QoS-aware Service Composition in Dynamic Service Oriented Environments

December 3rd 2009
Middleware '09, Urbana-Champaign, USA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

INRIA

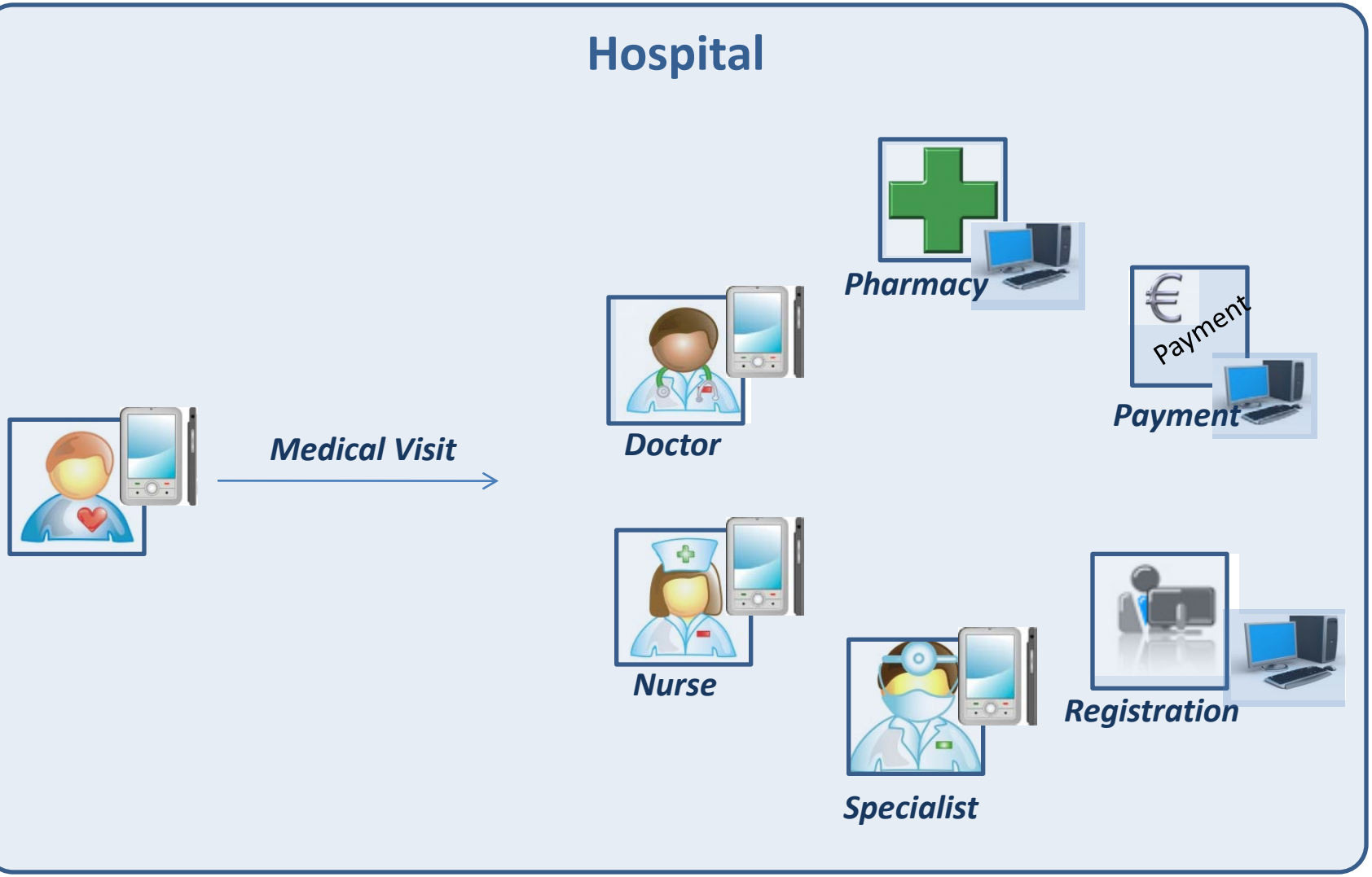
centre de recherche PARIS - ROCQUENCOURT

Nebil Ben Mabrouk, Sandrine Beauche, Elena Kuznetsova, Nikolaos Georgantas and Valérie Issarny

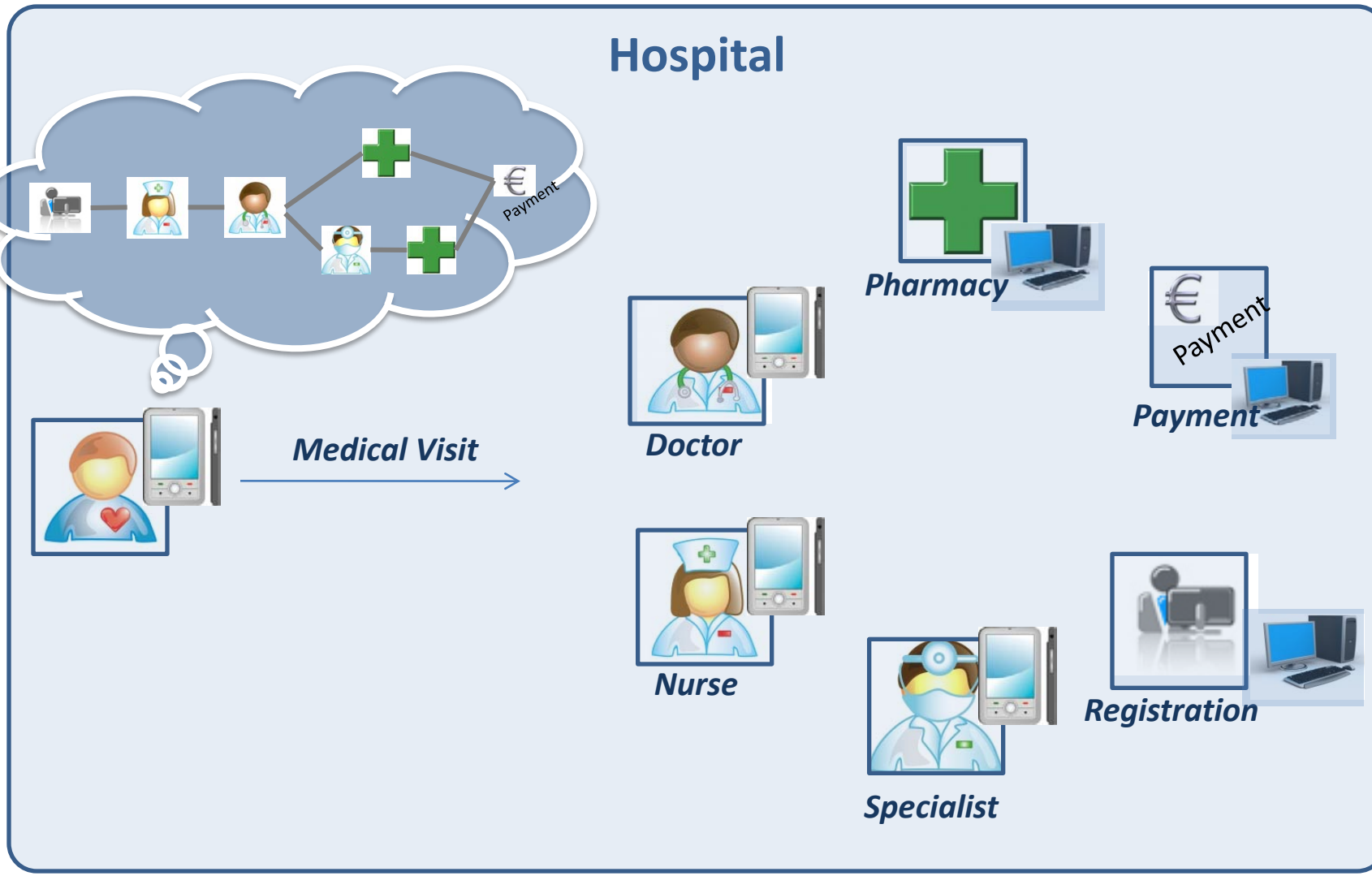
ARLES project-team
INRIA Paris-Rocquencourt France

Motivating scenario, objective and challenges

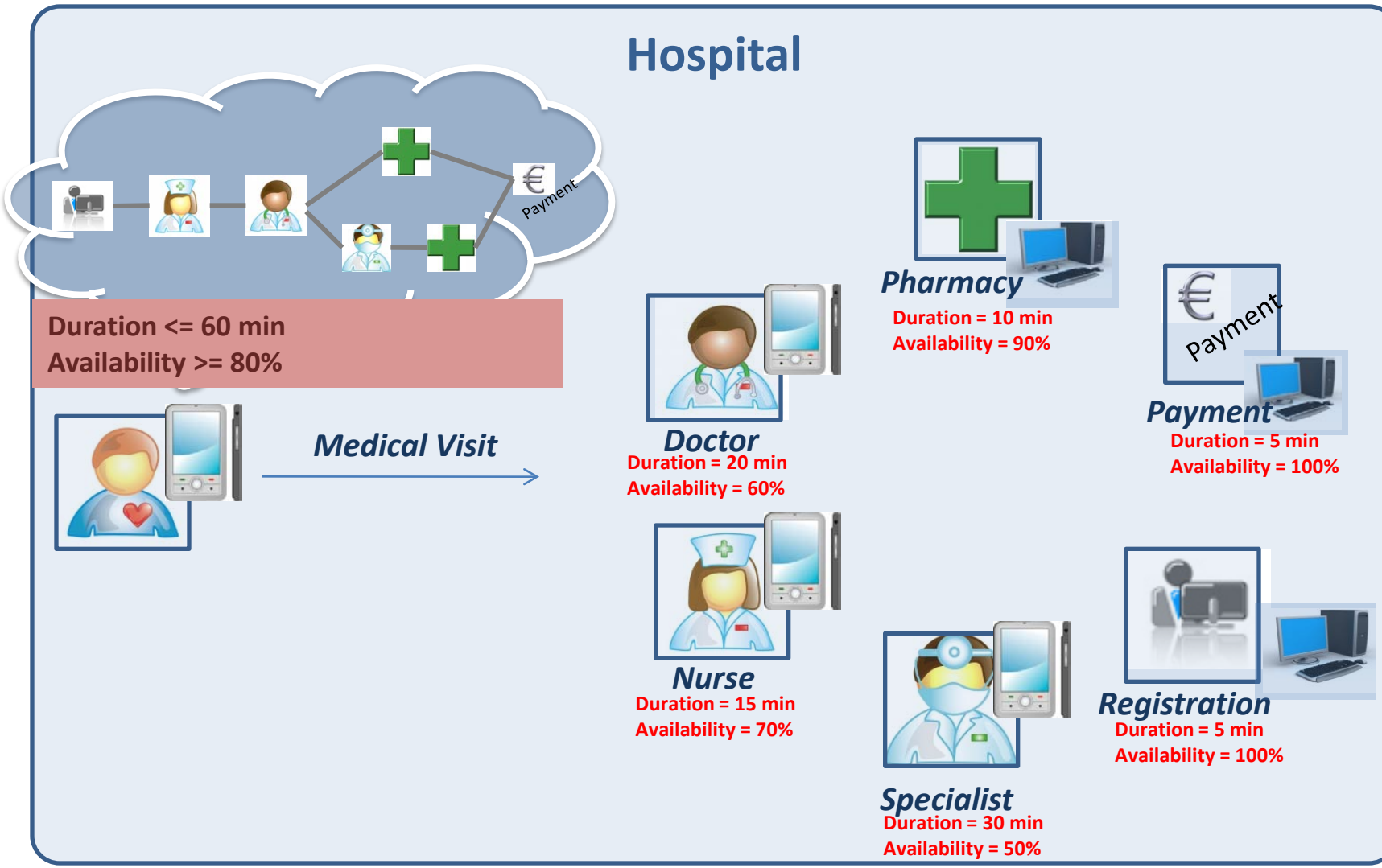
Motivating Scenario



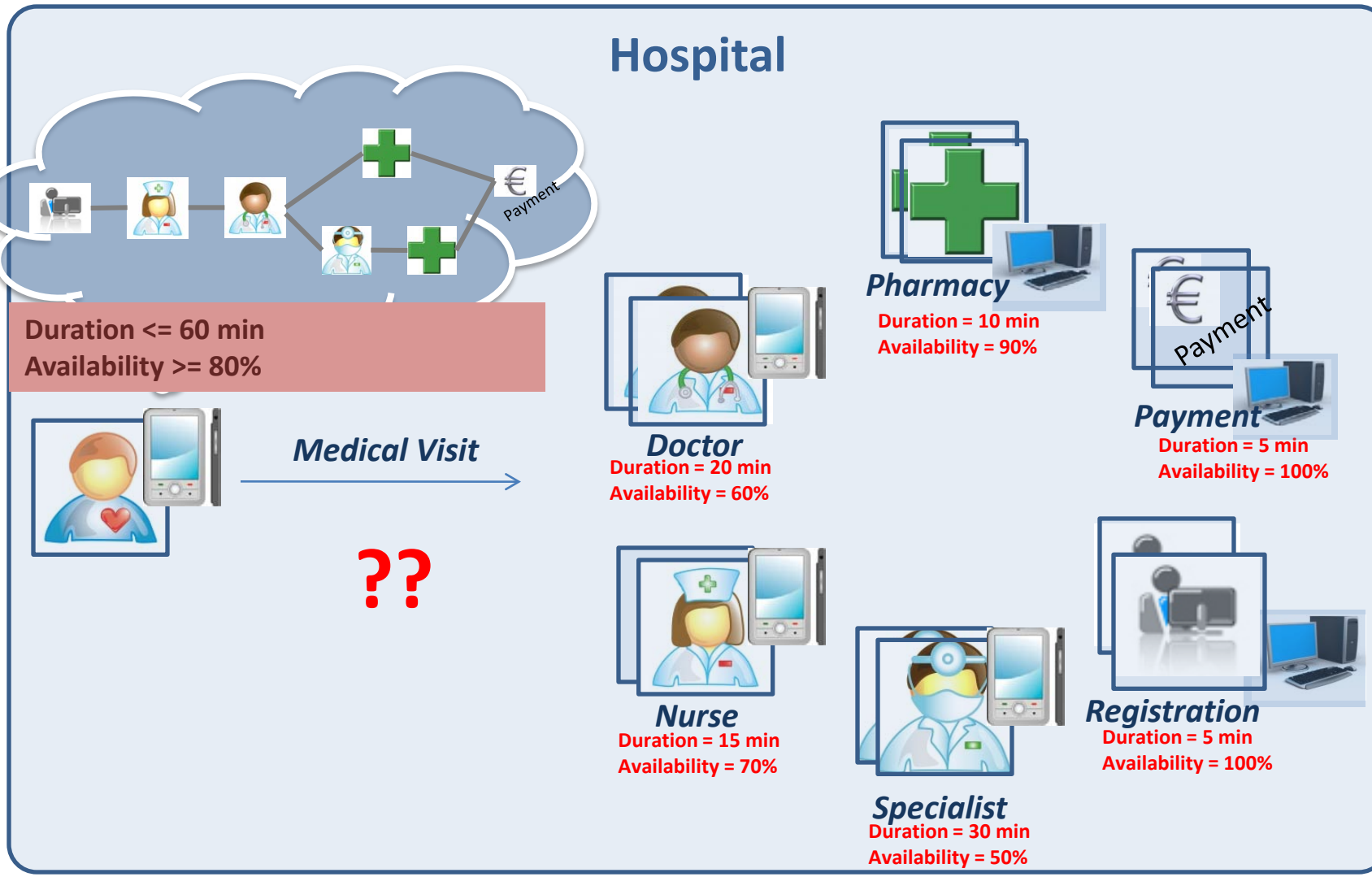
Motivating Scenario



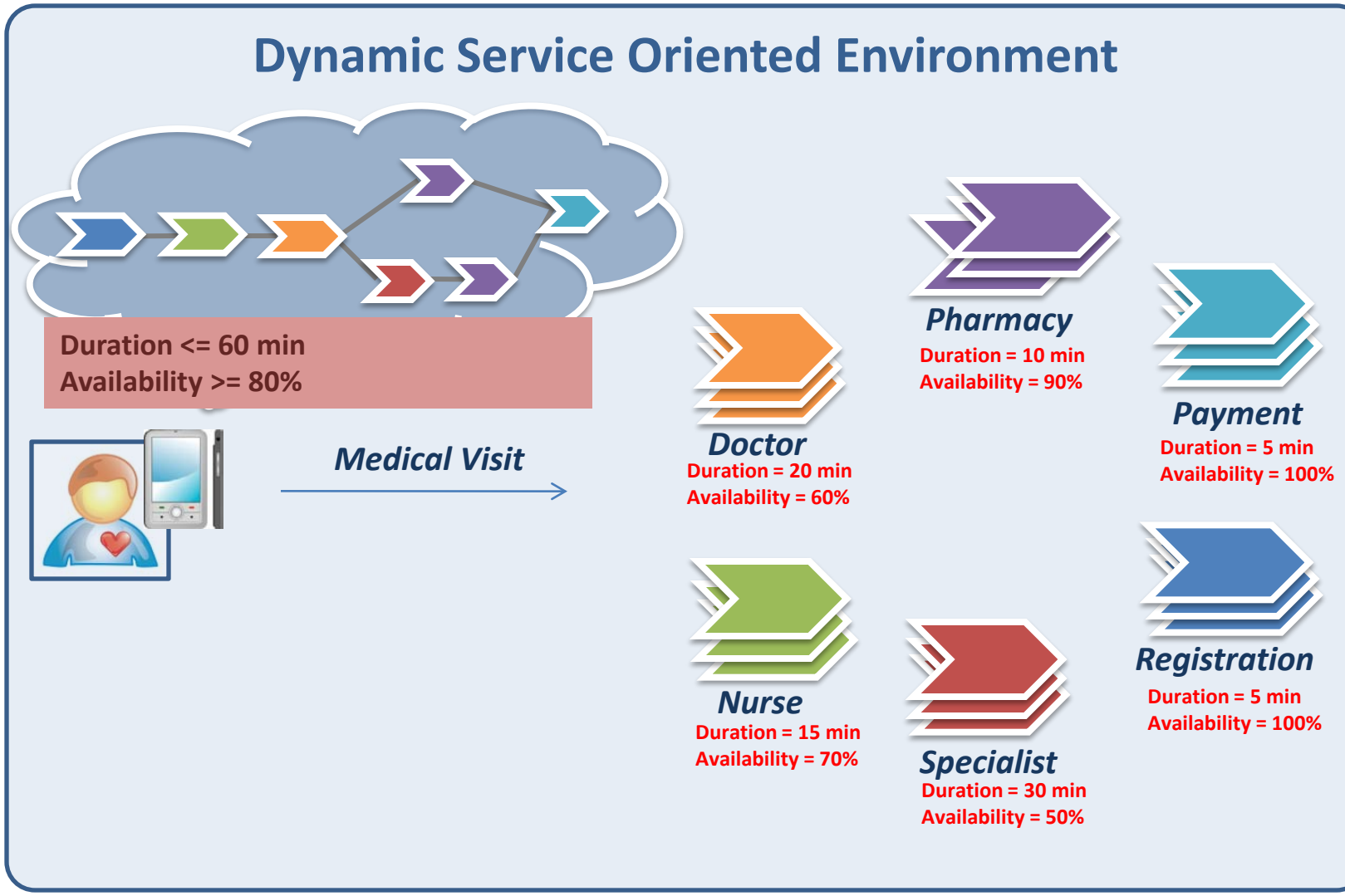
Motivating Scenario



Motivating Scenario



SOC Paradigm



Objective and Challenges

- **Objective**
  - *Selecting the **best** service compositions (i.e., in terms of QoS) able to fulfill user QoS requirements.*
- **Challenges**
  - ***Computational complexity**: QoS-aware service composition under global QoS constraints is NP-hard.*
  - ***Dynamic environments:***
    - *The user request should be fulfilled on the fly **=>** The time available for service selection is limited**.***
    - *Services can disappear or fail frequently **=>***

      Select many alternative service compositions to cope with the environment dynamics

Approach overview

QoS model

- **Generic QoS model**
  - ***Cross-domain / Domain-specific QoS properties***
    - *Cross-domain: e.g., duration, cost, availability, reliability.*
    - *Domain-specific: e.g., doctor's rating.*
  - ***Quantitative QoS properties***
    - *Negative: properties that we want to minimize (e.g., duration).*
    - *Positive: properties that we want to maximize (e.g., availability).*

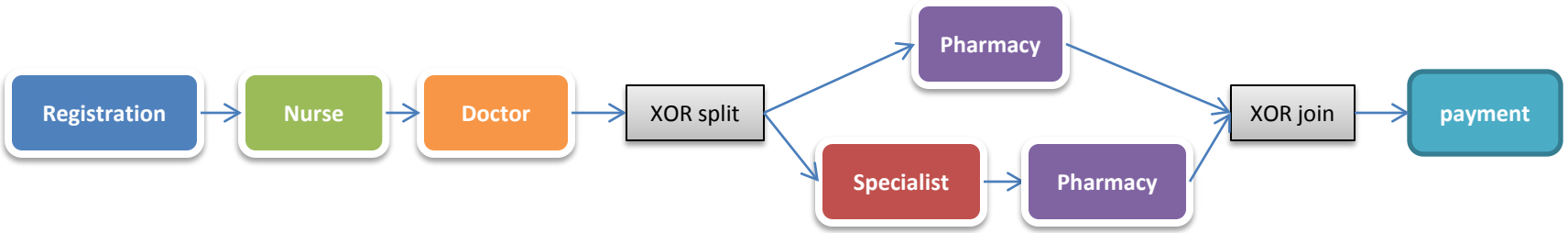
Composition model

- **Composition Patterns:**
  - ***Sequence***
  - ***AND*** *(parallel execution)*
  - ***XOR*** *(conditional execution)*
  - ***LOOP*** *(iterative execution)*
- **Computing QoS of composite service**
  - ***Pessimistic approach***

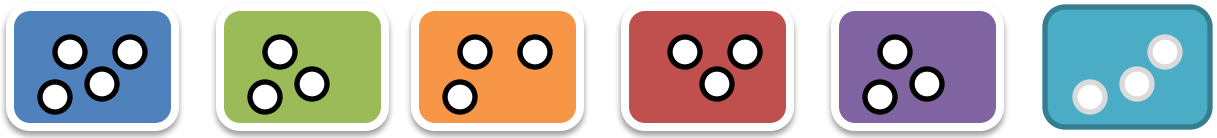
QoS attributes	Composition Patterns			
	Sequence	AND	XOR	Loop
Availability (av)	$\prod_{i=1}^{n} av_i$	$\prod_{i=1}^{n} av_i$	$min(av_i)$	av_i^k
Reliability (rl)	$\prod_{i=1}^{n} rl_i$	$\prod_{i=1}^{n} rl_i$	$min(rl_i)$	rl_i^k
Cost (c)	$\sum_{i=1}^{n} c_i$	$\sum_{i=1}^{n} c_i$	$max(c_i)$	$c \times k$
Throughput (th)	$min(th_i)$	$min(th_i)$	$min(th_i)$	$th \times k$
Duration (d)	$\sum_{i=1}^{n} d_i$	$max(d_i)$	$max(d_i)$	$d \times k$

Algorithm Overview

- **Input**
  - ***Abstract user task*** *(composed of abstract activities)*



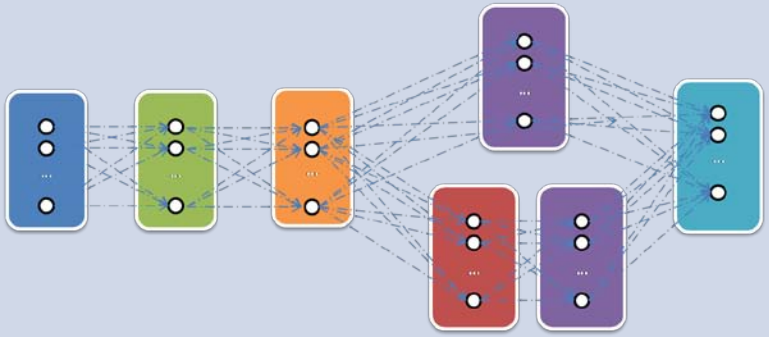
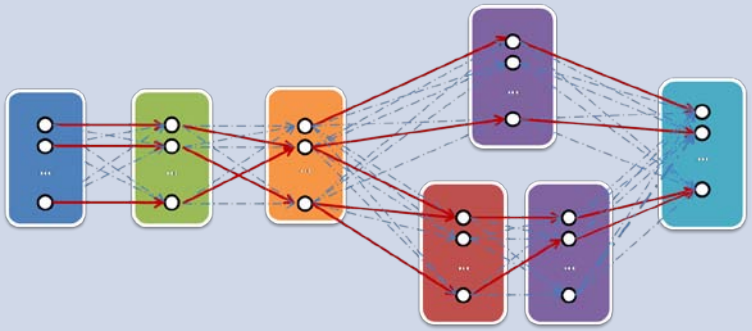
  - ***A set of service candidates for every activity in the task***



  - ***A set of n global QoS constraints imposed on the whole task***
    - $QoS_1 \leq U_1$ *(e.g., duration ≤ 60 min)*
    - …
    - $QoS_n \leq U_n$

Algorithm Overview

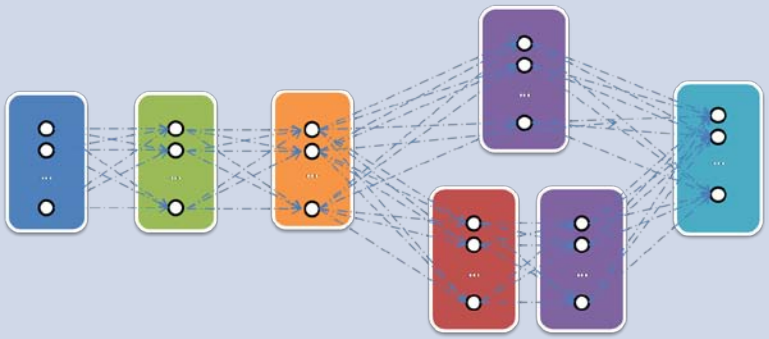
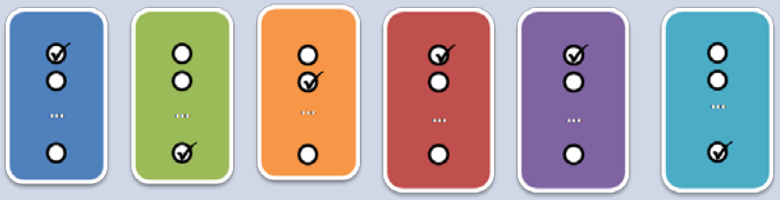
Design rationale:

Brute-force-like Algorithms	Heuristic Algorithms
	
Explores all possible compositions ✓Optimal selection × High computational cost	Explores a limited number of compositions ✓Low computational cost × Lower optimality

➢ **Brute-force-like algorithms are inappropriate for our purpose, we rather need a heuristic.**

Algorithm Overview

Design rationale:

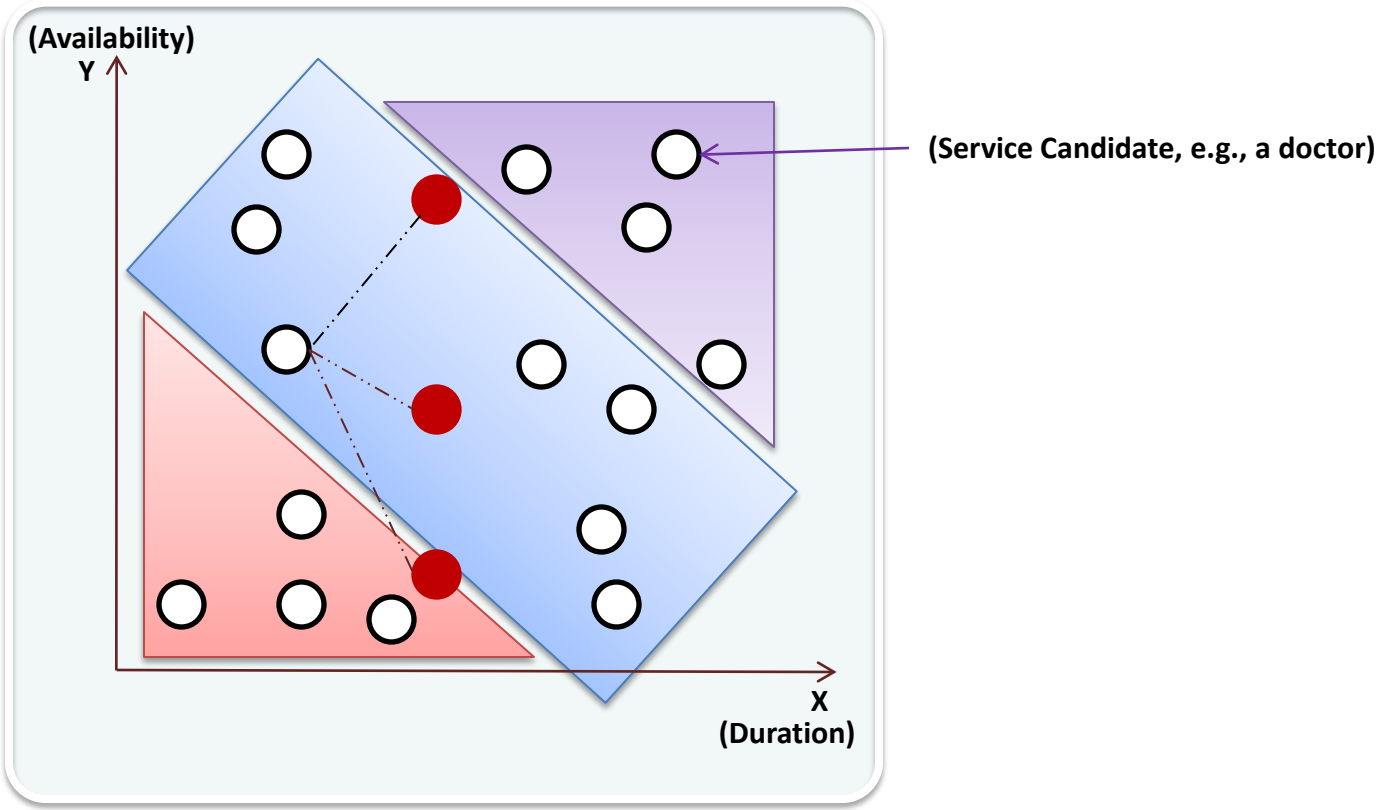
Global Selection	Local Selection
	
✓ Handles global QoS constraints × High computational cost	✓ Low computational cost × Does not guarantee QoS at the global level

➢ **Combine global and local selection approaches**

Local and global selection phases

Local Selection Phase

- **Performed for every abstract activity individually**
- **Based on clustering techniques, i.e., K-Means**



Doctor Activity

Local Selection Phase

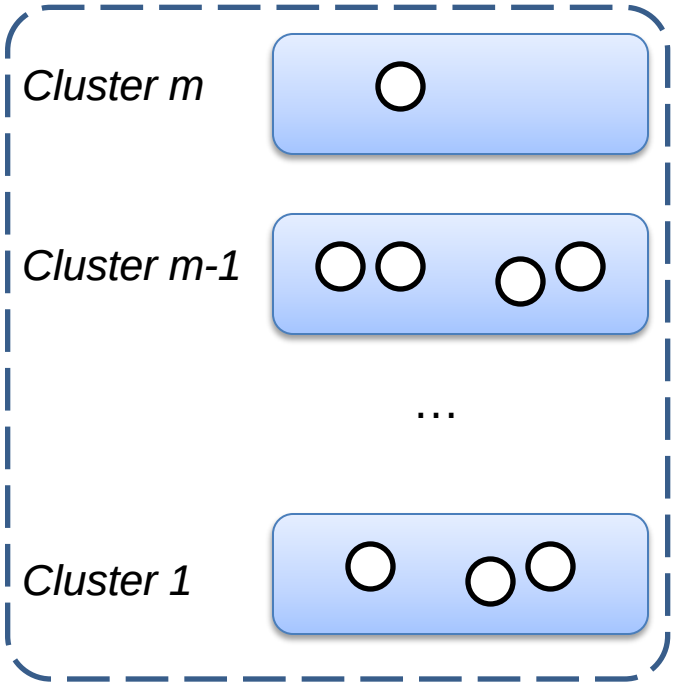
- **Proceeds through two main steps**
  1. *Scaling:*
     - *Normalizes QoS values associated with negative and positive QoS attributes.*
     - *$QoS_{Si} = \langle q_{s1}, q_{s2,..}q_{sn} \rangle$ where $0 \leq q_{si} \leq 1$ $(1 \leq i \leq n)$*
     - *All services are data points within n-dimensional $[0,1]^n$ hypercube.*
  2. *Clustering (K-Means)*
     - ***Input**:*
       - ***m:** Number of QoS levels (i.e., clusters)*
       - *Set of service candidates*
     - *Clustering based on the **n-dimensional Euclidian distance**:*

$$\sqrt{\sum_i (q_{ci} - q_{si})^2}$$

     - ***Output**: m clusters*

Local Selection Phase

- **Services' Selection**



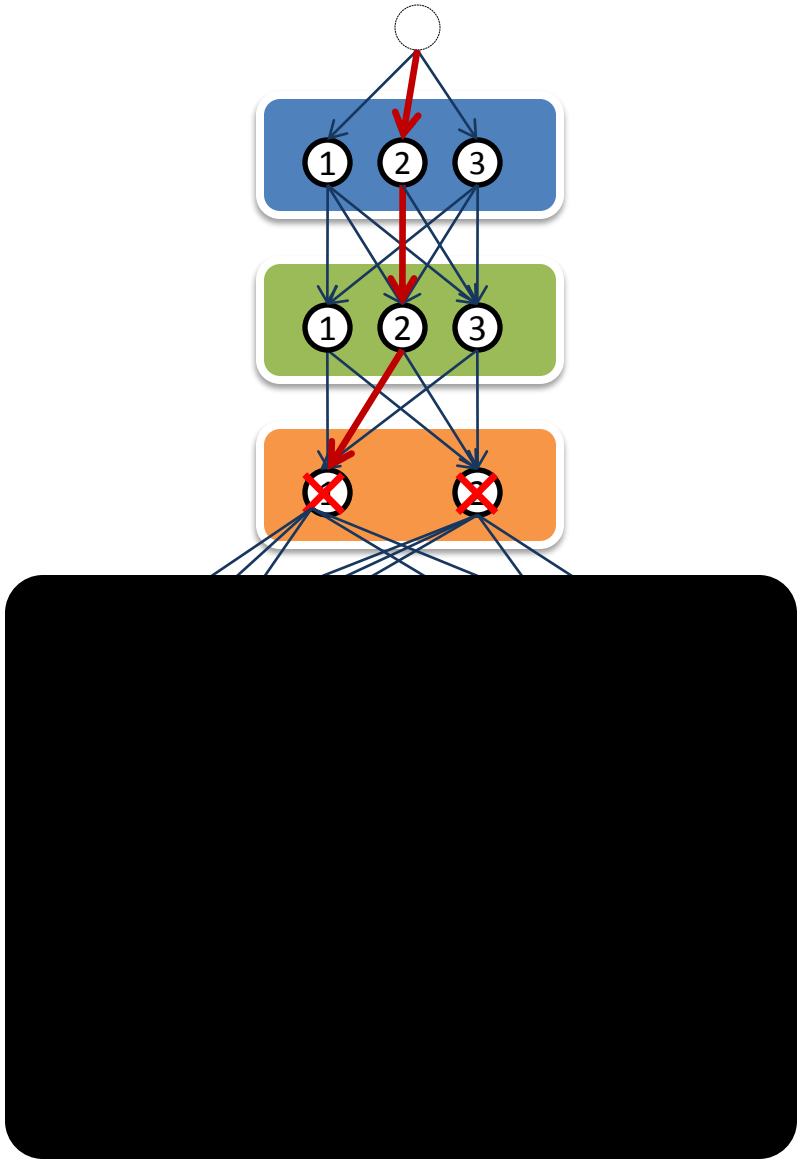
Doctor activity

- $S_i \longrightarrow f_i$
- $f_i = r/t \times qos_{si}$ *where*
  - r : *number of services in the cluster to which* S_i *pertains.*
  - t : *total number of service candidates*
  - qos_{si} : *average of QoS values*
- *Fix a utility threshold* τ
- *Select services with* $f_i \geq \tau$

➔ *The utility threshold* τ *manages the trade-off between the timeliness and the optimality of the algorithm.*

➔ τ *is fixed by the administrator of the service oriented environment (e.g., the hospital).*

Global Selection Phase



- *Explore the search space formed of the services resulting from the local selection phase.*
- *Services are sorted wrt their utilities μ in the descending order.*
- *Pruning the search space using incremental computation.*
- *Pruning the search space using utility approximation.*
- *The selected service compositions are ranked wrt their QoS utilities.*

Experimental setup

- **Hardware:**
  - *CPU: AMD Athlon 64 X2 Dual Core TK-55.*
  - *RAM: 1.80 GB.*

- **Software:**
  - *OS: Windows XP SP2*
  - *JVM: JDK 6 update 12*

- **Input Data:**
  - *Process generator:*

    Generates abstract user tasks based on randomly chosen composition patterns.

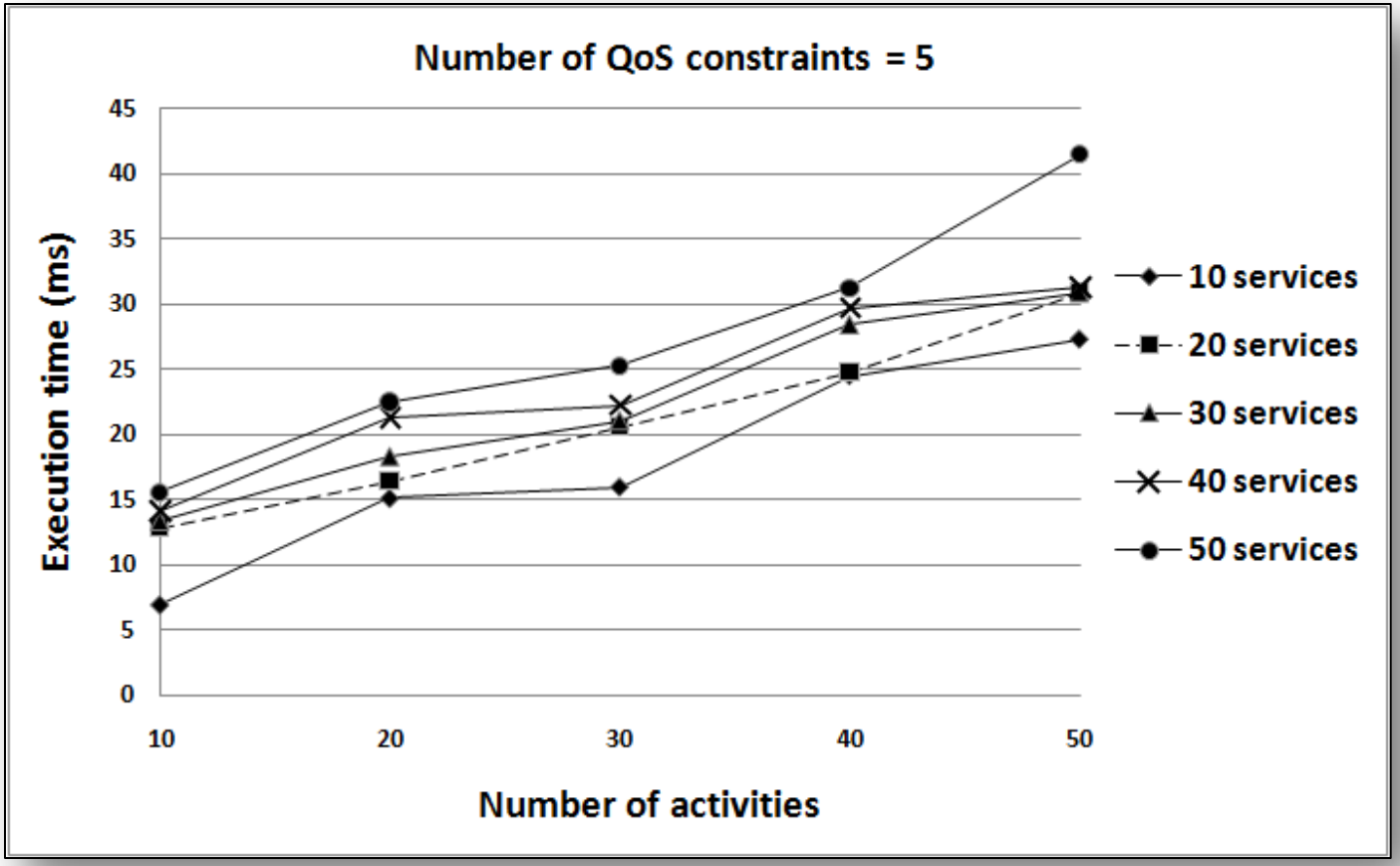
  - *QoS values' generator:*

    Generates QoS values of service candidates based on QoS of real web services **[Almasri et al. 2007].**

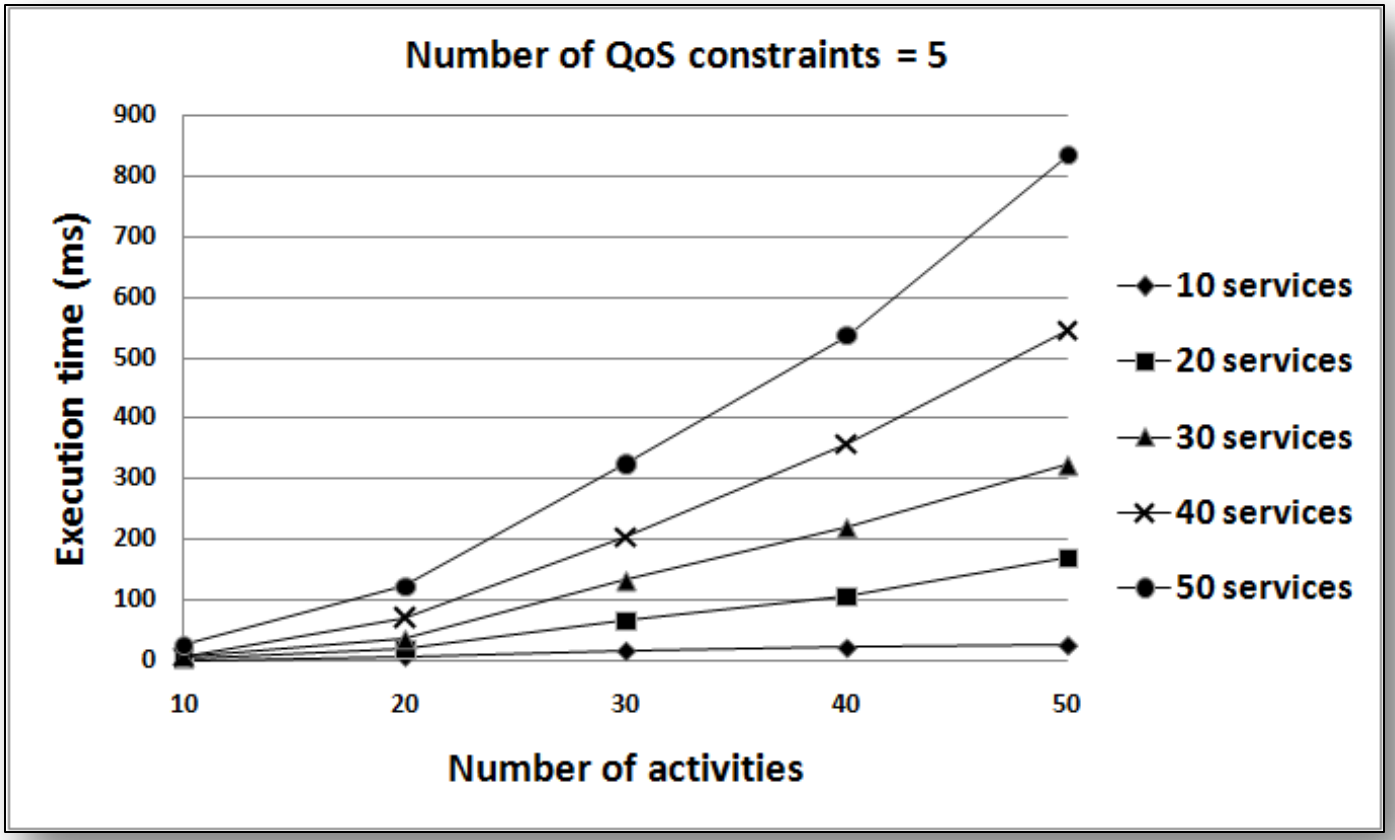
Experimental setup

- **Metrics**
  - *Execution time: We measure execution times of the local selection and global selection separately.*
  - *Optimality* = $\mathcal{U}_{max} / \mathcal{U}_{opt}$
    - $\mathcal{U}_{opt}$ *: the optimal utility given by the brute-force algorithm*
    - $\mathcal{U}_{max}$ *: the best utility yielded by our heuristic algorithm*
  - **Runs:**
    - *20 executions per configuration.*
    - Configuration:
      - *number of activities in the process*
      - *number of services per activity*
      - *number of QoS constraints.*

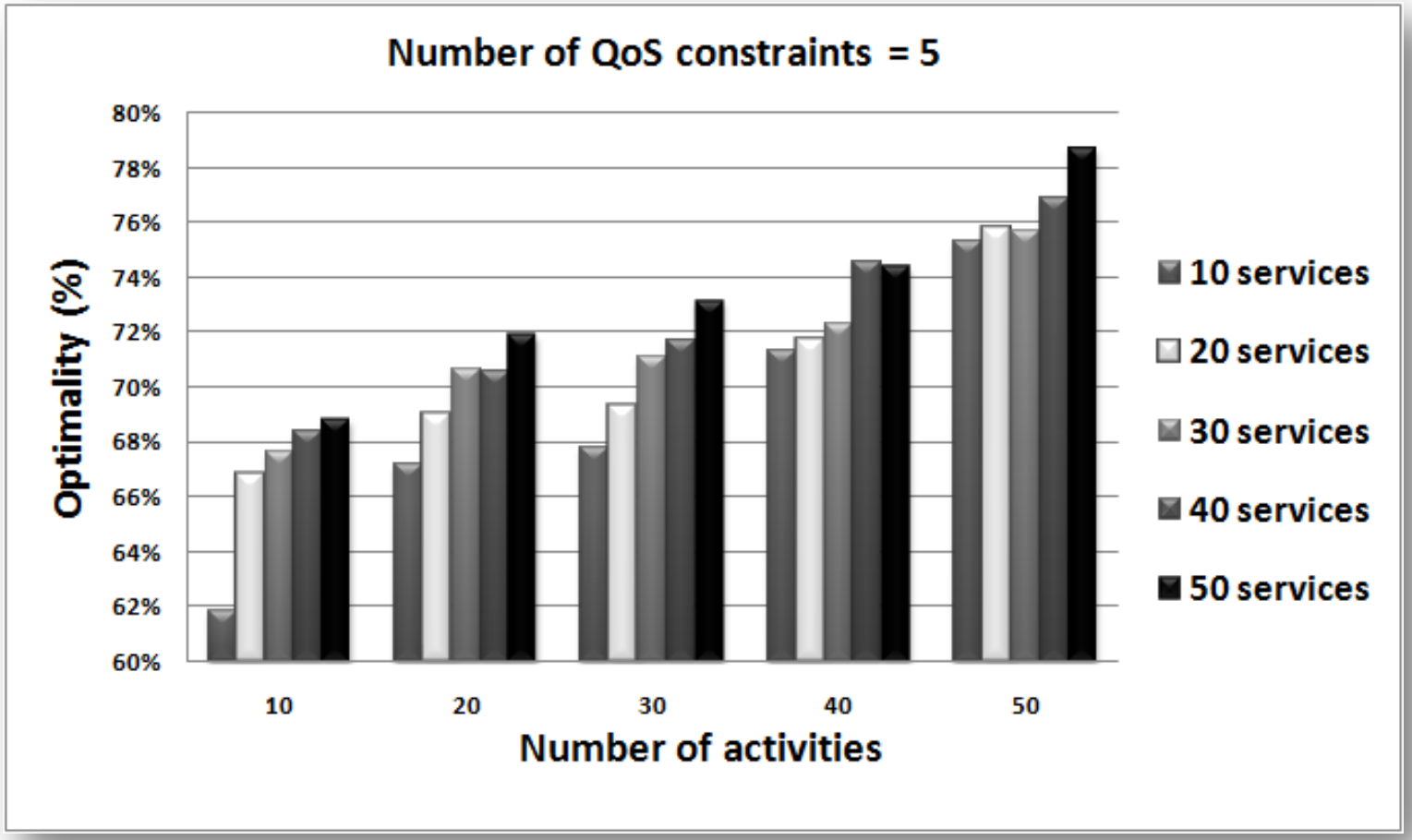
Execution Time: Local Selection



Execution Time: Global Selection



Optimality


Number of QoS constraints = 5
Optimality (%)
80%
78%
76%
74%
72%
70%
68%
66%
64%
62%
60%
10
20
30
40
50
Number of activities
10 services
20 services
30 services
40 services
50 services

Conclusion

- *We presented an efficient QoS-aware selection algorithm for interactive dynamic service environments.*
- *We investigated clustering techniques for services' selection.*
- *The proposed algorithm makes part of our work addressing QoS-aware middleware for dynamic service oriented environments.*

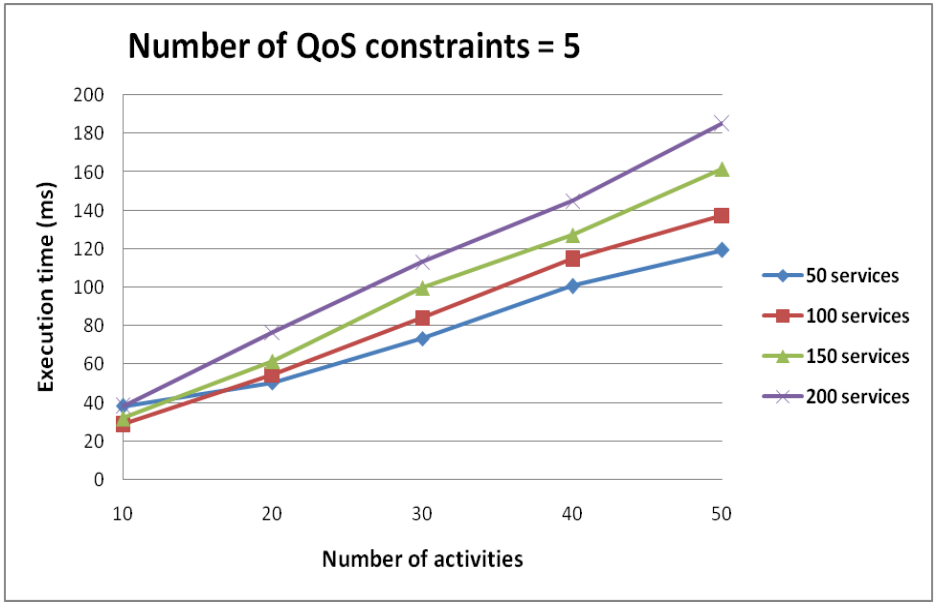
- **Ongoing and future work:**
  - *Investigating other ways of using clustering techniques for QoS-aware service composition.*
  - *Enhancing experimentations:*
    - *Investigate other aggregation approaches (i.e., optimistic and mean value).*

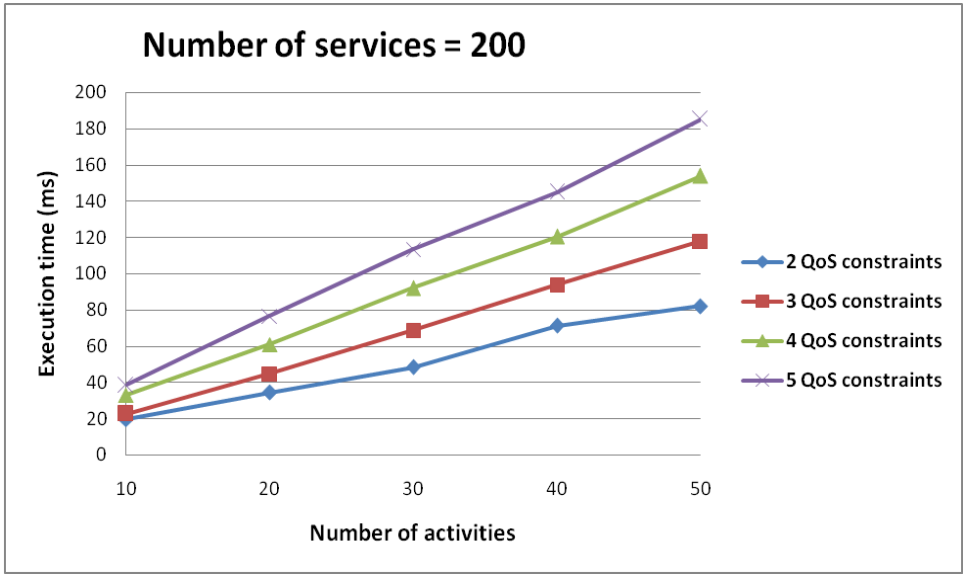
End

Thank you for your attention

Questions?

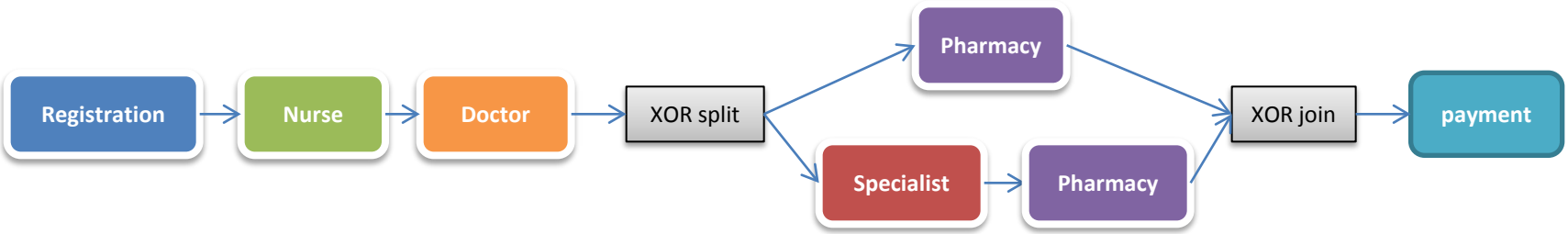
Total Execution Time





Composition model

- **Composition Patterns:**



- **Computing QoS of composite service**
  - ***Pessimistic approach***

QoS attributes	Composition Patterns			
	Sequence	AND	XOR	Loop
Availability (av)	$\prod_{i=1}^{n} av_i$	$\prod_{i=1}^{n} av_i$	$min(av_i)$	av_i^k
Reliability (rl)	$\prod_{i=1}^{n} rl_i$	$\prod_{i=1}^{n} rl_i$	$min(rl_i)$	rl_i^k
Cost (c)	$\sum_{i=1}^{n} c_i$	$\sum_{i=1}^{n} c_i$	$max(c_i)$	$c \times k$
Throughput (th)	$min(th_i)$	$min(th_i)$	$min(th_i)$	$th \times k$
Duration (d)	$\sum_{i=1}^{n} d_i$	$max(d_i)$	$max(d_i)$	$d \times k$