Human Factors in Security Administration: Brainstorming the Research Directions

Konstantin Beznosov
University of British Columbia

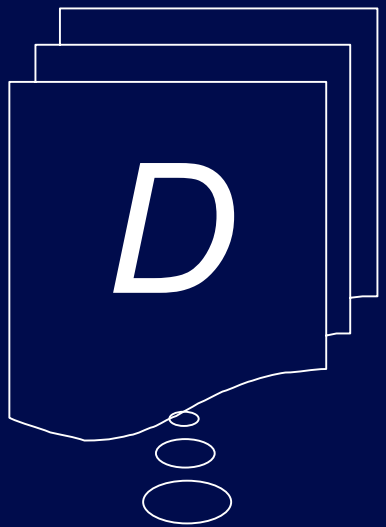
Outline

- State of the practice
- State of the art
- Research directions ideas

State of the Practice

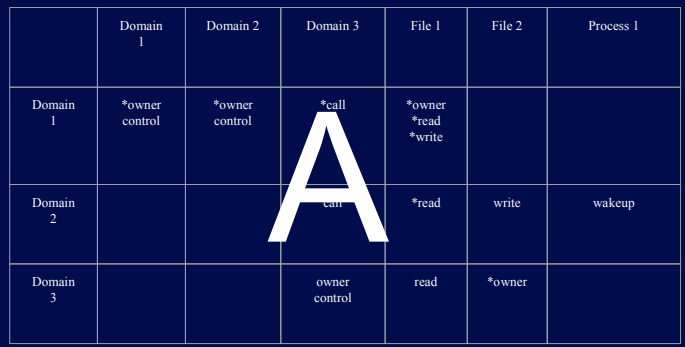
Classical Access Control Solution

Domains



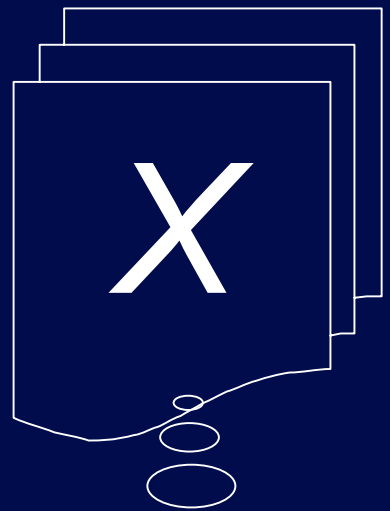
Have access to objects

Access Matrix



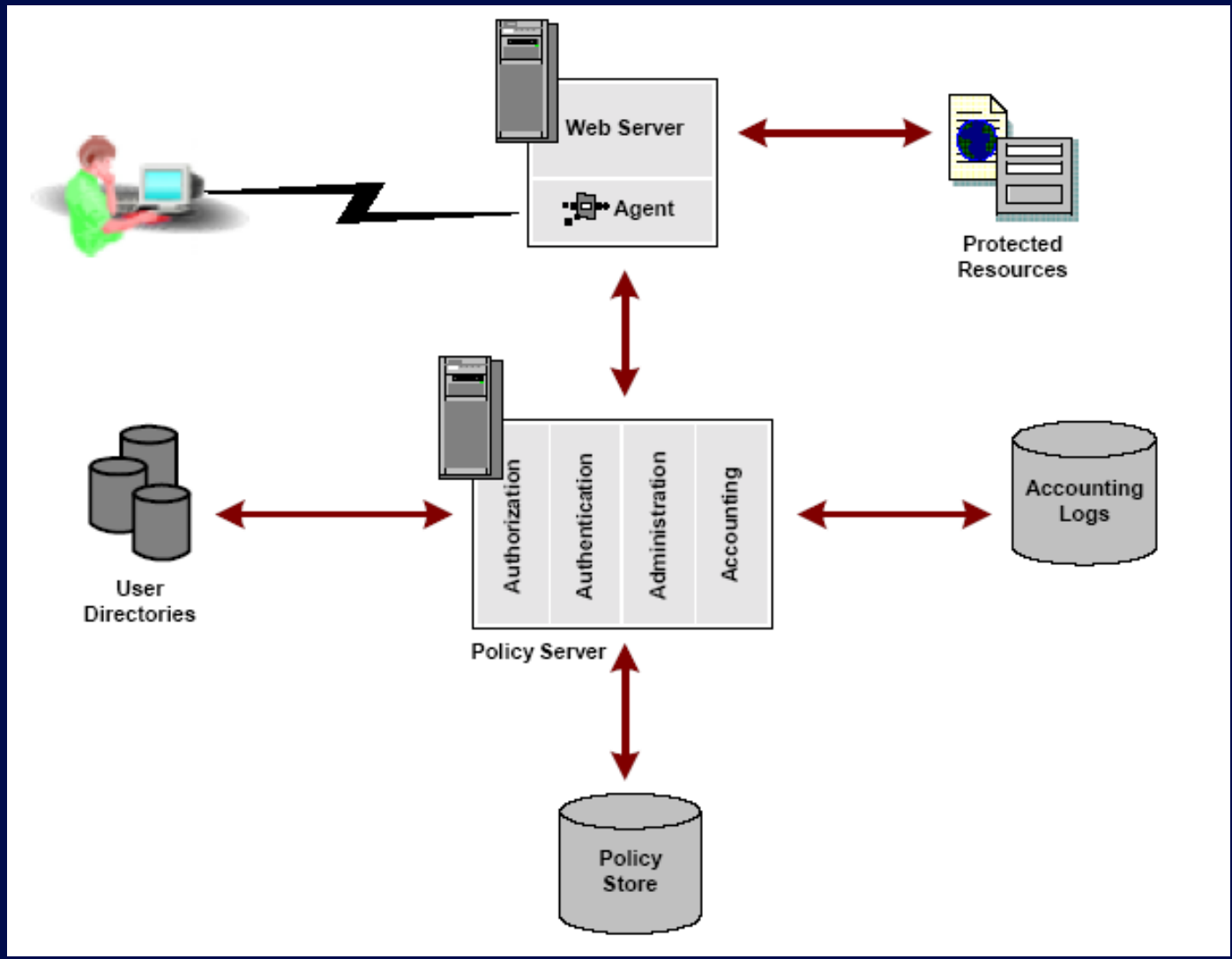
	Domain 1	Domain 2	Domain 3	File 1	File 2	Process 1
Domain 1	*owner control	*owner control	*call	*owner *read *write		
Domain 2			call	*read	write	wakeup
Domain 3			owner control	read	*owner	

Objects

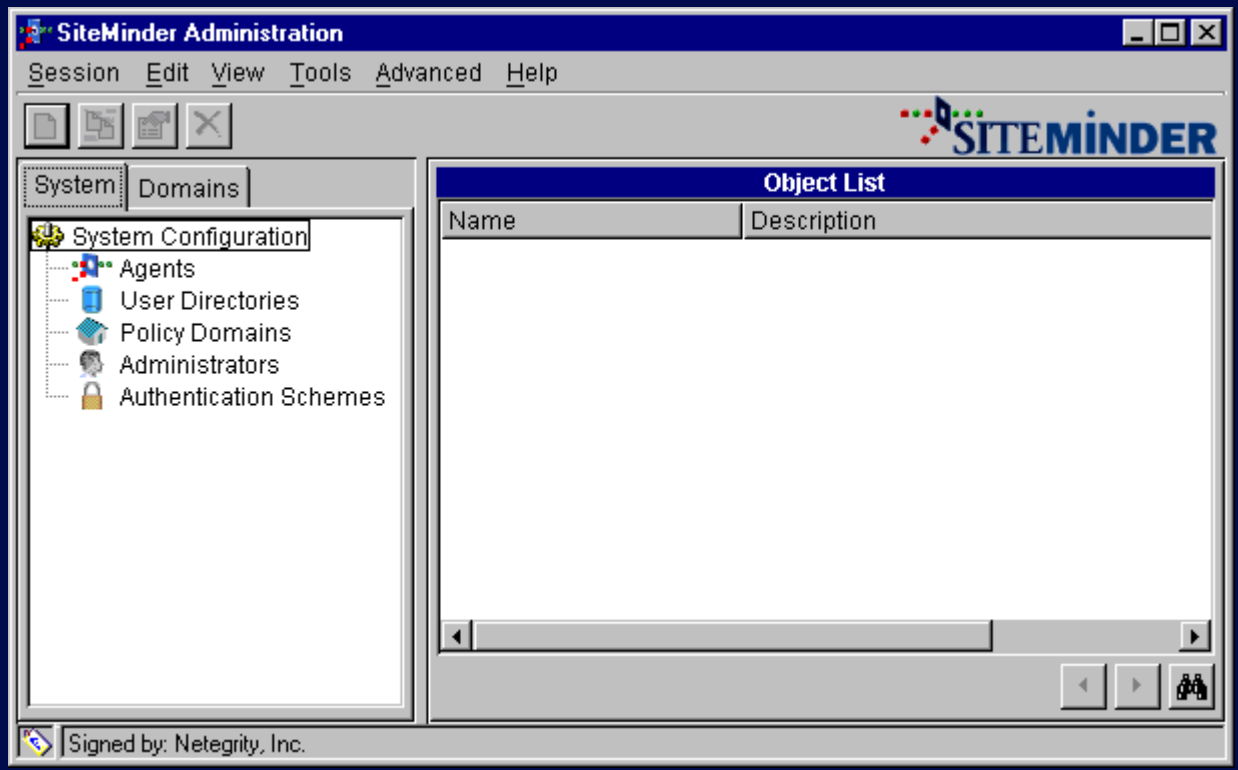


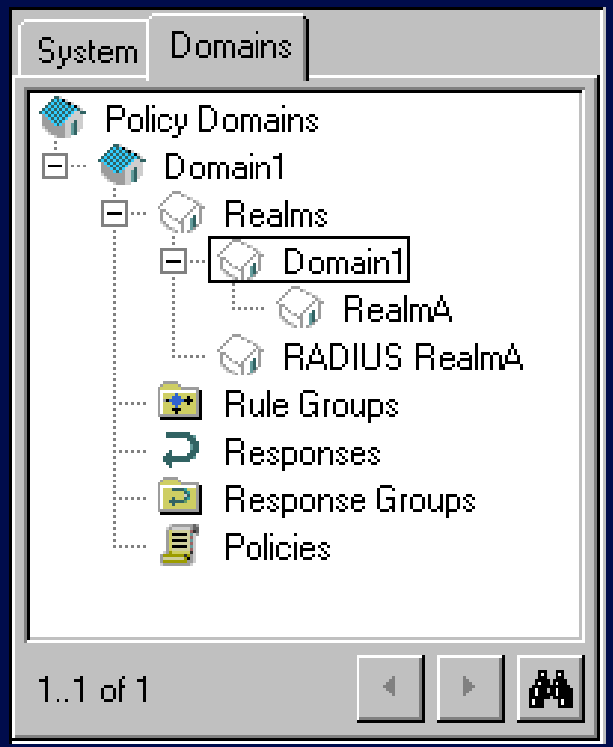
To be protected

Enterprise-scale authentication/authorization server

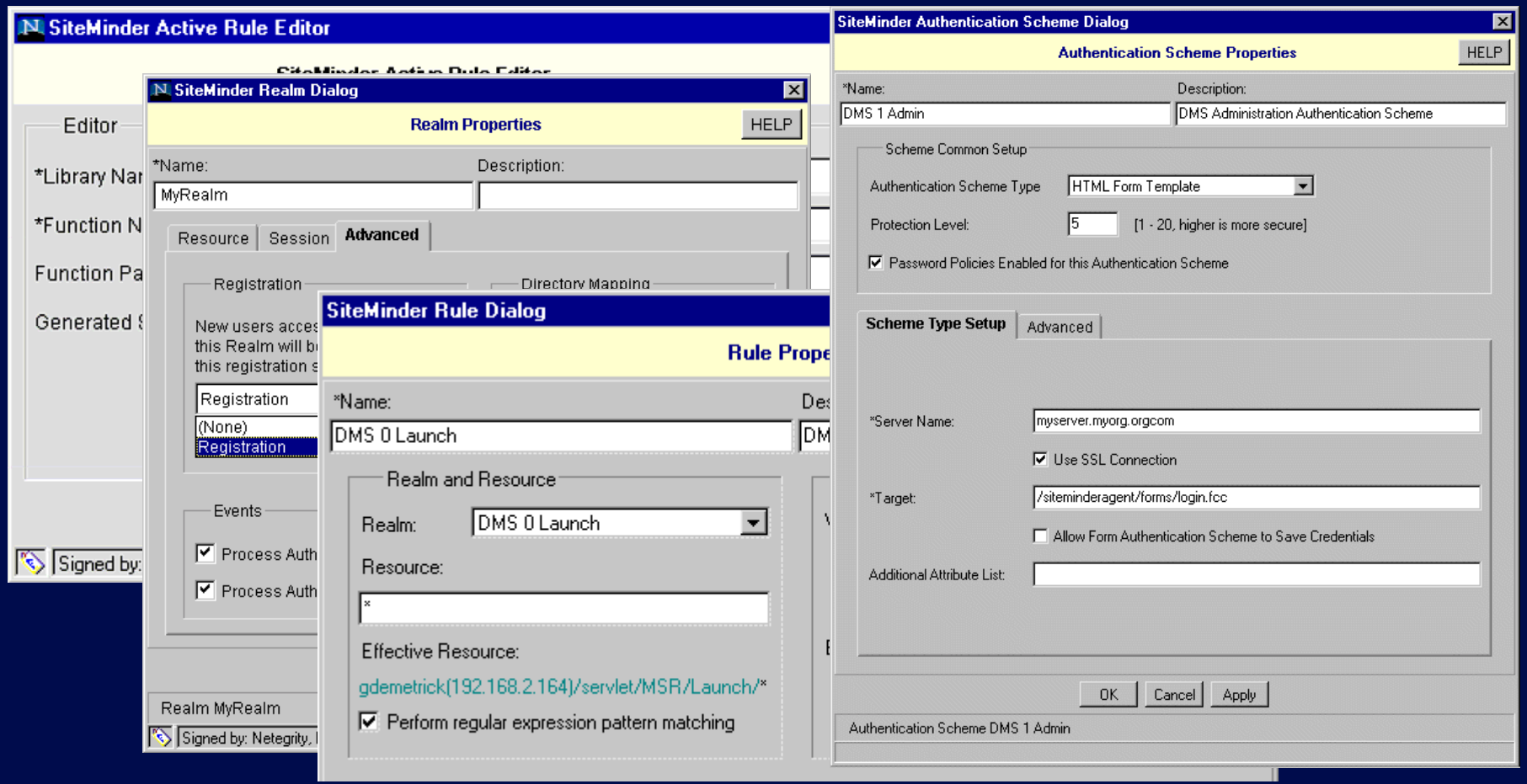


Everything starts with simple directory tree like structure

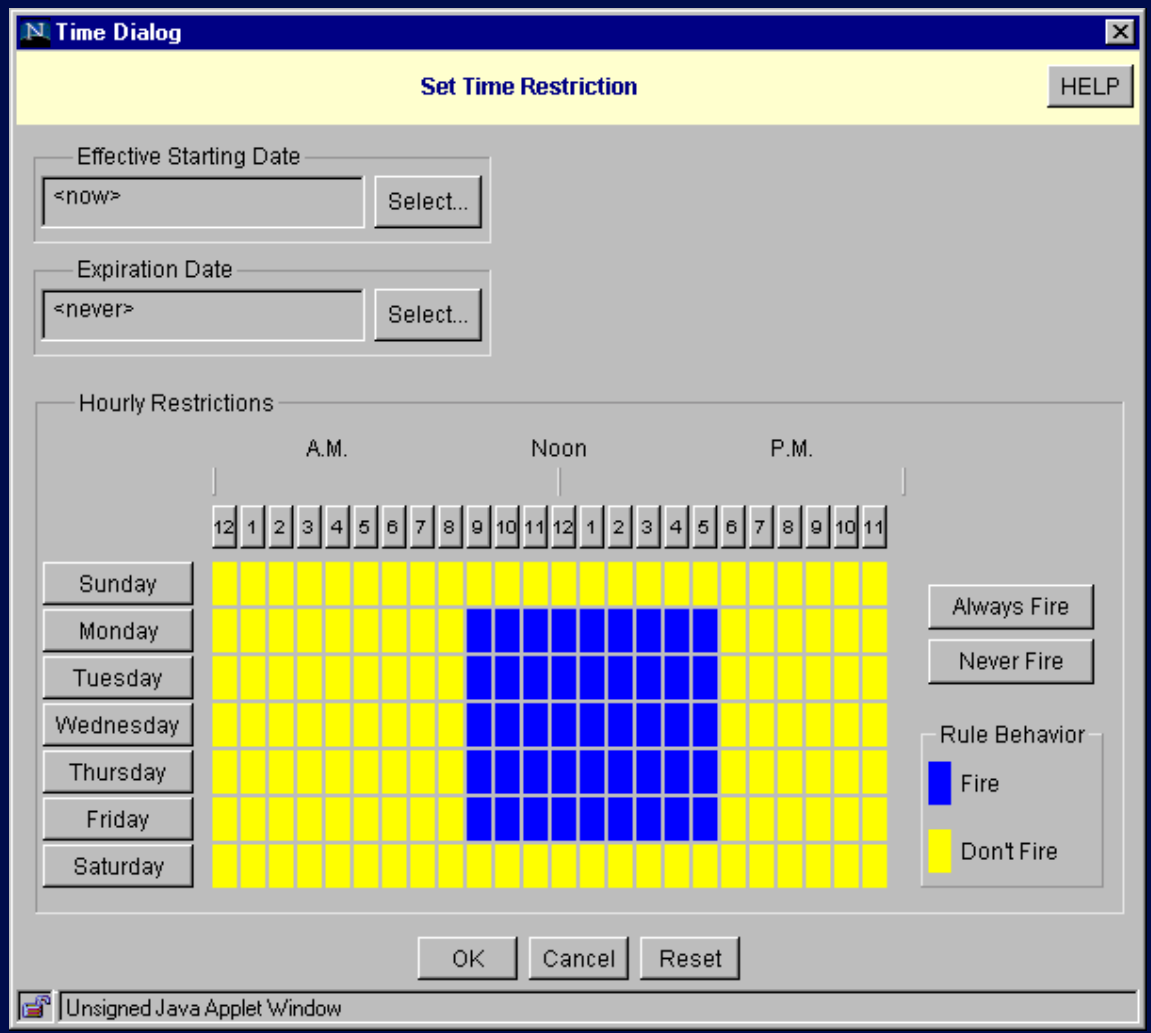




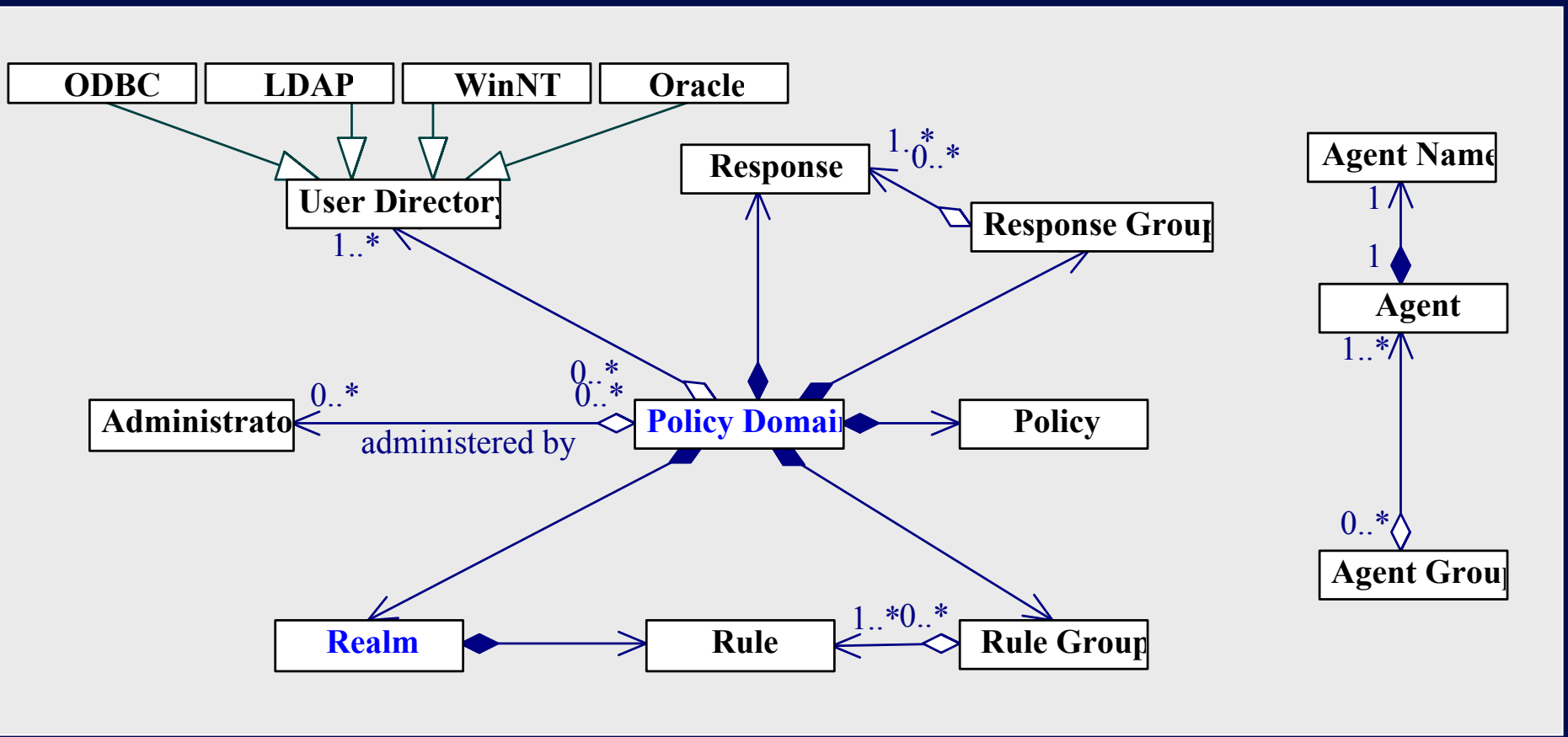
Then continues with simple forms to fill out ...



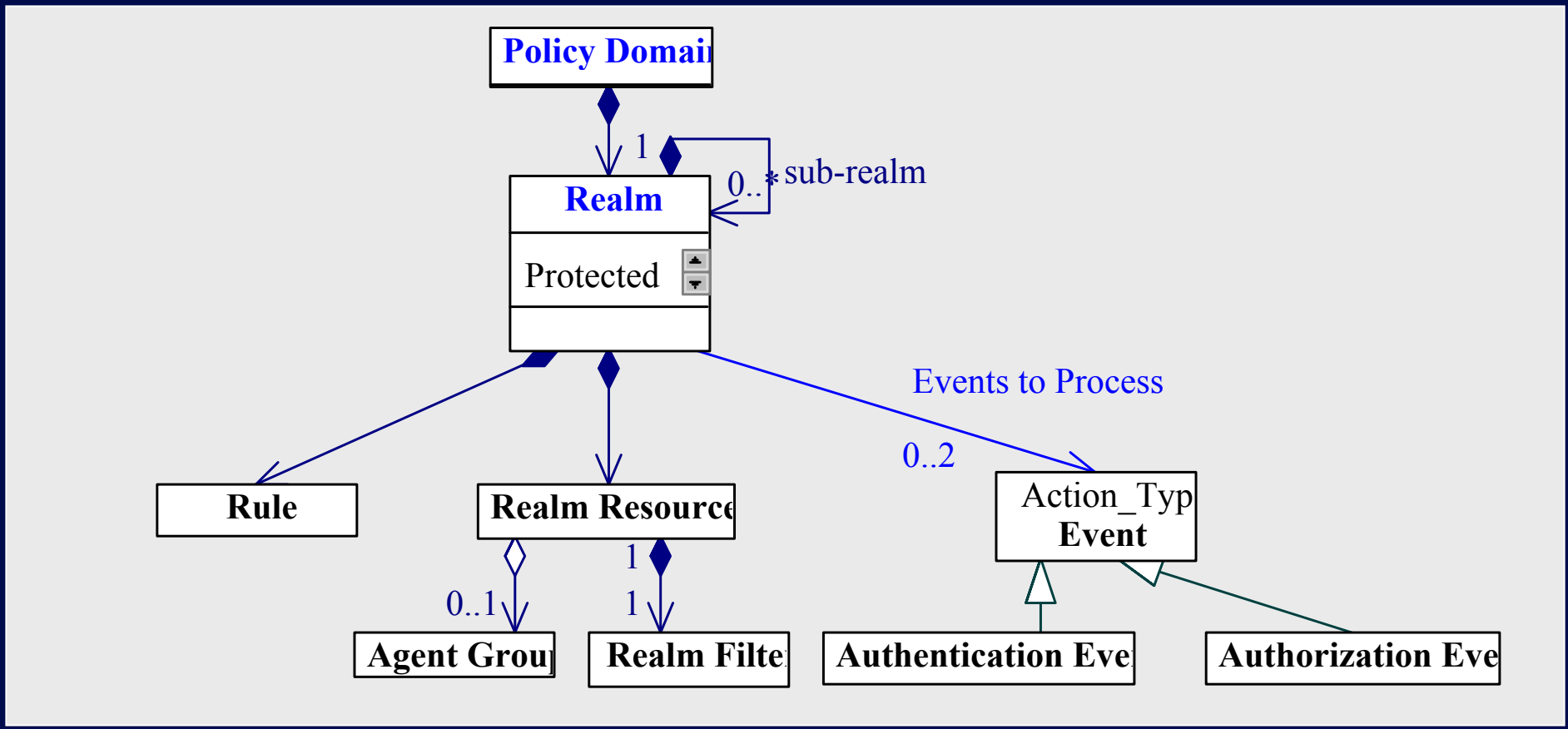
… or select


Time Dialog
Set Time Restriction
HELP
Effective Starting Date
<now>
Select...
Expiration Date
<never>
Select...
Hourly Restrictions
A.M.
Noon
P.M.
12 1 2 3 4 5 6 7 8 9 10 11 12 1 2 3 4 5 6 7 8 9 10 11
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Always Fire
Never Fire
Rule Behavior
Fire
Don't Fire
OK
Cancel
Reset
Unsigned Java Applet Window

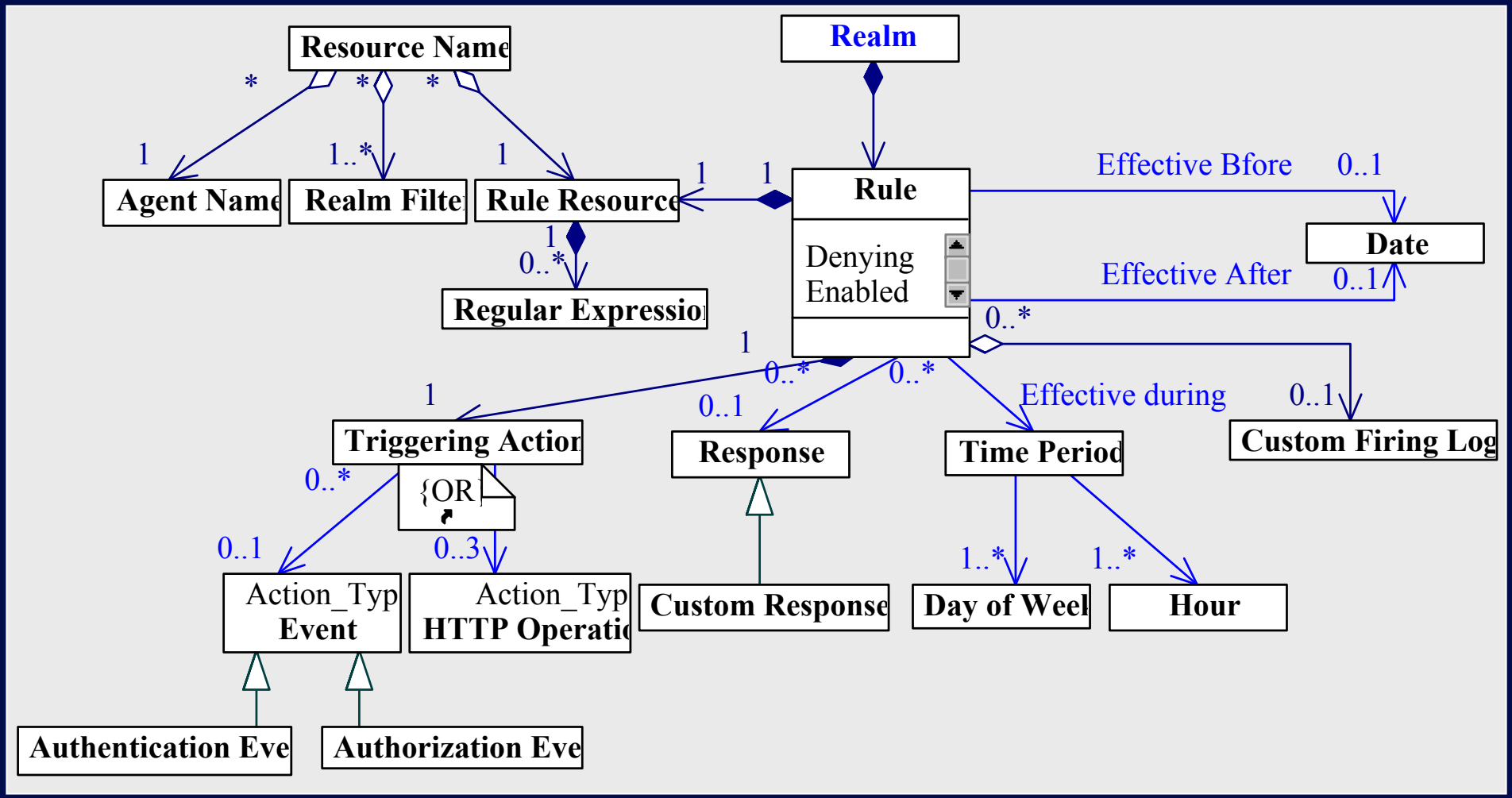
But the mental model is complex ...



… and even more …



... complex


Resource Name
Realm
Agent Name
Realm Filte
Rule Resource
Regular Expressio
Rule
Denying
Enabled
Effective Bfore
0..1
Date
Effective After
0..1
Triggering Actior
{OR}
Response
Time Period
Effective during
0..1
Custom Firing Log
Action_Typ
Event
Action_Typ
HTTP Operati
Custom Response
Day of Weel
Hour
Authentication Eve
Authorization Eve

So what?

- Steep learning curve
- Hard to fit real world into the model
- Easy to make costly mistakes
  - "friendly" DoS
  - inadvertent hard to catch vulnerabilities
- Hard to test
  - Expensive to test required scenarios
  - No "what if" scenarios to test before changing
  - Hard to perform complete testing
- Lacks domain-specific abstractions

State of the Art (Science, Engineering?)

Security Usability Comes in …

Waves:

- First, mid 1970s: **acknowledging**
  - [Saltzer and Schroeder, 1975]
- Second, late 1980s: **evaluating**
  - [Karat, 1989], [Mosteller, W. S. and Ballas, 1989]
- Third, late 1990s -- early 2000s: little bit of everything
  - **acknowledging, raising** [Schultz, et al., 2001]
  - **evaluating** [Jendricke and Markotten, 2000], [Whitten, 1999]
  - **suggesting** how to address [Holmstrom, 1999], [Patrick and Kenny, 2003], [Whitten and Tygar, 2003], [Yee, 2002]
  - **building community** (HCI & Security Systems Wrkshp, 2003)

with 14 year period!

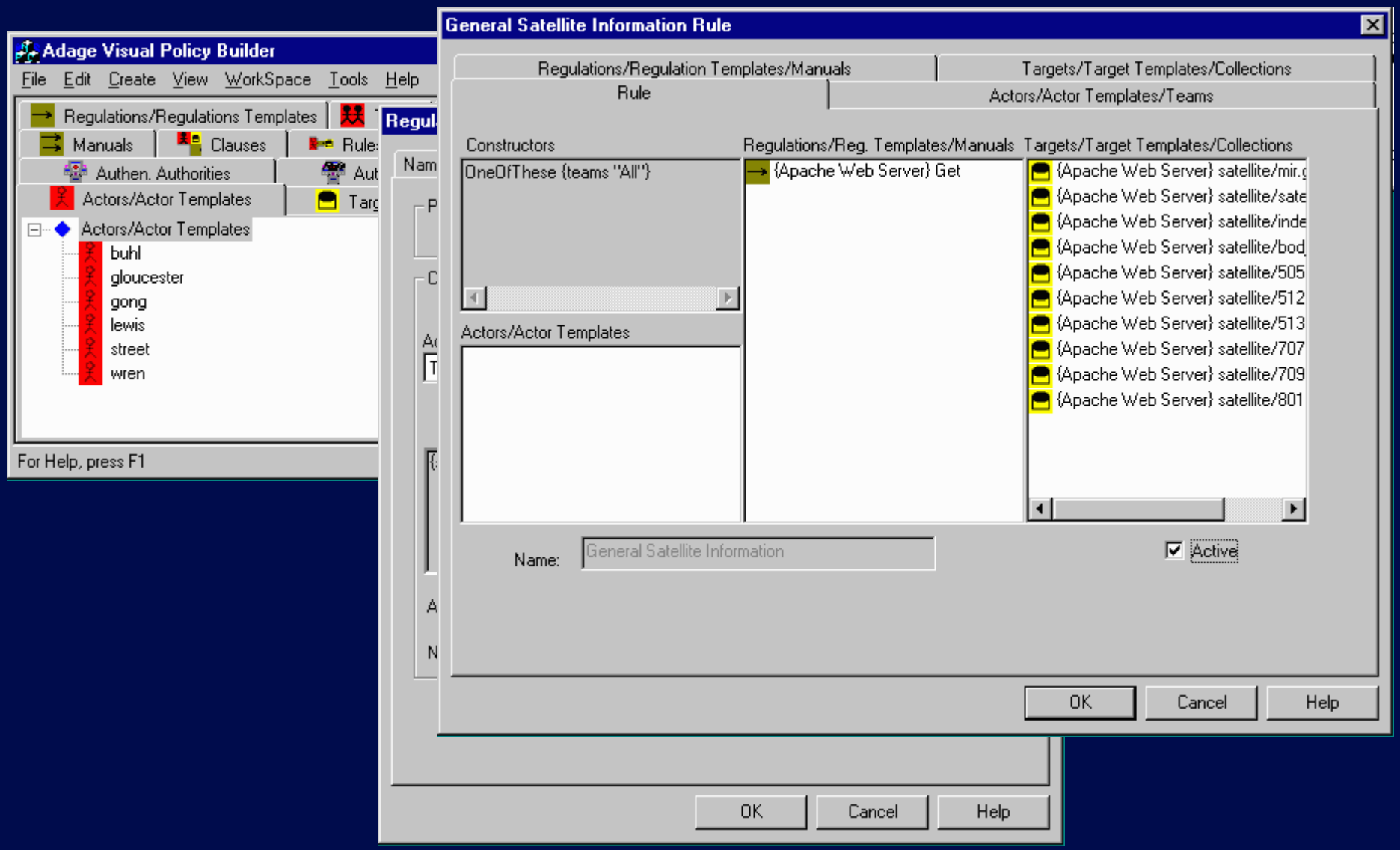
What kind of waves?

What about security administration and HCI?

One published study [Zurko, Simon, Sanfilippo, 1999], [Zurko, Simon, 1996]

- Experimental authorization system **Adage**
  - trial-and-error approach with feedback gained through:
    - contextual interviews,
    - verbal protocols, and
    - affinity mapping techniques for interpreting and categorizing notes on the subject's actions,
    - discount usability testing
    - lab testing
- Admin UI usability studies
- Results
  - admins are willing to learn the UI
  - difficult to
    - display events such as role conflicts
    - provide good feedback of admin actions' effects
    - to comprehend system model | difficult to visually display the global system overview
  - Would be nice to have
    - "test" mode could be very useful
    - ability to query the state of the system

Adage admin GUI Screenshots



Research Directions Ideas

What can be done at UBC?

1. Develop distributed secure application admin UI
   A. Improve visualization of security information presented to administrators
   B. Develop support for opportunistic and incremental task planning
   C. Employ newly suggested principles
      - designing interactions with security mechanisms [Yee, 2002]
      - Safe Staging (creating learnable security software) [Whitten and Tygar, 2003]
2. Compare usability with existing technologies
   – CORBA, EJB, COM+, (ASP).NET