. Filter with capability (Question: Does customer need C1?)

Service	Capabilities	Configurations
S1	C1, C3	G1(o11, o12)
S2	C1,C2	G1(o11,o12,o13), G2(o21,o23)
S3	C1,C3	G1(o11,o12)
S4	C1,C2	G1(o12,o13), G2(o21,o22)
S5	C2,C3	…
S6	C3	…
S7	C2	…

Candidate Service Filtering Example

- Utilize common capability and configuration as filtering condition to narrow down candidate services search space.
- E.g. Filter with capability (Answer: User needs C1. Next question: Does customer need C2?)

Service	Capabilities	Configurations
S1	**C1**, C3	G1(o11, o12)
S2	**C1**,C2	G1(o11,o12,o13), G2(o21,o23)
S3	**C1**,C3	G1(o11,o12)
S4	**C1**,C2	G1(o12,o13), G2(o21,o22)
S5	C2,C3	…
S6	C3	…
S7	C2	…

Candidate Service Filtering Example

- Utilize common capability and configuration as filtering condition to narrow down candidate services search space.
- E.g. Filter with capability (Answer: User needs C2. Next question: What about G1? We have o11, o12, o13)

Service	Capabilities	Configurations
S1	**C1**, C3	G1(o11, o12)
S2	**C1**,**C2**	G1(o11,o12,o13), G2(o21,o23)
S3	**C1**,C3	G1(o11,o12)
S4	**C1**,**C2**	G1(o12,o13), G2(o21,o22)

Candidate Service Filtering Example

- Utilize common capability and configuration as filtering condition to narrow down candidate services search space.
- E.g. Filter with configuration (Answer: User select o11 for G1)

Service	Capabilities	Configurations
S2	**C1,C2**	G1(**o11**,o12,o13), G2(o21,o23)
S4	**C1,C2**	G1(o12,o13), G2(o21,o23)

How to pick the next capability and configuration?

- Effectiveness Model: Quantify the effectiveness of obtaining the wanted service. Extra interaction can further reduce search space, but requires more questions.
- Find a sequence $Q = (Q_1, Q_2, \ldots, Q_n)$ of filtering conditions to rule out unsatisfied services.

$$eff(Q) = \sum_i eff(Q_i)$$

- **Goal**: Maximize the effectiveness of a sequence of questions. (user behavior unpredictable)

$$eff(Q) = \arg\max_Q \sum_i eff(Q_i)$$

- **Strategy**: Greedy, maximize effectiveness for each step.

$$eff(Q)^* = \sum_i \arg\max_Q eff(Q_i)$$

How to quantify effectiveness?

- **Intuition**: Quantify the filtering condition according to the number of candidate services that might be ruled out.

$$eff(Q_i) = \overline{n}_{prune}$$

- Capability: $eff(Q_i) = \overline{n}_{prune} = p(yes)n_{yes} \times p(no)n_{no}$
- Numeric Configuration:
  - Assume the user's input satisfy a certain distribution by calculating the number of services that can be pruned if user inputs the mean of the distribution

$$eff(Q_i) = \overline{n}_{prune} = p(input > \mu)n_{>\mu} + p(input \leq \mu)n_{<\mu}$$

- Multiple choice configuration:
  - Calculate the average number of services that can be pruned for each option

$$eff(Q_i) = \overline{n}_{prune} = \frac{1}{|Q_i|}\sum_{o \in Q_i} p(input = o)n_o$$

Semantic Query Engine

- Include the SPARQL query templates that would be used during conversation.
- Including:
  - Retrieve function via service category.
  - Retrieve service via service function.
  - Retrieve capability via function.
  - Retrieve configuration via function.
  - Retrieve configuration parameter via configuration.
  - Retrieve dependent service via given service.
  - Retrieve service profile via service.
  - …

Future Work

- System intelligence
  - Expand the semantic understanding
  - Enhance inference with OWL based reasoner

- Performance
  - Semantic query:
    - Replace Jena reasoner to support larger scale
    - Fully materialize inference triples offline

  - Distributed Storage:
    - Use distributed rdf3x

Conclusion

Cloud Services Marketplace is the appropriate ecosystem to support interactive service retrieval for customer by providing

- Conversational service acquisition
- Automatic resolution of dependencies
- Simultaneous service filtering and configuration of services