Architecture for Migratory Adaptive User Interfaces

Takuto YANAGIDA and Hidetoshi NONAKA
Hokkaido University, Japan

1. Introduction

Background

- Constant improvements in technology have spawned a variety of platforms used for interactive services, and that has created users' new demands.
  1. To be able to use services through different devices and modalities in accordance with certain contexts.
  2. To be able to change devices and take tasks from one device to another (user interface migration [3]).

[3] R. Bandelloni and F. Paternò. Flexible interface migration. In IUI 04, pages 148–155, 2004.

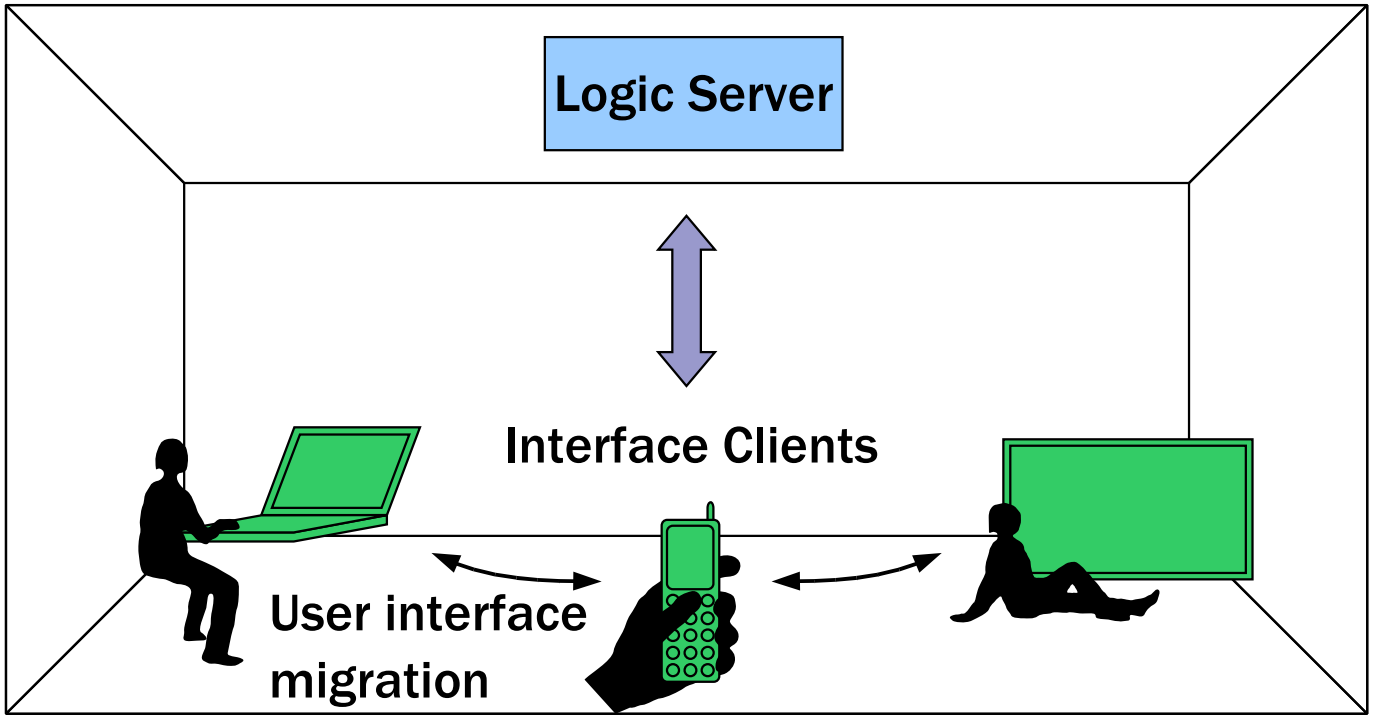
Problems

- Conventional ways of associating devices and services does not meet the users' demands because of costs and inadequate separation between services and devices.
- Developing multiple versions for each platform is expensive for the developers in terms of time, money, and maintaining the version consistency.

Point of our proposal

- We improve interface client/logic server (ICLS) [18], which supports UI migration, and offers adaptive UIs for devices and services.
- The target of ICLS is dialog-based interactive services, in which some input facilities are used on some dialogs updated as state transitions.

[18] T. Yanagida, H. Nonaka, and M. Kurihara. User-preferred interface design with abstract interaction description language. In IEEE International Conference on Systems, Man and Cybernetics, 2006.



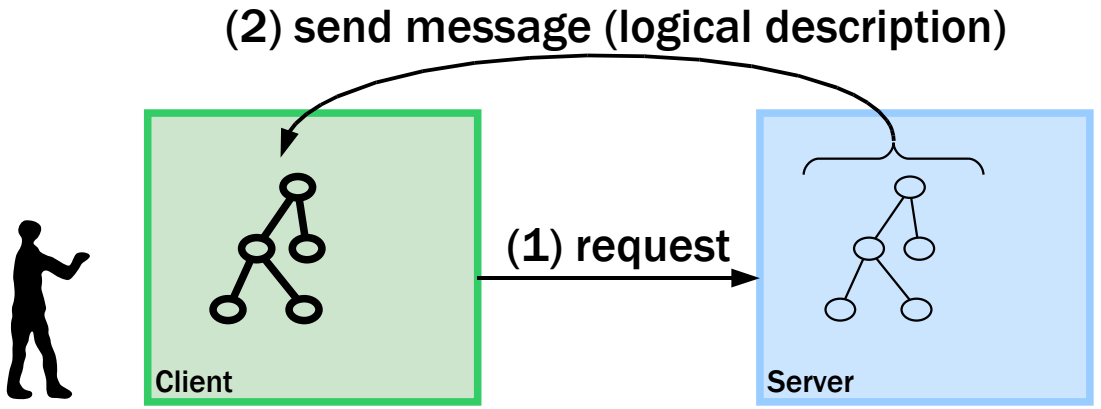
- Users of ICLS-based services possess interface clients, and connect them to logic servers when they use the services provided by the servers.

- ICLS has a mechanism for attaching clients to existing sessions for UI migration.
- When the users do not disconnect the previous client after attaching a new client, they can utilize the two clients simultaneously.

2. Architecture

2.1. Clients and servers

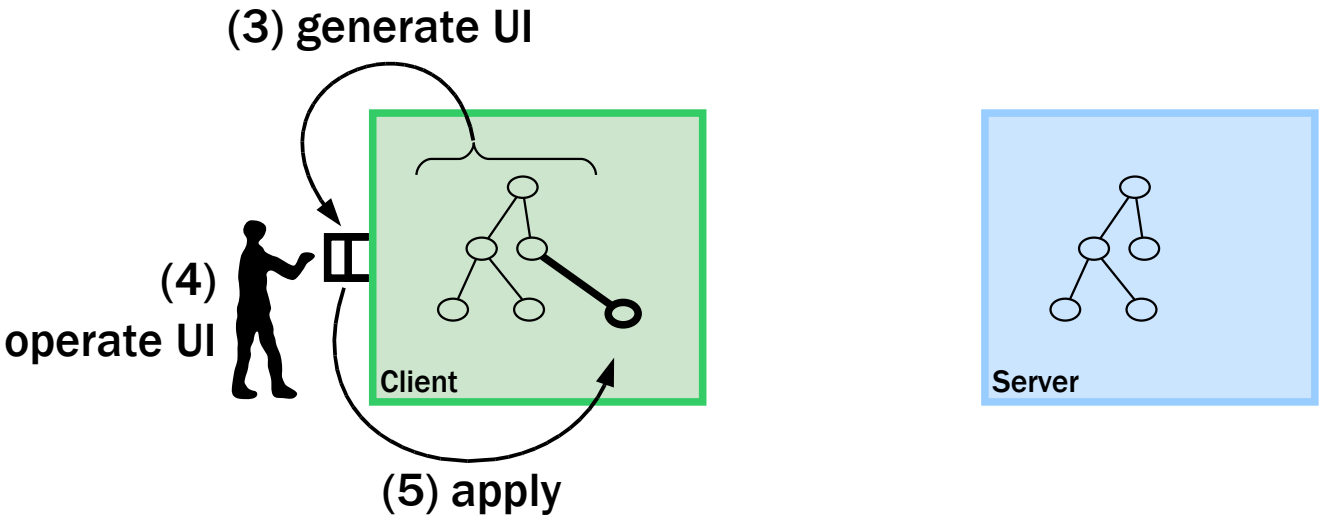
- An example of the flow of a session between a client and a server for a reminder service, which manages users' scheduled tasks (7 steps).



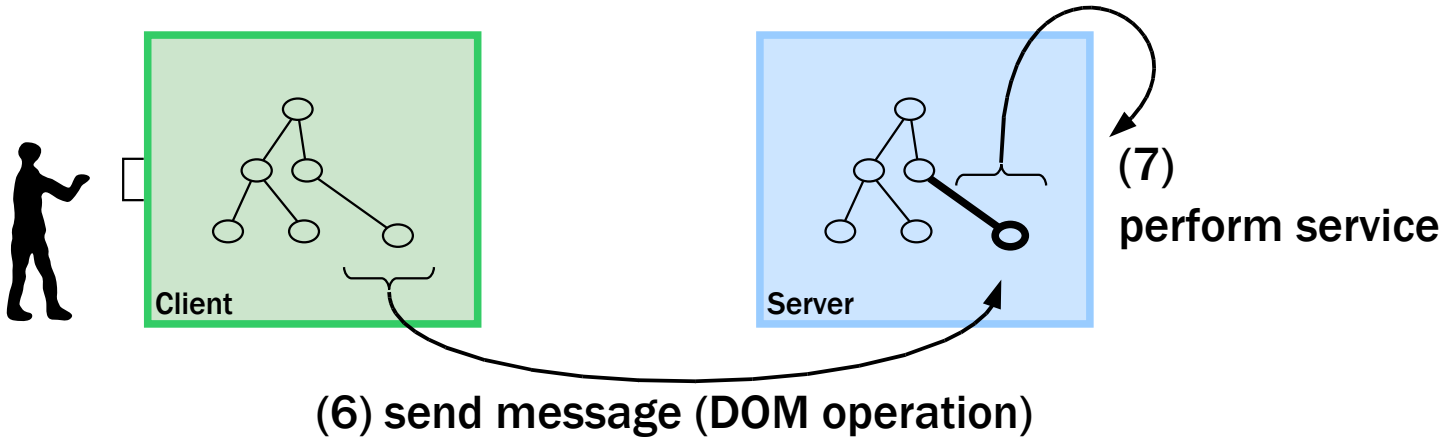
1. A service session starts with a request from a client to a server.
2. After the session starts, the client receives a logical description from the server.
   - Logical descriptions are written in abstract interaction description language (AIDL), an application of XML.

- A server and its clients in session maintain the same structured DOM* tree of XML.
- A DOM tree constructed with AIDL contains not only a UI structure but also a current state of a generated UI.

* the DOM (document object model): a standard object model for representing XML.



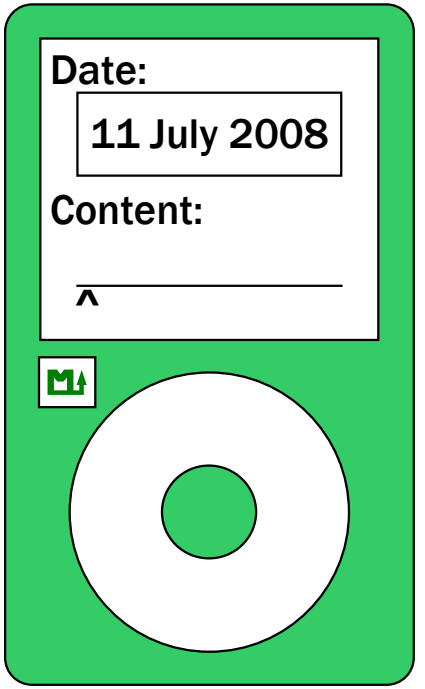
3. The client constructs a DOM tree, and generates a UI based on the tree.
4. The user operates the client.
5. The client changes the DOM corresponding to the user's operations.



6. The client sends these DOM changes as messages to the server.

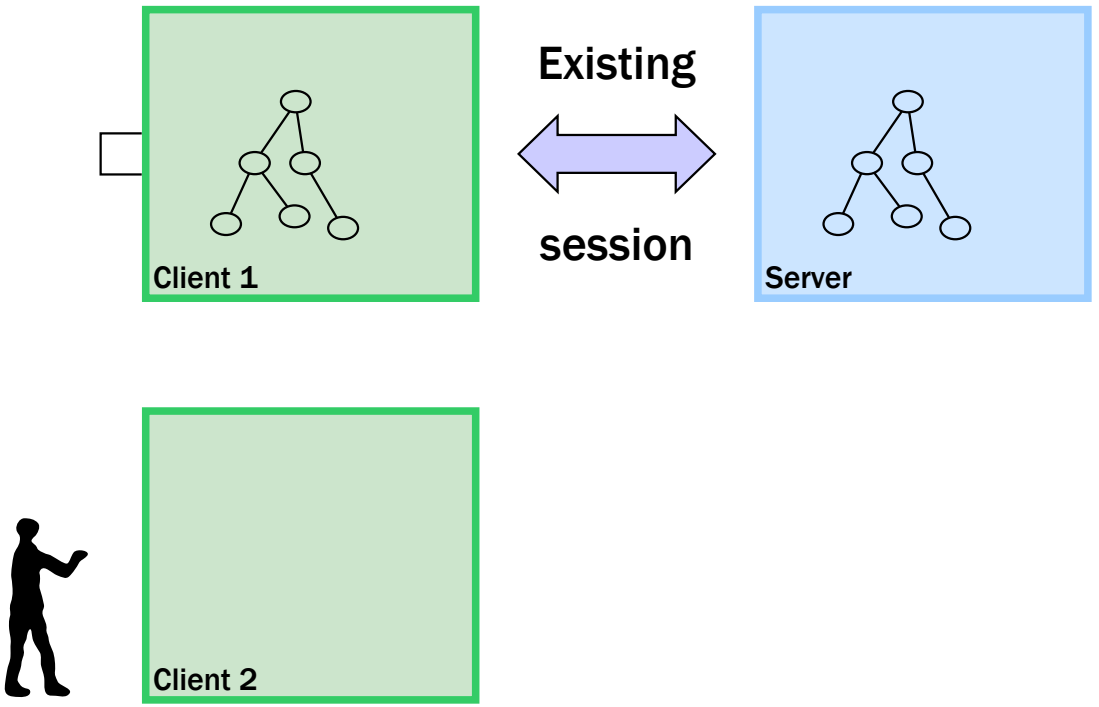
7. The server performs the service according to the message from the client.

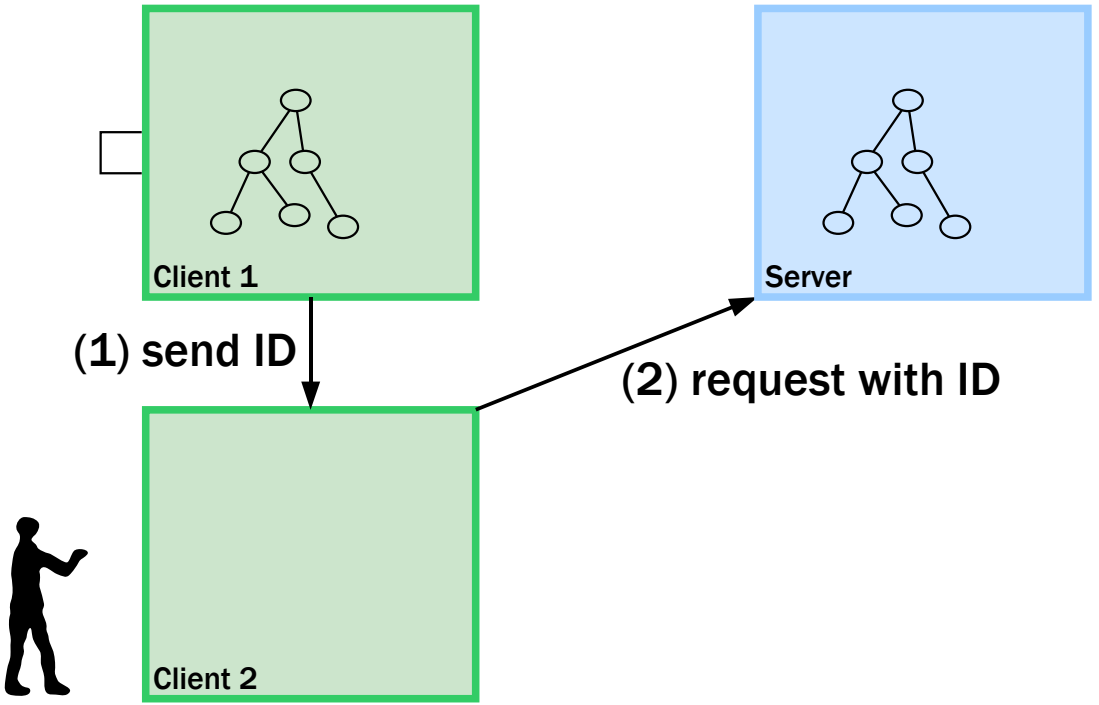
- *The user can check his scheduled task list using a device like a music player which has a small screen and a wheel control, on the train, on the way to his office.*
- *The user can set the time limit for a task item easily using the wheel control, but he might feel it's a nuisance to fill the content field of the item using the same control.*



Migration and simultaneous use

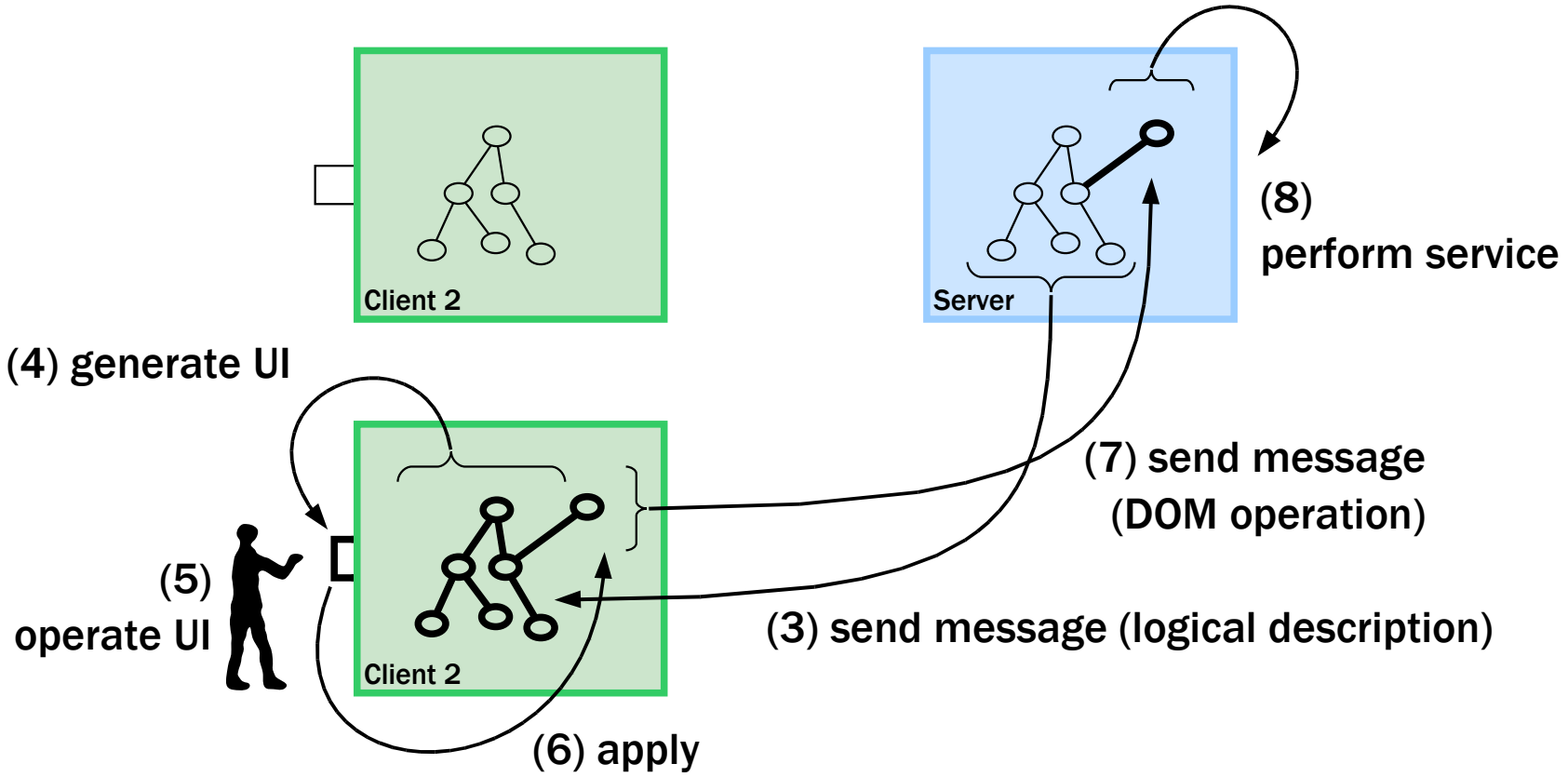
- During the time a user is accessing a service through one client with a session, the user can access the same service through another client.





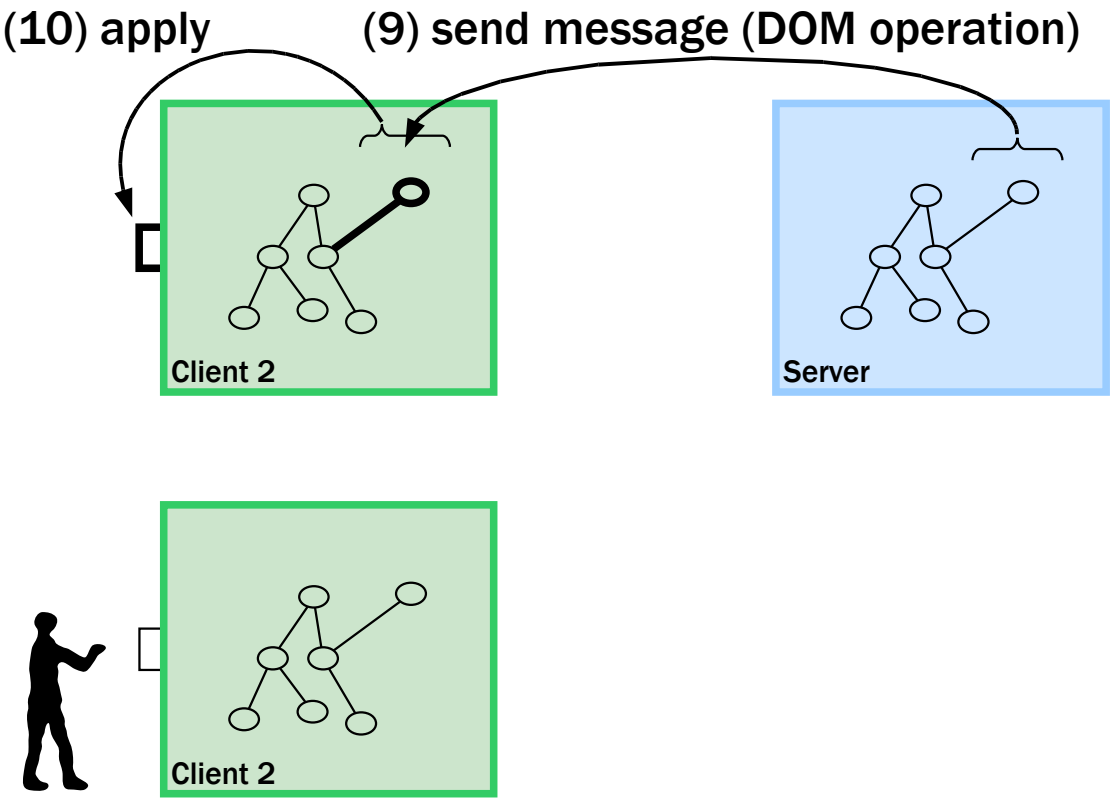
1. A new client (client 2) obtains a session ID from the existing client (client 1).
2. A simultaneous service session starts by a request with an ID from client 2 to the server.

- DOM trees constructed with logical descriptions contain not only UI structures but also current states of generated UIs.
- Receiving a DOM tree that is being used by another client means receiving the whole information about all of the UIs in a session.



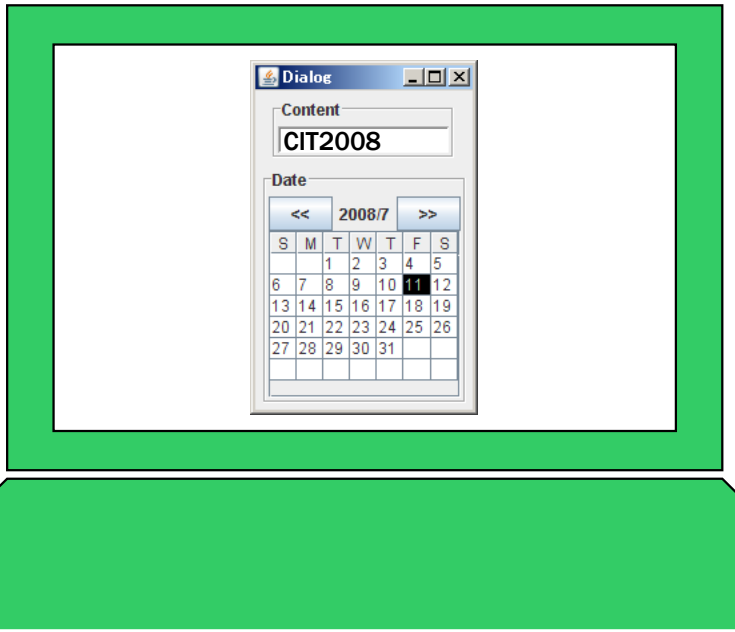
3. Client 2 receives from the server a logical description used in the session.

4.- 8. Client 2 does the same UI generation process and sends the user's operation to the server.



9. The server broadcasts the received message to another client.
10. Client 1 receives the message, and applies it to its DOM tree and UI.

- *Accessing the server with the same session through a PC, the user can continue to type in the content of the item, without having to restart of the service or reenter the time data.*



2.2. Protocol

- Interface clients and logic servers communicate with a protocol, tree structure synchronize protocol (TSSP), where changes in DOM trees are serialized as messages.
- The protocol provides the way to send whole or partial trees as documents written in AIDL in order to synchronize two or more DOM trees in each client and server.

Message

- Messages exchanged between a client and a server are categorized into two types:
  - Just sending entire AIDL documents, and
  - Serialized change operations consist of
    - a command (`INSERT`, `ERACE`, or `REPLACE`),
    - an XML path (such as "`/group[0]/selection[1]`"),
    - a piece of an AIDL document inserted or replaced.

Attach to session

- Attaching an existing session is performed by passing a session ID to a server.
- When a client sends a request with an ID, the server returns the DOM tree of the session identified with the ID as an AIDL document.
- Clients have to communicate with each other to obtain this ID from other clients.

Collision of messages

- When multiple clients exist in the same session, the priority of their messages retains the consistency of their DOM trees.
- When a server receives messages conflicting among clients and/or the server,
  - the state of the server takes priority over the clients',
  - the message that arrives first takes priority.

3. Logical description language

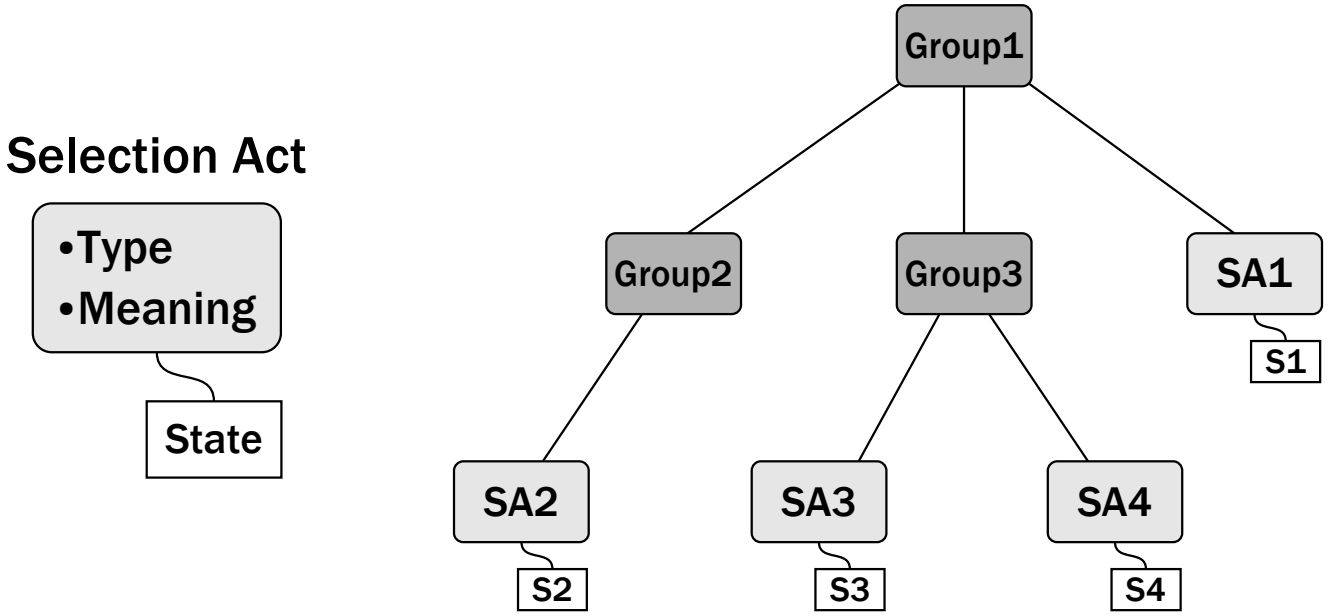
AIDL

- Abstract interaction description language (AIDL) is an application of XML, we developed, describes interaction structures, presentations, and task models of services.
- It supports the clear separation of interface clients and logic servers, as XML is a web standard, and has no device- or modality-specific contents.

```
<aidl:pane>
  <aidl:dialog>
    <aidl:selection aidl:meaning="http://.../LampPowerState">
      <aidl:description aidl:caption="Power" />
      <aidl:state>http://www.example.com/On</aidl:state>
      <aidl:resources>
        <aidl:choice aidl:uri="http://www.example.com/Off">
          <aidl:description aidl:caption="Off" />
        </aidl:choice>
        <aidl:choice aidl:uri="http://www.example.com/On">
          <aidl:description aidl:caption="On" />
        </aidl:choice>
      </aidl:resources>
    </aidl:selection>
  </aidl:dialog>
  <aidl:knowledge>
    <rdf:RDF>
      <rdf:Description rdf:about="http://.../LampPowerState">
        <rdfs:subClassOf rdf:resource="http://.../PowerState" />
      </rdf:Description>
    </rdf:RDF>
  </aidl:knowledge>
</aidl:pane>
```

Interaction model

- UI structures and their current states described in AIDL are abstracted as selection acts, which represent the essential function of UI elements.
- A selection act consists of three elements:
  - a type (a set of choices),
  - a meaning (a purpose of a selection in a service), and
  - a state (a current state of selection).



- Selection acts are grouped with other selection acts and groups, and make an interaction tree.
  - In AIDL documents, selection acts and groups are expressed as XML nodes, and
  - the nodes are added or removed by clients and servers.

4. Implementations

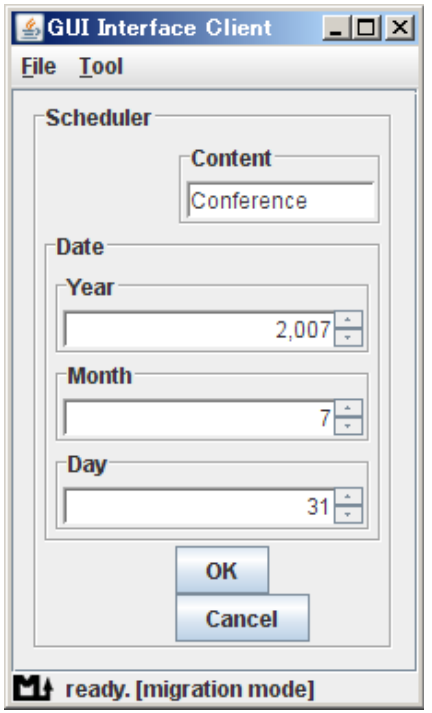
Framework

- We implemented ICLS as a framework, which is a class library written in Java (JDK 1.6) with a semantic web library Jena [8].
- Although various protocols for sending text are available as a low protocol, we adopted TCP/IP as a default.
- We developed three interface clients and two logic servers (a reminder service and a remote controller of a virtual appliance).

[8] Hewlett-Packard Development Company, L.P. Jena.

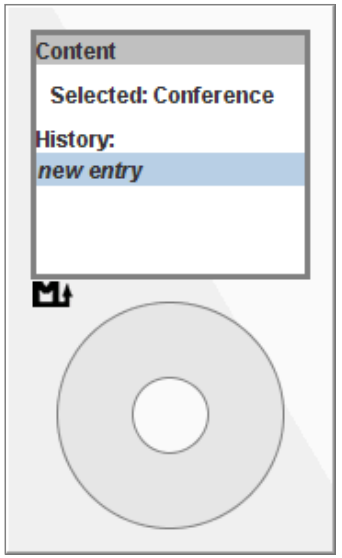
Clients and servers

a)

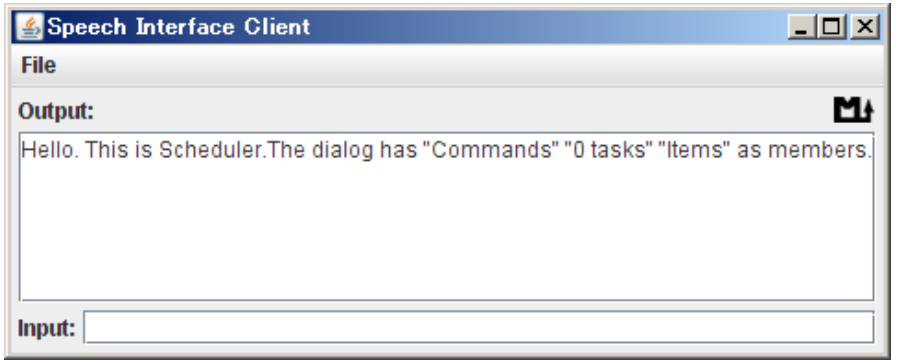


a) The GUI client adopts Swing as a GUI toolkit, and it generates GUI dialog boxes based on AIDL documents.

b)



c)



b) The mobile client has a small screen and a wheel control, and offers hierarchical menu UIs.

c) The client of voice UI simulator expresses voice communication with strings.

5. Discussion

Design principles

- We adopted following three design principles:
  - Since it is difficult to separate the whole UI process from service codes, we have separated only device- or modality-specific processes as the interface clients.
  - We intend to have the clients utilize their resources for the generation of various UIs, and
  - We did not add any scripting function to the architecture in order to retain its design simplicity.

Possibility of various UI

- In the above implementations, we have adopted tentative rules for generating UIs from logical descriptions, but at least for the GUI client, we have provided a specific solution.

6. Related work

Model-based UI design

- As a general solution, a broad range of research has been proposed, and almost all of it employs a model-based UI design [7, 16], which commonly utilizes logical descriptions.

[7] J. Eisenstein, J. Vanderdonckt, and A. Puerta. Applying model-based techniques to the development of UIs for mobile computers. In IUI '01, 2001.
[16] J. M. Vanderdonckt and F. Bodart. Encapsulating knowledge for intelligent automatic interaction objects selection. In CHI '93, 1993.

Flexible interface migration

- A web migratory interface system has been proposed in [3].
  - It targets arbitrary web applications and performs a reverse engineering of existing web pages in order to obtain their logical information.
  - This approach has the benefit of being able to handle web applications, but it does not suit our purpose.

[3] R. Bandelloni and F. Paternò. Flexible interface migration. In IUI 04, pages 148–155, 2004.

7. Conclusion and future work

Conclusion

- We presented interface client/logic server (ICLS) architecture, which offers migratory and simultaneous interfaces.
- Simultaneous interfaces as the extension of migratory interfaces is a new technique on the UI fields we presented.

Future work

- The current implementation of GUI interface client uses some layout managers implemented in Swing, and it has no intelligent mechanism for widget layout.
- Here, we are exploiting achievements in the field of artificial intelligence, especially constraint satisfaction problems (CSPs), and prototyping some clients.