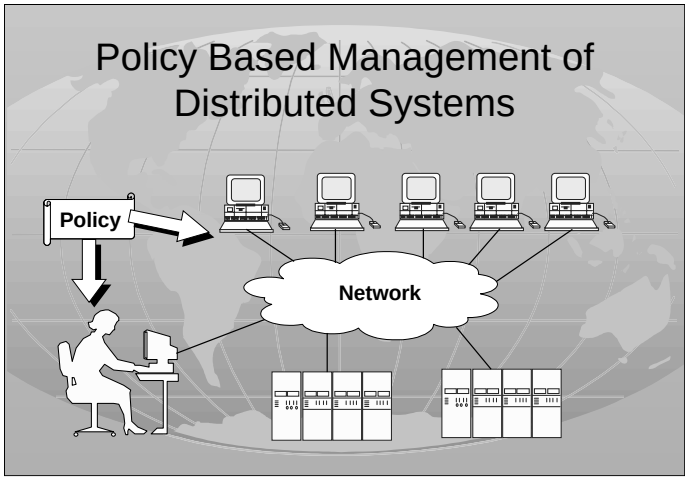
Policy Agents: Licensed to Manage

Morris Sloman
Department of Computing
Imperial College
London

Although the title of my talk is
Policy Agents: Licensed to Manage

An alternative title which reflects what I will be talking about could be

Policy Based Management of Distributed Systems



How to manage distributed computer systems which are (potentially) distributed around the world,
but interconnected by a network.

Management of both hardware and software

Contents

- What is management & why it is need
- Domains as a means of grouping objects
- Management policy which applies to objects
- Detecting conflicts which occur between policies
- Management roles in an organisation
- Object-oriented techniques for reuse of role and policy specifications
- Finally - brief overview of important future directions

Management: What?

- Monitoring
  - Current state
  - Events

Monitoring is essential for all aspects of management

Events are notification messages generated when something of interest happens eg

- Component fails or
- A threshold is reached
  eg discarded messages >5

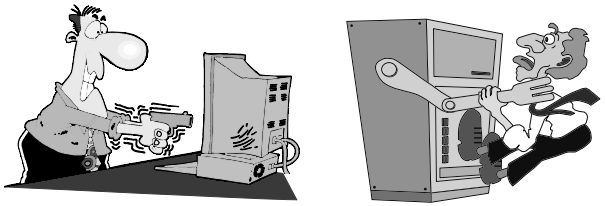
Management: What?

- Managers:
  - interpret policy
  - make decisions

Managers interpret the policy which influences their decisions.

Management: What?

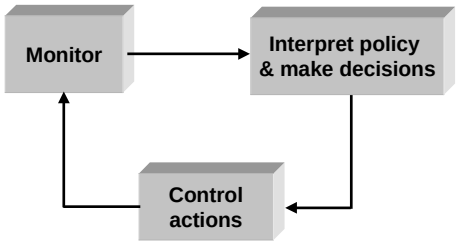
- Perform management actions



Wishful thinking!

Reflects what managers would often like to do!

Management Control Loop

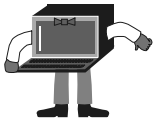


A standard management control loop reflecting the activity of managers

Managers

- People

- Automated agents

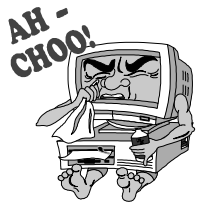


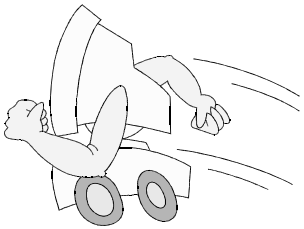
Our management model caters for both people and automated manager agents

I now want to go on to talk about why management is needed

Management : Why?

- Fault handling
- Performance Optimisation





Fault handling includes

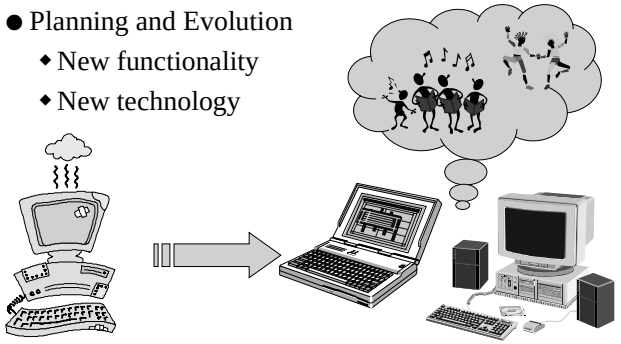
Diagnosis of what has caused the fault

Taking action to rectify it

Performance optimisation - making hardware or software run faster or more efficiently

Management: Why?

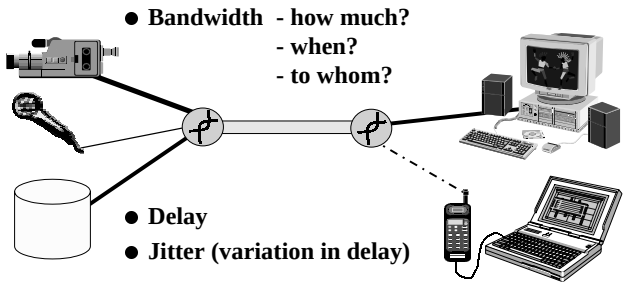
- Planning and Evolution
  - New functionality
  - New technology



New functionality may imply installing some new all singing all dancing software.

Upgrade the system to make use of new technology - mobile systems or multimedia

Quality of Service (QoS) Management



QoS management is about making sure the system meets the requirements of the users, customers

Particularly important for Mulitmedia systems which transfer video and sound

Bandwidth - diameter of an “pipe”

Talking on the telephone with long delays can be very off-putting

Variations of delay between voice and video streams can lead to incorrect lip synchronisation

These are example characteristics of QoS which must be managed

Our Dependence on Distributed Computing

● Distributed computer systems are critical for functioning of many organisations:

Banks **Transport** **Telecommunications**

Banks - all funds are transferred electronically around the world now. Easy to withdraw money from your account in UK when on holiday in USA

Transport - reservations and actual control of planes, trains and traffic lights

Use of mobile phones only became practical with distributed computers systems

Manufacturing for interaction with suppliers

These systems cannot be stopped for management action but must keep working 24 hours a day, 365 days a year

Why is Management of Distributed Systems Difficult?

- Large scale
- Multiple organisations
- Required availability
- Distribution
- Security

Large scale - may be millions of objects eg mobile phones

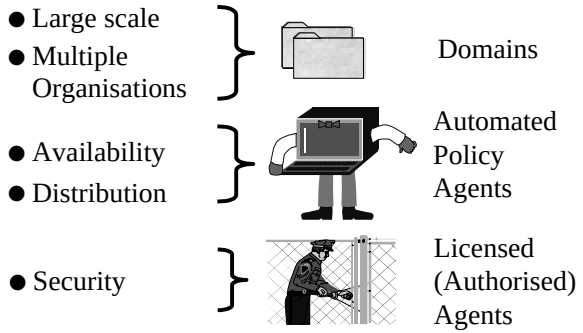
Roam to a different country - connect to a different service provider

Cannot shut down system to reconfigure correct a fault

Physical distribution - sometime in different countries around the world - difficult to get a consistent state or to perform an update of all components at the same time

Management is very powerful so system must be protected from unauthorised users performing management actions

Management Solutions



Domains permit grouping of objects to simplify management or reflect organisational structure

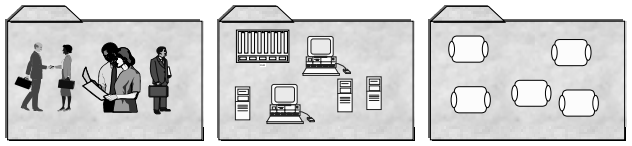
Multiple automated agents can be replicated for availability and distributed in every computer around the system

Security is important because agents and human managers have to be licensed ie authorised to perform particular management actions to protect the system being managed

Contents

Domains ➔ Grouping

A **domain** is a collection of objects which have been explicitly grouped together for management purposes e.g. to apply a common policy



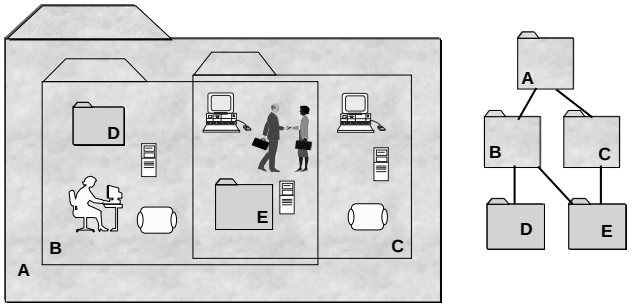
People | **Hardware Components** | **Software Components**

A domain is very similar to a directory or folder on your personal computer but can group more than just files.

Can partition large systems to assign responsibility

Domains ➔ Hierarchy

- **Subdomains & overlapping domains**



Object can be members of more than 1 domain - overlapping.

A subdomain is a domain in a parent domain

Domain hierarchy can reflect:

Organisational structure - companies, departments

Network structure - LANs in a campus, building, floor etc.

Contents

Management Policy

- Policy influences behaviour of objects
- Need to specify and modify policies without coding into automated agents
- Policies are persistent
- But can be dynamically modified

Change policy to change behaviour

Traditional systems - policy is coded into programs of controllers or agents

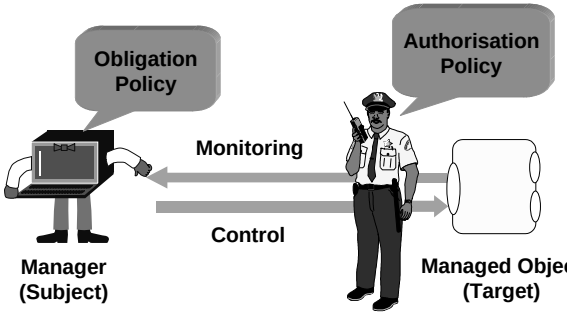
To change policy have to edit, recompile, stop agent and reload new program

Request to backup staff PCs is not a policy but a 1 off command

A persistent policy would say “archiver must back up staff PCs every night at 1 am”.

Can be changed to 3 am

Policy



We are interested in 2 types of policy -obligation and authorisation

Obligation - what managers must do

Authorisation - what managers are permitted to do

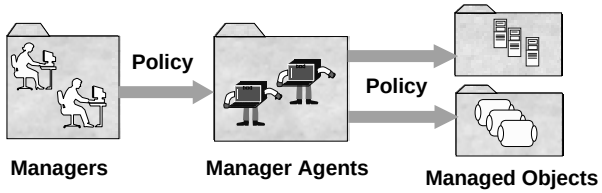
Managers receive monitoring information and perform control actions - called subjects

Managed object are things being managed - called targets

Software objects have clearly defined interfaces which allow actions or operations to be performed on them eg a file has an interface for performing operations read, write, delete, rename etc

Authorisation policy is used by a security agent to protect target objects.

Domains and Policies



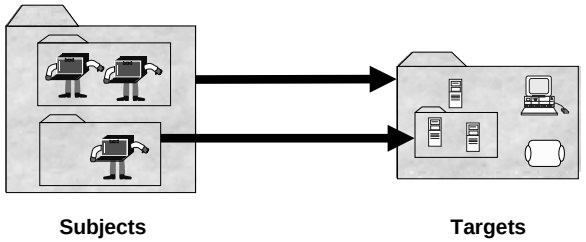
- **Impractical to specify policy for individual objects in large systems with many objects**
- ➔ **specify policy for domains**
- **Can change domain membership without changing policy**

Domains provide scope for policy

Domains give the flexibility to add and remove objects from the domains without changing the policy specification.

Policy defines a relationship between managers (subjects) and the target managed objects.

Policy Propagation



Policy normally propagates to subdomains

So can specify policy for all students and it applies to 1st, 2nd and 3rd year students.

Authorisation Policy

- Defines what a subject is permitted or not permitted (prohibited) to do to a target
  - Permitted operations
- Protect target objects from unauthorised management actions
  - ➔**Target based** interpretation and enforcement
- Not specific to management

Also what monitored information can be received by a manager

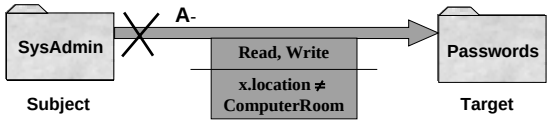
Managers cannot be trusted to interpret authorisation policy

Need authorisation policy to specify how to control access to resources by all users not just managers.

Authorisation Examples

A+ AGroup {videoconf(bw=4, priority=3)}
AGroupNY + DGroupBoston
when (16.00 < time < 18.00)

A- x:SysAdmin {read, write} Passwords
when x.location ≠ ComputerRoom



Textual and graphical representation

specify:

- subject
- target
- actions
- constraints - global eg time
- or related to attributes of the subject or target

Negative Authorisation

- Reflect laws and organisational policies

  A- lecturers {strangle} students
  A- staff {talk} press
- Used for revocation of access rights

  A- Joe Bloggs {any} StudentFiles
  when 1:6:1998 < date < 8:6:1998

Wish to suspend student from using computer system for a limited period as a punishment, for doing something they should not do.

Default Authorisation

- **Default Negative**

  Everything forbidden unless explicitly authorised
  A- x:SysAdmin {read, write} Passwords
  when x.location ≠ ComputerRoom

  ⇩

  A+ x:SysAdmin {read, write} Passwords
  when x.location = ComputerRoom
- **Default Positive**

  Anything permitted unless explicitly forbidden

Can sometimes change -ve authorisation to +ve authorisation by changing the constraint

Default negative is generally recommended - particularly for networked environments

Default positive or permissive can be used in closed trusted environments - home computers

Obligation Policy

- Defines what activities a subject must (or must not) do.
- Assumes well behaved subjects with no freedom of choice.
- **Subject based** ➔ subject interprets policy and performs actions on targets
- Event triggered positive obligation

We do not model freedom of choice although some policy researchers do.

Freedom to over-ride obligation policy

Event - component failure or error rate exceeded a threshold is used to trigger a management action

Obligation Examples

O+ at 01.00 archiver {backup} StaffPcs

O+ on 3*LoginFail(userid)
SecurityAgent{disable(userid),
log(userid), notify (sysadmin,userid)} users

Negative Obligation

- Needed as a restraint on actions which is **subject** rather than **target** based

  O- tutors {TellResults} students
  when date < FinalExamMeeting

  O- x:schedulers {assign} WorkPool
  when x.state = standby

Used where targets do not wish to be protected from subject actions

Staff are permitted to talk to students - impractical to implement as a normal authorisation policy

Schedulers must restrain action based on local state - ie in standby mode

They are authorised to assign jobs to computers in WorkPool

Finance Director Policy Refinement

O+ FD {/* maintain £50K < balance < £100K */ } accounts

O+ on overspend (id) FD {freeze (id)} accounts/personal

O+ on XtraVote (x) FD {allocate (0.2*x)} accounts/equipment

A+ FD {freeze, unfreeze, allocate, deallocate} accounts

I am financial director in the Department but I am going on sabbatical so Bob with his well known interest in intelligent agents would like to replace me with one.

Need considerable intelligence to interpret high level state based policies

Obviously I fit the bill but does a policy agent?

No not intelligent enough

Need to refine high level (abstract policies) into implementable event triggered ones

Policy Notation

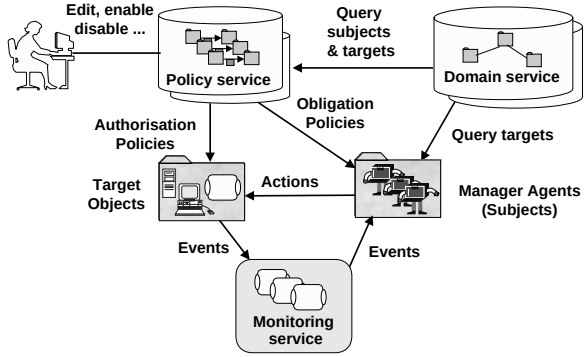
- Precise specification of subjects, targets, actions and constraints for authorisations and obligations
- Needed for both:

Automated agents

Human managers

- Clear specification of responsibility, rights and duties ➔ "job description"

Policy Implementation



1 Administrator edits or creates policies in the policy service

2. Policy service uses target domain to query domain service to determine to which target objects authorisation policies should be sent.

3.Policy service uses subject domain to determine to which agents, obligation policies should be sent.

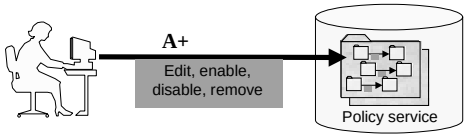
4. Managed objects emit events which are disseminated by the monitoring service to agents, where they trigger obligation policies

5. Manager agents determine target object by querying domain service

6. Agents invoke operations on domain service

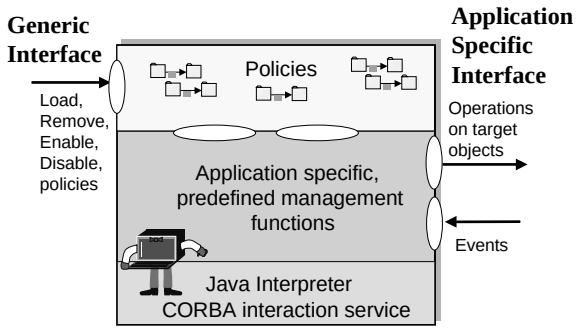
Protecting Policies

- Policy objects can be included in domains and can be protected by authorisation policies



Can thus specify which managers are authorised to access and modify policies

Obligation Policy Agent

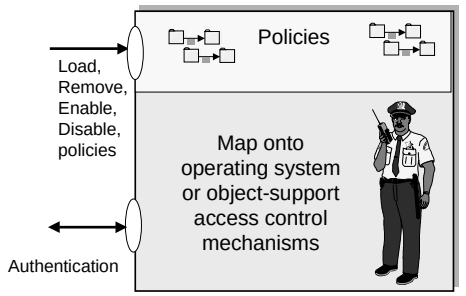


A policy agent is specialised for a particular management application:

- Security agent
- Configuration agent
- Accounting agent

It includes compiled or pre-loaded code specific to the management application and then policies can be loaded to tailor behaviour

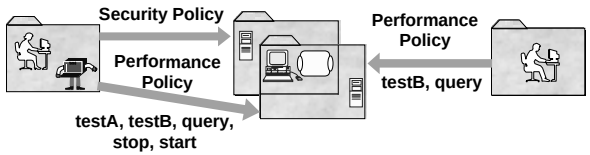
Authorisation Agent



Authorisation agent intercepts operations requests and checks whether they should be permitted or rejected or can use OS to perform checking.

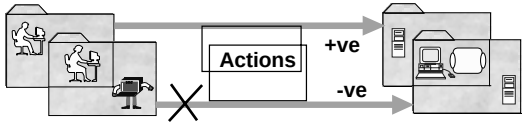
Contents

Multiple Policies May Apply

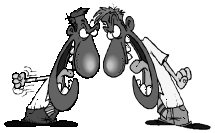


- An object can be a member of multiple domains (overlap)
- Multiple policies can apply to single domain
- Need conflict detection and resolution

Modality Conflicts



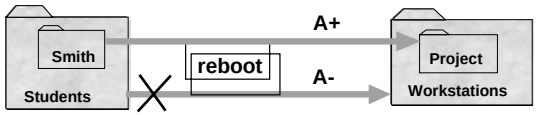
- Potential conflict from overlap of subjects, targets and actions
- 3 types: O+/O-, A+/A-, O+/A-
- Detected by syntactic analysis



Do not need to understand policies to detect potential conflict - just detect overlap of subject, target and actions.

O- / A+ is not a conflict

Example Conflicts

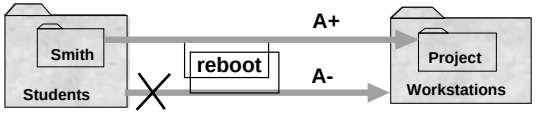


- A- Students {reboot} workstations
- Exception:
  A+ Smith {reboot} workstations/project
- O+ at 01:00 archiver {backup} StaffPcs
  No authorisation policy to permit actions

If there is a default negative authorisation and security administrator has forgotten to specify the required authorisation

Precedence

- Can resolve some conflicts automatically by specifying precedence, e.g.:
  - **Negative** policies override
    Does not permit positive exceptions to negative policies.
  - More **specific** policies override



Negative policy precedence is needed for revocation of access rights.

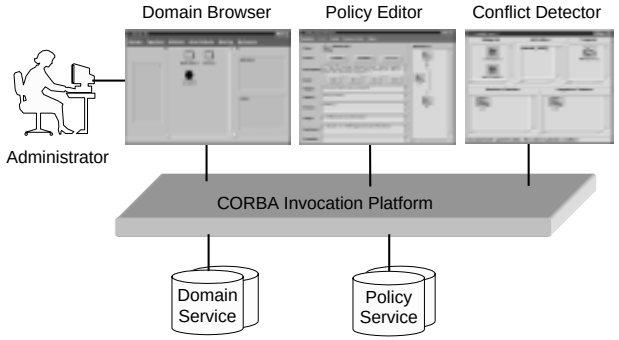
More specific policies taking precedence is possibly more intuitive in many situations

Currently only implement more specific precedence but need a to be able to specify a variety of precedence relationships

Meta-Policies

- Policies about policies
- Specify application specific conflicts such as conflicts of duties or interests
  - E.g. same person is not permitted to approve payment and sign payment cheque
- Currently specified as Prolog predicate on permitted policies in a domain

Tool Support



Domain browser supports navigation of the domain structure

Policy editor for creating and disseminating policies

Conflict detector for analysing policies

Policies are objects which can be included in a a domain - can use authorisation policies to control access to them.

Contents

User Representation Domain

- Persistent representation of a registered user
- URD is subject of policies applying to a specific person
- At login adapter object created to represent and act on behalf of person in system
  ➔ command interpreter

Need to represent users in the system

Commands could be to select objects, double clicking to launch a program etc. or selecting management operations from a menu in a window.

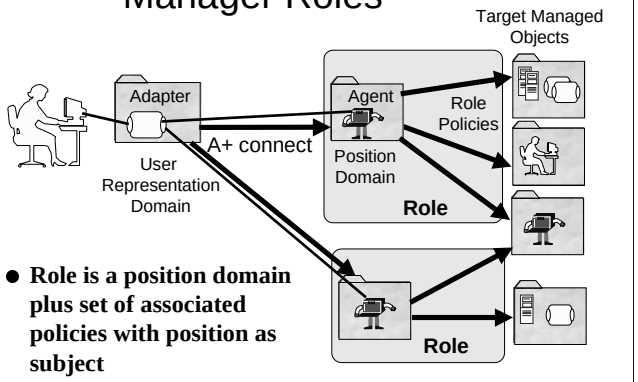
For management, roles are more important than specific users

Roles

- Role is the duties and rights related to a **position** in an organisation
- Eg finance director, personnel manager, ward nurse, surgical nurse
- Specify policy in terms of **roles** rather than **persons**

➔ do not have to respecify policies when person assigned to new role

Manager Roles



- **Role is a position domain plus set of associated policies with position as subject**

Assign user to role by means of an authorisation policy permitting connections from URD to agent in position domain

User logs on, create adapter - policy propagates to adapter

Set up connection from adapter to agent

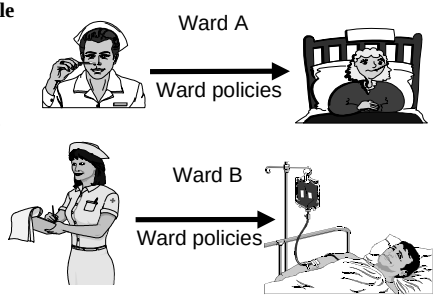
Agent performs action on behalf of user but using role policies.

User may be assigned to additional role eg Finance director and project manager.

Can have separate window for each role with menus reflecting what each role is permitted to do.

Role Instances

- **Multiple nurse role instances**
- **Different persons assigned to roles**
- **Different patients**
- **Similar policies**
- ➔ **Role Class**
- **Reuse of role specification**



Role class is rather like a mould used to create many instances of the particular role.

Each instance has different position domain and applies to different target objects

Need policy templates

Policy Template

- Policy specification with **placeholder** for subject or target or both
- Actual subjects or targets are defined when a policy instance is created from template

O+ on x.HighTemp D {administer (analgesic)} x:t

O+ on x.HighTemp NurseA{administer (analgesic)}x:WardA

O+ on x.HighTemp NurseB {administer (analgesic)}x:WardB

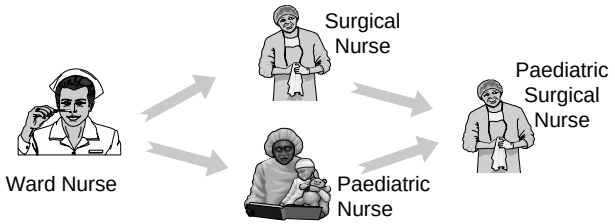
Policy template is also like a mould but for policies

Different subjects, targets or both but the same actions and constraints

Can be used to create policies that are not part of roles.

Role Specialisation

- **Derive new role specifications from existing ones**
- **Add or refine policies**
- ➔ **Inheritance**



Inheritance can be used to derive a specification of Surgical Nurse from Ward Nurse - similarly Paediatric Nurse and both of these can be used to derive a Paediatric Surgical Nurse role.

Before getting to the conclusion of the talk, I would like to show you some example Special Agent Roles brought to you by our own Hollywood producer Keng Ng

Special Agent Roles

Jeff Kramer, Jeff Magee and I have been working together in the Distributed Software Engineering (DSE) section for about 20 years.

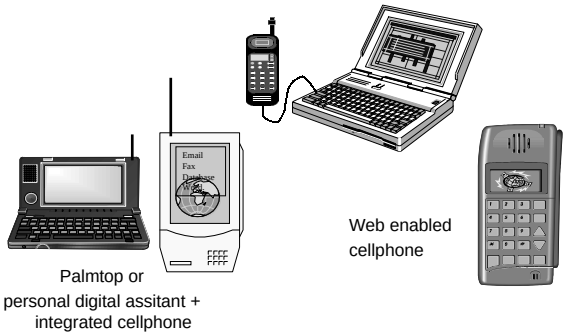
They are quick to shoot me down when the role of professor goes to my head!

Contents

Management: What and Why?
Domains
Management Policy
Policy Conflicts
Management Roles & Object-Orientation
Future Directions

I would like to talk about some of the issues relating to the management of future networks and distributed systems

Mobile Users



Mobility will make management particularly of QoS more difficult.

What adaptation policies are needed?

Policies needed for supporting ‘visiting’ users in ‘foreign’ environments

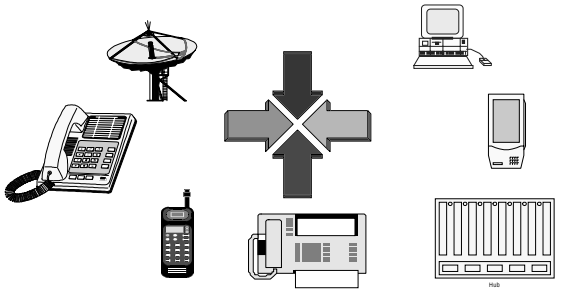
- What services or resources they can use when visiting a hotel or another organisation.
- Eg visitor role

Filtering policies - what information flows to mobile users eg remove attachments, convert a fax to text.

Mobility Issues

- **Makes management, particularly of QoS, more difficult – what adaptation policies needed?**
- **Policies needed for supporting 'visiting' users in 'foreign' environments**
  - **What services or resources they can use?**
  - **➔ Visitor Role**
- **Filtering policies for information flows to mobile users – eg convert fax to text, remove attachments**

Convergence of Telecommunications & Computing



There has been a realisation that Telecomms is a distributed system, and that management can use standard distributed processing techniques.

I have been pushing this for at least10 years

Telecoms providers can make use of cheaper hardware and software from computing industry.

Convergence: Why?

- Open telecommunications
- No longer only 2 players – users and service providers
- Multiple value-added service providers
- Rapid deployment of new value-added services
- Cheaper infrastructure, hardware and software technology

Legal requirement to open up the Telecomms market in USA

Multiple service providers

- Basic communications eg BT
- Video transmission from another provider
- Video conferencing from 3rd provider

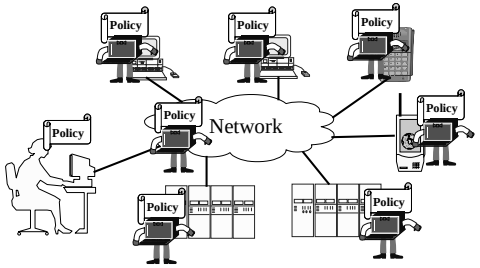
New service in timescales of months or days rather than years - must be provided by software

Programmable Telecommunications

- Adaptable network and communication components
- Dynamically change behaviour to cater for new services
- Management programmable?
- Customer programmable?
- Message programmable - active networks?
- Reliability and security issues.

Network & Distributed Systems Management

- Policy agents: licensed to manage



Policy agents, licensed to manage, are a key concept for management of future systems

- within components of the network
  - switches, routers;
- in the systems connected to the network and
- for human managers.

Thank you for your attention

I hope this talk has left you
"stirred, not shaken!"

Additional Information

Information on papers, projects etc. available from

http://www-dse.doc.ic.ac.uk/~mss