Trade-off Analysis between Security Policies for Java Mobile Codes and Requirements for Java Application

Haruhiko Kaiya

Shinshu University, JAPAN

Sep. 11, 2003

http://www.cs.shinshu-u.ac.jp/~kaiya/

Background and Motivation

- Mobile codes are useful,
  - e.g., constructing services on the fly, reuse.
- but sometimes dangerous.
  - e.g., threats to valuable resources.
- Requirements Analysis Method for Mobile codes applications is needed.
- First step
  - Java mobile codes only, for simplicity.
  - Security policies.

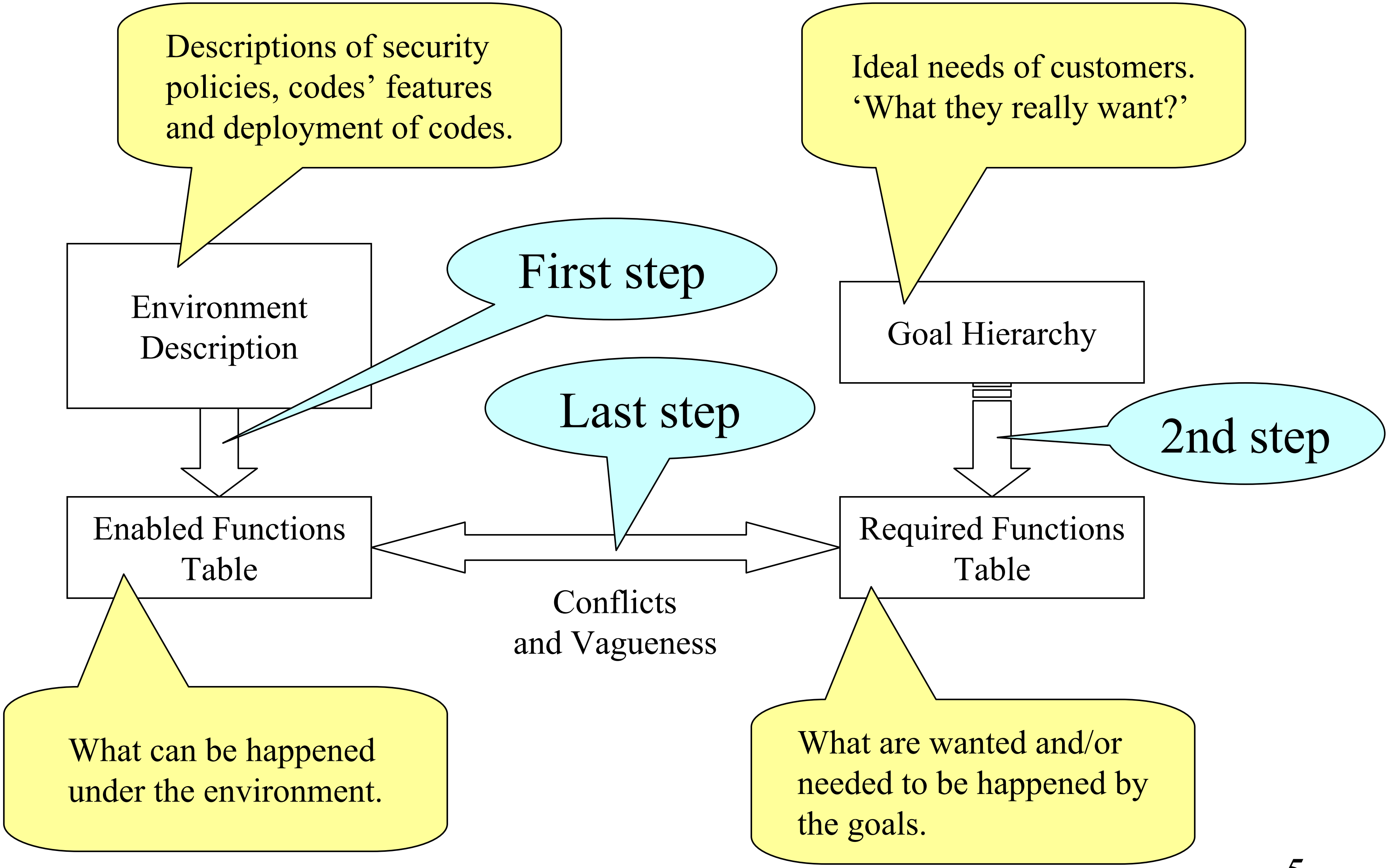
Role of Security Policies in Java

- Restrict the functions of mobile codes.
- Incomplete Policies.
  - allow inadequate and/or malicious functions
  - hard to find them – anti-requirements, which show ‘something should not happen!’.
- Complete Policies for Java.
  - cannot avoid inadequate/malicious functions completely,
  - because access controls are applied not to each code but to each location.

Research Issue and Strategy

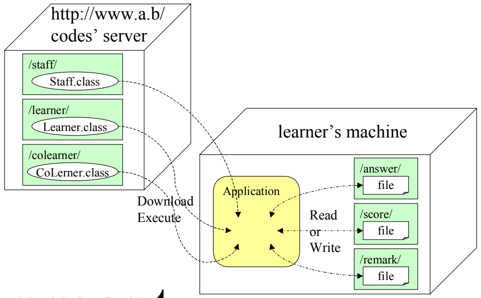
- Get feasible requirements specification
  - for an application using Java mobile codes
  - under an environment.
    - An environment means security policies, functions of each mobile code, and deployment of mobile codes.
- Compromise differences between goals for application users and the environment.
  - Abandon several goals for application users so as to meet the environment, if possible.
  - Modify several parts of the environment so as to meet the goals, if possible.

Analysis Framework



Descriptions of security policies, codes' features and deployment of codes.
Ideal needs of customers. 'What they really want?'
First step
Environment Description
Goal Hierarchy
Last step
2nd step
Enabled Functions Table
Required Functions Table
Conflicts and Vagueness
What can be happened under the environment.
What are wanted and/or needed to be happened by the goals.

Environment and Enabled Functions



```
// Policies for Java

grand codeBase
  "heep://www.a.b/staff/" {
     permission java.io.FilePermission{
        "/answer/*", "read";
     permission ………….
```

Environment
(deployment and policies and code functions)

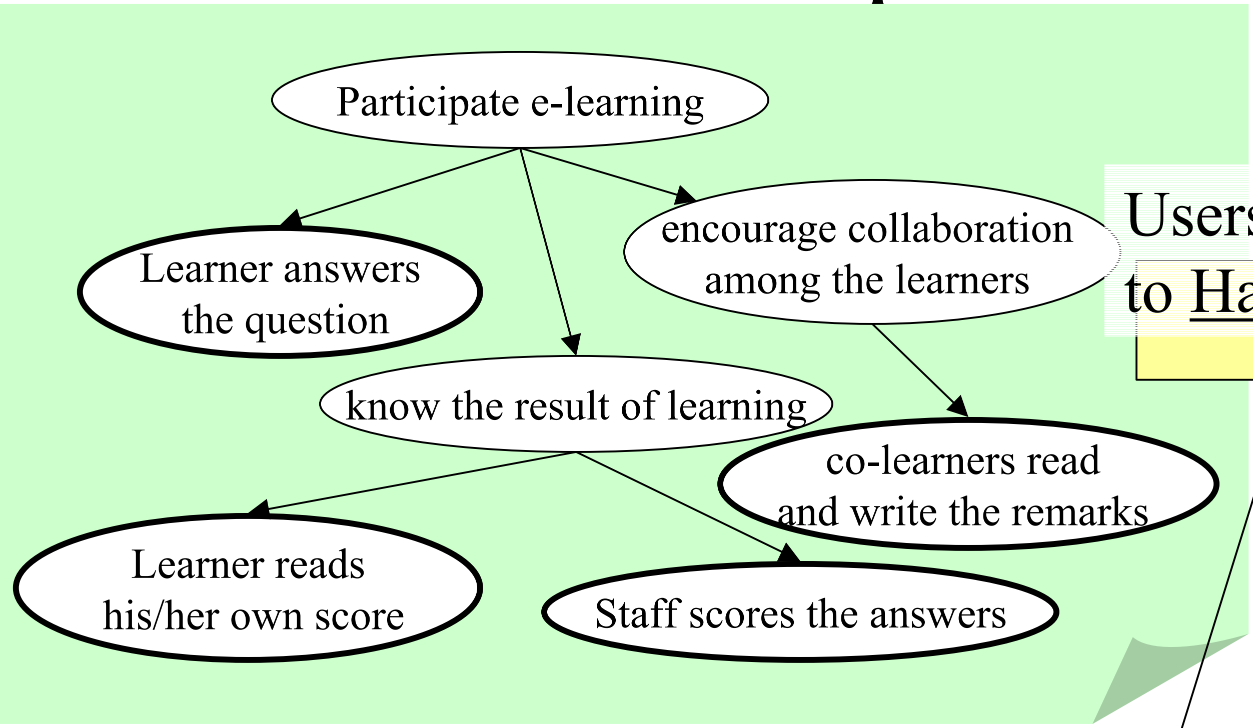
A Code X can handle
a Resources Y.

	both in Staff.class			
	Teacher	Admin.	CoLearner	Learner
Answer	r+ w-	r+ w-	-	r+ w+
Score	r+ w+	r+ w+	-	r+ w-
Remark	r+ w+	r+ w+	r+ w+	r+ w+

Resources

Codes

Goals and Required Functions



Participate e-learning

Learner answers the question

know the result of learning

encourage collaboration among the learners

co-learners read and write the remarks

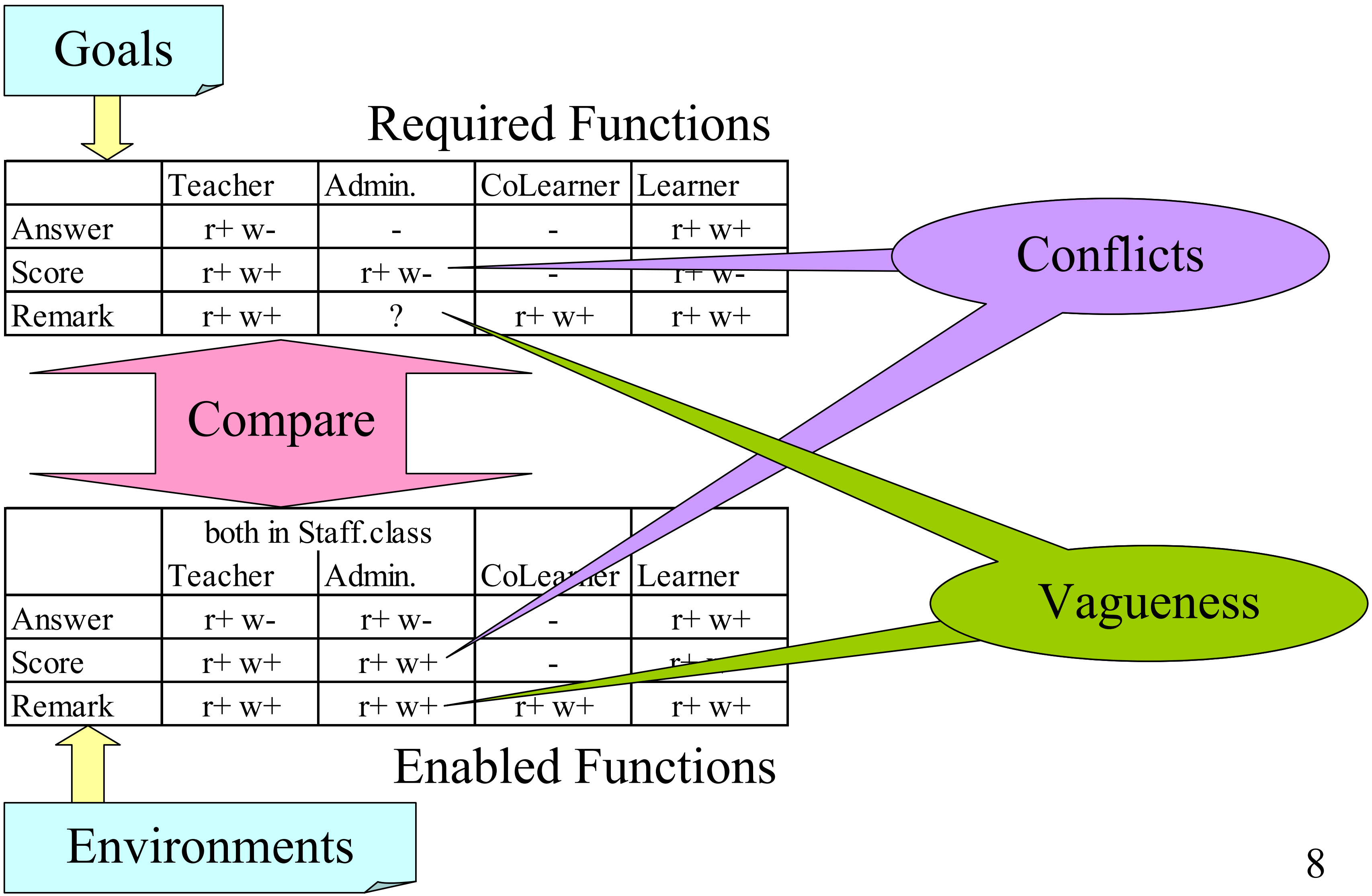
Learner reads his/her own score

Staff scores the answers

Users want a Code X to Handle a Resource Y.

	Teacher	Admin.	CoLearner	Learner
Answer	r+ w-	-	-	r+ w+
Score	r+ w+	r+ w-	-	r+ w-
Remark	r+ w+	?	r+ w+	r+ w+

Conflicts and Vagueness



Goals

Required Functions

	Teacher	Admin.	CoLearner	Learner
Answer	r+ w-	-	-	r+ w+
Score	r+ w+	r+ w-	-	r+ w-
Remark	r+ w+	?	r+ w+	r+ w+

Compare

	both in Staff.class Teacher	 Admin.	CoLearner	Learner
Answer	r+ w-	r+ w-	-	r+ w+
Score	r+ w+	r+ w+	-	r+ w-
Remark	r+ w+	r+ w+	r+ w+	r+ w+

Enabled Functions

Environments

Conflicts

Vagueness

Supporting Tools

- Goal Oriented Requirements Analysis.
  - decompose and convert abstract goals to concrete goals (functions).
  - Get required functions.
- Security Policy Checker and Generator.
  - check which code can be executed or not under an environment.
  - Get enabled functions.

Current and Next Works

- Current
  - A method has been designed.
  - CASE tools are partially implemented.
- Next
  - Completing and integrating tools.
  - Finding realistic examples for our method.

Future Works

- Support *user-centric style* access control.
  - ‘Who runs the application?’
  - JASS (Java Authentication & Authorization Services)
  - Now *code-centric style* only.
- Beyond the security mechanism for Java.
  - too simple to be used in general.
- Handle conflicts among stakeholders.
  - AGORA

I want to show
the next slide in

12th IEEE International Requirements Engineering conference

September 6-10, 2004
Kyoto, Japan

(if possible)