

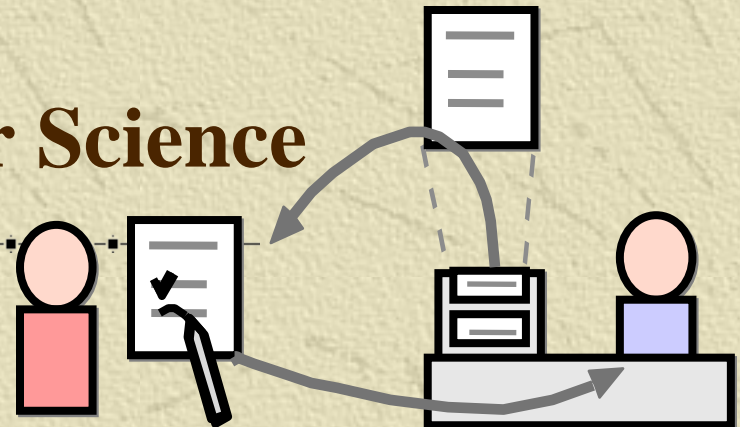
ICICWS'09 :

**The 2009 IAENG International Conference
on Internet Computing and Web Services
Hong Kong, 18-20 March, 2009**

End-user Initiative Requirement Definitions Based on Web Service

**Takeshi Chusho, Noriyuki Yagi
and Katsuya Fujiwara**

**Department of Computer Science
Meiji University
Kawasaki, Japan**



Background of Research

- **Goal**

IT contributes to saving resources and environmental preservation for a sustainable society.

- **Problem**

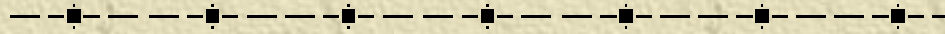
Application software is required and then the fund is needed for its development by IT professionals.

- **Solution**

The end-user initiative development of application is indispensable for the solution of this dilemma.

The First Example

- A thrift store -



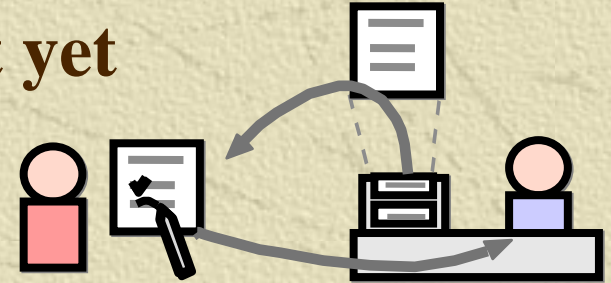
- Many thrift store sells limited goods to limited customers in a local area.
- The number of goods and the number of customers will increase if **business professionals** develop the application for the Web site and open the Web site.



The Second Example

- Service counters -

- Many service counters have not yet support the Internet usage, because of lack of funds.



- If **business professionals** at a service counter can develop the application for a Web site, resources are saved because of the paperless system and reducing the cost of electricity by not using elevators when going to the actual counter.



The Third Example

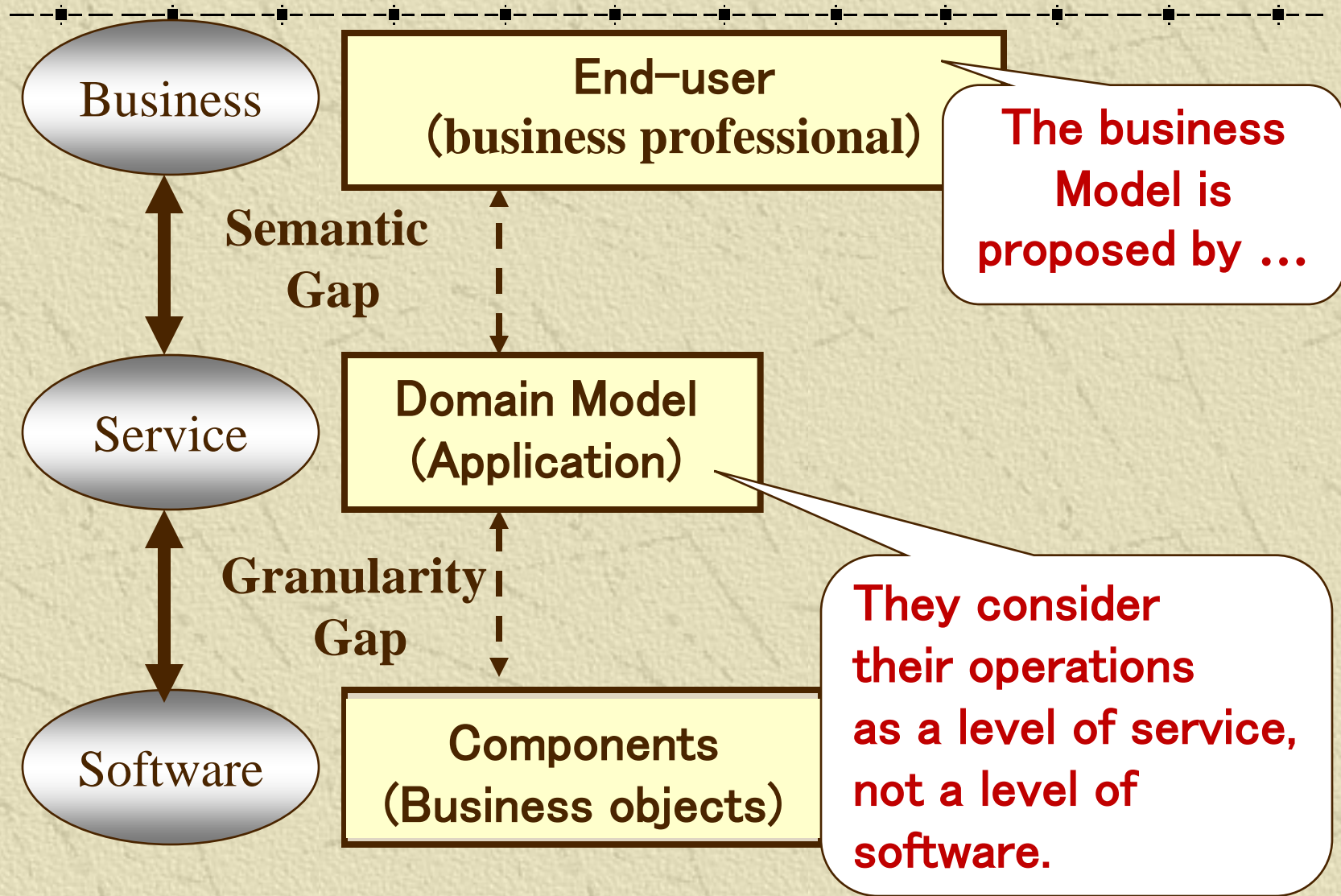
- Online shops -



- Customers may want to buy goods from the nearest shop to reduce carbon dioxide (co2) emission in transportation.
- It must be useful to open a Web site where you can search several online shops and display the transportation distance.
- This application may be developed by **end-users** if both shopping sites and an online map site is integrated by using recent mash-up tech.

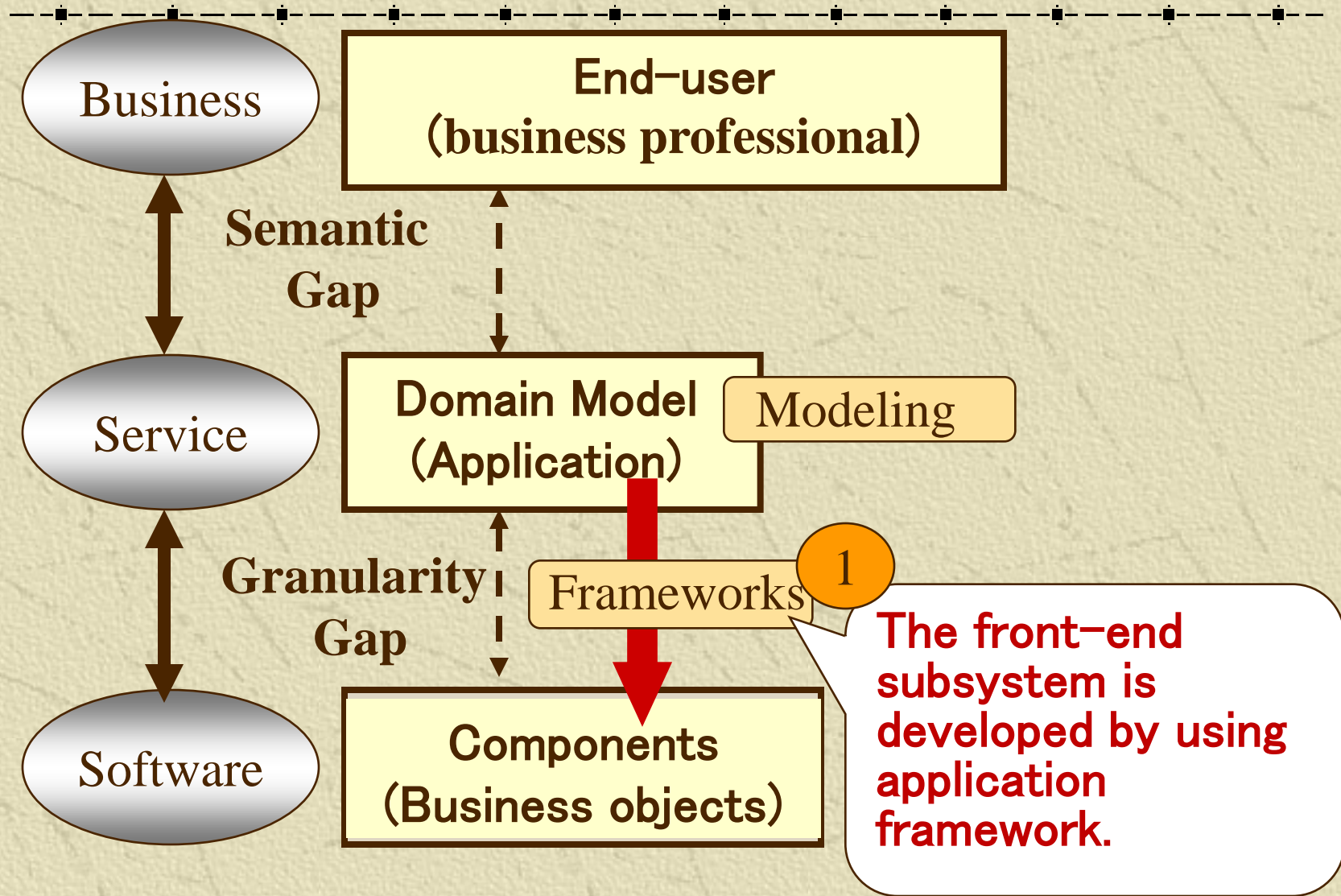
Enduser-Initiative Approach

- how to make web applications -



Our Primary Approach (1)

- **Component-based software engineering** -



Application Framework

- In our **UI (user-interface) driven approach**, the visual forms are defined first and the framework is used.
- The business logic depending on the application is defined by the form definitions.
- The other business logic is embedded into the framework.

An Application Building Procedure

(1) Service definitions :

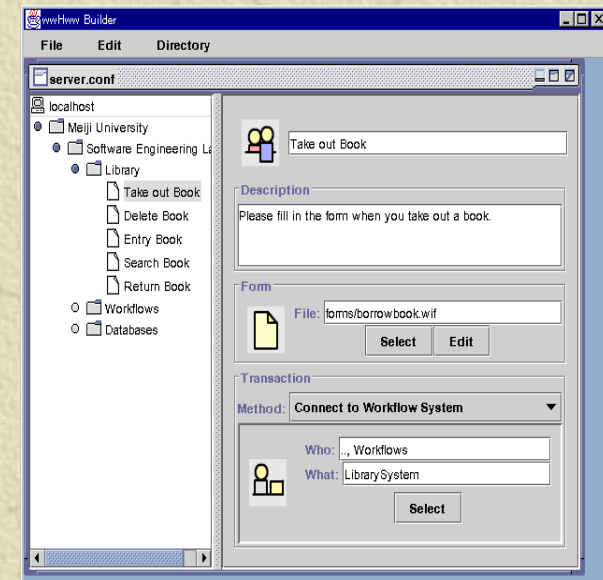
Services at the window, are defined.

(2) Form definitions :

Electronic forms for these services are defined with navigation information.

(3) Registration :

These definitions are registered into the corresponding servers.



The Browser for System Definitions

(1) Service definitions

(2) Form definitions

(3) Registration

The screenshot displays the 'wwwHww Builder' application interface. The main window is titled 'server.conf' and shows a directory tree on the left. The 'Library' folder is expanded, and the 'Take out Book' file is selected. The right pane shows the configuration for this service, with three sections highlighted by red boxes:

- Service Definition:** A header with a person icon and the text 'Take out Book'. Below it is a 'Description' field containing the text: 'Please fill in the form when you take out a book.'
- Form Definition:** A section titled 'Form' with a document icon. It includes a 'File' field with the value 'forms/borrowbook.wif' and 'Select' and 'Edit' buttons.
- Registration:** A section titled 'Transaction' with a 'Method' dropdown menu set to 'Connect to Workflow System'. Below this are 'Who' and 'What' fields. 'Who' contains '..., Workflows' and 'What' contains 'LibrarySystem'. A 'Select' button is located at the bottom of this section.

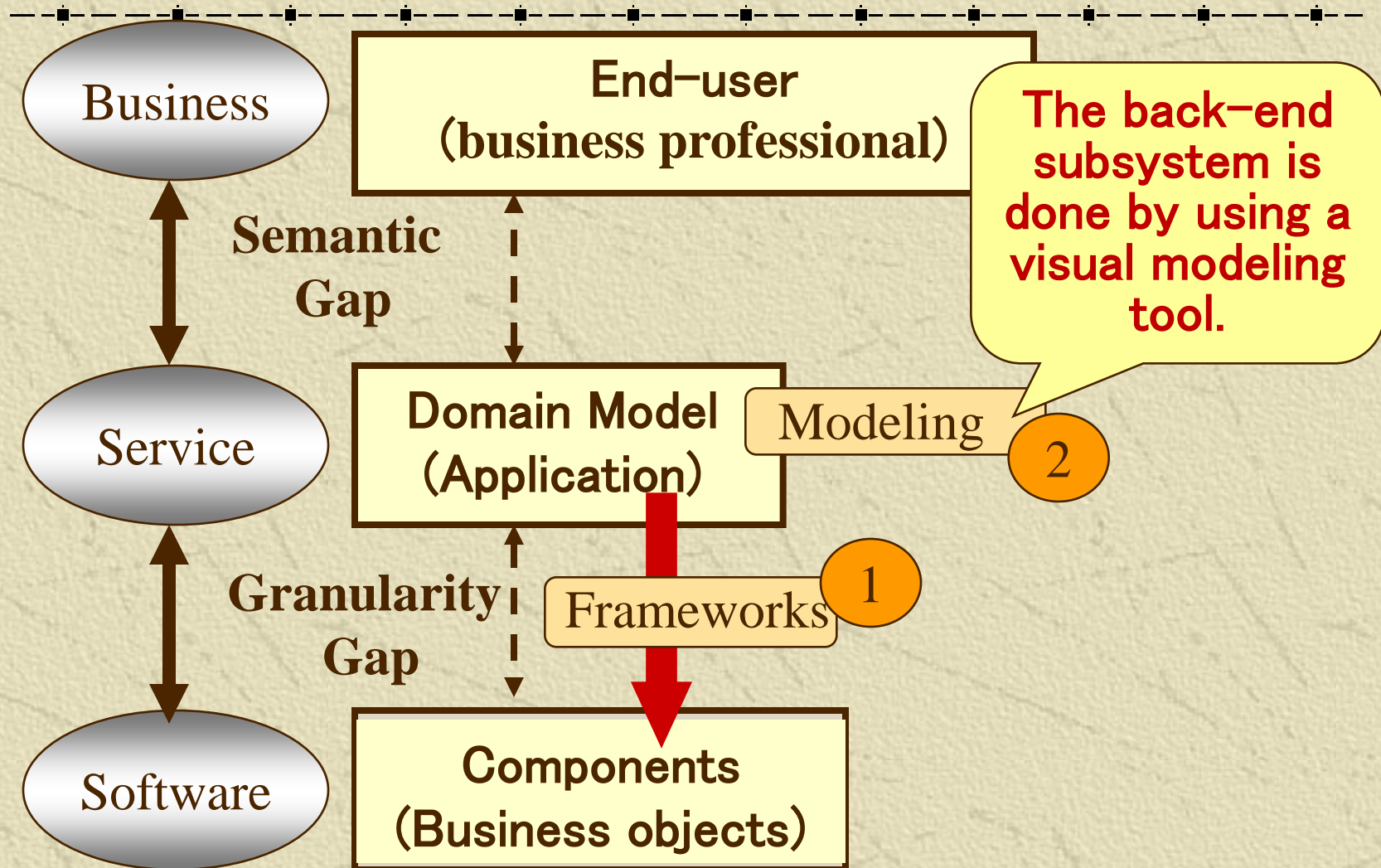
Results of Experiences

- Application Framework -

-
- **Form design implies requirement definitions.**
 - **Most functions are directly mapped into forms.**
 - **End-users need an IT professional's assistance for the UI to be implemented in JSP, components to be newly developed and a complicated DB management system.**

Our Primary Approach (2)

- Component-based software engineering -



Visual Modeling

- In our **model-driven approach**, the back-end subsystem of work flow is developed by using a visual modeling tool.
- This solution is given as a formula of **“a domain model = a computation model.”**
That is, one task in a domain model of cooperative work corresponds to one object in a computation model.
- **It is not necessary for the end-user to convert a domain model into a computation model.**

An Experience

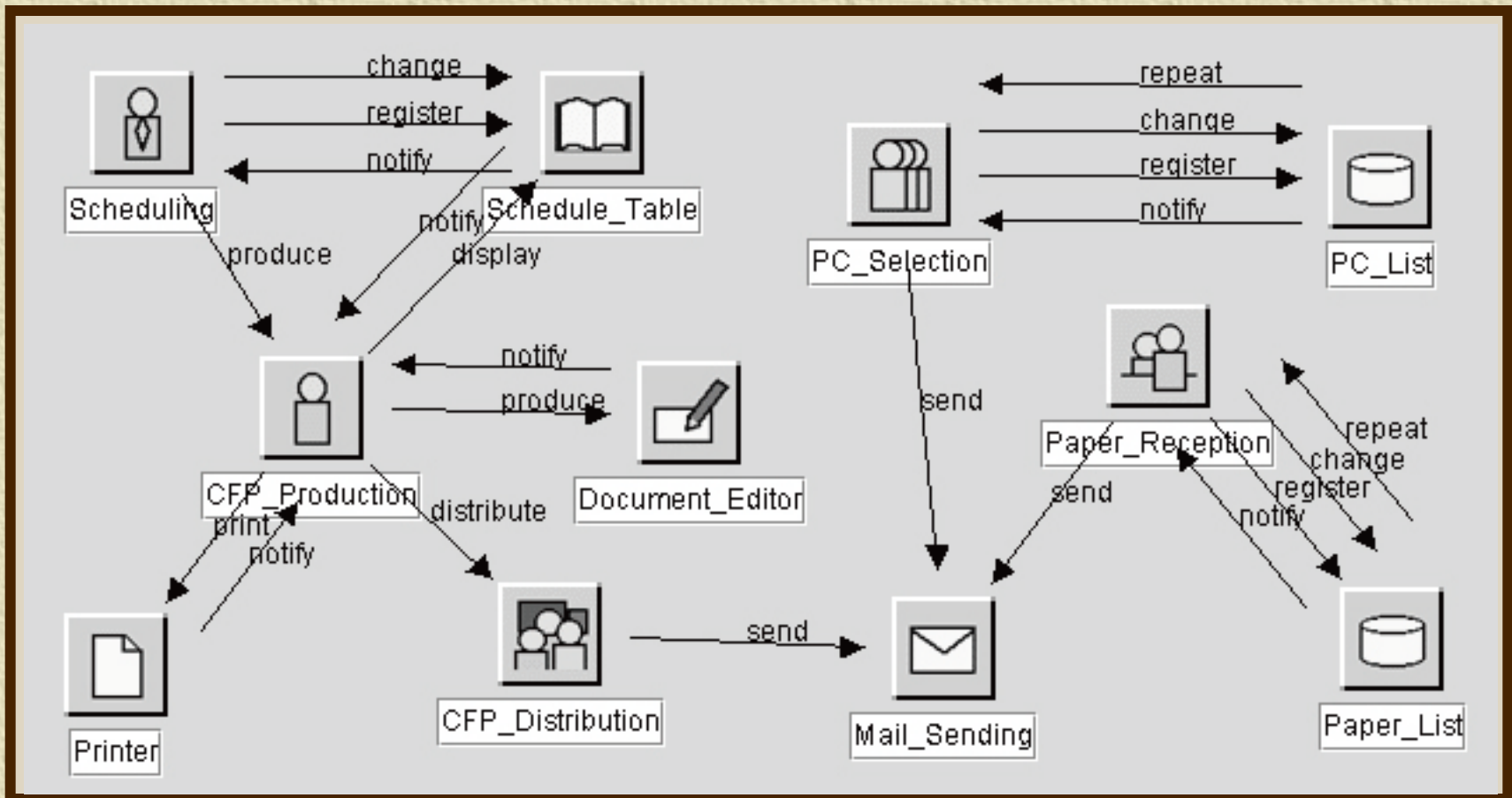
- with the visual modeling tool -

-
- **The system behavior is expressed as a message-driven model.**
 - **User interfaces and the transition diagram are generated automatically, and are used for confirmation of external specifications of the application.**
 - **The system behavior is verified by using a simulation tool.**

An Example of a Domain Model

- while introducing eleven kinds of objects -

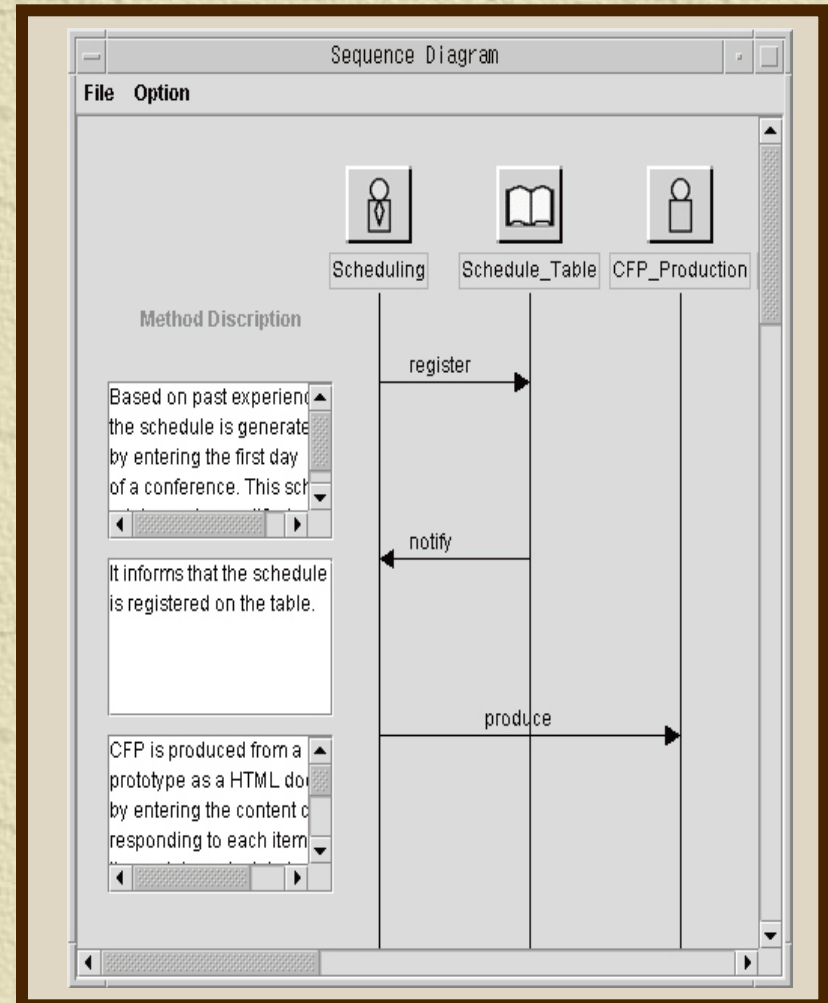
The program chair's role of a academic conference



Requirements Validation

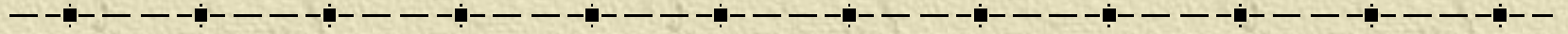
- Modeling and Simulation -

The simulation is executed for validation of the requirement definitions, both on the domain model and on the sequence diagram, while displaying traces of the message flows.



Results of Experiences

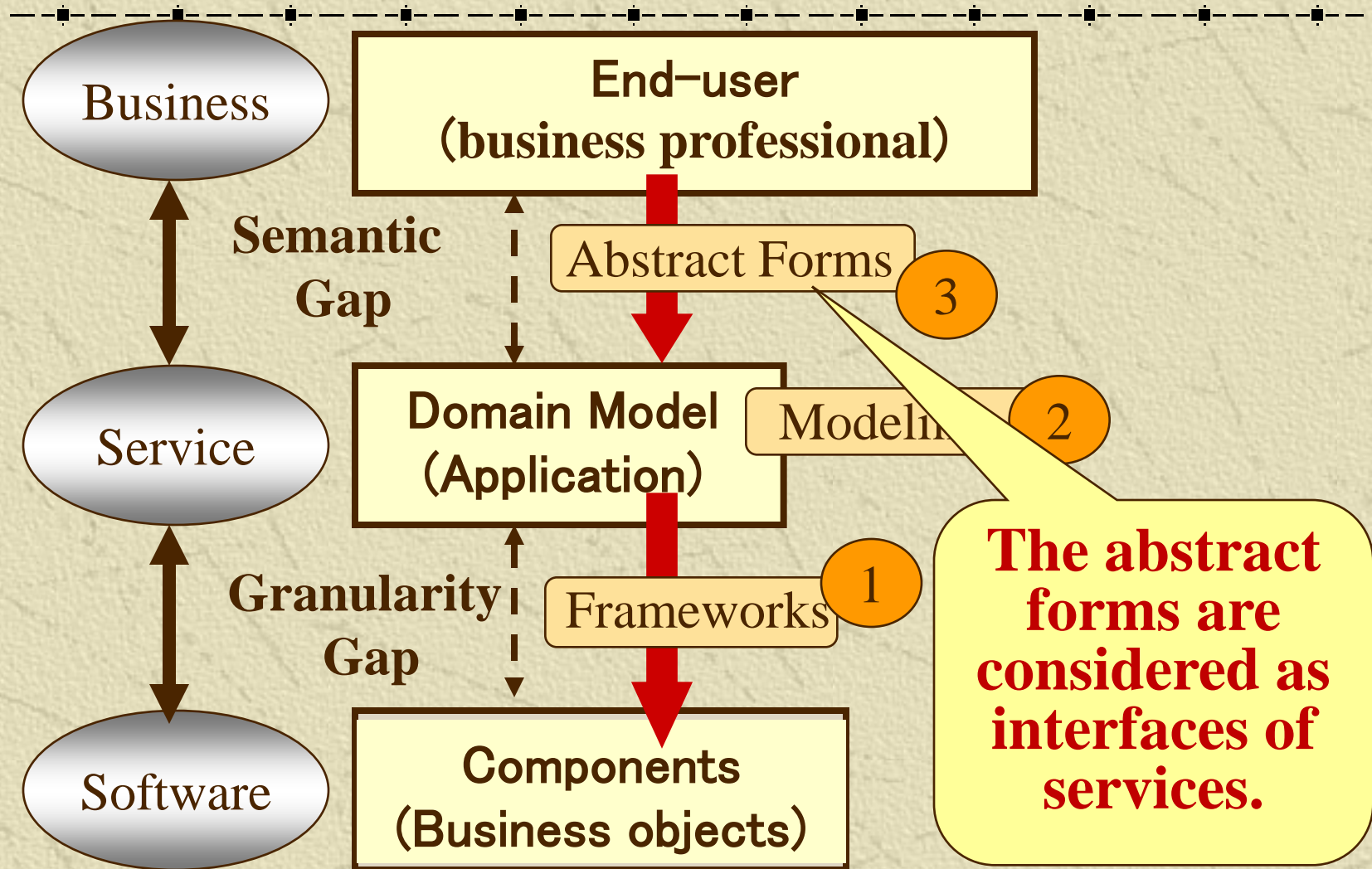
- Visual modeling -



- **The visual modeling process was confirmed through a feasibility study.**
- **End-users need an IT professional's assistance for components to be newly developed and complicated DB management systems.**

Our Recent Approach

- Form-driven approach -



Metaphors for Web Services

- Service counter -

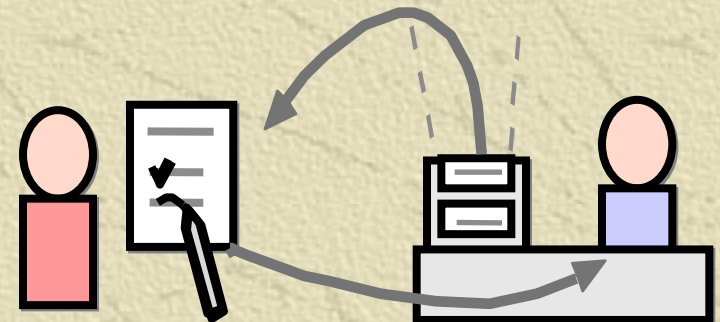
- **Target Domain**

- * A typical distributed system : **service counters**

- * This is not limited to the actual service counter.

(Ex.) SCM can be considered as

combination of the virtual service counters.

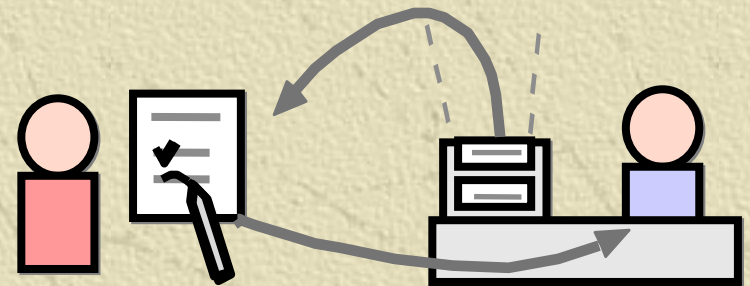


Interfaces for Web Services

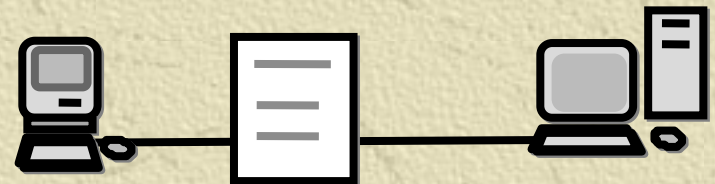
- Forms -

- The service counter receives service requests from clients to service providers.
- Forms are considered as the interface.
- Our concept : **"One service = One form"**

(a) The actual service counter



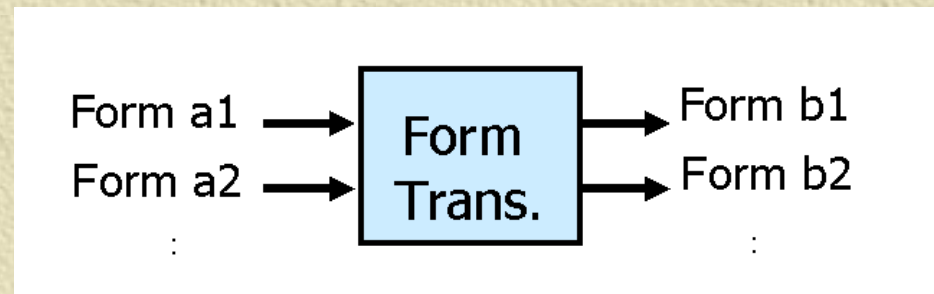
(b) The Web service



Web Service Integration

- **Form Transformation**

- * From input forms into output forms



- **Our Goal**

- * Business professionals make an application by the form transformation.

- Actually, most of these forms are not visual forms, but **abstract forms**.

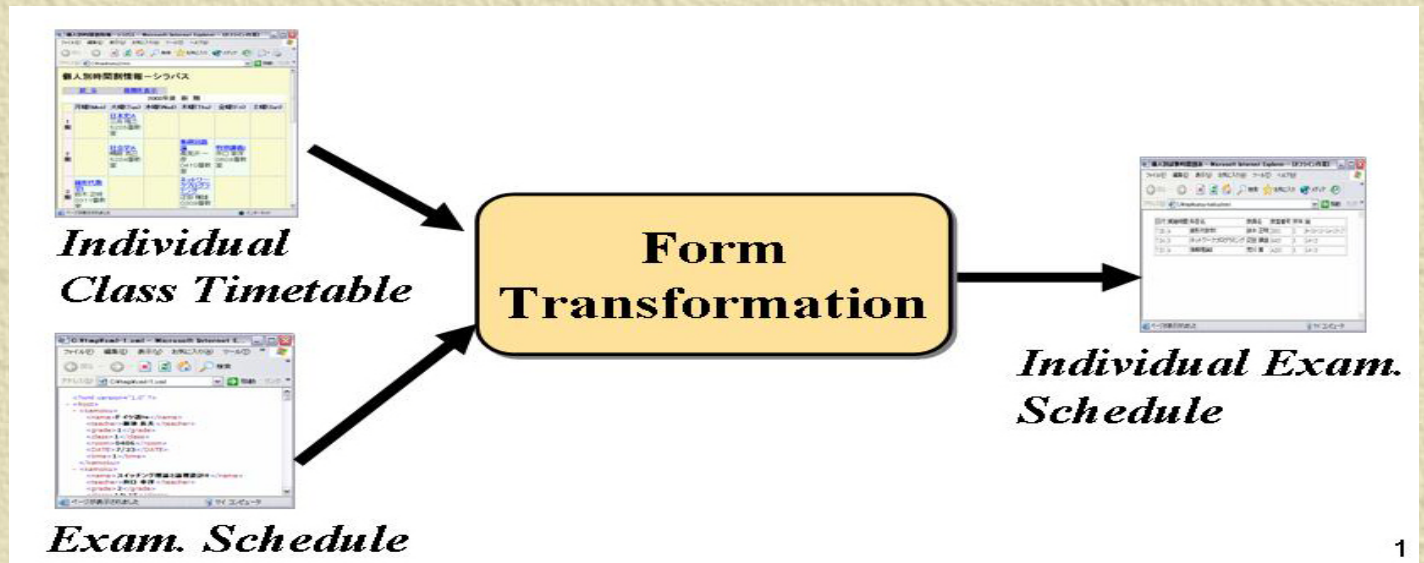
Enduser-Initiative Approach

- **IT skills are not required of end-users because end-users consider such Web service integration as **work flow with visual forms**.**
- **Our previous two approaches are unified.**
 - # **The frameworks as a special case where the abstract forms are actually visual forms.**
 - # **The visual modeling as a special case where the message flow is used instead of the form flow.**

Form Transformation in XSLT

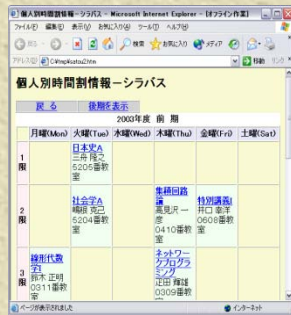
- Basic XML merger -

- This approach is applied to develop an application which generates **an individual examination schedule** from the individual timetable for classes and the examination schedule.



System Architecture

End-users describe the procedure
in a script language by using a visual tool.



HTML

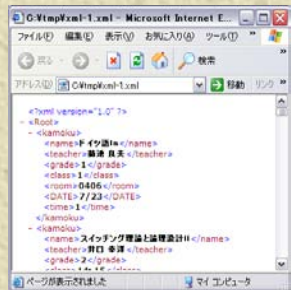
HTML→XML



*Individual
Timetable for classes*

*Individual Exam
Schedule*

XML



XML

Basic
merger

XML

XML→HTML

Exam. Schedule

HTML

Form Transformation by Mapping

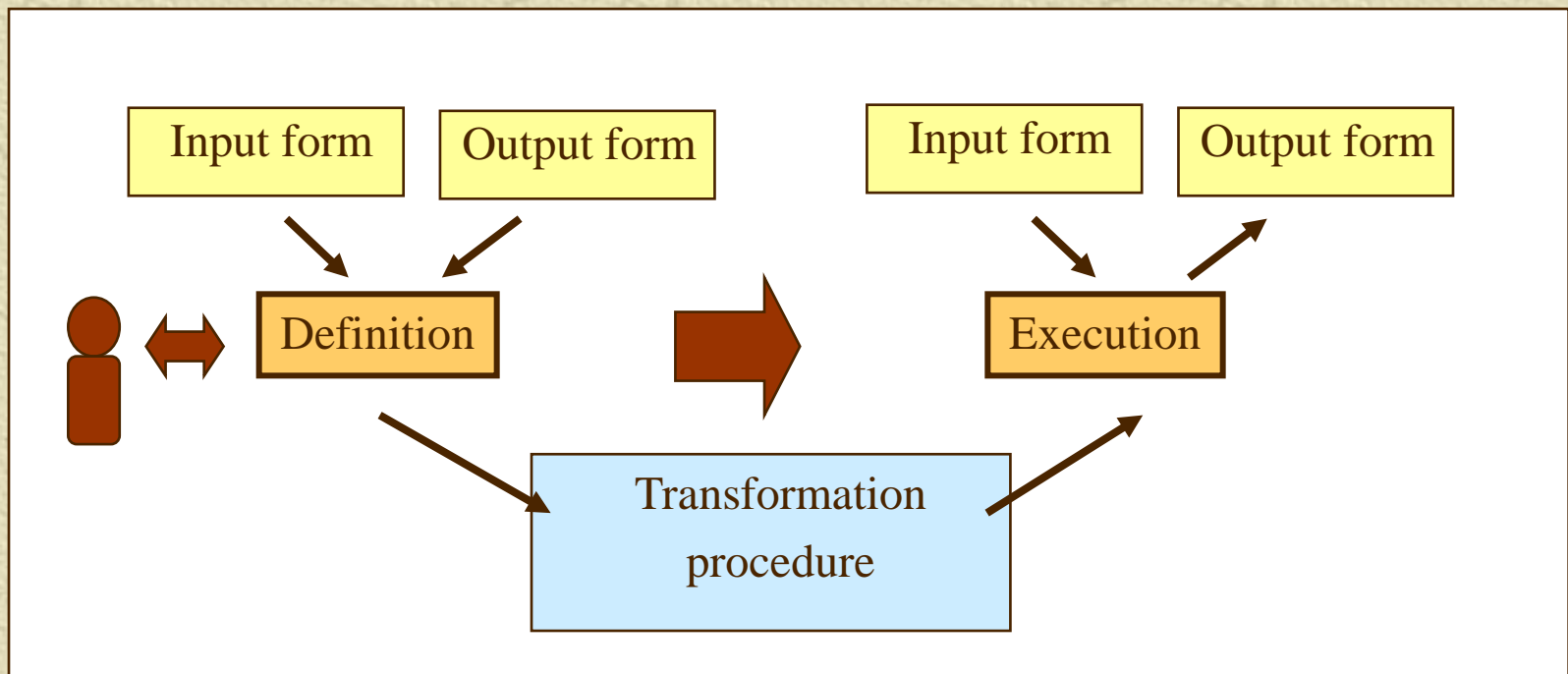
- Form-to-form transformation (FTFT) -

- The form transformation by **mapping** is easier than the form trans. in XSLT since the end-users need not to learn XSLT.
- The form transformation procedure is defined **by only mouse manipulations** to relate items in input forms to items in output forms.

A Visual Tool for FTFT

- Definitions and Executions -

- After defining of the procedure, the form transformations from input forms into output forms is executed.



An Example of Form-to-form transformation (FTFT)

- **A Web application for the reuse of laboratory equipment was selected.**
- **In our university, a lot of secondhand equipment such as PCs are thrown away although many of them can still be used.**
- **If the reuse site is open, the available but unnecessary equipment will be reused by someone.**

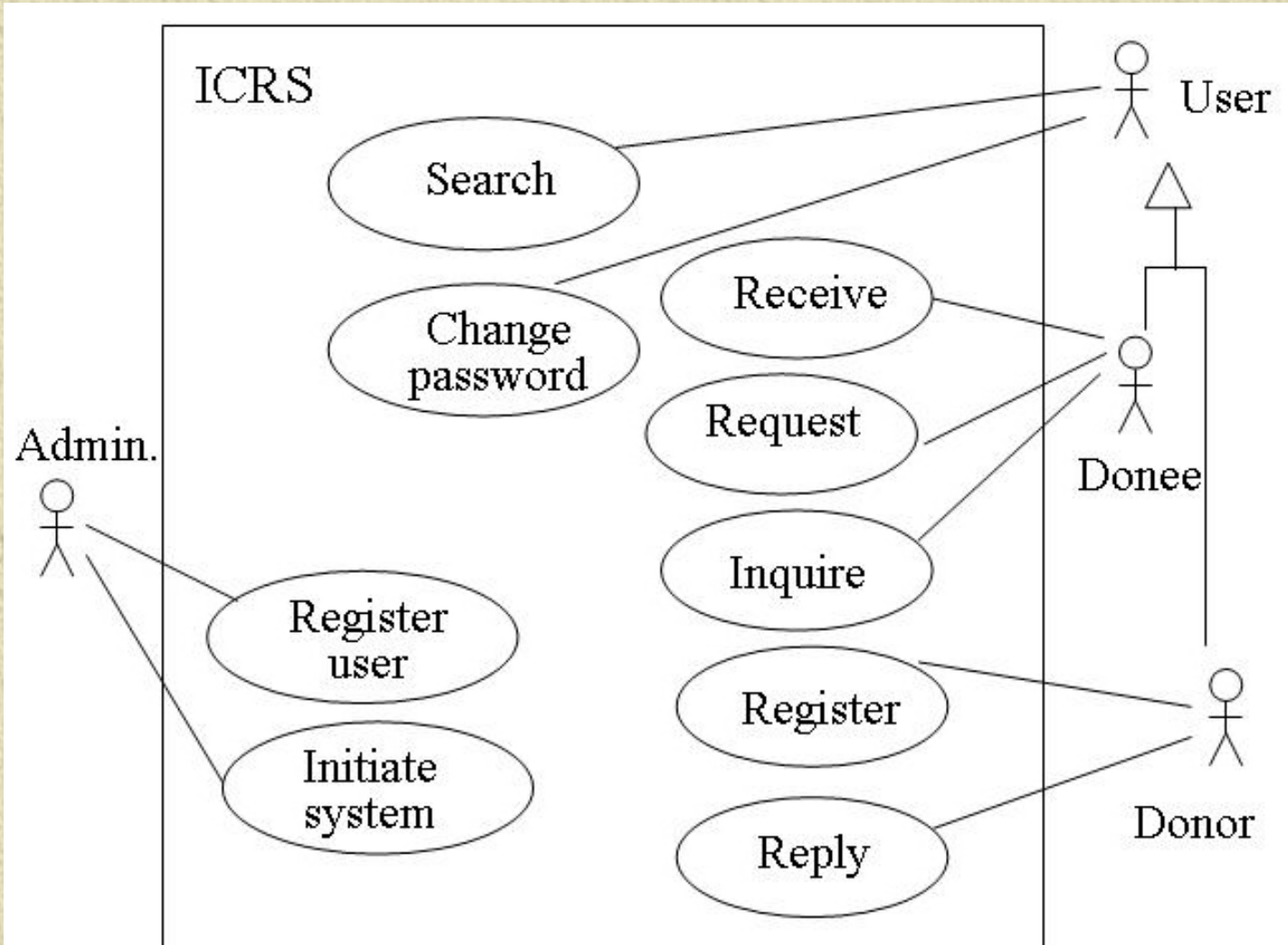
Main Rules for This System

- for less troubles -

-
- 1. Users are limited to members who have mail addresses which are managed by the university.**
 - 2. The equipment should be free for avoiding illegal dealing of the property.**
 - 3. The site administrator takes no responsibility for any troubles since this site is supported by volunteers.**

Main Functions

- A usecase diagram -

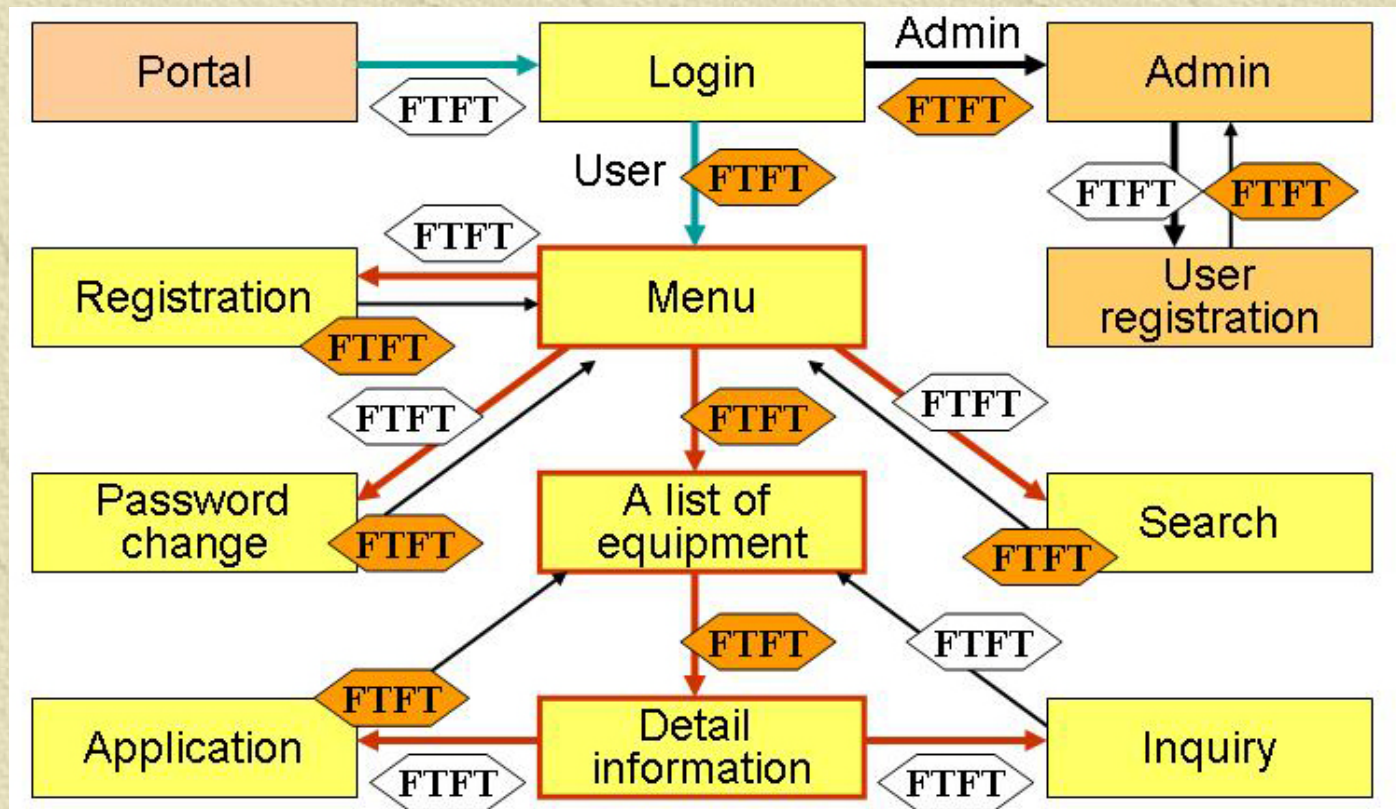


User Interfaces

- UI transition diagram -

FTFT with DB accesses via abstract forms.

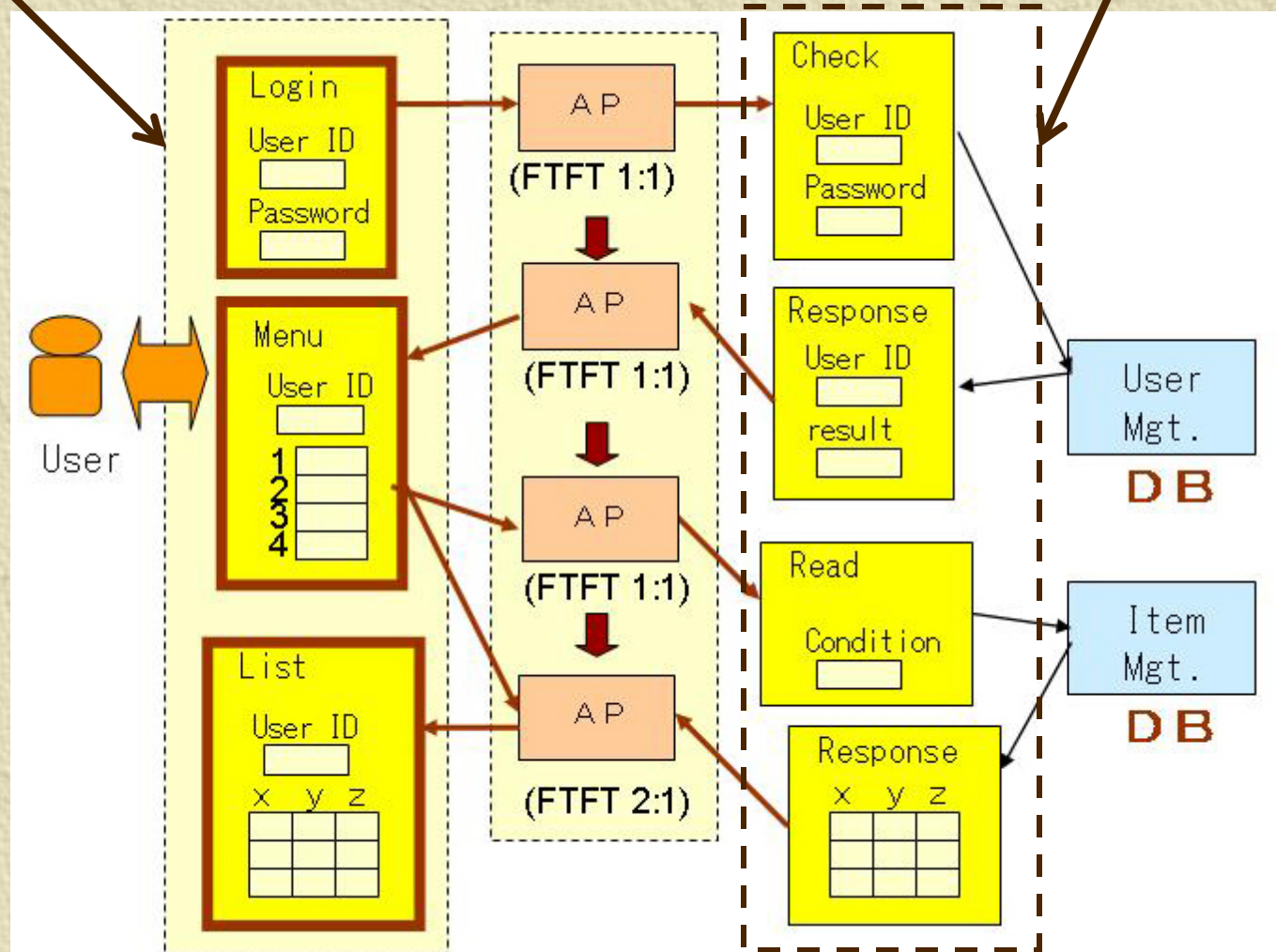
FTFT between visual forms



A Part of Form Flow

An actual visual form

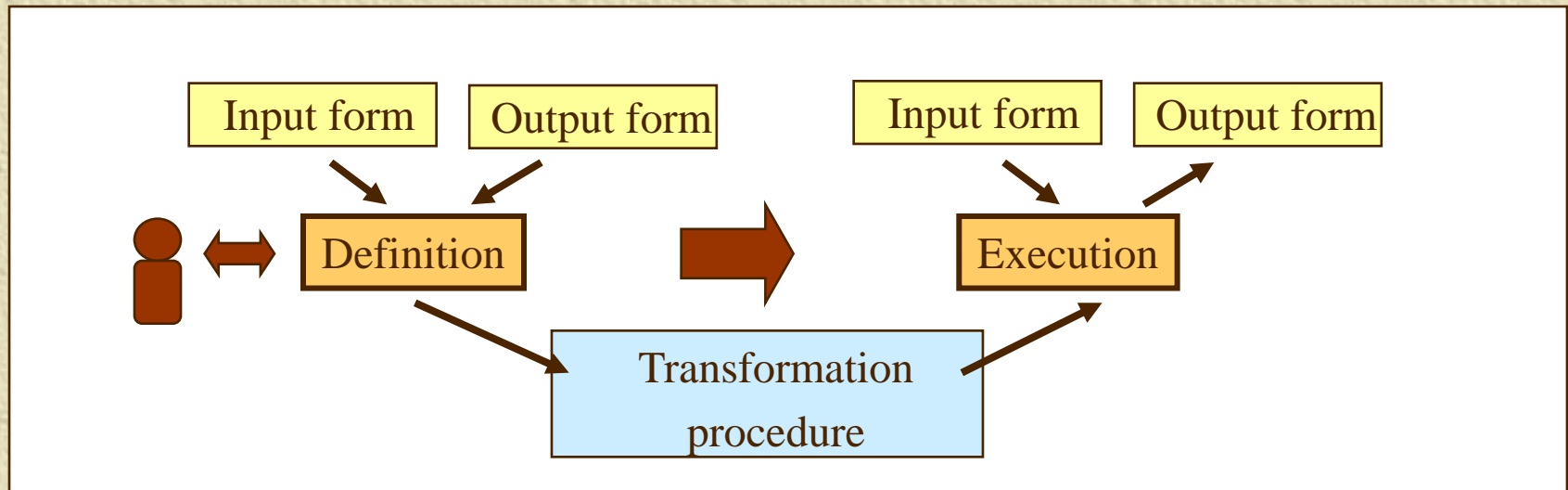
An abstract form



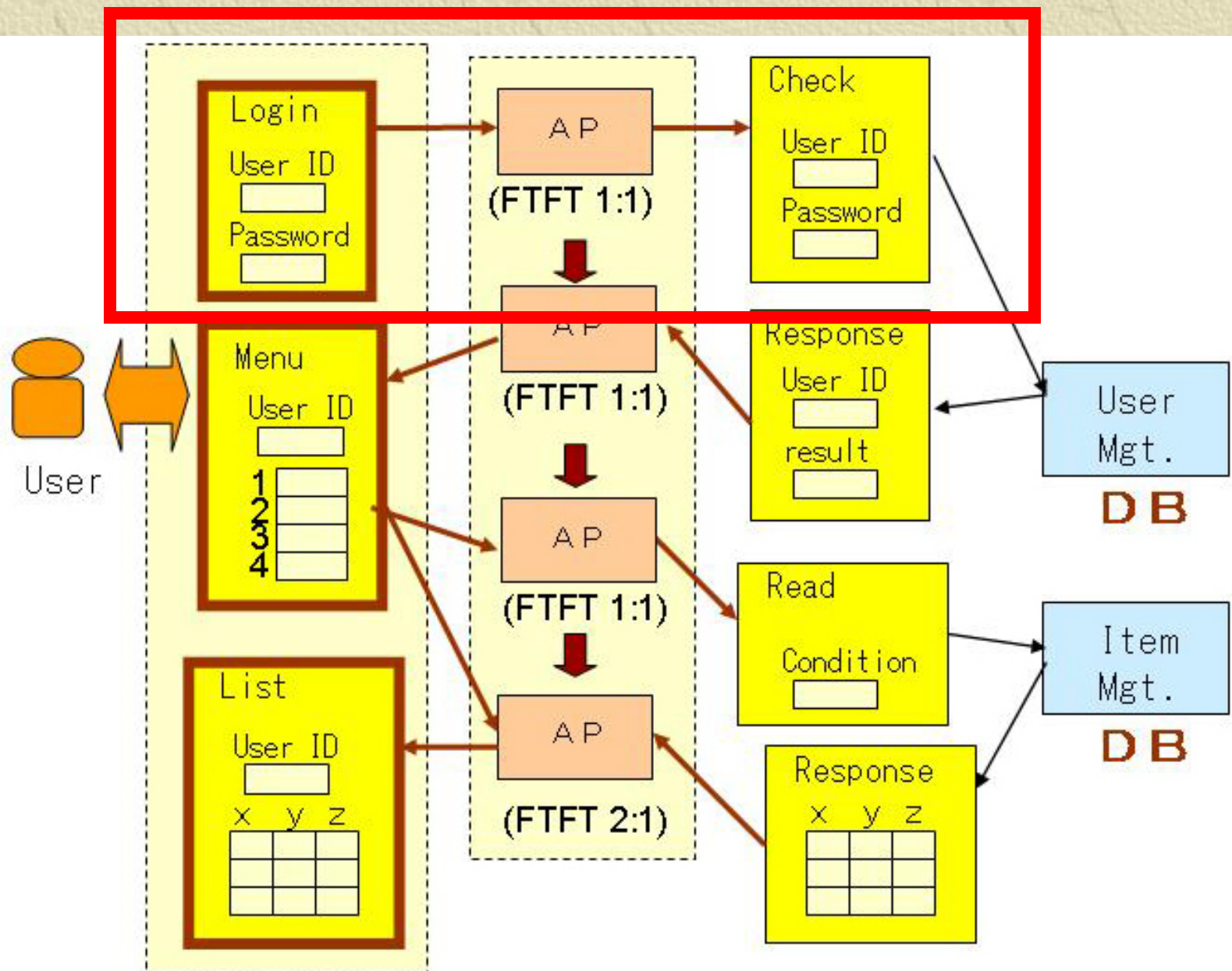
A Visual Tool for FTFT

- Definitions and Executions -

- The UI in HTML and JavaScript.
- The generated procedure in XML.
- The interpreter in Java.



An Example of FTFT



An Example of FTFT Definition

- by using the tool -

An input form

A parette with buttons

An output form

The screenshot displays a software interface for defining FTFT. It features two input forms and a button palette. The first form, titled "Login Form", contains two input fields: "User ID" and "Password". The second form, titled "Check Abstract-Form", also contains "User ID" and "Password" fields. To the right is a palette of buttons including "Undo", "Init", "=", "x", "y", "z", another "=", "f", "g", "h", and another "=". Below the forms is a table with two columns: "Order" and "Item".

Order	Item
1	Login.UserID
2	EQUAL
3	Check.UserID
4	INIT
5	Login.Password
6	EQUAL
7	Check.Password
8	INIT

An Example of FTFT definition

- by mouse manipulation -

** Input Forms **

Login Form

User ID 1

Password 5

** Output Forms **

Check Abstract-Form

User ID 3

Password 7

Undo

4 Init 8

2 =

x

y

z

6 =

f

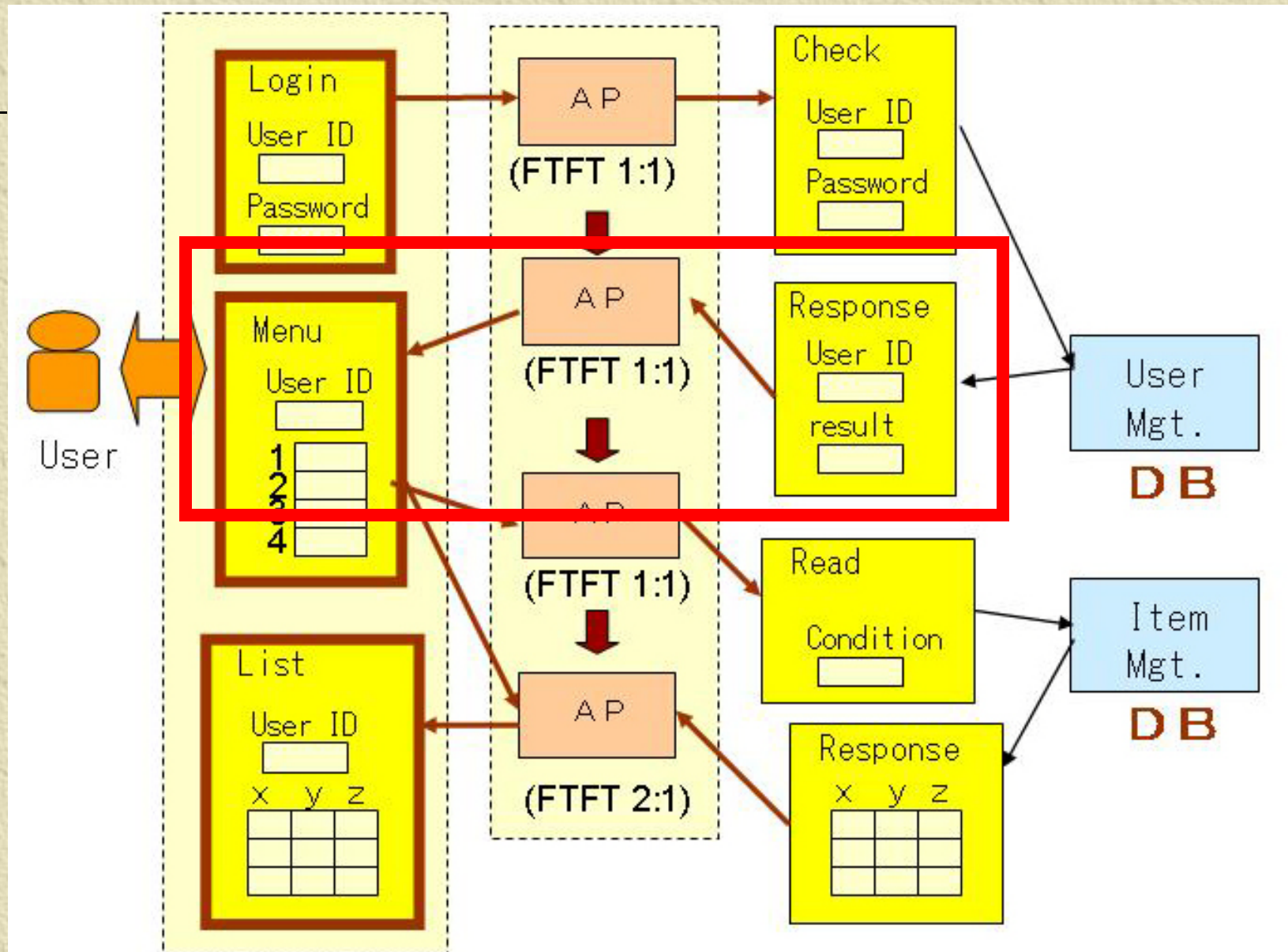
g

h

=

Order	Item
1	Login.UserID
2	EQUAL
3	Check.UserID
4	INIT
5	Login.Password
6	EQUAL
7	Check.Password
8	INIT

Another example of FTFT



Functions for Complex Business Logic

** Input Forms **

Response Abstract-Form

User ID **1**

Result **5**

** Output Forms **

Menu Form

User ID **3**

List **11**

[5,6,7,8]
List = f (Result)
[9,10,11,12]

Undo

4 Init **8** **12**

2 =

x y

z

6 =

7f **9** g

h

10 =

Order	Item
1	Response.UserID
2	EQUAL
3	Menu.UserID
4	INIT
5	Response.Result
6	EQUAL
7	FUNCf
8	INIT
9	FUNCf
10	EQUAL
11	Menu.List
12	INIT

Conclusions

- ✦ The enduser-initiative requirement definition method based on **abstract forms** are proposed.
- ✦ The business logic can be defined as **the form transformation** from input forms into output forms.

