Towards Web Services Composition and Execution Using Software Agents

Zakaria Maamar[1], Quan Z. Sheng[2], Boualem Benatallah[2]

[1]College of Information Systems, Zayed University, Dubai, U.A.E.
Zakaria.maamar@zu.ac.ae

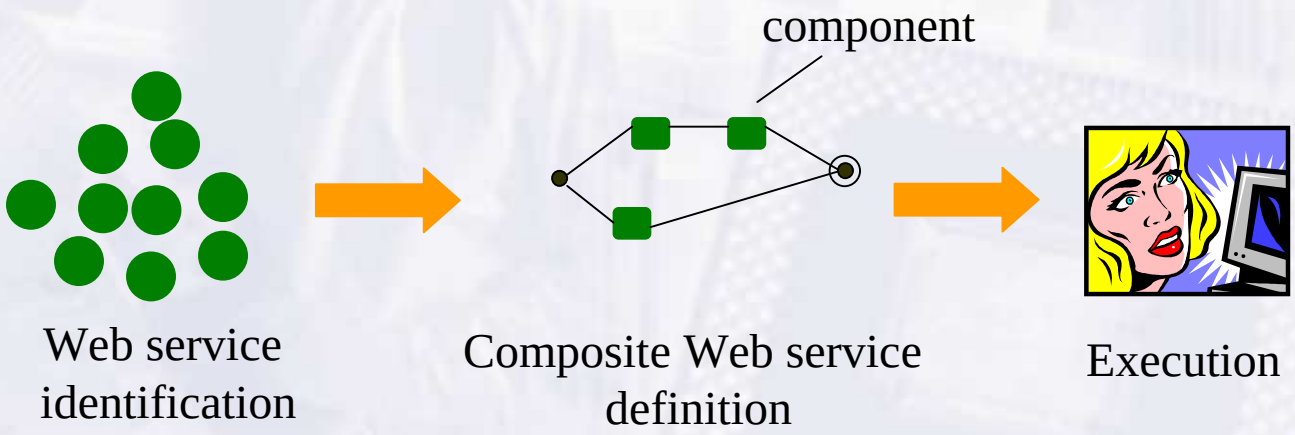
[2]School of Computer Science and Engineering,
The University of New South Wales, Sydney, Australia
qsheng, boualem@cse.unsw.edu.au

Agenda

- **Web Service Composition/Execution: What are the Challenges**
- **Agent-based Architecture**
- **Web Services Composition and Execution**
- **Future Work**

Web Services Composition/Execution: What are the Challenges

- In current Web services composition proposals (e.g., CMI, Mentor, CrossFlow, WISE), the development, provisioning, and execution of composite services is a sequential multi-stage process.



Multi-stage process for Web services composition and execution

Web Services Composition/Execution: What are the Challenges (Cont.)

- Such way of preparing composite services can not cater for the ***dynamic nature*** of Web services. E.g.,
  - Services may be *obsolete* for the execution (e.g., they are not available any more)
  - The status of services may be changed (e.g., the services are overloaded)
  - The *QoS* (Quality of Service) of services may be changed (e.g., price is increased, execution time becomes shorter)

Agent-based Architecture (cont.)

- We suggest an agent-based multi-domain (i.e., a *user domain* and a set of *provider domains*) architecture for the Web service composition and execution.

  A domain is a set of computing hosts on top of which services and software agents are deployed.

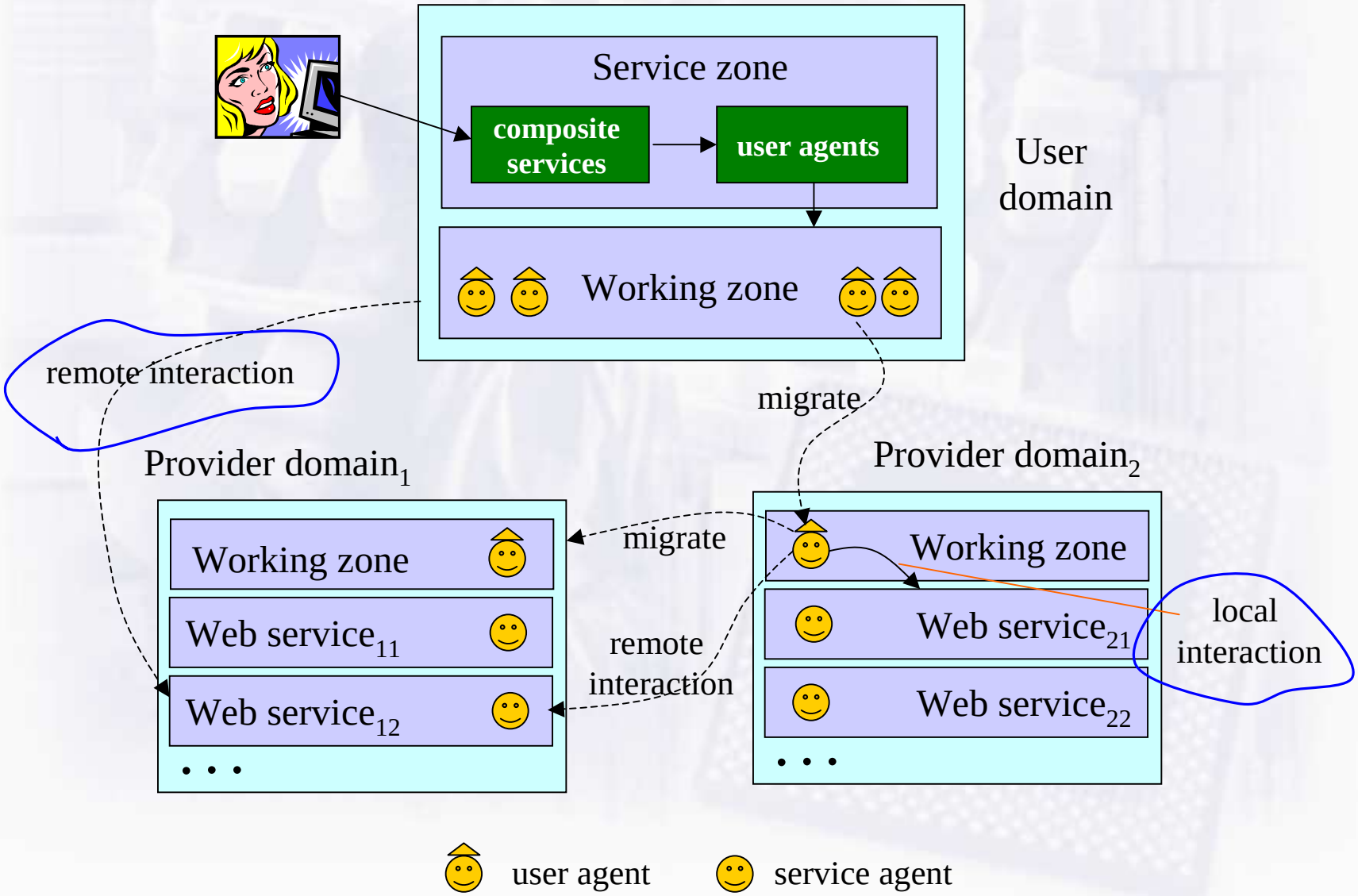
User domain

- User agents: act on behalf of users, decides and invokes component services.
- Service zone: composite services & user agents are developed and deployed.
- Working zone: user agents are resided.

Provider domain

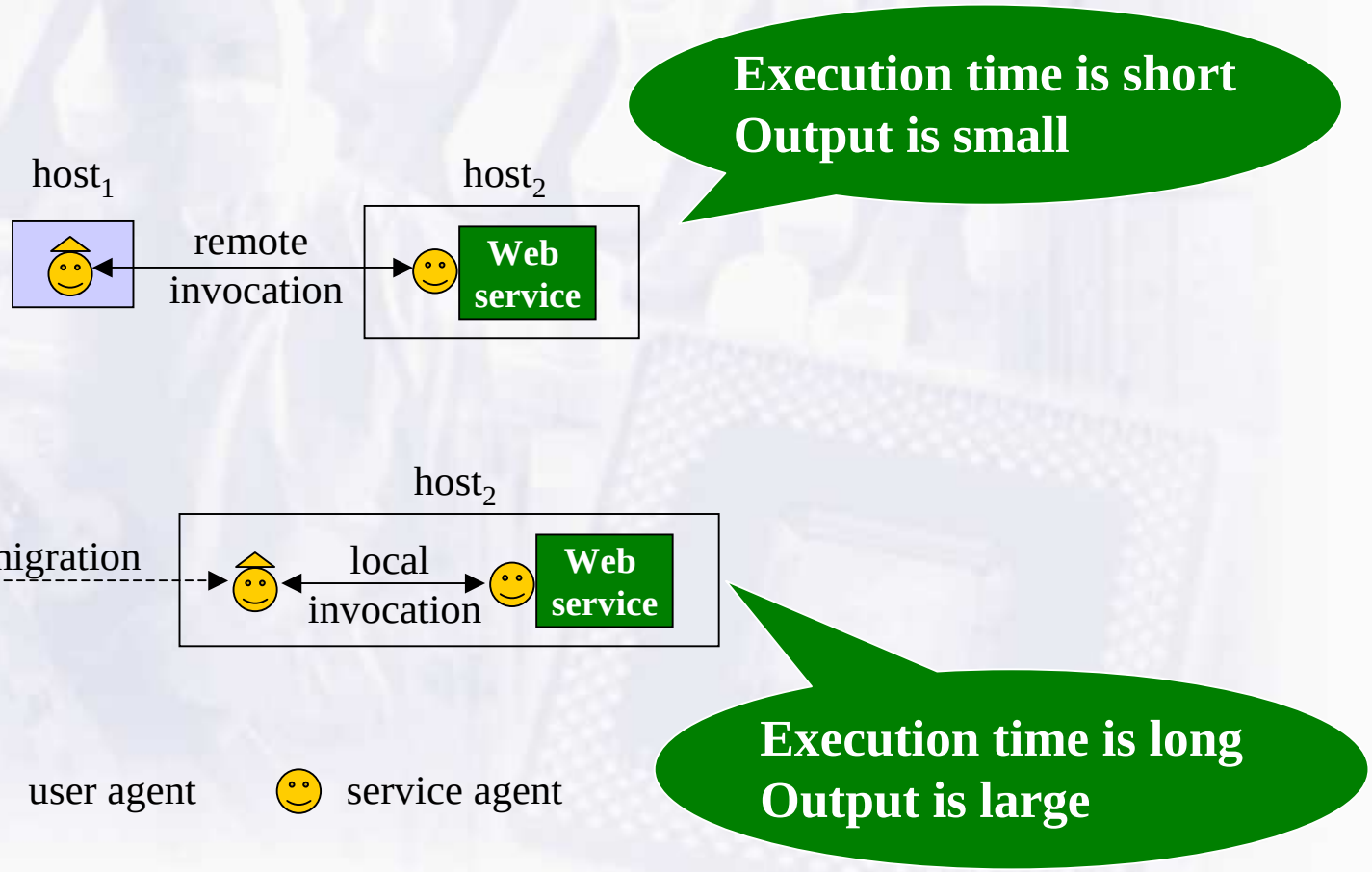
- Service agents: act on behalf of services, handle the invocation requests
- Working zone: receiving and hosting user agents
- Couple of Web services

Agent-based Architecture (Cont.)



Agent-based Architecture

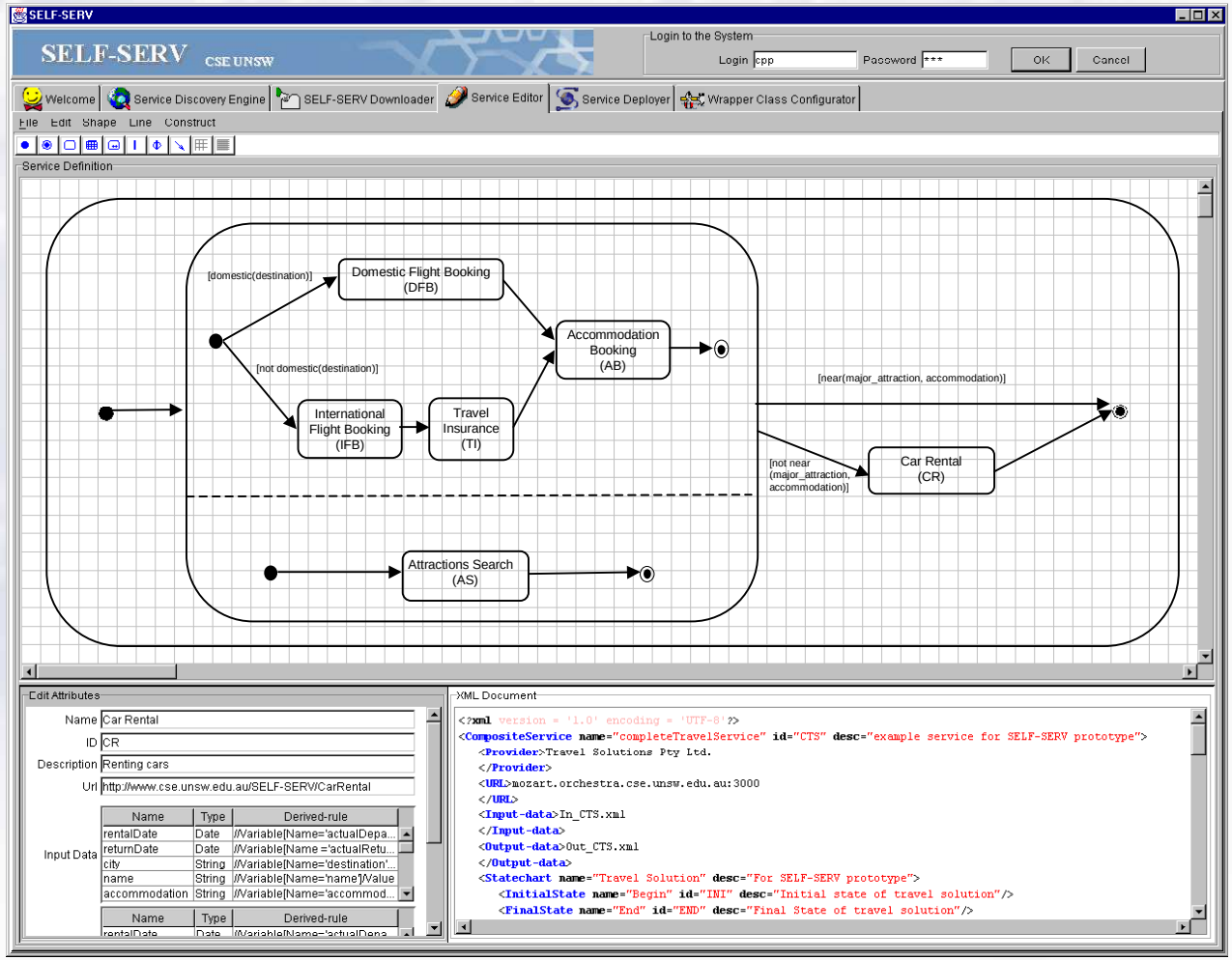
- A Web service can be invoked either ***remotely*** or ***locally***



Web Services Composition and Execution

Composition using statecharts

- A composite Web service schema is modelled using statecharts.



Self-Serv project: IEEE Internet Computing 7(1), VLDB02, ICDE02

Web Services Composition and Execution (cont.)

Service selection

- A task (e.g., flight ticket booking) could be offered by several service agents.
- When a user executes a composite service, a specific service first needs to be selected for each task of the composite service at run time, i.e., ***an execution plan***

Composite service

$$Sc = \{t_1, t_2, \cdots, t_n\}$$

Execution plan

$$P(S) = \{< t_1, sa_1, tp_1 >, < t_2, sa_2, tp_2 >, \cdots, < t_n, sa_n, tp_n >\}$$

$$tp_i \in \{local, remote\}$$

Web Services Composition and Execution (cont.)

Service selection

- The selection consider two criteria: execution cost and location of computing hosts.

- Introduction of location criterion aims at ***gathering in the same provider domain the maximum number of component services***. The benefits:

  - Reducing the number of remote interactions between domains.
  - Reducing the number of migrations of user agents.
  - Reducing the number of data exchanges between domains.

Web Services Composition and Execution (cont.)

Service selection

- Phase 1. Search service agents for all the tasks.

  $<t_i, SA_i>$, where $SA_i = \{sa_{i1}, sa_{i2}, ..., sa_{im}\}$

- Phase 2. Select service agent for a task, $<t_i, sa_{i,} tp_i>$

  phase 2.1: process first task, t_1. Only execution cost is considered, i.e., the service with the minimum execution cost will be selected.

  Phase 2.2: process remaining tasks, t_i, $i=2, 3, \ldots, n$. Both location and execution cost are considered. First consider location criteria, then execution cost.

Web Services Composition and Execution (cont.)

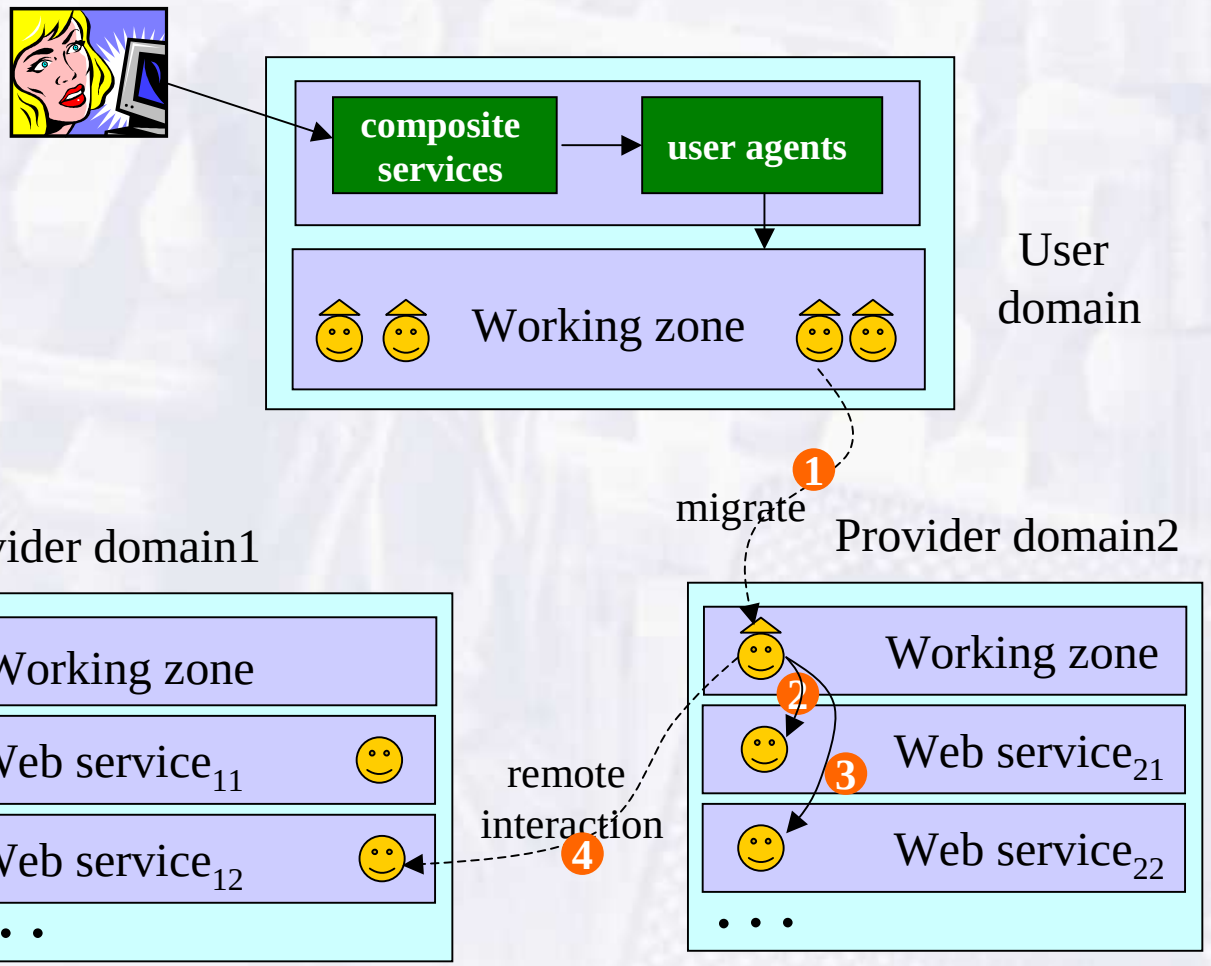
Service selection

for each $<t_i, SA_i>$, i=2,….., n //suppose $<t_{i-1}, sa_{i-1}, tp_{i-1}>$ already exists
begin
  A← Ø //set of service agents that are in the same domain as sa_{i-1}
  B← Ø //set of service agents that are in the same domain as user agent.
  C← Ø //set of service agents that are in other domains.
  for (j=1;j<$\|SA_i\|$;j++) //$sa_j \in SA_i$
  begin
    if domain(sa_j) = domain(sa_{i-1})
    then A ← A ∪ sa_j
    else if domain(sa_j) = domain(userAgent)
      then B ← B ∪ sa_j
      else C ← C ∪ sa_j
  end //A ∪ B ∪ C = SA_i
  if A ≠ Ø
    then contact service agents of A – go to Phase 2.1
  else if B ≠ Ø
    then contact service agents of B – go to Phase 2.1
  else contact service agents of C – go to Phase 2.1
end

Algorithm for service agent selection

Web Services Composition and Execution (cont.)

An example



$$P(S) = \{<t_1, sa_{21}, local>, <t_2, sa_{22}, local>, <t_3, sa_{12}, remote>\}$$

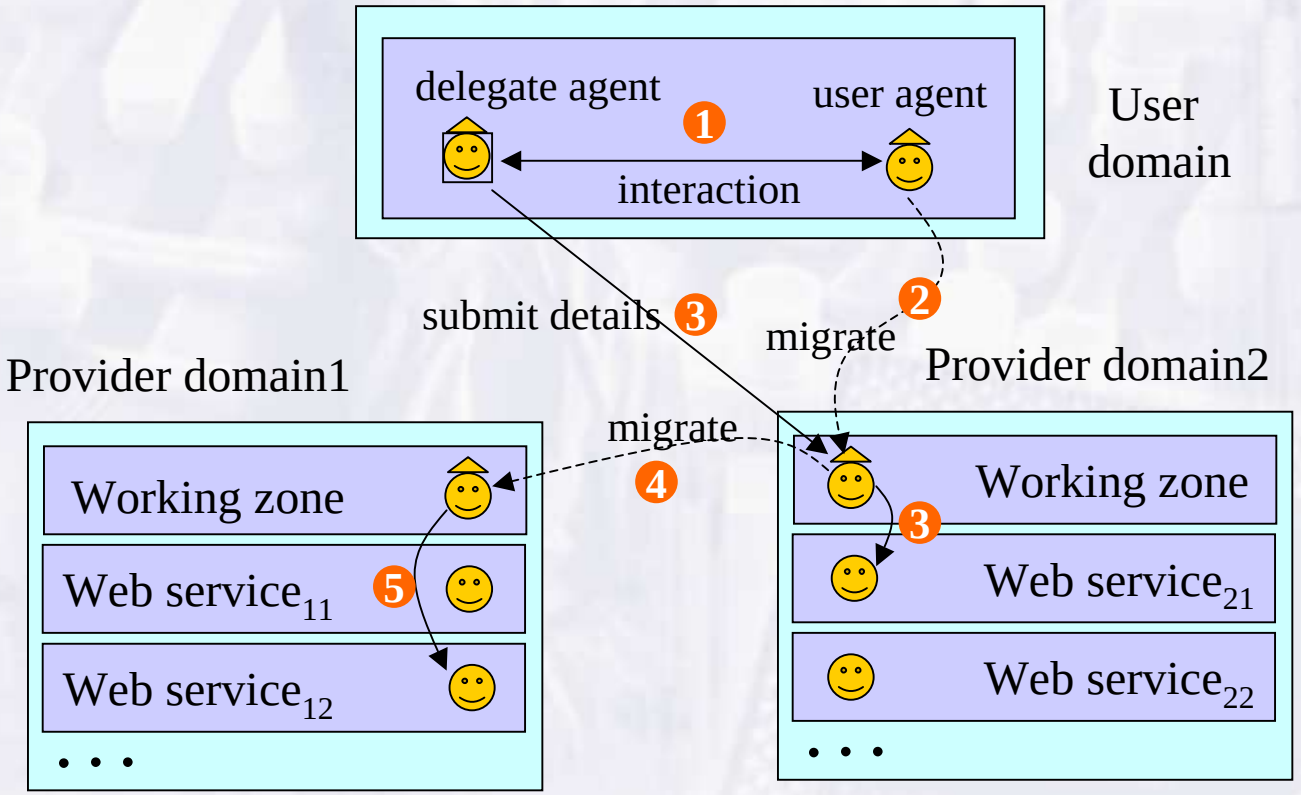
Web Services Composition and Execution (cont.)

Interleaving Composition & Execution

- Interleaving composition and execution (i.e., composition and execution are carried concurrently) has a couple of advantages:
  - Cater for the dynamic nature of the Web services.
  - Perform a reliable service execution.
- A user agent delegates a part of work to a *delegate agent*. The delegate agent prepares the services and submits the details to the user agent, which is always *one-step* ahead of the user agent.

Web Services Composition and Execution (cont.)

Interleaving Composition & Execution



Interleaving service composition and execution

Future work

- **Implementation the prototype**
- **Performance and scalability study**
- **Change management**

Finally…

Thanks a lot for your time.

Quan Z. Sheng
School of Computer Science and Engineering (CSE)
The University of New South Wales (UNSW),
Sydney 2052, Australia
qsheng@cse.unsw.edu.au
Tel: +61 2 9385 6908 Fax: +61 2 9385 4936
http://www.cse.unsw.edu.au/~qsheng