- The need for architecture driven service discovery
- Where does it fit in the SECSE methodology?
- Overview of the discovery process
- The technical framework
- Example (Demo)
Need for architecture-driven service discovery

architecture-driven service discovery =

discovery during service-centric systems design

WHY?

- To discover services that fit with elaborated system architecture and detailed design models, and constraints
- To identify which parts of system functionality will need to be implemented due to absence of services
- To support the formulation (and re-formulation) of architecture models of service-centric systems based on available services
ASD starts following the formulation of an initial design model specifying the main interactions (SIM) and structure (SSM) of a service centric system (this is typically driven by requirements and services discovered using them).

ASD identifies concrete service operations that could be deployed in these models and indicates how they may be integrated with them.
Overview of the discovery process

SSM/SIM

query specification/extraction

query execution

query reformulation

binding re-integration

Candidate bindings

service/operation selection
Overview of the discovery process

SSM/SIM

- Query specification/extraction
- Query execution
- Query reformulation

Candidate bindings

- Binding re-integration
- Service/operation selection

Specified as UML class and sequence diagrams
Overview of the discovery process

Specified as UML class and sequence diagrams

Specified as UML sequence diagrams annotated by stereotypes and OCL constraints
Overview of the discovery process

SSM/SIM

Specified as UML class and sequence diagrams

Query

Specified as UML sequence diagrams annotated by stereotypes and OCL constraints

Candidate bindings

Represented within the UML models

Service/operation selection

Query specification/extraction

Query execution

Query reformulation
Architecture-driven Service Discovery

Architecture of ASD prototype

Design Tool
(IBM's RSM)

Design Models
(+ list of services & service operation bindings)

Service Registry
(eXist database)

ASD prototype

Design Tool Integration Module

queries

Query Execution Engine

service mappings & descriptions
Query are specified in reference to UML system interaction models (sequence diagrams) by:

- **Using stereotypes** which signify the role of the different messages in the query. Examples of these stereotypes are:
  - "<<asd_query_message>>" which signifies the messages that invoke service operations that should be discovered
  - "<<asd_context_message>>" which signifies messages that may impose type constraints on the operations that should be discovered

- **defining additional OCL constraints** that should be satisfied by the operations and the services to be discovered
Technical framework: query execution

Filtering phase: Find set of service operations $V^S$ that satisfy hard query constraints

Optimisation phase:
Find an optimal morphism between the operations of $Q$ and $V^S$ that minimises an aggregate operation distance defined as:

$$\sum_{(v^Q_i,v^S_j) \in O(V^Q,V^S)} D(F,v^Q_i,v^S_j)$$

where:
- $D(F,v^Q_i,v^S_j) = \sum_{r \in F} w_r d_r(v^Q_i,v^S_j)$
- $d_r$ are distances defined over descriptions of operations appearing in different service facets which currently include:
  - an operation signature distance ($d_{f-sig}$)
  - operation constraint distances ($d_{f-con}$), and
  - a service behaviour distance ($d_{f-beh}$)
Example

- Design of a service centric in-car system that supports dynamic trip planning, updates to personal agendas if emerging trip requirements conflict with existing meetings, and contacting meeting attendees (given by FIAT)

Diagram:

1. Get Current Location
   - «return»
   - 2: Get Current Location : currentLocation

3. Get Location(destinationAddress)
   - «return»
   - 4: Get Location : destinationLocation

5. Calculate Trip Time(currentLocation, destinationLocation, localTime)
   - «return»
   - 6: Calculate Trip Time : arrivalTime

7: Check Schedule(userId, arrivalTime, meeting)
   - «return»
   - 8: Check Schedule : response

9: Remote Phone Connection(caller, calling, cost)
   - «return»
   - 10: Remote Phone Connection : rc

11: Set Appointment(meeting)
   - «return»
   - 12: Set Appointment : rc

Architecture-driven Service Discovery

City University
London
Conclusions

- Iterative service discovery process based on industrial modelling standards (UML) that can support both pure and hybrid service centric systems
- Flexible (and extensible) matching mechanisms with well-defined semantic foundations
- Support for integration of ASD results into system design models