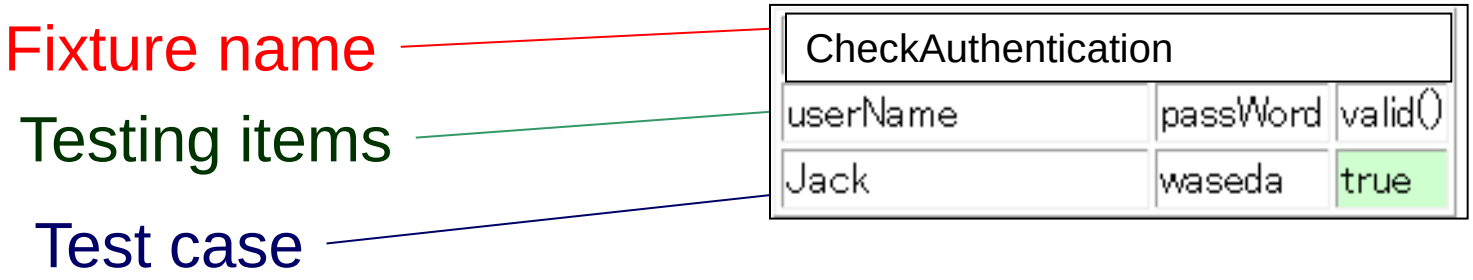
Acceptance Testing based on Relationships among Use Cases

Kariyuki Susumu, **Hironori Washizaki**, Yoshiaki Fukazawa
Waseda University
Atsuto Kubo, Aoyama Media Laboratory
Mikio Suzuki, TIS Inc.

http://www.washi.cs.waseda.ac.jp/

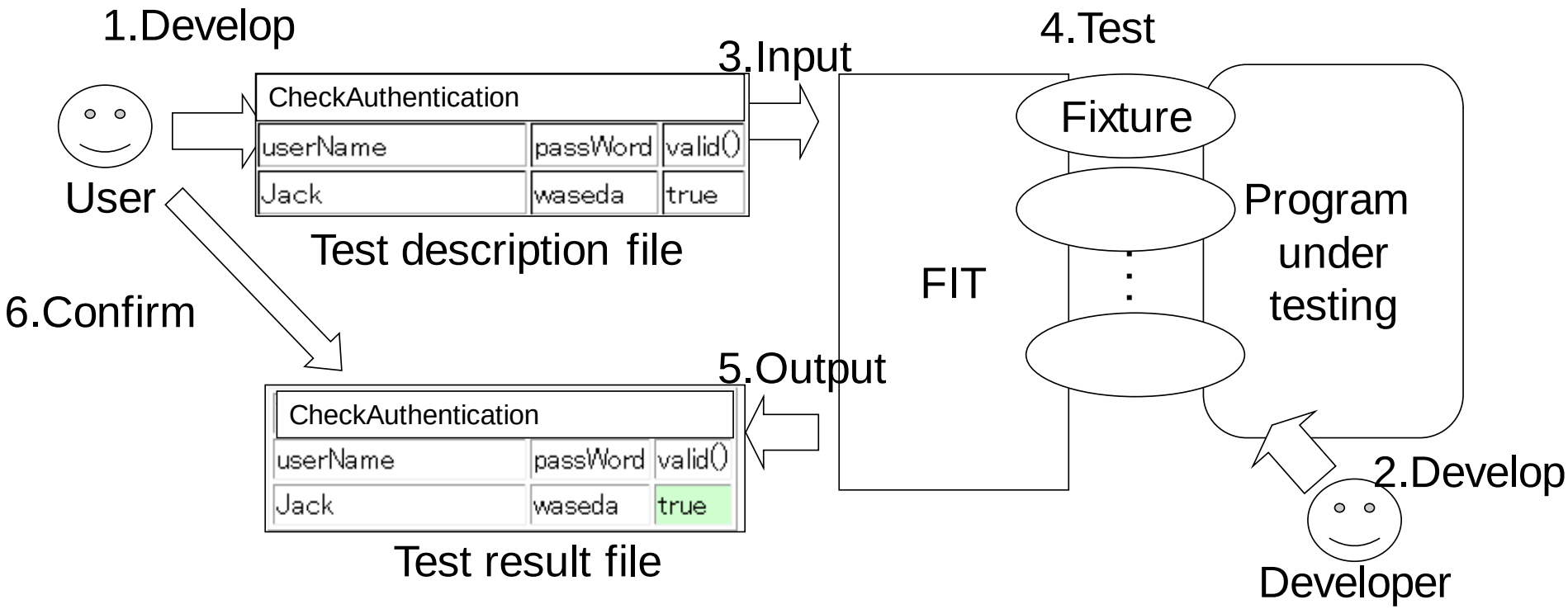
Acceptance testing and FIT

- Testing by users confirming whether the system performs the requirements
- Framework for Integrated Test(FIT): framework for acceptance testing



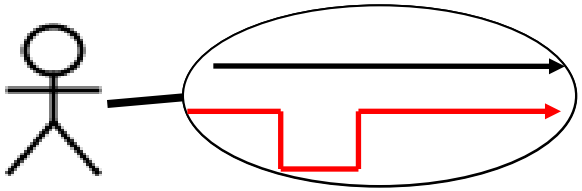

```
import fit.*;
public class CheckAuthenticationFixture extends ColumnFixture{
    public String userName;
    public String passWord;
        public boolean valid() {
                return true;
        }
}
```

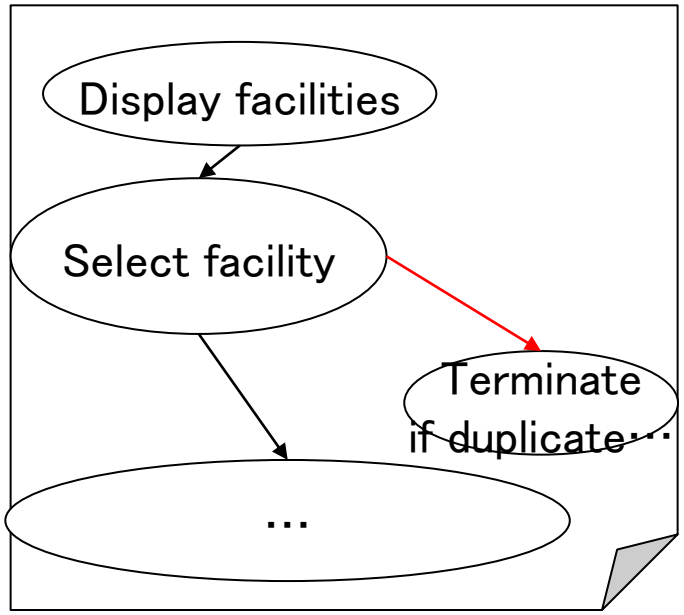
Overview of FIT



Use case

- descriptions of the external functions of a system from the viewpoint of actors
  - Used for defining test cases of functionality acceptance testing





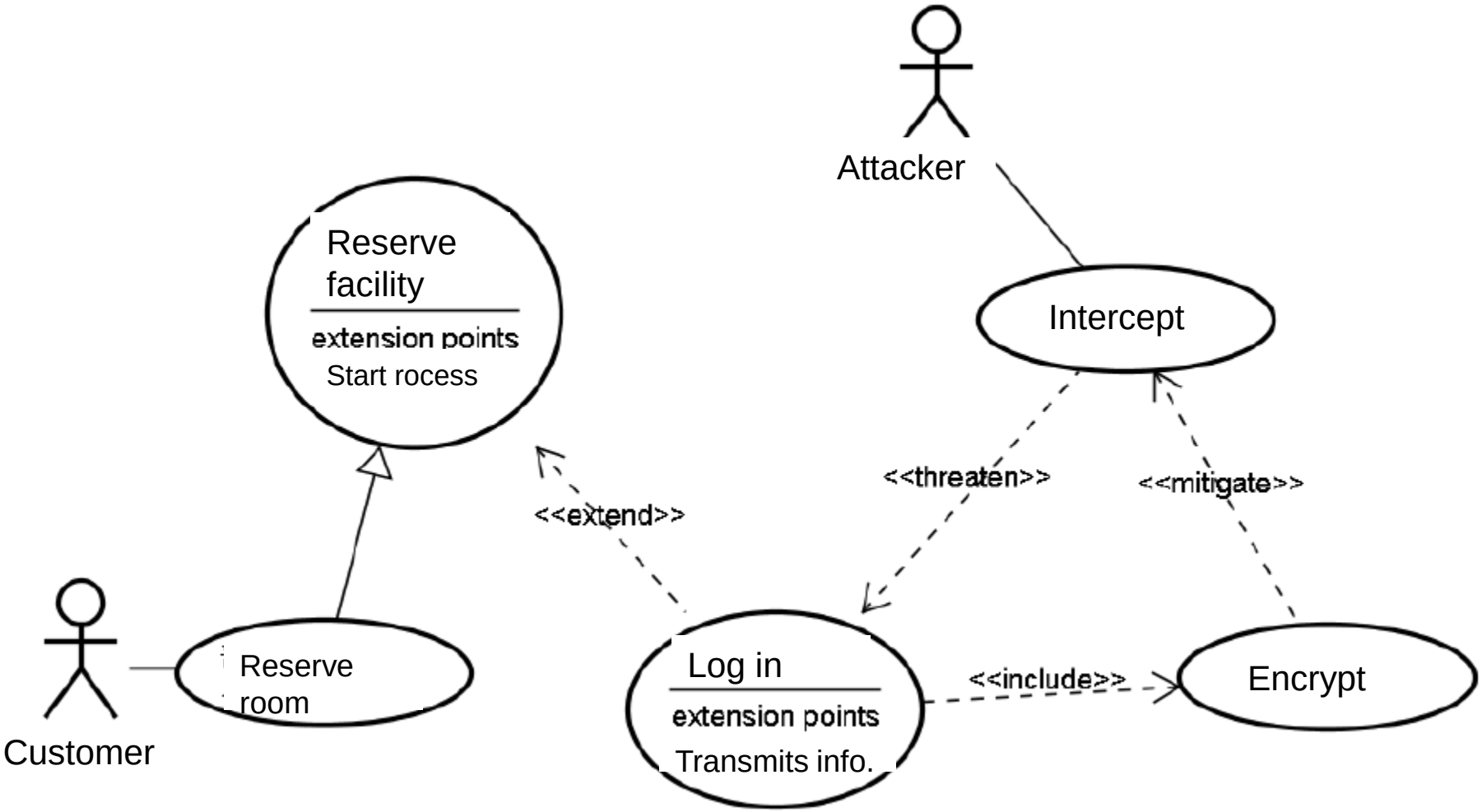
Use case: Reserve facility
Main flow name: Reserve facility
1. The system displays available facilities.
2. The customer selects a facility.
…
Alternative flow:
Duplicate application
…

Use case description

Multiple execution flows could exist in one use case.

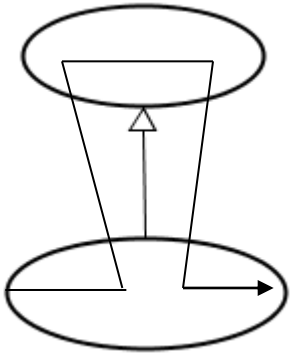
Relationships among Use Cases

- Include: including steps of another use case
- Generalization: “is-a” between use cases
- Extend: adding steps of another use case to existing use case

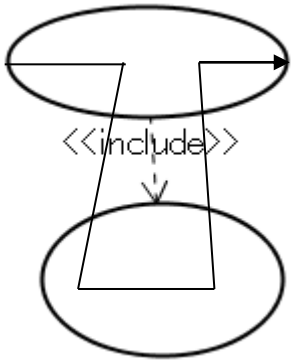


Relationships and changes in execution flows

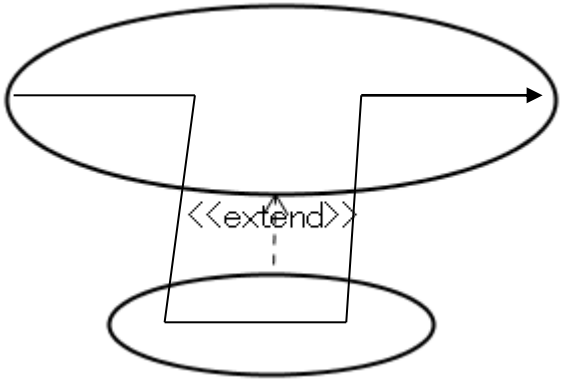
- Generalization



- Include



- Extend



Execution flows become complex as relationships among use cases become complex.

What's the problem?

In conventional approaches, execution flows of the use cases are manually identified.

- Problem 1: A widely accepted definition for the coverage of the acceptance test using test scenarios identified from use cases has not yet been established
- Problem 2: A complicated relationship between use cases may lead to the incomplete coverage of execution flows.

We propose solutions to these problems:

- **Solution for problem 1:** we define the coverage for the acceptance test targeting test scenarios identified from use case descriptions. Following this definition, the dependence of the judgment of acceptance test completion on the individual can be avoided.
- **Solution for problem 2:** We automatically identify the execution flows of use cases using a FIT-based-system for generating the test scenarios and skeleton codes for the acceptance test environment

[IBM06] Peter Zielczynski, Traceability from Use Cases to Test Cases
http://www.ibm.com/developerworks/rational/library/04/r-3217/?S_TACT=105AGX90&S_CMP=content

Coverage of acceptance test using use cases

Step coverage: C'_0

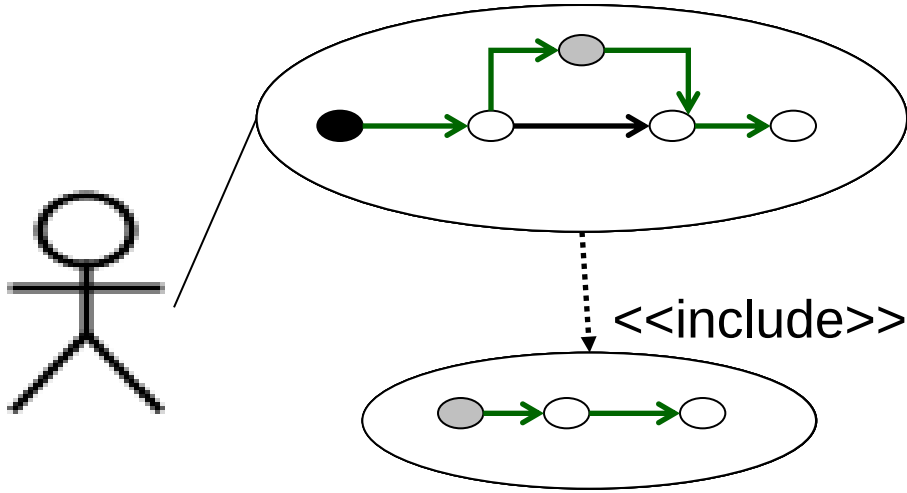
$$C'_0 = \frac{\#steps\_executed}{\#All\_steps}$$

Branch coverage C'_1

$$C'_1 = \frac{\#Branches\_tested}{\#All\_branches}$$

All-execution-flows coverage C'_∞

$$C'_\infty = \frac{\#Flow\_tested}{\#All\_flows}$$



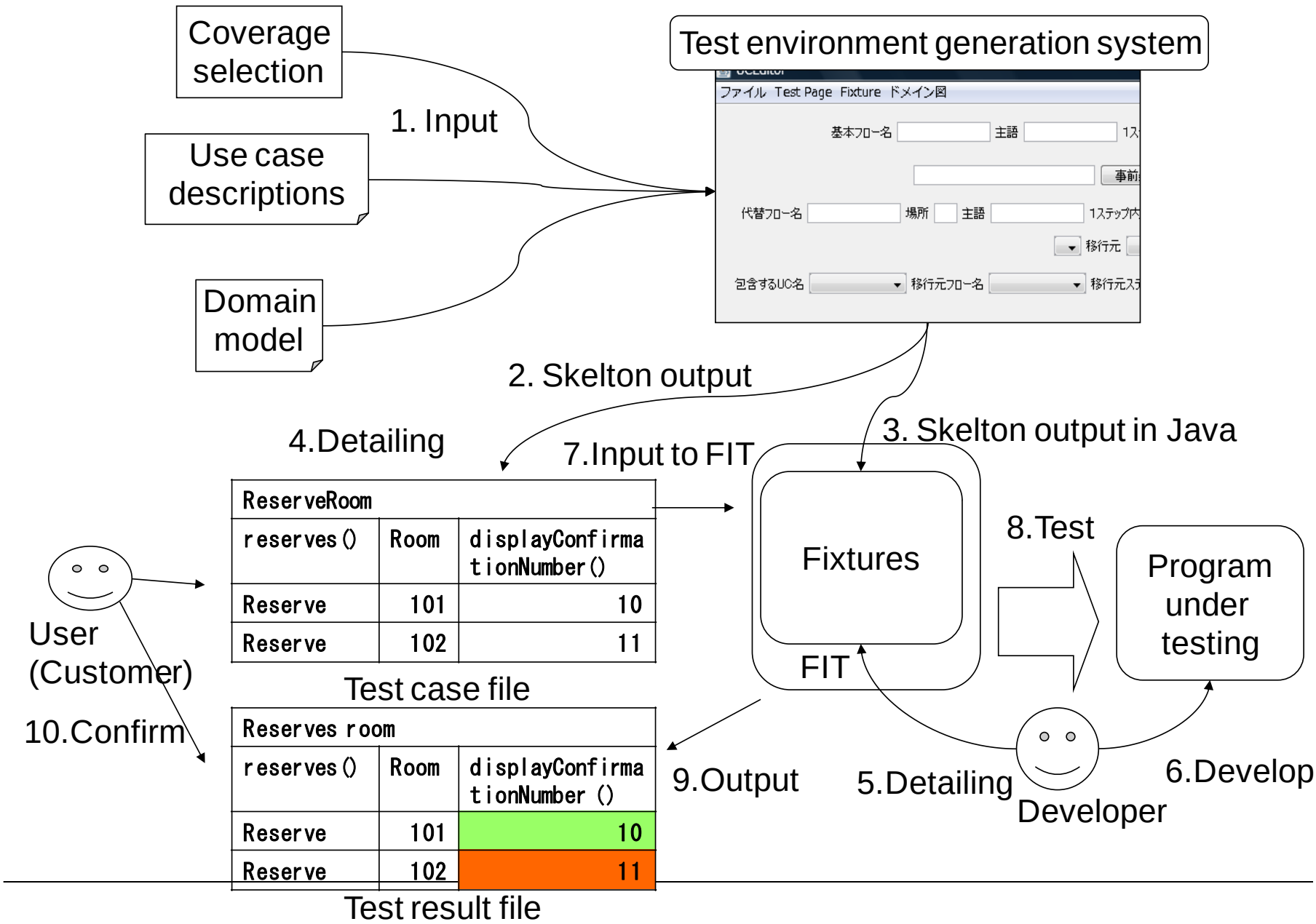
If the green flow has been executed:

C'_0=100%

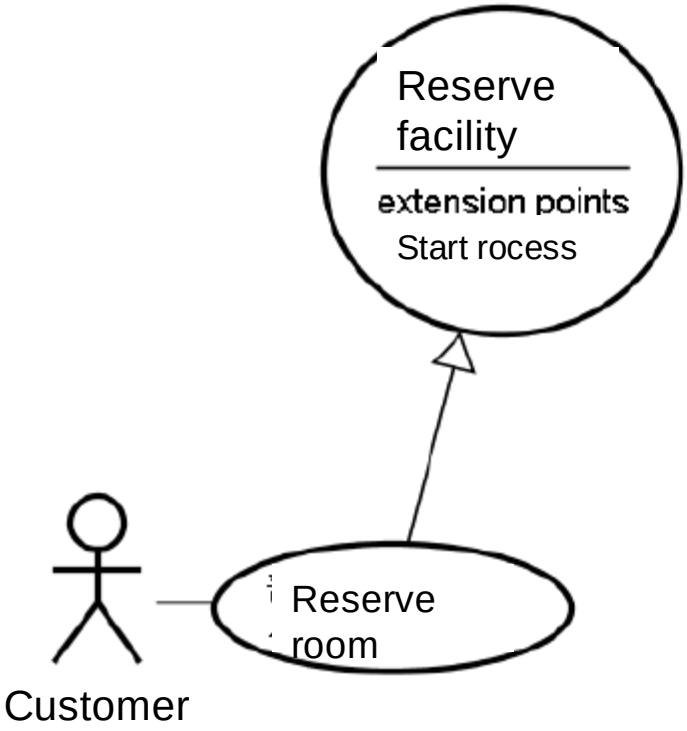
C'_1=50%

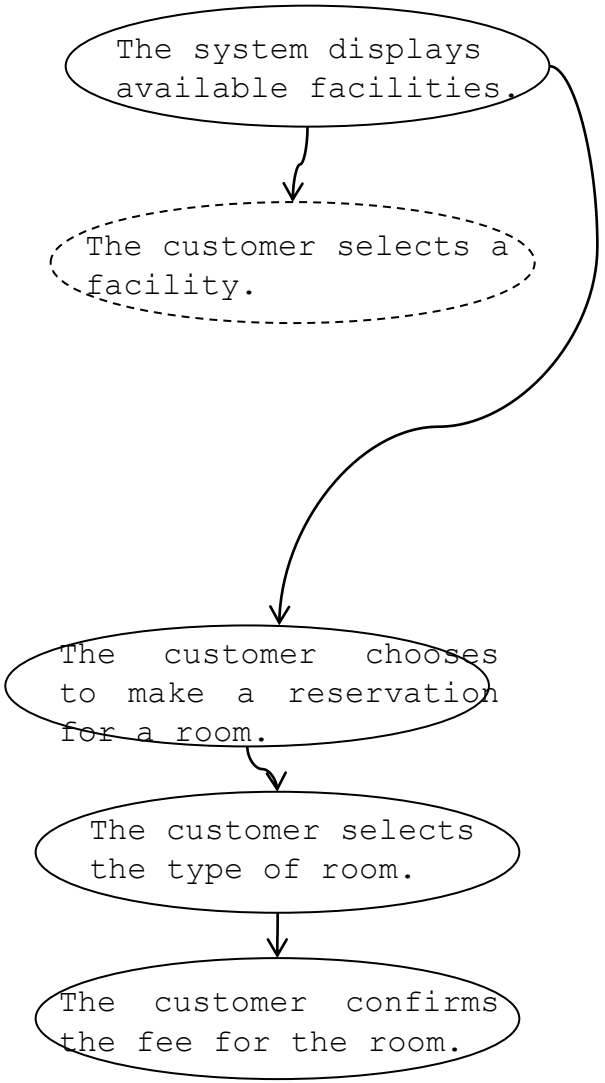
C'_∞=50%

Overview of test environment generation system



Changes in execution flow: generalization



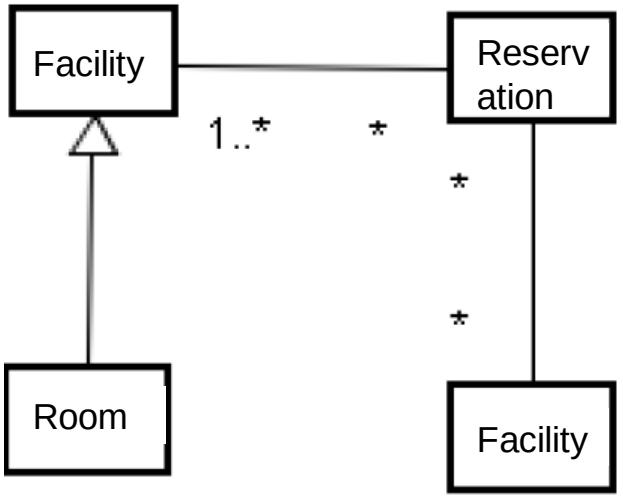


Generating skeletons of test scenarios

- Generation of skeleton test scenarios
  - One table for one execution flow
  - Specify domain entities by referring to the domain model
- Detailing of the test scenarios

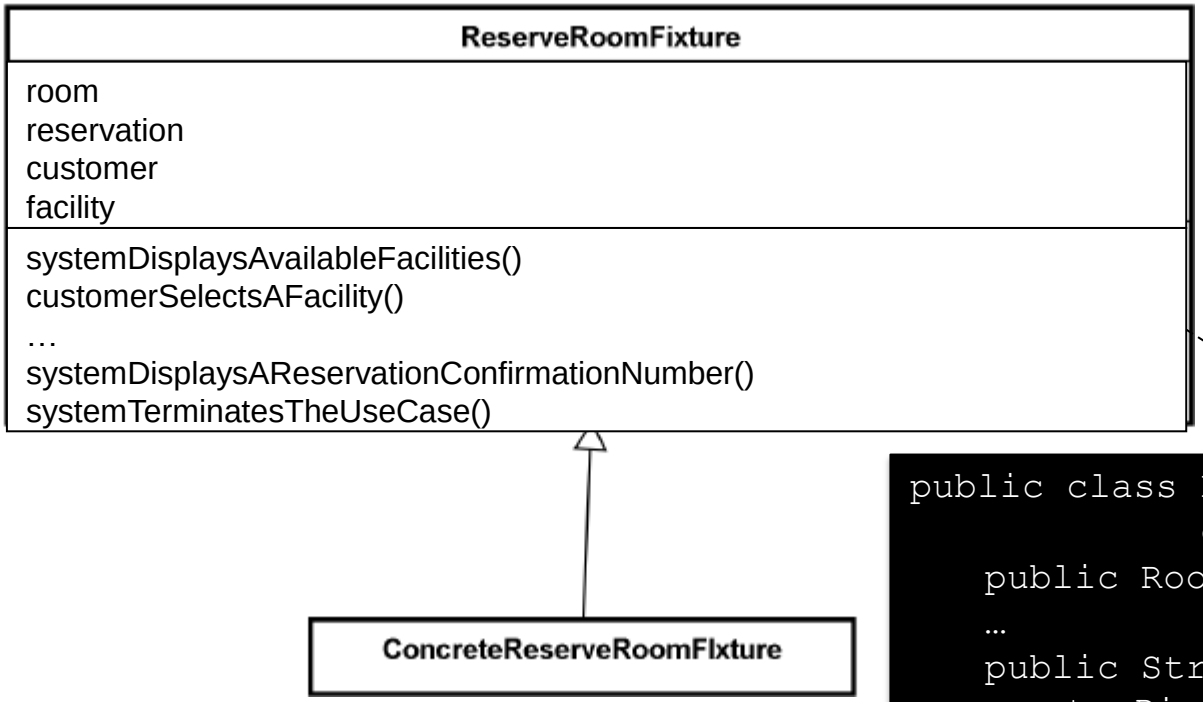
ReserveRoom				
systemDisplaysAvailableFacilities()	Room	customerChoosesToMakeAReservationForARoom()	…	systemTerminatesTheUseCase()
Pass	null	pass	…	pass
Pass	null	pass	…	pass

ReserveRoom				
systemDisplaysAvailableFacilities()	Room	customerChoosesToMakeAReservationForARoom()	…	systemTerminatesTheUseCase()
List	101	Reserve	…	exit
List	201	Reserve	…	exit



Generating fixtures

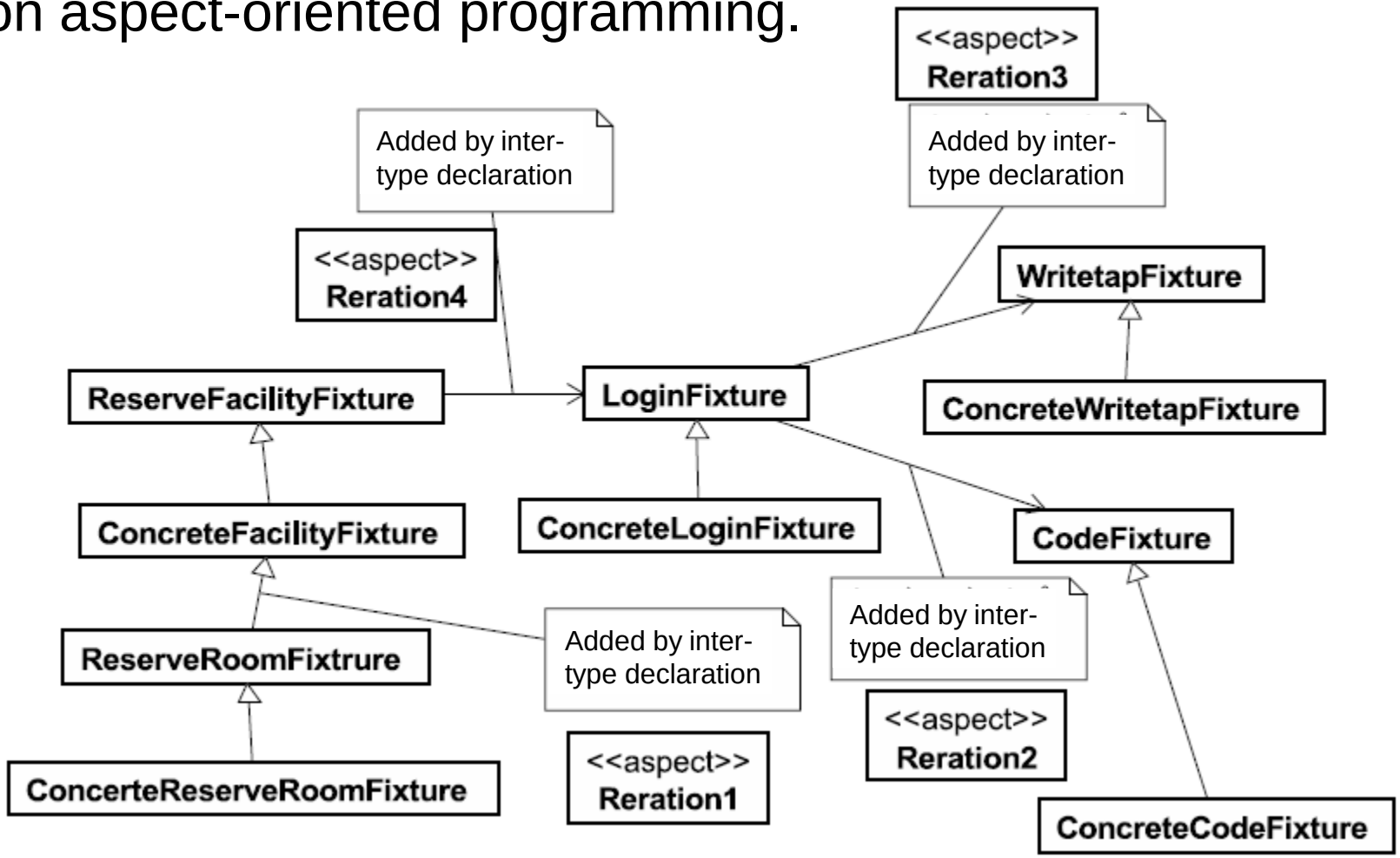
- Use case Abc -> AbcFixture
- Methods correspond to steps in use case description
- Attributes correspond to domain entities




```
public class ReserveRoomFixture
             extends ColumnFixture {
    public Room room;
    …
    public String
     systemDisplaysAvailableFacilities() {
        return “pass”;
    }
    …
```

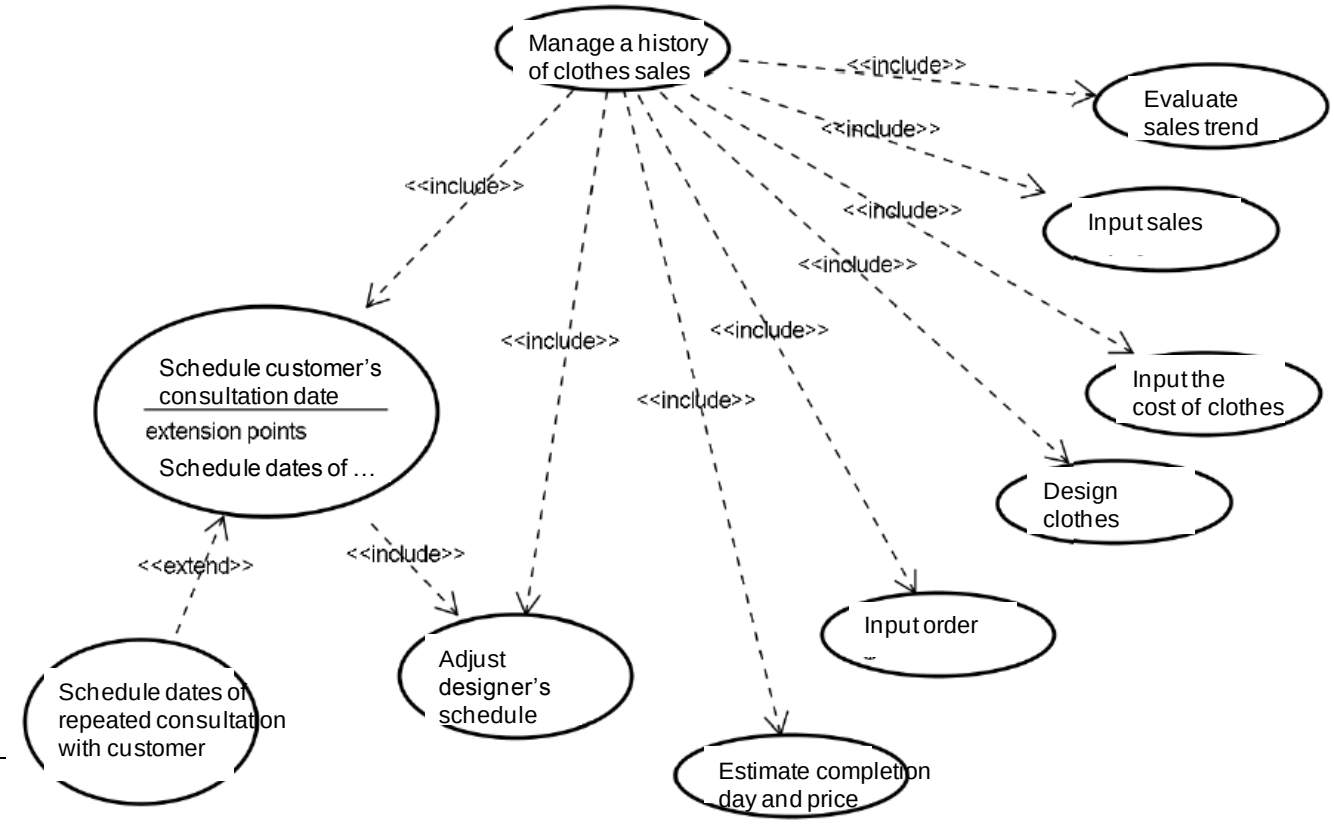
Adding fixture relationships

Relationship between fixtures corresponding to use case relationship is defined by an inter-type declaration based on aspect-oriented programming.



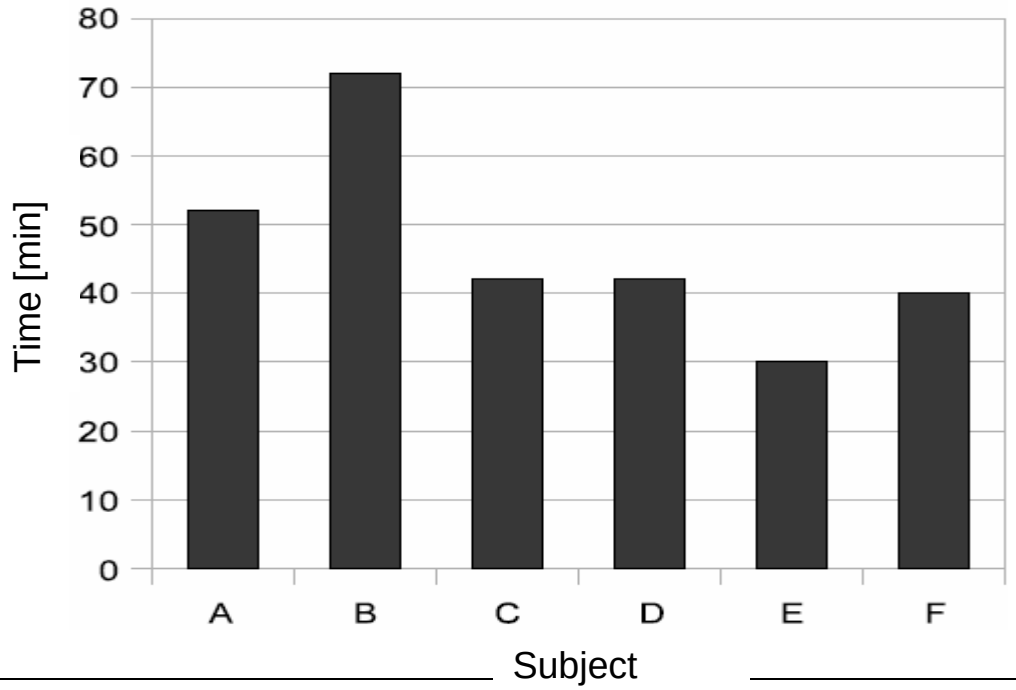
Experiment on identifying execution flows

- A comparative experiment was carried out using the conventional(manual) and proposed techniques
  - Identify all execution flows in "Manage …"
  - List the test scenarios using branch coverage
- Subjects: six students with two years UML experience
- Statistics: 10 use cases, 47 steps, 32 flows



Result of comparative experiment

- Subjects listed 0-2 incorrect flows in conventional technique
  - In proposed technique, flows are automatically identified; errors and incomplete coverage can be prevented.
- Subjects took 30-70 min in conventional technique
  - These times can be almost eliminated using the proposed technique.



Summary and future work

- Contribution
  - Three coverages for the acceptance test based on the test scenarios identified from use cases
  - Technique of automatically generating skeleton test scenarios and skeleton test programs using FIT
  - We confirmed that there is a possibility of calculating incomplete coverage without our technique
  - By using our technique,
    - acceptance test completion can be determined without depending on individual judgments
    - efficiency of the acceptance test can be increased by partial automated generation.
- Future work
  - Larger experiments
  - Generation of test cases in addition to execution flows