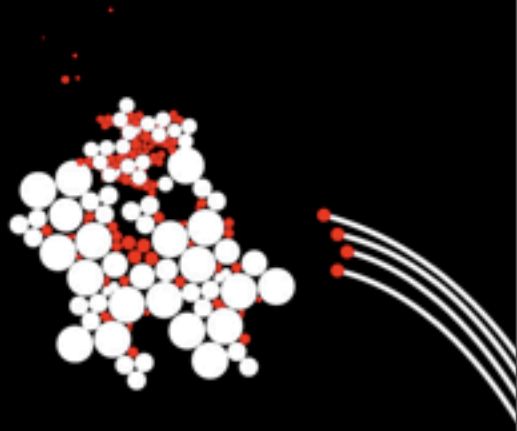
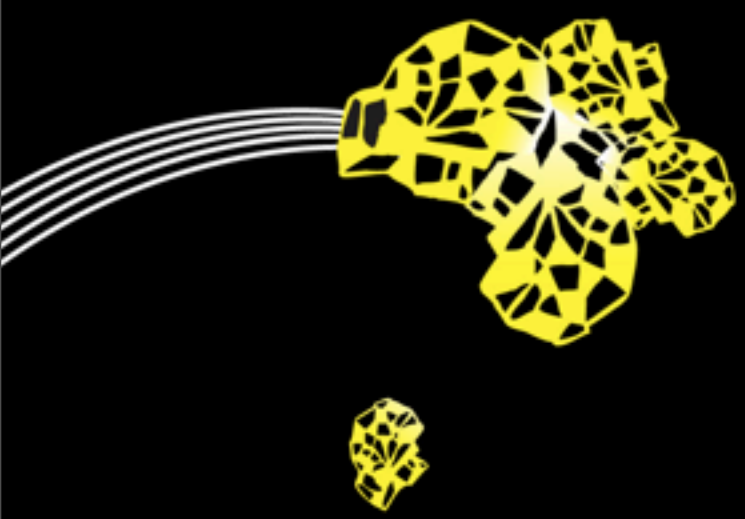
UNIVERSITY OF TWENTE.

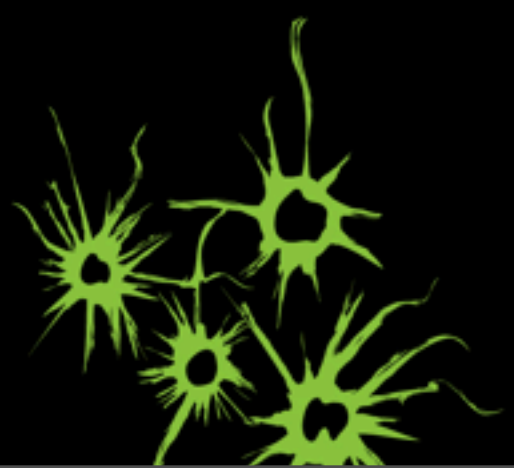


User-centric dynamic service composition

Luís Ferreira Pires, University of Twente

Joint work with Eduardo Gonçalves da Silva and Marten van Sinderen





Contents

- Vision
- Motivation
- Application scenarios
- DynamiCoS
- Evaluation framework
- User types
- Current and future work

Computing waves

There have been five computing waves since the start of computing in the 50's (source: ICT 2010 keynote presentation of Hermann Hauser)

1. Mainframes
2. Minicomputers
3. Workstations
4. Personal computers
5. Mobile (handheld) computers

- We are actually in the transition from the fourth to the fifth wave!

Trends in computing applications

Technology push

- Computers are getting smaller and faster, and are pervading into the users' environments

Market pull

- Users want to have personalised on demand service provisioning
- Service provisioning requirements are moving from 'production for the masses' to 'production for the individual'
- Special techniques have to be devised for that, since the development of different products for each individual normally does not scale!

Services on the Internet

- There are many (automated) services being offered nowadays through the Internet
- Services can be accessed through webpages (with a browser) or via programmatic interfaces (web services)
- The number of services available on the Internet are expected to grow exponentially in the next years

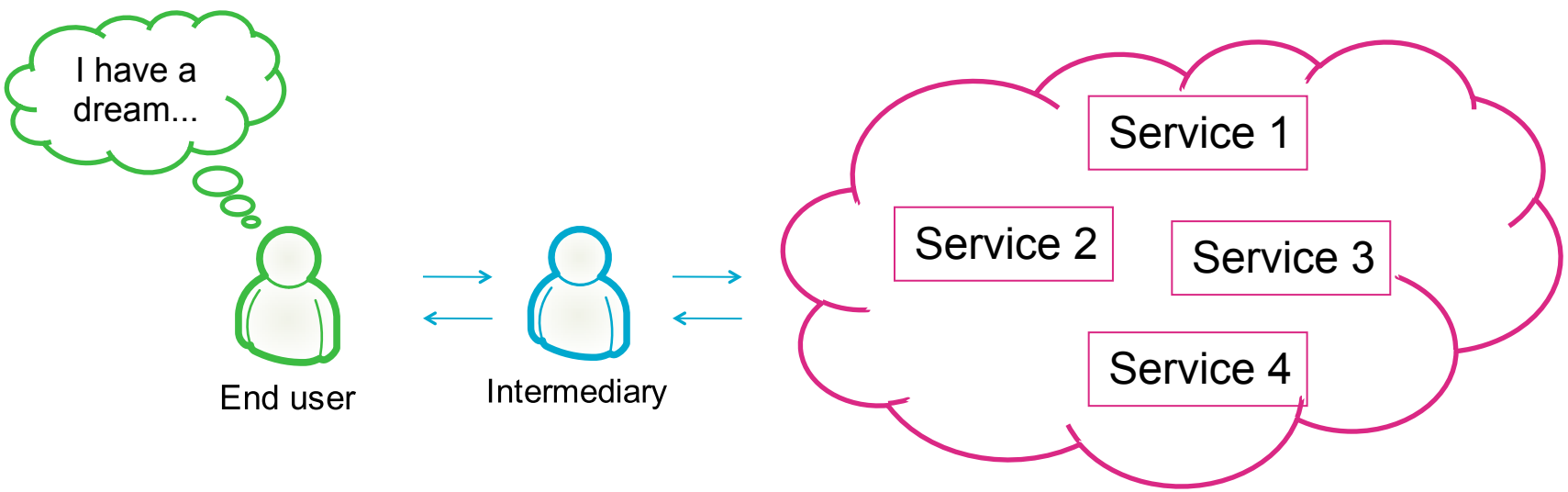
- Although there are many services available, personalisation and on demand requirements of end-users may adaptation of available services to the goals of the end-users

Services proliferation

- Proliferation of network-based context-aware resources and applications available as services
- Smaller and faster devices and the availability of fast networks allow users to access these services from any place at any time

Intermediaries

- Intermediaries between end-users and available services should be able to adapt these services to the goals of the end-users



User-centric service provisioning

Intermediaries are often necessary to

- Find the proper services
- Adapt the services according to the users goals
- Compose the services in case no single service supports the users goals

Examples

- User wants to eat at a restaurant close to his location and go to the theatre afterwards
- An elderly person living alone has his vital signs monitored and warnings are issued if he needs to change position or take some medicine

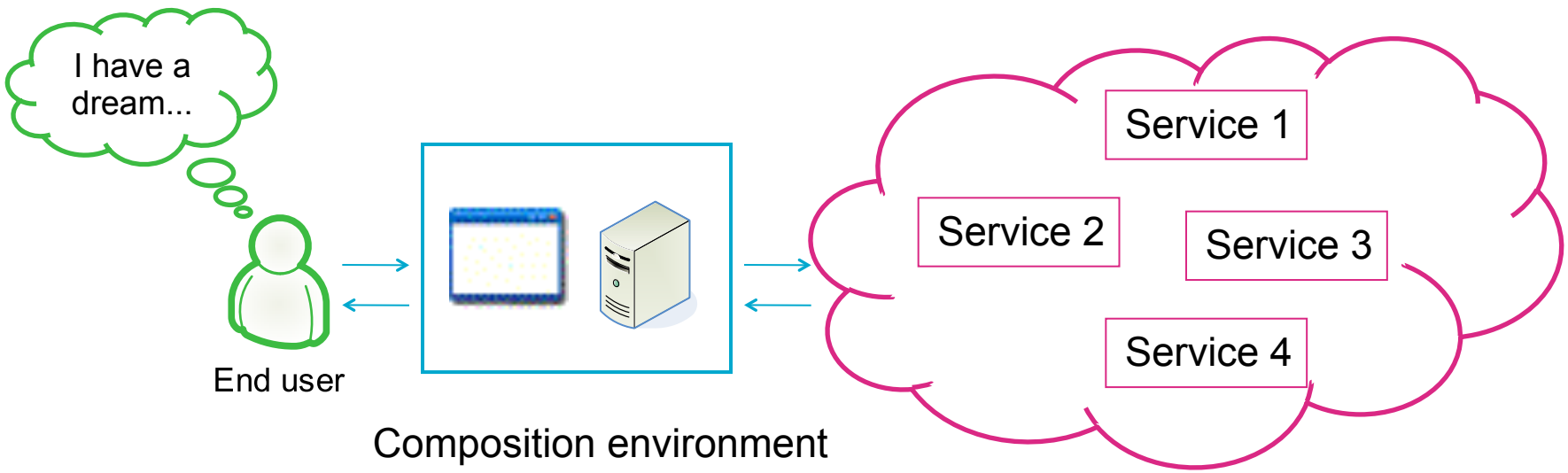
User-centric service composition

- A lot of potential (social and economical) benefits have been identified for service composition
- There has been a lot of interest from research to service composition
- Interest is shifting now towards user-centric and dynamic service composition

Our research goal

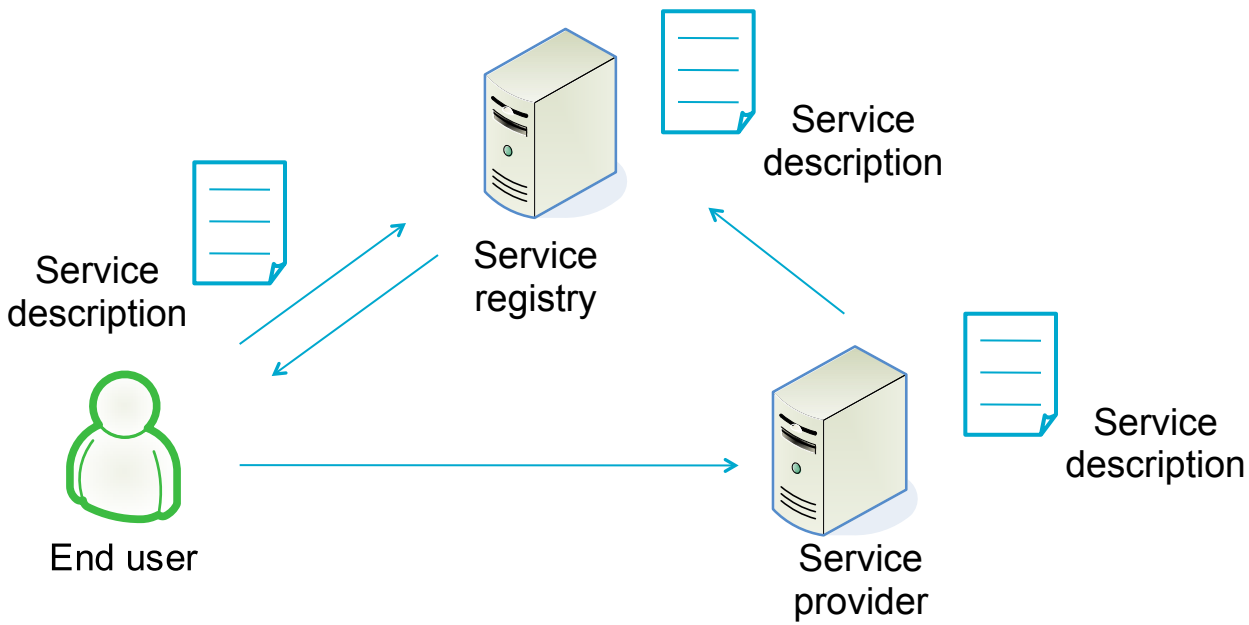
Devise mechanisms to compose services on demand (dynamic composition) taking into account the goals of individual services (user-centric) in order to achieve personalised (user-centric) service delivery

User-centric dynamic service composition



Some basic assumptions

- We rely on the design principles of Service-Oriented Architecture (SOA)

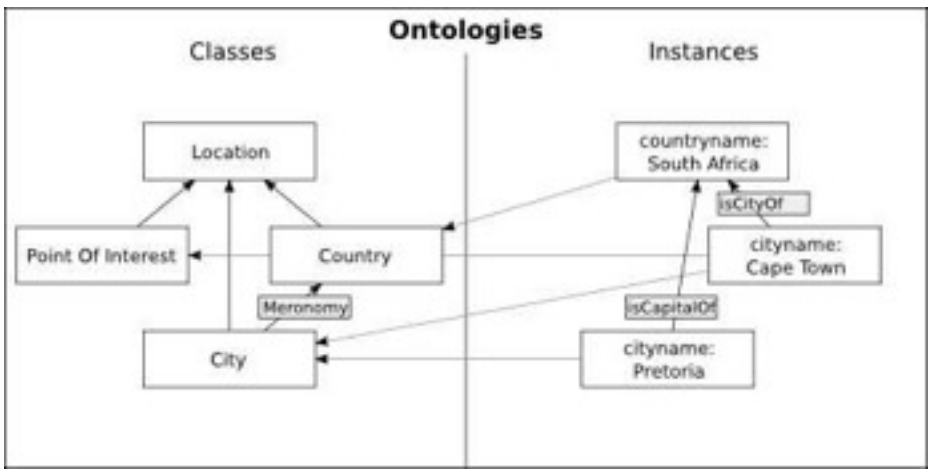


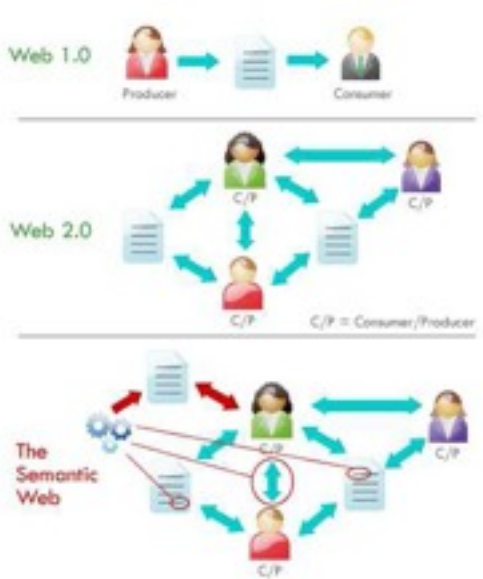
Some basic assumptions

- We aim offering an ‘as transparent as possible’ experience to the user with respect to the underlying technologies → automatic service composition

Some basic assumptions

- The supporting system has to be able to “understand” what the user wants and “reason” to find services that may fulfil the user requirements → ontologies and services annotated with semantics can help!





Demystifying ontologies

- Introduced by the ancient Greek philosophers who were trying to argue about the existence of things (individuals, universals, substance, etc.)
- Meant for classifying ‘everything that exists’
  → one single ontology!
- Official definition: ‘study of conceptions of reality and the nature of being’

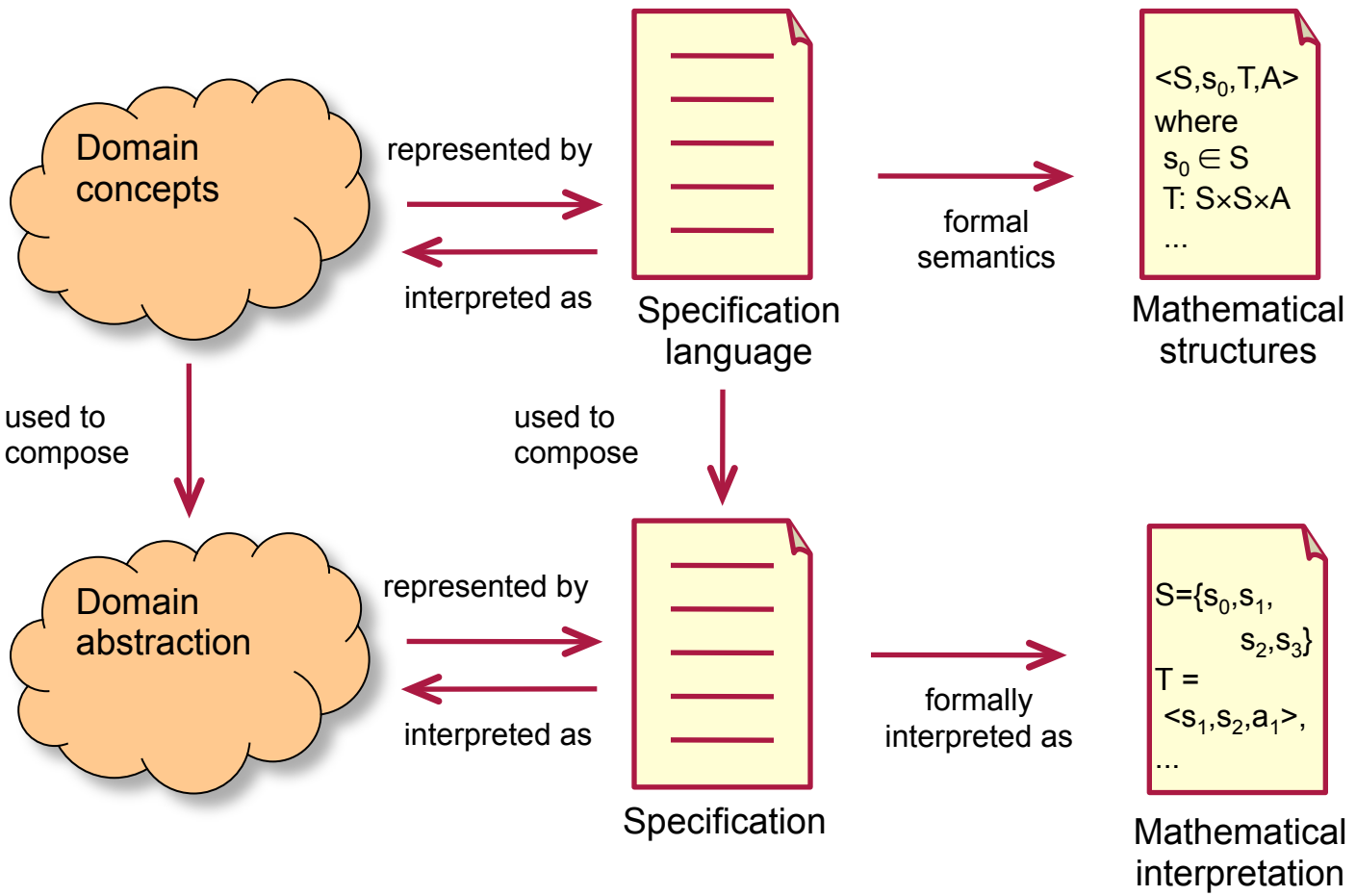
Ontologies in Computer Science

- In Computer Science
  - First introduced in the area of Artificial Intelligence to perform reasoning
  - Currently being used in many areas like Databases, Software engineering and the Semantic web

- More or less standard definition

  'A formal specification of a conceptualisation'

Ontologies and formalisation

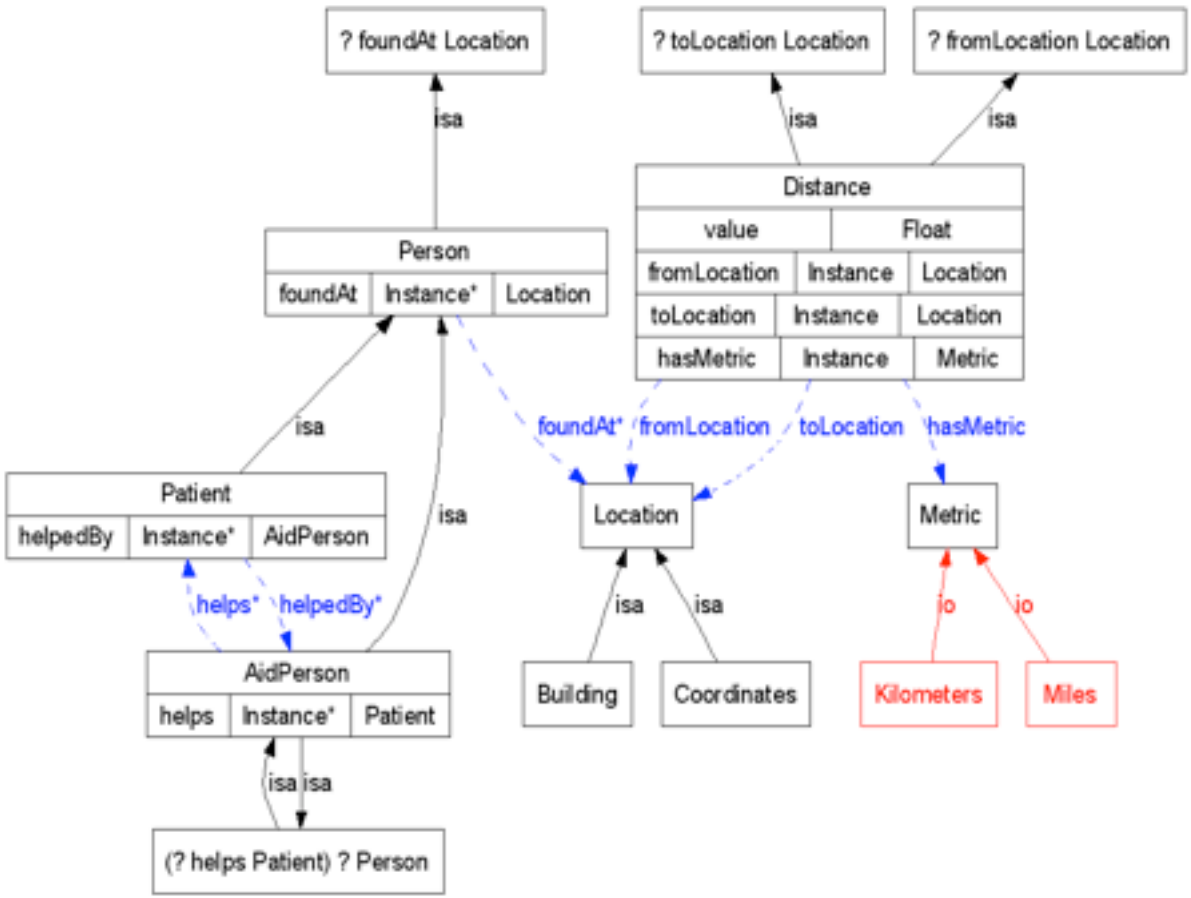


Ontology Web Language (OWL)

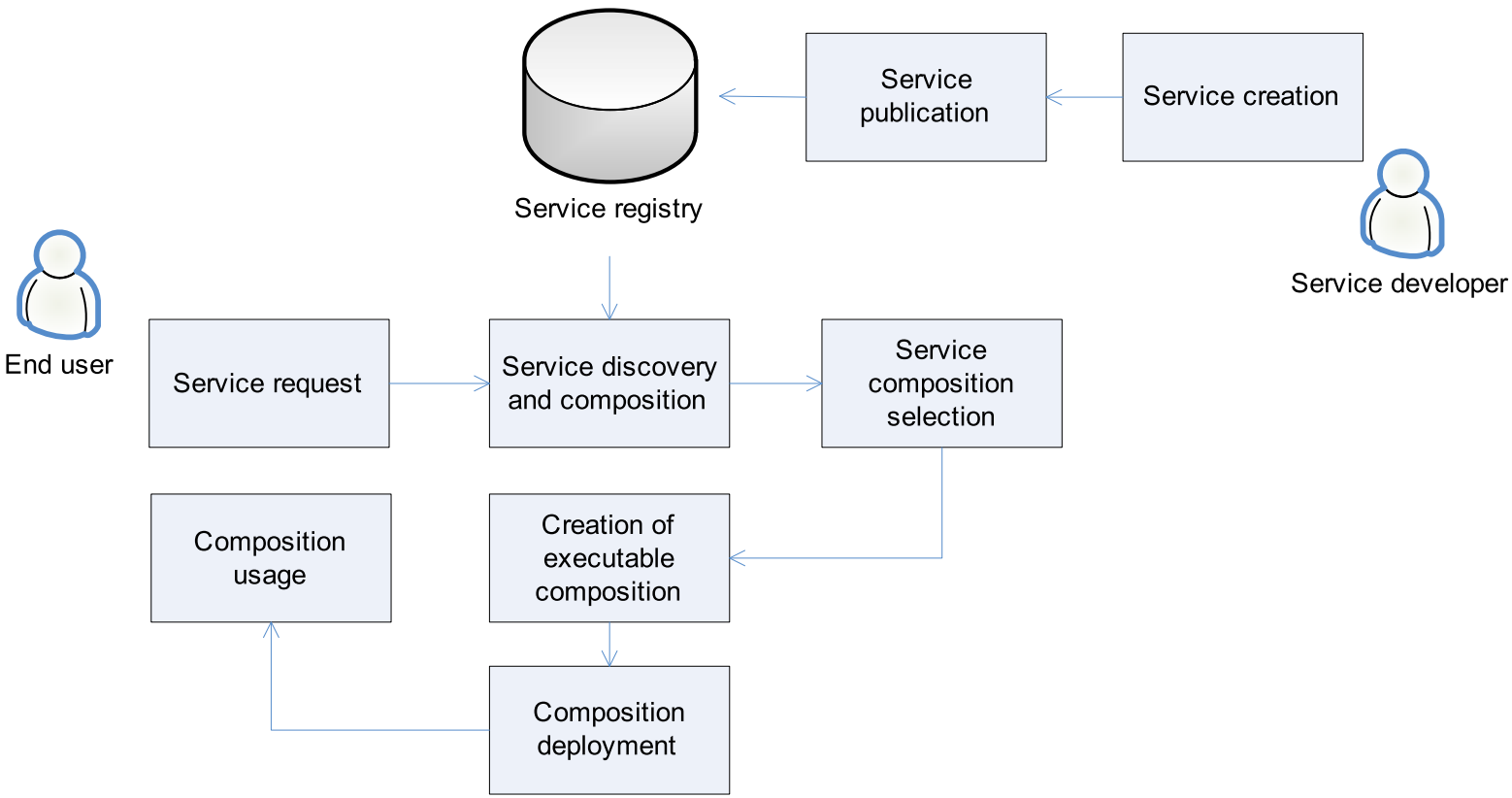
In practise, we use ontologies specified in OWL

- Developed to offer support for ontology definition (serialisation)
- OWL documents are valid XML documents
- OWL has been buildt on top of RDFS
- Meant to be intuitive for human beings
  → based on frame-based and object-oriented languages
- OWL has three variants, namely OWL Lite, OWL DL and OWL Full

Ontology (fragment) example: Person and Location



Dynamic service composition life-cycle



Stakeholders

Two main stakeholders to consider

End-user

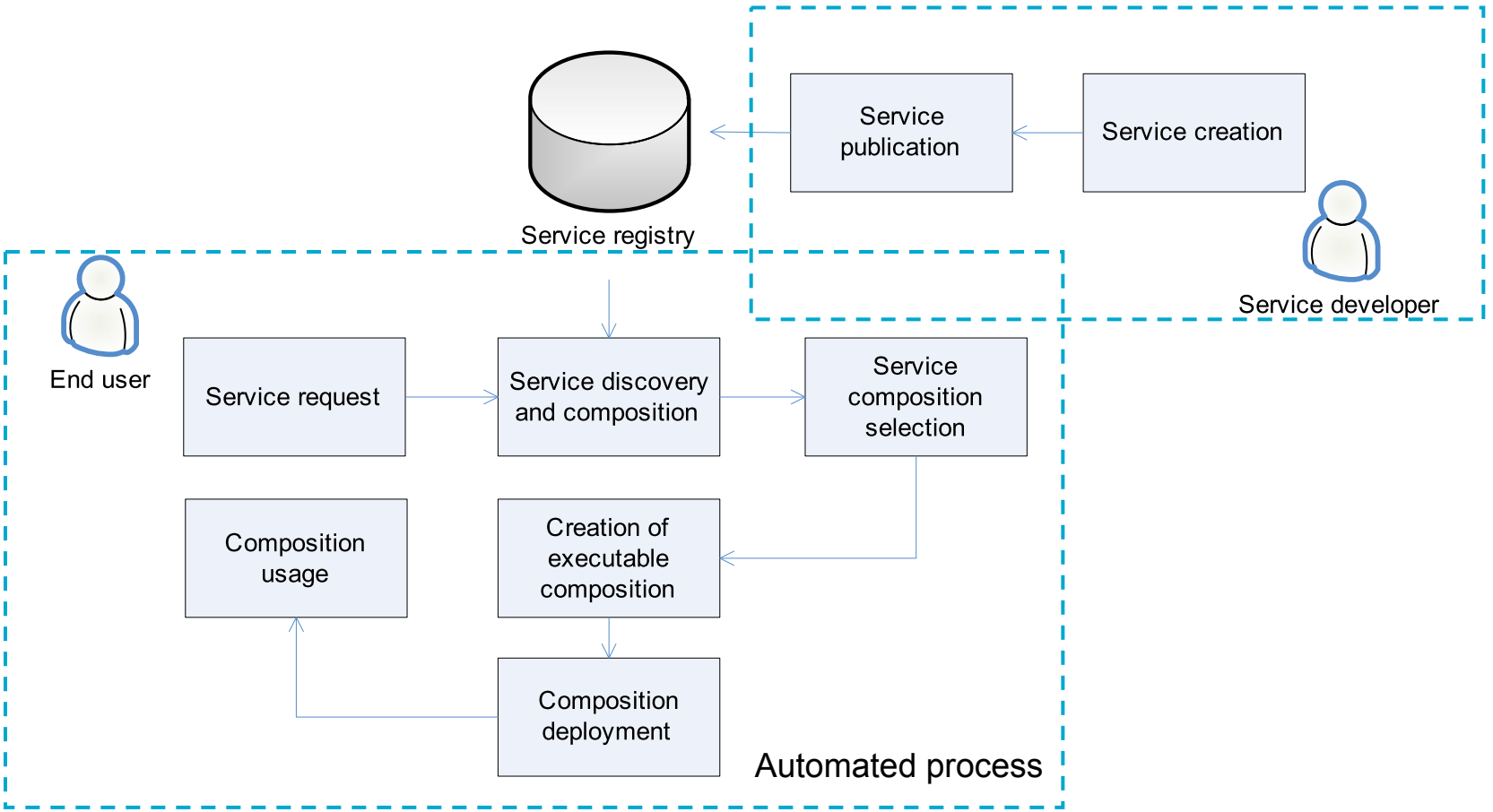
- Individual who is supposed to use the service being composed

Service developer

- Responsible for creating and publishing candidate component services (services that serve as components in a composition)

- Automated service composition is performed to fulfil the requirements of the end-user

Dynamic service composition life-cycle



DynamiCoS framework

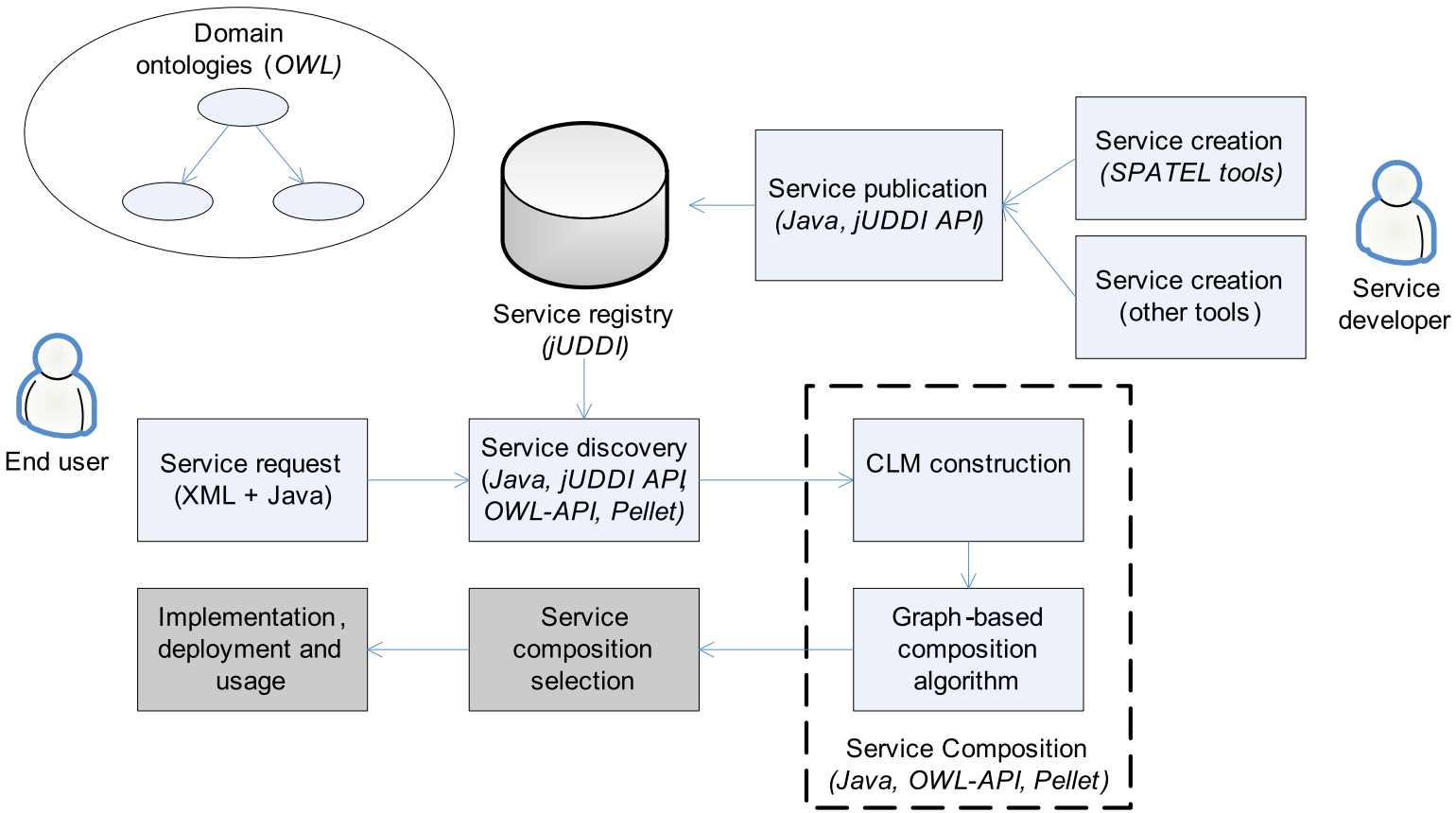
- Framework developed at the University of Twente to support dynamic service composition
- Originated from our contribution to the IST FP6 SPICE project http://www.ist-spice.org/
- Developed to support the dynamic service composition life-cycle discussed before

DynamiCoS framework: design principles

Design principles

- Neutral with respect to service and service composition description languages
- Use ontologies to help automate parts of the composition process
- Pre-processing of service connections by using a Causal Link Matrix (CLM), extended with information about non-functional properties of services
- Graph-based algorithm for automatic service composition

DynamiCoS overview

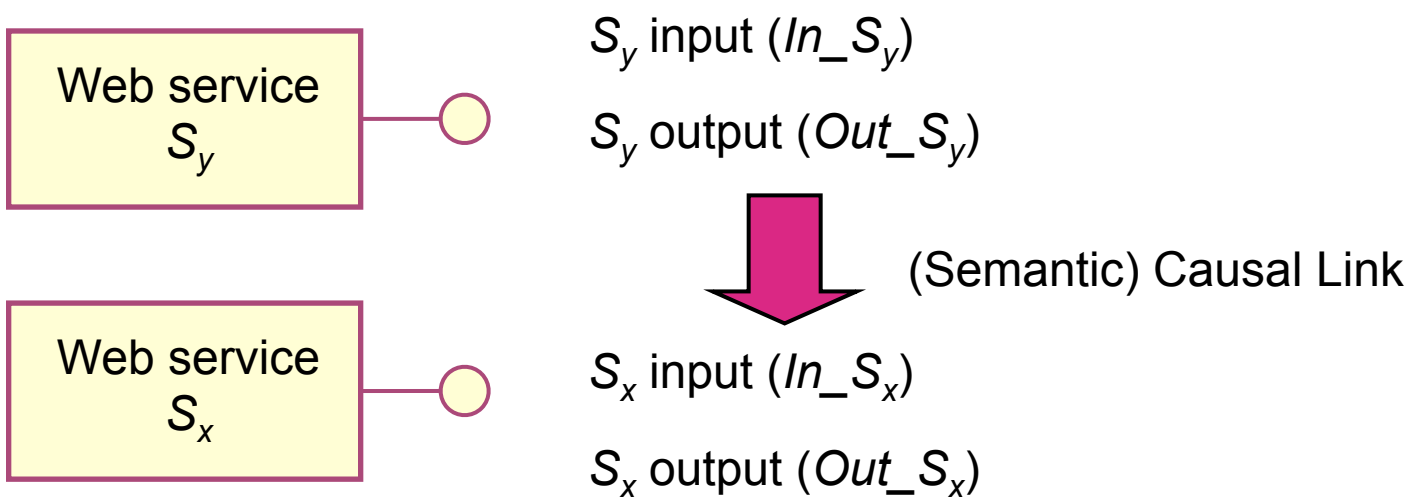


Semantic annotations

- For time being limited to request-response component services
  → ignore more complex interface behaviours
- Annotations of functional and non-functional aspects:
  - Inputs (I)
  - Preconditions (P)
  - Outputs (O)
  - Effects (E)
  - Goals (G)
  - Non-functional properties (NF)
  - Domain ontologies (Ont)

Service composition

- A service composition is in principle a sequence of services that have semantically related interfaces (outputs and inputs of consecutive services are related)



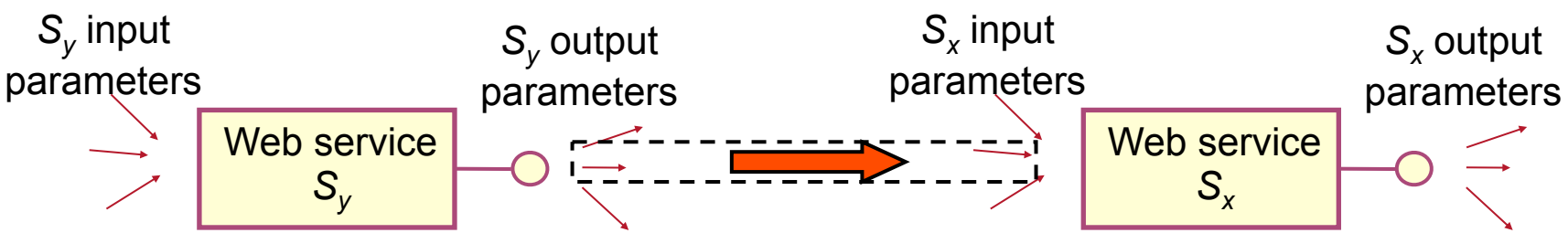
Causal link

- Causal link or semantic connection is valued by the Semantic Similarity function

  $Sim_T (Out\_S_y, In\_S_x)$

- Given outputs and inputs of two services, the semantic similarity function returns their semantic matching (Exact, Plug-in, Subsume or Fail)
- A causal link is formally defined as the triplet

  $(S_y ; Sim_T (Out\_S_y, In\_S_x) ; S_x)$



extended Causal Link Matrix (CLM+)

- CLM+ is a CLM [Lécué et al 2006] extended with Non-Functional properties
- A CLM+ is defined as
  - Rows (service inputs): r_i are labeled by $Input(S_{WS})$ described in the ontology T
  - Columns (service outputs): c_i are labeled by $Output(S_{WS})$ described in the ontology T
  - Each entry $m_{i;j}$ of a CLM M is defined as a set of triplets $(S_y, score, q_{Sy})$, where

    $score = (S_y, Sim_T (Out\_S_y, In\_S_x), q_{Sy})$

    and q_{Sy} represents some non-functional properties of S_y

extended Causal Link Matrix (CLM+)

$(S_y, score, q_{Sy}) = (S_y, Sim_T (Out\_S_y, In\_S_x), q_{Sy})$

$$\mathcal{M} = \begin{pmatrix} \emptyset & \emptyset & \{(S_1,\sqsubseteq,1)\} & \emptyset & \emptyset \\ \emptyset & \emptyset & \{(S_2,\sqsubseteq,4)\} & \emptyset & \emptyset \\ \emptyset & \emptyset & \{(S_1,\sqsubseteq,1),(S_2,\sqsubseteq,4)\} & \emptyset & \{(S_3,\equiv,1),(S_4,\equiv,3)\} \\ \emptyset & \emptyset & \emptyset & \emptyset & \{(S_3,\equiv,1),(S_4,\equiv,3)\} \\ \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \end{pmatrix}$$

$Input(S_{WS})$

$Output(S_{WS})$

Semantic graph-based composition

- Graph-based composition algorithm using the services represented in the CLM+
- Compositions are constructed ‘backwards’ from requested Outputs to requested Inputs
- Composition alternatives are represented as graphs
- Requested non-functional properties can be used to prune the inappropriate branches during the composition process
- Algorithm is language-independent
  → it can be used with different description languages

Example: E-Health scenario

- Insurance companies and hospitals aim at provide support to users with some health problem requiring assistance

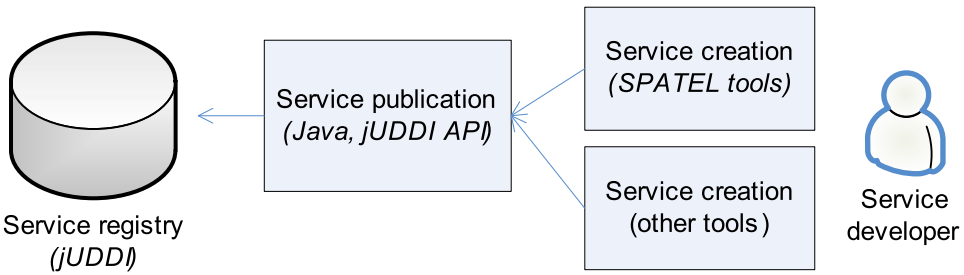
Services

- *FindHospital*, which finds the nearest hospital given a location
- *FindDoctor*, which finds a doctor given a hospital and a medical speciality
- *LocateUser*, which locates a user given his telephone location
- *MakeMedicalAppointment*, which makes an appointment between a patient and a doctor of a specific hospital

Service request

- Make a medical appointment at the nearest hospital

Service creation and publication



- Services are possibly created by different developers, described in different (semantic) languages
- For each language there must be an interpreter in the composition environment → SPATEL interpreter has been implemented
- Language-neutral service representation in the framework

  s = <ID, I, O, P, E, G, NF>

Service request

End user

Service request
(XML + Java)

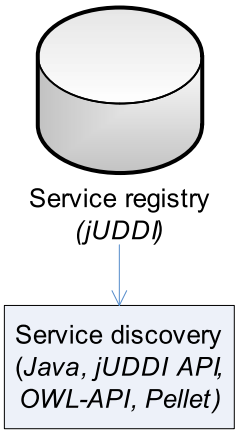
- Allows users to specify declaratively the desired service
  → (G, IOPE,NF) of the service
- (G, IOPE, NF) are semantic references to the framework ontologies
- Service request interface can vary, as long as it collects the required information
- We defined an XML-based representation for service requests

Service request: example

- Service request “Make a medical appointment in the nearest hospital” is translated to the XML fragment

```
<ServiceRequest>
  <input>IOPTypes.owl#CellNumber</input>
  <output>IOTypes.owl#MedicalAppointment</output>
  <goal>Goals.owl#FindLocation</goals>
  <goal>Goals.owl#FindHospital</goals>
  <goal>Goals.owl#FindDoctor</goals>
  <goal>Goals.owl#MedicalAppointment</goals>
</ServiceRequest>
```

Service discovery



- Service discovery based on service request parameters (*G, IOPE, NF*)
- Pure goal-based (*G*) discovery can be made
- Partial matches can be returned, such as, for example

  $RequestedConcept \supseteq DiscoveredConcept$

Service discovery: example

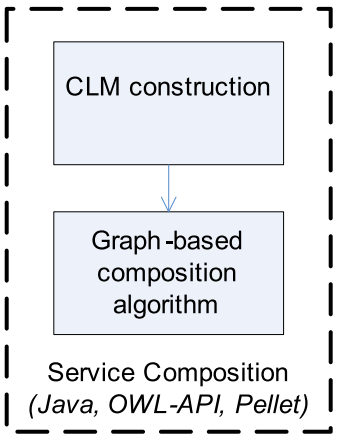
- Considering that a pure goal-based service discovery is performed, the services that semantically match the goals are retrieved

  `<goal>Goals.owl#FindLocation</goals>`
  `<goal>Goals.owl#FindHospital</goals>`
  `<goal>Goals.owl#FindDoctor</goals>`
  `<goal>Goals.owl#MedicalAppointment</goals>`

- The following set of services are discovered

Service	Input	Output	Goal
locateUser	IOTypes.owl#CellNumber	IOTypes.owl#Coordinates	Goals.owl#FindLocation
findHospital	IOTypes.owl#Coordinates	Core.owl#Hospital	Goals.owl#FindHospital
findDoctor	IOTypes.owl#MedSpeciality Core.owl#MedicalPlaces	Core.owl#Physician	Goals.owl#FindDoctor
makeMedAppointment	Core.owl#Physician Core.owl#Patient	IOTypes.owl#MedicalAppointment	Goals.owl#MedicalAppointment

Service composition



- Service composition is represented as a graph $G = (N,E)$ where N are services and E coupling between services
- Coupling (O → I) are semantic compositions or causal links (≡, ⊑, ⊒, ⊥)
- Two-step composition process
  1. CLM creation, 2. Graph composition algorithm

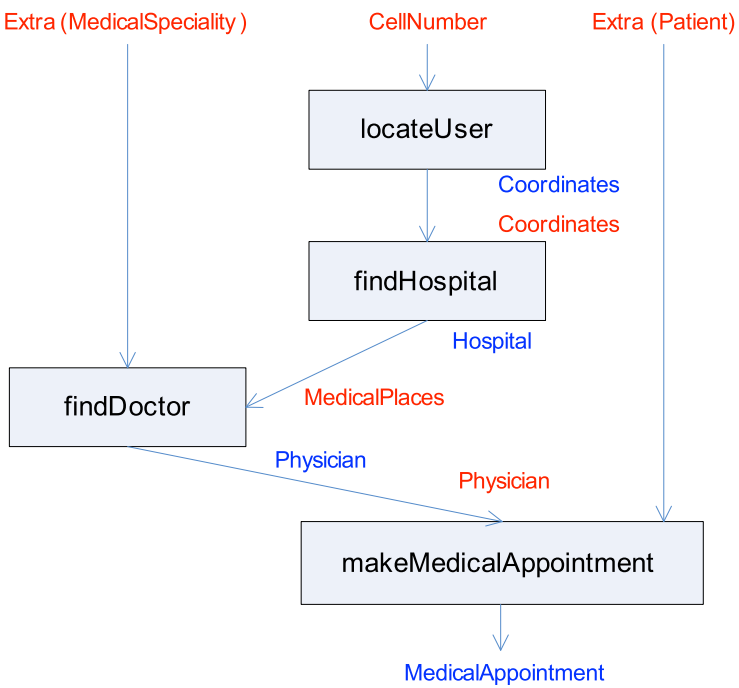
Service composition: example

- Created CLM

	cellNumber	Coordinates	MedSpeciality	MedPlaces	Patient	Physician	MedAppoint
CellNumber	0	$S1, \equiv$	0	0	0	0	0
Coordinates	0	0	0	$S2, \sqsubseteq$	0	0	0
MedSpeciality	0	0	0	0	0	$S3, \equiv$	0
MedPlaces	0	0	0	0	0	$S3, \equiv$	0
Patient	0	0	0	0	0	0	$S4, \equiv$
Physician	0	0	0	0	0	0	$S4, \equiv$

Service composition: example

- Resulting composition graph, built starting from the final goal until all sub-goals are fulfilled (backwards chaining)



Evaluation of service composition approaches

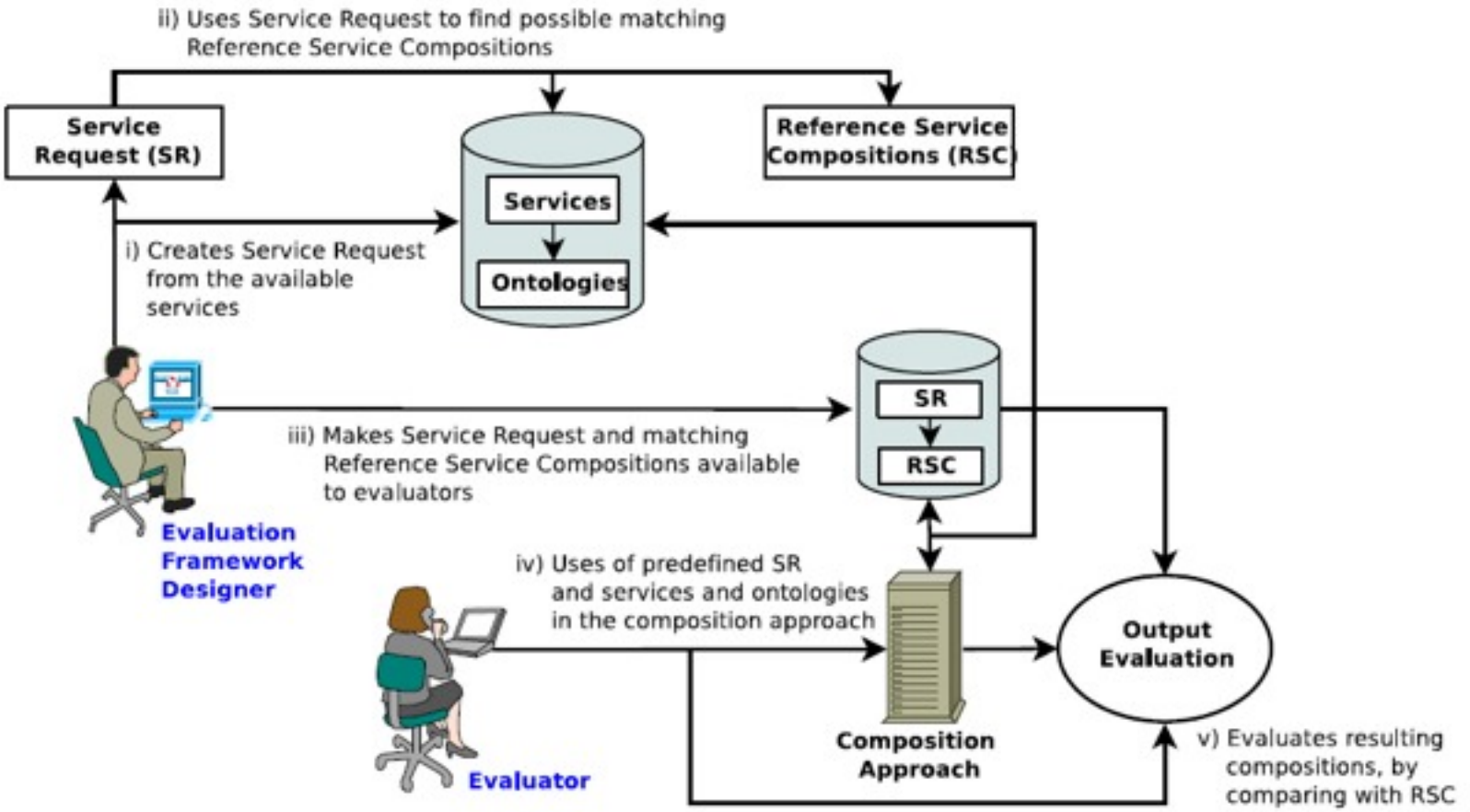
- With the emergence of the semantic web, many semantic service composition approaches appeared
  → How to evaluate them objectively?
- Nowadays we are actually evaluating these approaches in an “ad-hoc” manner
- Many times reviewers of our papers have (understandable) criticism like “Why you don’t compare your approach with approach X?...”
- We have worked for a while on a methodology to evaluate semantics-based service composition approaches

Ideas for objective evaluation

- Common service collections and ontologies to all evaluators
- Generation of common evaluation scenarios
  - Service requests
  - Matching service compositions
- Evaluation metrics: common way to report the results

- We also assume that the composition approaches are automated, i.e., they find matching service compositions given a service request, or set of requirements, specified by a user/developer

Evaluation architecture



Requirements: corpus of semantic services

- Large collections of “realistic” semantically annotated services
  → requires a collection of ontologies to annotate the semantic services

Two alternatives

1. Existing semantic services: S3-Contest (OWL-S, SAWSDL) [multiple ontologies], SWS-TC (OWL-S) [one ontology], OPOSSum (gathers existing collections and allows submit new services)
2. Automatically generated services: WS-Challenge has used automatically semantic services
   → very difficult to create “realistic” semantic services!

- Alternative 1 is being preferred and is receiving more attention

Requirements

- Service collections and service requests (SR) common to evaluators
- For each SR at least one matching reference service composition (RSC) must exist

Alternatives

1. Top-down: introduce services that yield compositions in a collection
2. Bottom-up: inspect the existing collection for compositions

Metrics

Confusion-matrix based

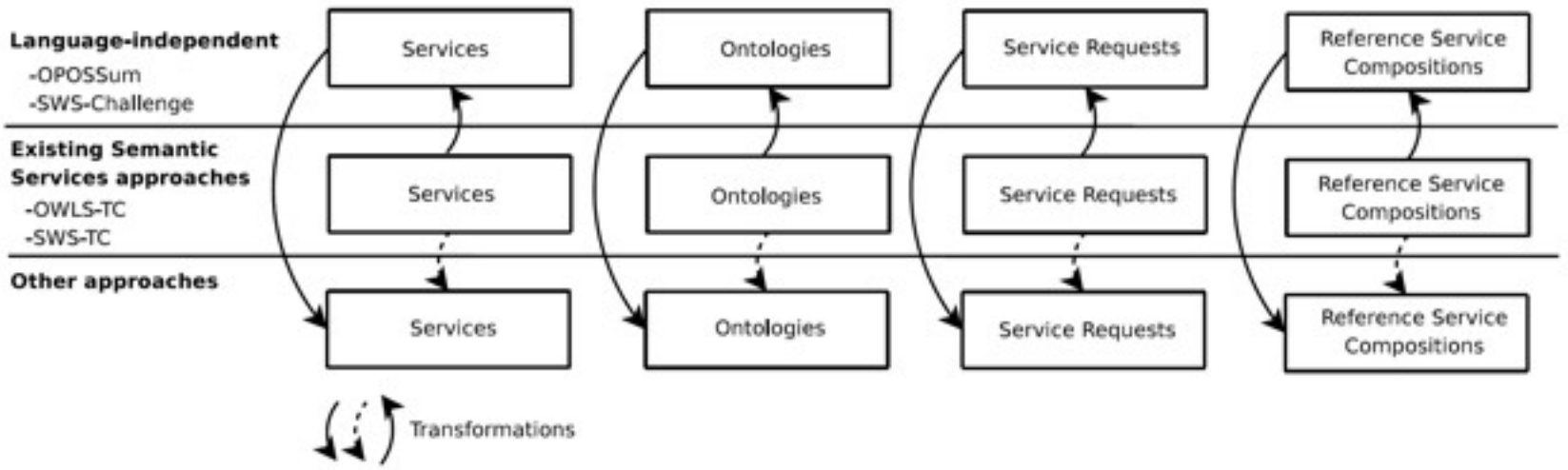
- Measure the quality of the composition found by an approach

Time-based

- Measure how fast a composition approach is, and how it scales when the number of services in the registry change

There may be other metrics, and other aspects, to consider when evaluating an approach, we focuses only on these two

Necessary artefacts



- Designer defines the reference services, ontologies, service requests and reference service requests, and evaluators have to make sure to translate them to their approach description formalisms (languages)

Confusion-matrix based metrics

		Actual Values	
		P	N
Classified Values	P'	True Positives (TP)	False Positives (FP)
	N'	False Negatives (FN)	True Negatives (TN)

$$\textit{Precision or Positive Classified Values}(PPV) = \frac{|TP|}{|TP| + |FP|}$$

$$\textit{Recall or True Positive Rate}(TPR) = \frac{|TP|}{|TP| + |FN|}$$

$$\textit{False Discovery Rate}(FDR) = \frac{|FP|}{|FP| + |TP|}$$

$$\textit{Positive Accuracy}(Acc^{TP}) = \frac{|TP|}{|TP| + |FN| + |FP|}$$

$$\textit{Signal to Noise Ratio}(SNR_{FP}^{TP}) = \frac{PPV}{FDR} = \frac{|TP|}{|FP|}$$

Time-based metrics

- It is difficult to perform time-based comparison of different approaches, amongst others due to different hardware and communication means (middleware and network)

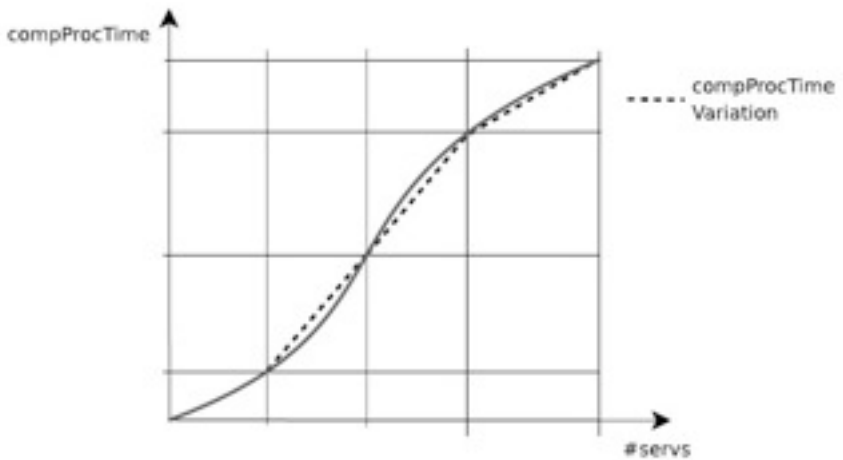
Two metrics to overcome this

1. Composition Processing Time (*compProcTime*): time taken to perform the whole process
2. Scalability: processing time variation when the number of services varies

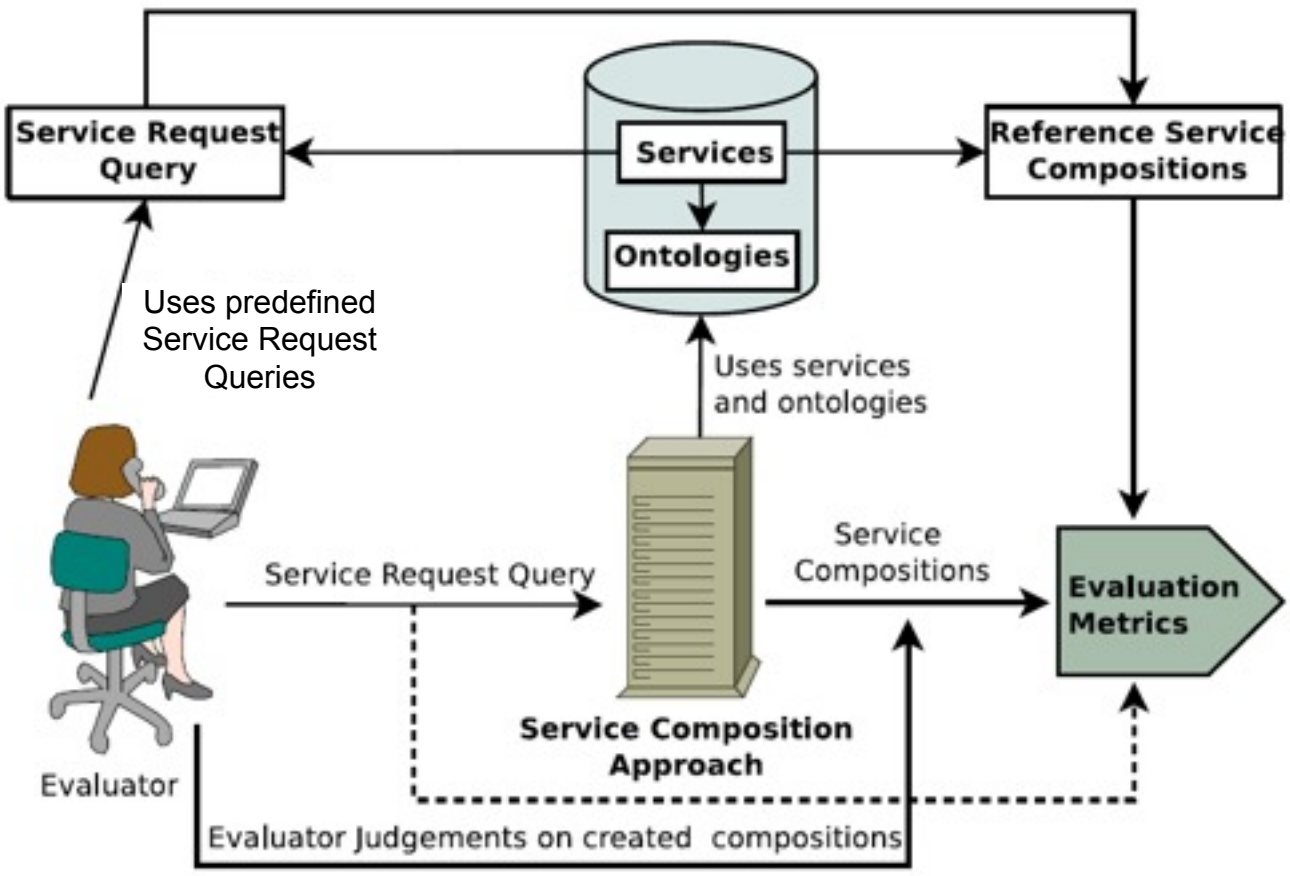
Scalability metrics

$$Scalability = \left(\frac{\partial compProcTime}{\partial \#servs} \right)^{-1}$$

$$\simeq \frac{1}{N-1} \left(\sum_{i=2}^{N} \frac{compProcTime(i) - compProcTime(i-1)}{\#servs(i) - \#servs(i-1)} \right)^{-1}$$



Evaluation methodology

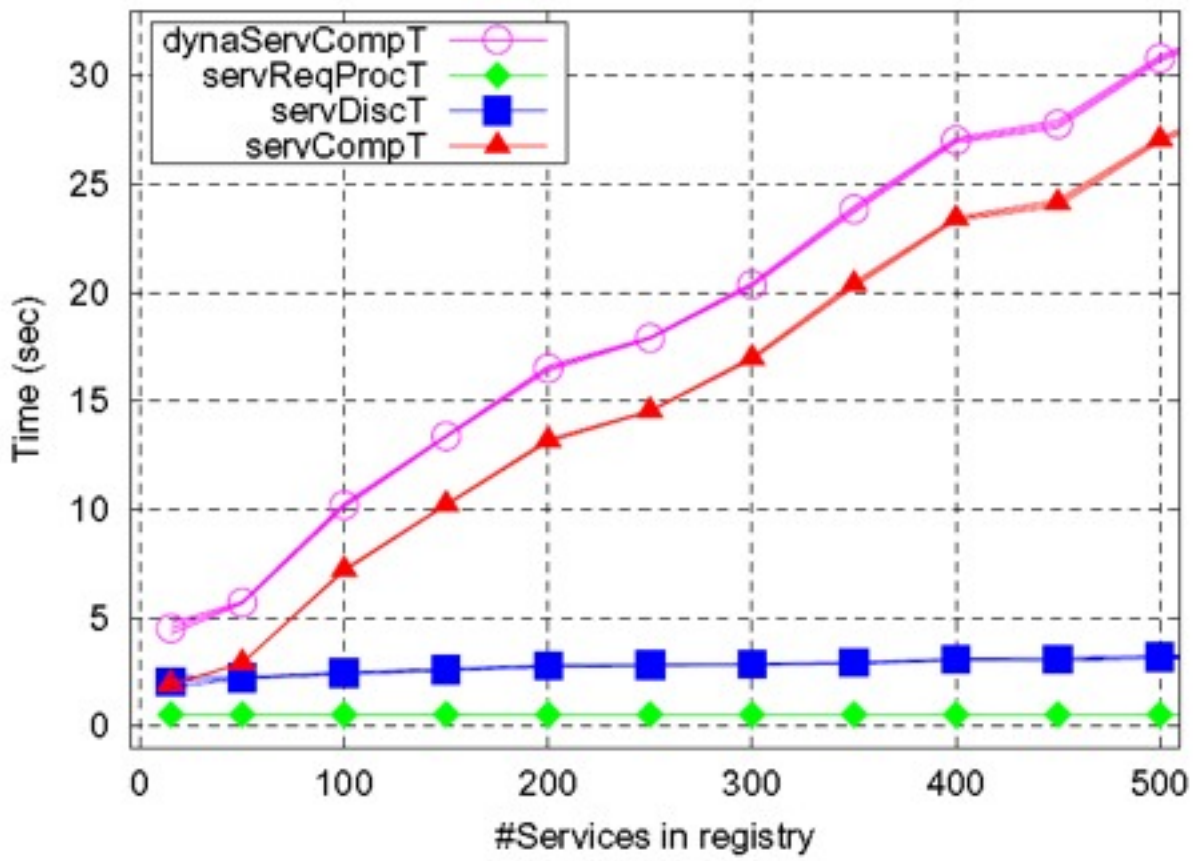


Results

- We first reported on a (fake) example to demonstrate how the framework could be applied to evaluate service composition approaches
- We lacked the man-power to do actual evaluation on existing frameworks
  - Install and learn all frameworks seems to be a big problem

- We evaluated the time-based metrics of our framework though

DynamiCoS scalability



Different end-users

- Most approaches make explicit or implicit assumptions about the knowledge and skills of the end users
- However different composition support should match these knowledge and skills → there is no 'one size fits all'
- We identified different types of users

Type of End-user	Domain Knowledge	Technical Knowledge
Layman	No	No
Domain Expert	Yes	No
Technical Expert	No	Yes
Advanced	Yes	Yes

Current work: support to different end-users

- Based on these end-users we re-designed DynamiCoS to allow different usage workflows to be supported using the same core components of the original DynamiCoS framework
- Developed two case studies (with two Master students) using different workflows
  - E-government: physically impaired citizen asking for a parking permit
  - Lifestyle: person planning a day-out (movies, shows, dinners, etc.)

- Case studies show that the approach of reusing the DynamiCoS components is feasible → reported in the Master theses of these students

Conclusions

- We have developed automated service composition support based on semantic service descriptions
- We have developed methods to evaluate automated service composition solutions and we applied some of these methods to our prototype
- We identified user types and re-designed our service composition support in order to make it more flexible
  → work in progress, to be reported more extensively soon!

Some references

- Gonçalves da Silva, E.M., Ferreira Pires, L. and van Sinderen, M.J. Towards runtime discovery, selection and composition of semantic services. Elsevier Computer Communications (in press, to appear)
- Gonçalves da Silva, E.M., Ferreira Pires, L. and van Sinderen, M.J. On the Design of User-centric Supporting Service Composition Environments. In: ITNG 2010
- Gonçalves da Silva, E.M., Ferreira Pires, L. and van Sinderen, M.J. Supporting Dynamic Service Composition at Runtime based on End-user Requirements. In: UGS 2009 (ICSOC 2009 workshop)
- Gonçalves da Silva, E.M., Ferreira Pires, L. and van Sinderen, M.J. A Framework for the Evaluation of Semantics-based Service Composition Approaches. In: ECOWS 2009
- Gonçalves da Silva, E.M., Martínez López, J., Ferreira Pires, L. and van Sinderen, M.J. Defining and Prototyping a Life-cycle for Dynamic Service Composition. In: ACT4SOC 2008 (ICSOFT 2008 workshop)