Services, Features and Policies

Stephan Reiff-Marganiec
University of Leicester

evolution: features

- Market requires new functionality
- New functionality is provided by features
- Features are add-ons to basic service
  - call waiting, conference calling
  - (idea exists in office software, cars, ...)
- They can be deployed
  - in the network (IN, POTS)
  - on the end-device (IPTel, MobileComms)
- Development hindered by FI problem and lack of good SE frameworks **(more later ...)**

(re)configuration: policies

- Features
  - allow only minimal end-user configurability
  - do not consider user’s context
  - do not support high-level goals
- Policies
  - can provide all of the above
  - require appropriate languages, supporting architectures and development processes
  - can lead to policy conflict

(again more later)

[Reiff-Marganiec, Turner: FIW 2003; Reiff-Marganiec, Turner: FORTE 2002; Reiff-Marganiec: Objects, Agents, Features (LNCS) 2004]

evolution on steroids: services

- Today
  - Web Services
  - Grid services
  - Service Oriented Architectures

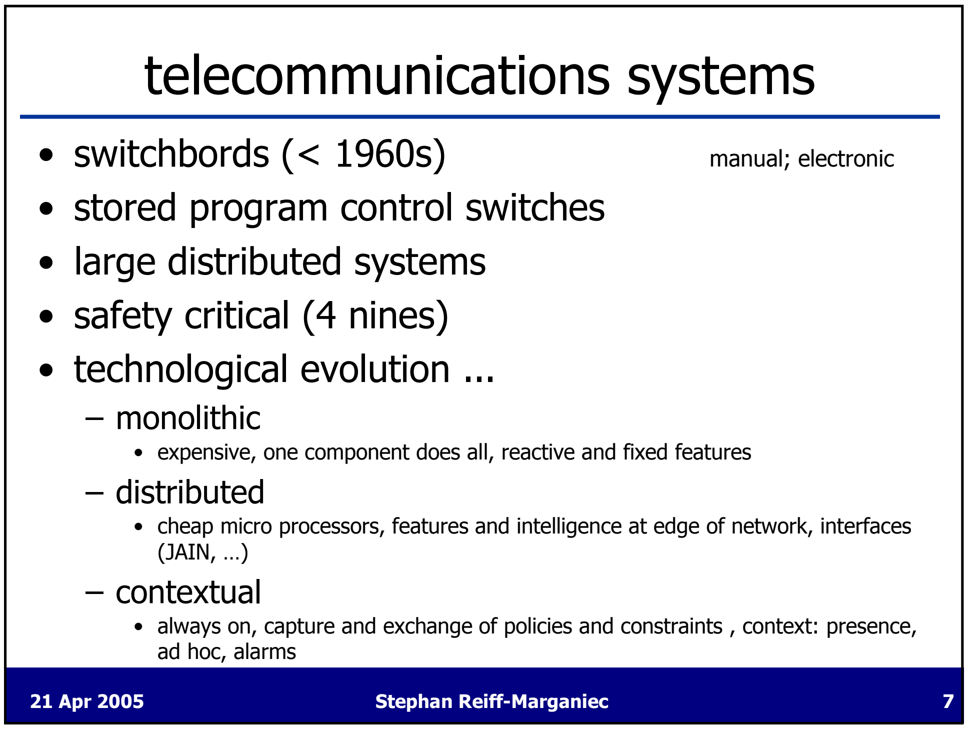
- Can provide new functionality
- No basic service needed!
- Systems can be assembled/ configured at runtime

we will look only at basics of SoA, L. Baresi will do more …

outline

- Features and Services
- Policies
- Policies for end-user services in telecommunications
- End-user services in service oriented architectures

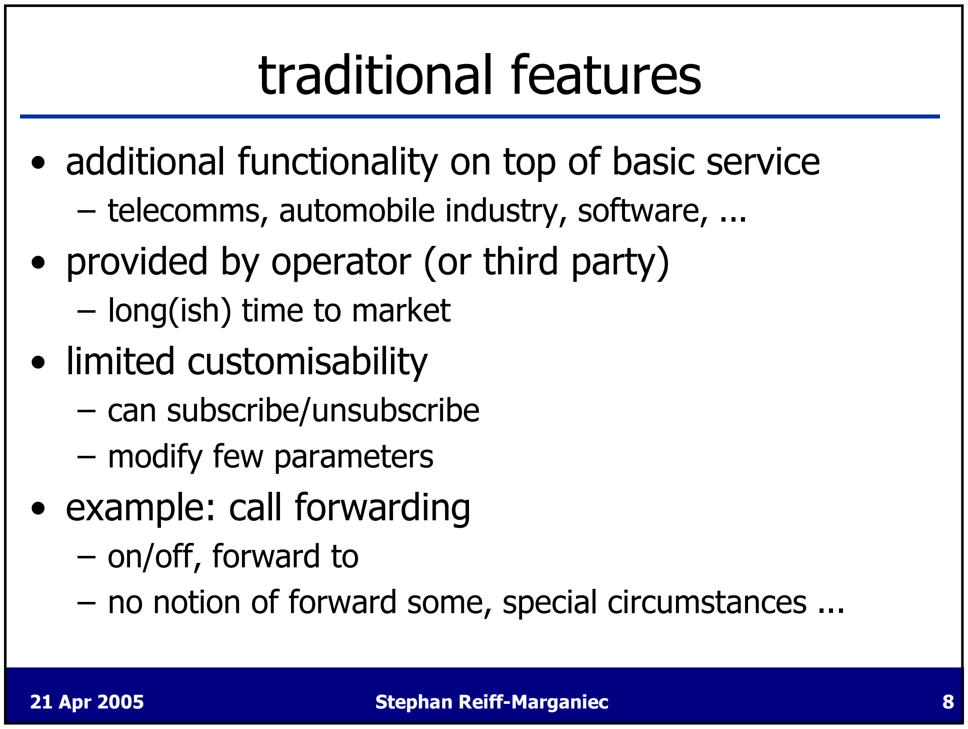
Features and Services



telecommunications systems

- switchbords (< 1960s) manual; electronic
- stored program control switches
- large distributed systems
- safety critical (4 nines)
- technological evolution ...
  - monolithic
    - expensive, one component does all, reactive and fixed features
  - distributed
    - cheap micro processors, features and intelligence at edge of network, interfaces (JAIN, …)
  - contextual
    - always on, capture and exchange of policies and constraints , context: presence, ad hoc, alarms

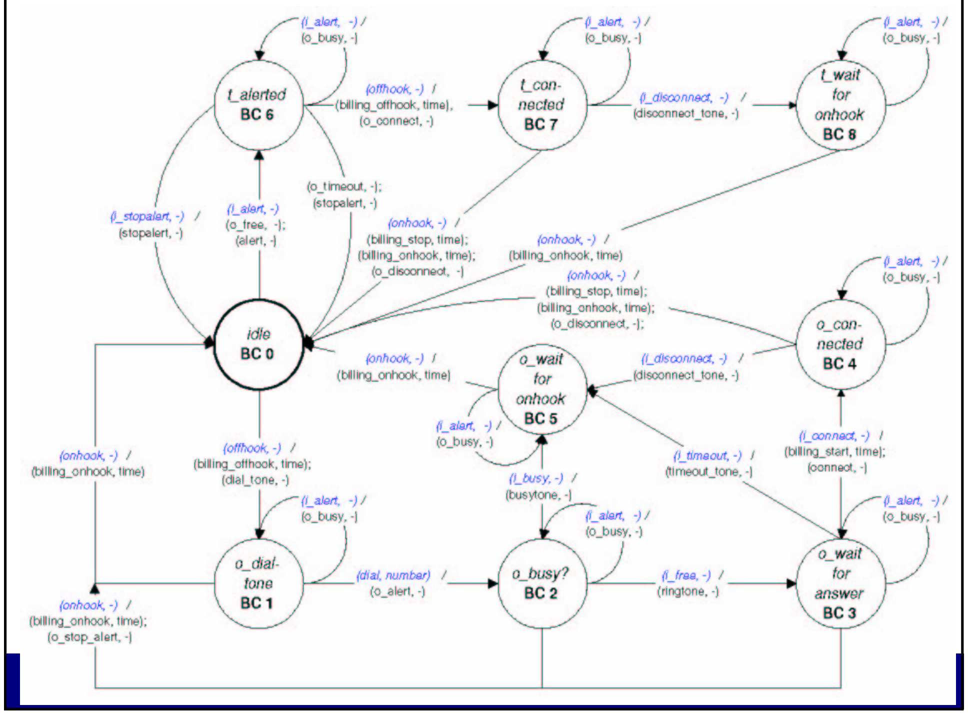
21 Apr 2005 Stephan Reiff-Marganiec 7



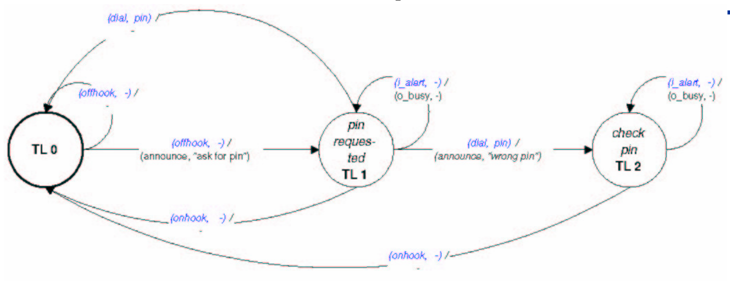
traditional features

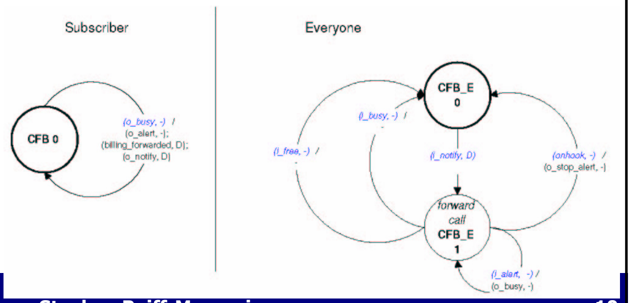
- additional functionality on top of basic service
  - telecomms, automobile industry, software, ...
- provided by operator (or third party)
  - long(ish) time to market
- limited customisability
  - can subscribe/unsubscribe
  - modify few parameters
- example: call forwarding
  - on/off, forward to
  - no notion of forward some, special circumstances ...

21 Apr 2005 Stephan Reiff-Marganiec 8


t_alerted BC 6
t_connected BC 7
t_wait for onhook BC 8
idle BC 0
o_connected BC 4
o_wait for onhook BC 5
o_dial-tone BC 1
o_busy? BC 2
o_wait for answer BC 3
(i_alert, -) / (o_busy, -)
(offhook, -) / (billing_offhook, time), (o_connect, -)
(i_disconnect, -) / (disconnect_tone, -)
(o_timeout, -); (stopalert, -)
(i_stopalert, -) / (stopalert, -)
(i_alert, -) (o_free, -); (alert, -)
(onhook, -) / (billing_stop, time); (billing_onhook, time); (o_disconnect, -)
(onhook, -) / (billing_onhook, time)
(onhook, -) / (billing_stop, time); (billing_onhook, time); (o_disconnect, -);
(onhook, -) / (billing_onhook, time)
(i_disconnect, -) / (disconnect_tone, -)
(onhook, -) / (billing_onhook, time)
(offhook, -) / (billing_offhook, time); (dial_tone, -)
(i_busy, -) / (busytone, -)
(i_timeout, -) / (timeout_tone, -)
(i_connect, -) / (billing_start, time); (connect, -)
(onhook, -) / (billing_onhook, time); (o_stop_alert, -)
(dial, number) / (o_alert, -)
(i_free, -) / (ringtone, -)

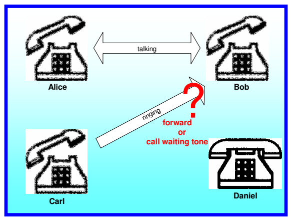
example features


TL 0
pin requested TL 1
check pin TL 2
(dial, pin) /
(offhook, -) /
(offhook, -) / (announce, "ask for pin")
(i_alert, -) / (o_busy, -)
(dial, pin) / (announce, "wrong pin")
(onhook, -) /
(onhook, -) /


Subscriber
Everyone
CFB 0
(o_busy, -) / (o_alert, -); (billing_forwarded, D); (o_notify, D)
CFB_E 0
forward call CFB_E 1
(i_busy, -) /
(i_free, -) /
(i_notify, D)
(onhook, -) / (o_stop_alert, -)
(i_alert, -) / (o_busy, -)

the feature interaction problem ...

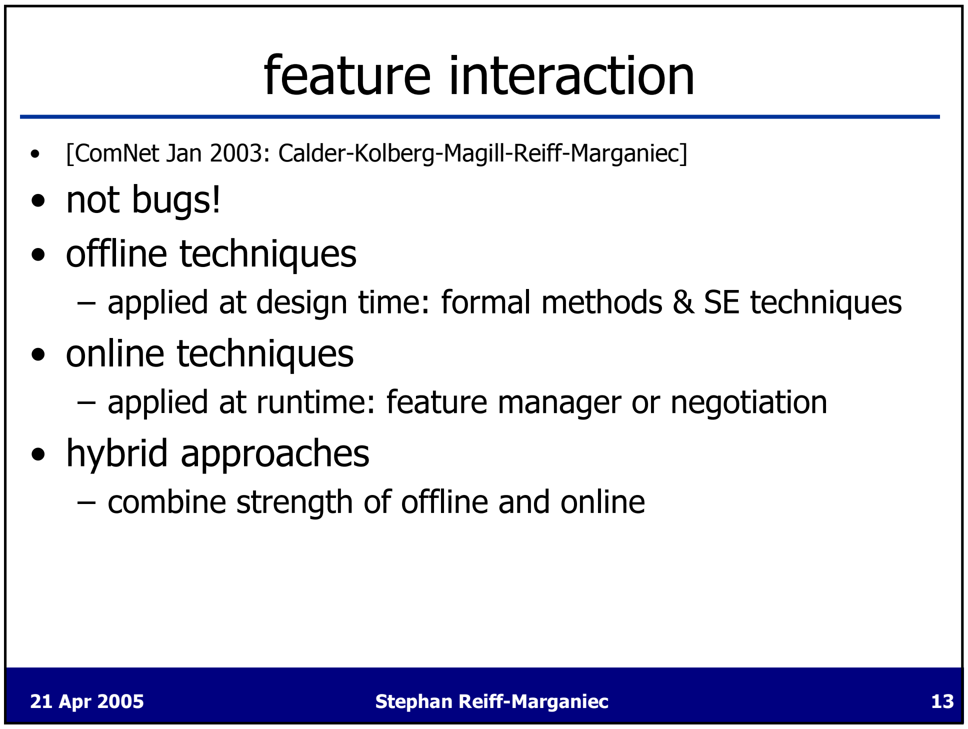
- *features are added functionality for a basic service that are independently (of each other) working correctly*
- *e.g.*
  - *Call Forwarding, 1471, Call Waiting*
  - *equation editor, graphics plugin*
  - *car alarm, accident escape system*
  - *burglar alarm, climate control*



- *however, two or more features together might not work correctly ⇨ the feature interaction problem*

... and worse

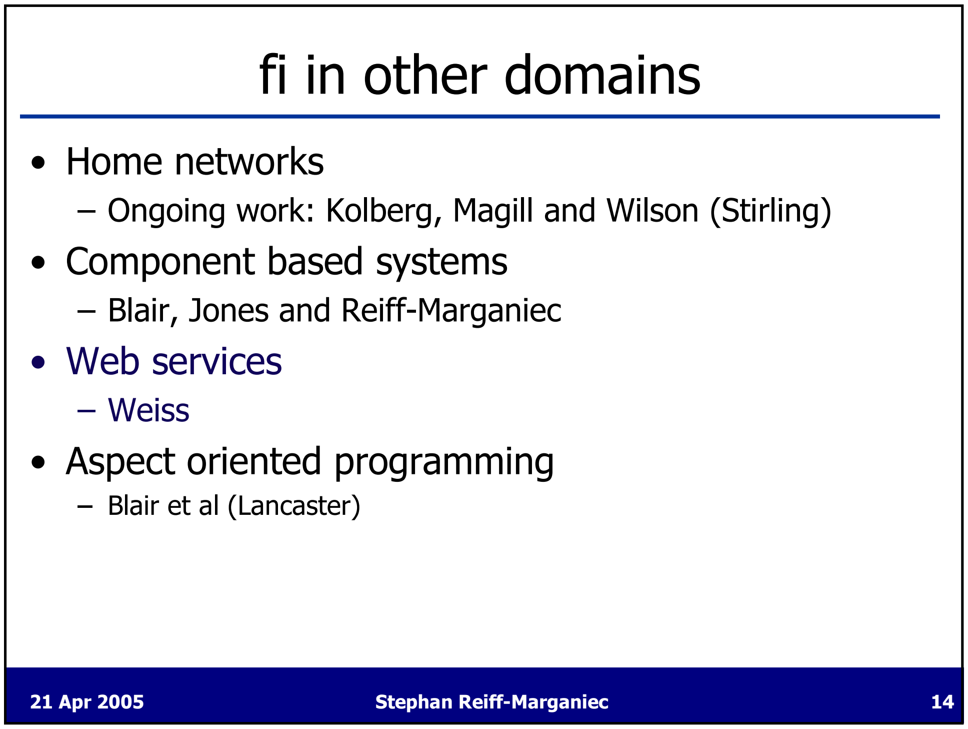
- legacy telecomms systems
  - large distributed systems
  - evolving over time
  - fragile code
  - no reliable documentation
- deregulated market
  - no design time information about third party features
  - short development periods
  - features presence might only be recognized at runtime



feature interaction

- [ComNet Jan 2003: Calder-Kolberg-Magill-Reiff-Marganiec]
- not bugs!
- offline techniques
  - applied at design time: formal methods & SE techniques
- online techniques
  - applied at runtime: feature manager or negotiation
- hybrid approaches
  - combine strength of offline and online

21 Apr 2005 Stephan Reiff-Marganiec 13



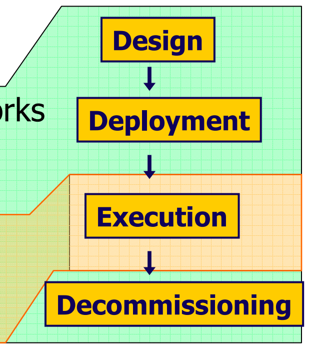
fi in other domains

- Home networks
  - Ongoing work: Kolberg, Magill and Wilson (Stirling)
- Component based systems
  - Blair, Jones and Reiff-Marganiec
- Web services
  - Weiss
- Aspect oriented programming
  - Blair et al (Lancaster)

21 Apr 2005 Stephan Reiff-Marganiec 14

handling fi

- Feature Interaction must
  - be detected
  - be resolved
- requires
  - software engineering frameworks
    - that allow automatic detection,
    - and suggest concrete solutions
  - runtime environments
    - that allow automatic detection,
    - and automatic resolution


$$\begin{tikzcd}\text{Design} \arrow[d] \\ \text{Deployment} \arrow[d] \\ \text{Execution} \arrow[d] \\ \text{Decommissioning}\end{tikzcd}$$

service

- “Unit of work done by a service provider to achieve desired result for consumer”

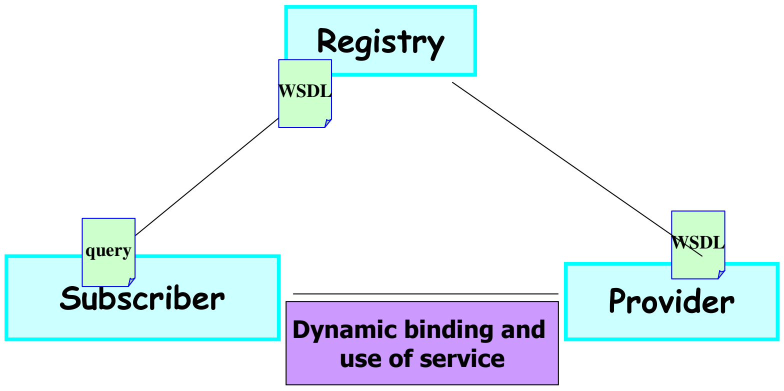
  [eforce; www.eforceglobal.com]

- A service is logical manifestation of some resource combined with some business logic
- Service interaction is facilitated by message exchange
- Reuse at business level

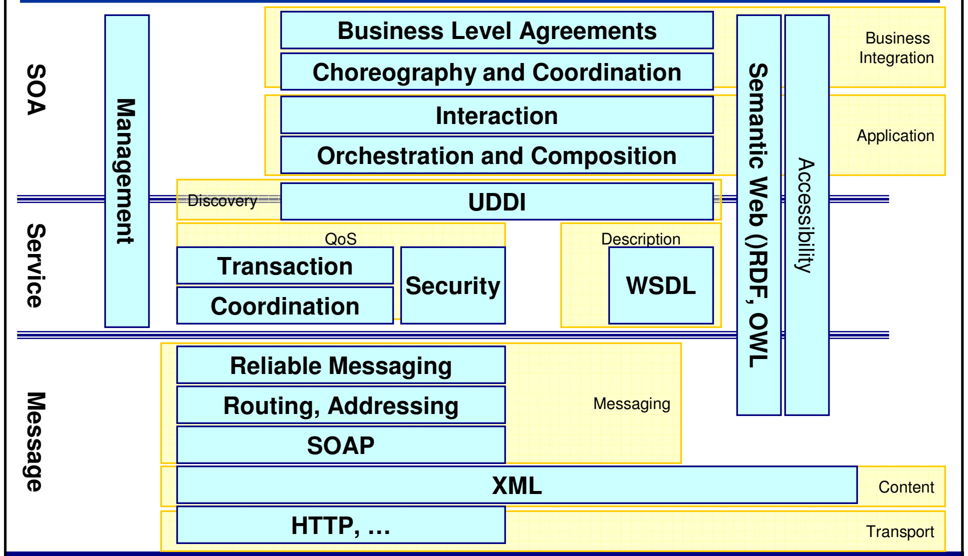
soa

- Service oriented Architecture
  - Application architecture with all functions defined as independent services with well defined interfaces which can be called in defined sequences
  - It's a way of thinking about building software
- Keywords: loosely coupled, event-driven, assembly and integration

publish-find-bind



ws: the "standards"



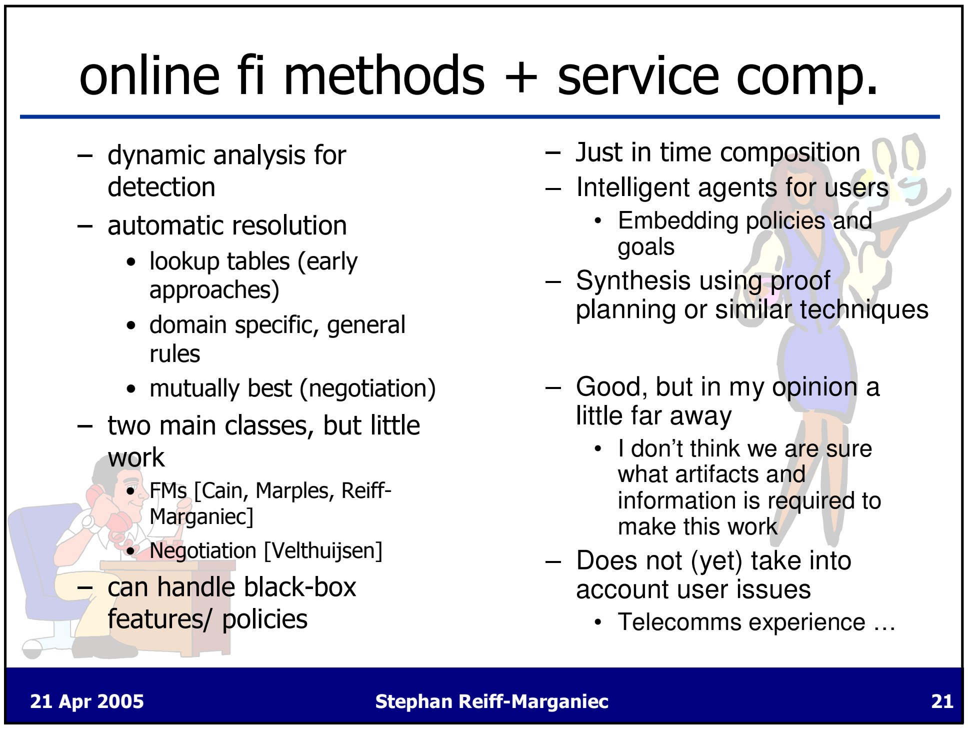
offline fi methods + service comp.

- static analysis detects problems
  - (FM, Testing, Design Principles)
- resolution by redesign
- good if details are known (intra-company, …)
- for policies automatic methods can be used at upload time, user then can redefine policies

- not suitable when design details are unavailable (open market)

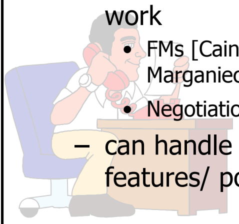
- Manually compose well understood services
- BPEL etc
- User needs to understand technical details
- Some tool support possible

- Not good if we want just in time composition
- Self healing systems require much effort
- What about end-user goals?

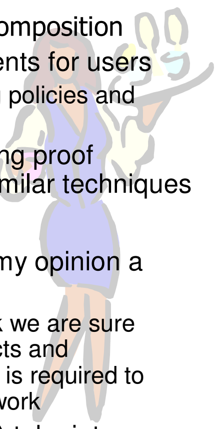


online fi methods + service comp.

- dynamic analysis for detection
- automatic resolution
  - lookup tables (early approaches)
  - domain specific, general rules
  - mutually best (negotiation)
- two main classes, but little work
  - FMs [Cain, Marples, Reiff-Marganiec]
  - Negotiation [Velthuijsen]
- can handle black-box features/ policies

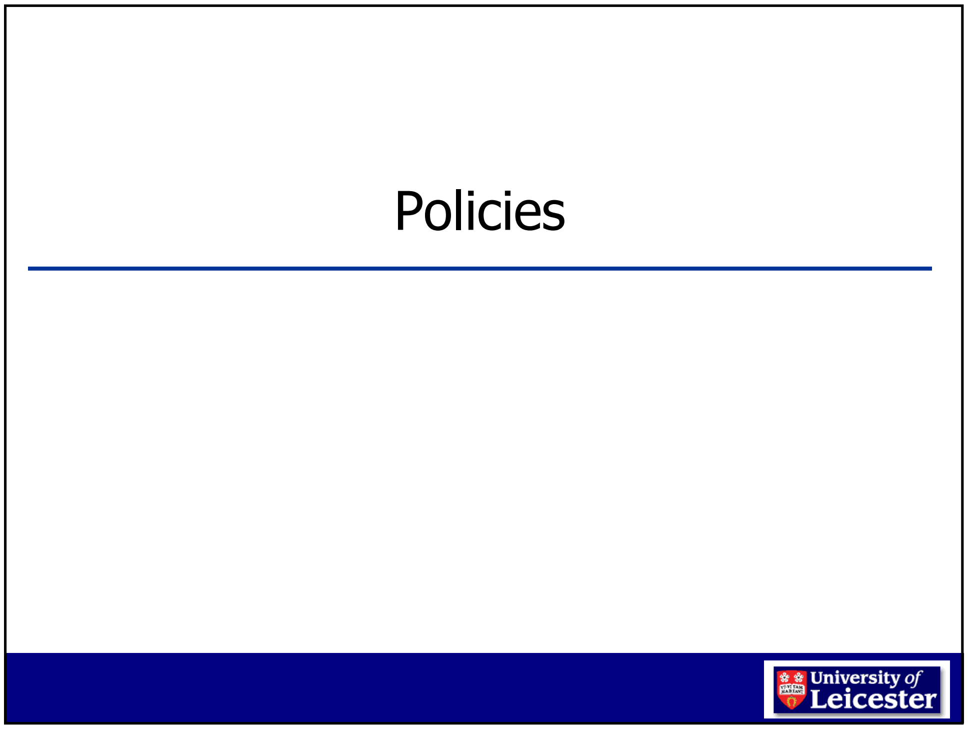


- Just in time composition
- Intelligent agents for users
  - Embedding policies and goals
- Synthesis using proof planning or similar techniques



- Good, but in my opinion a little far away
  - I don't think we are sure what artifacts and information is required to make this work
- Does not (yet) take into account user issues
  - Telecomms experience …

21 Apr 2005 Stephan Reiff-Marganiec 21



Policies

University *of* Leicester

policies

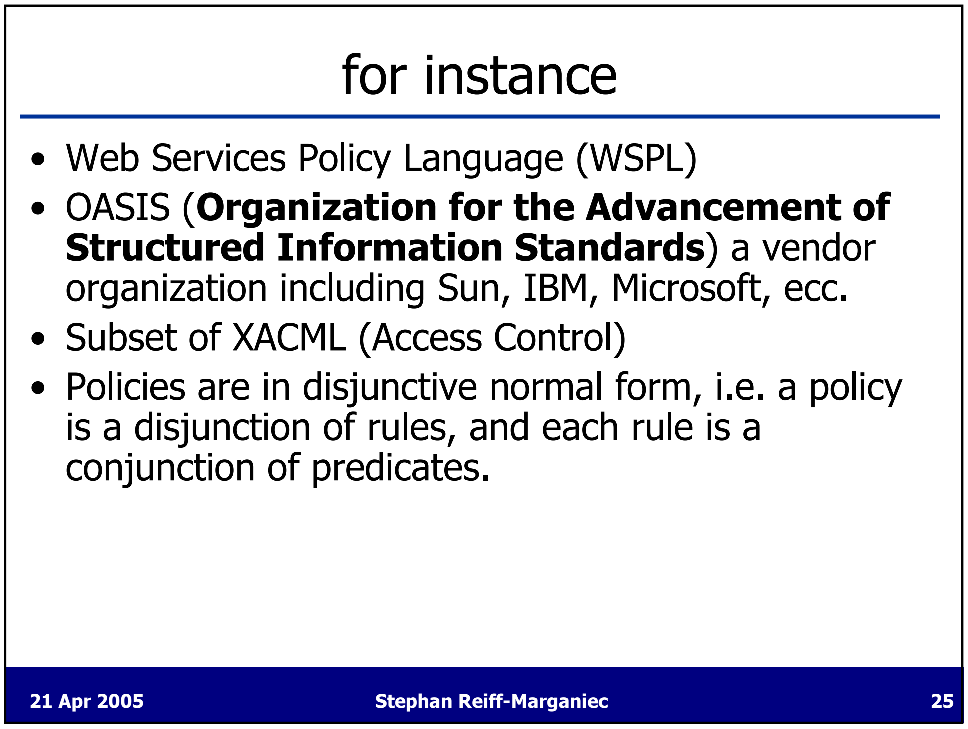
information which can be used to modify the behaviour of a system (Lupu, Sloman, 1999)

matches Mark Ryan's understanding of feature:
description of how to transform the base system

- policies are used in the context of
  - multimedia and distributed systems
  - agent based systems
  - systems management
  - security
  - quality of service management
- usually for access control

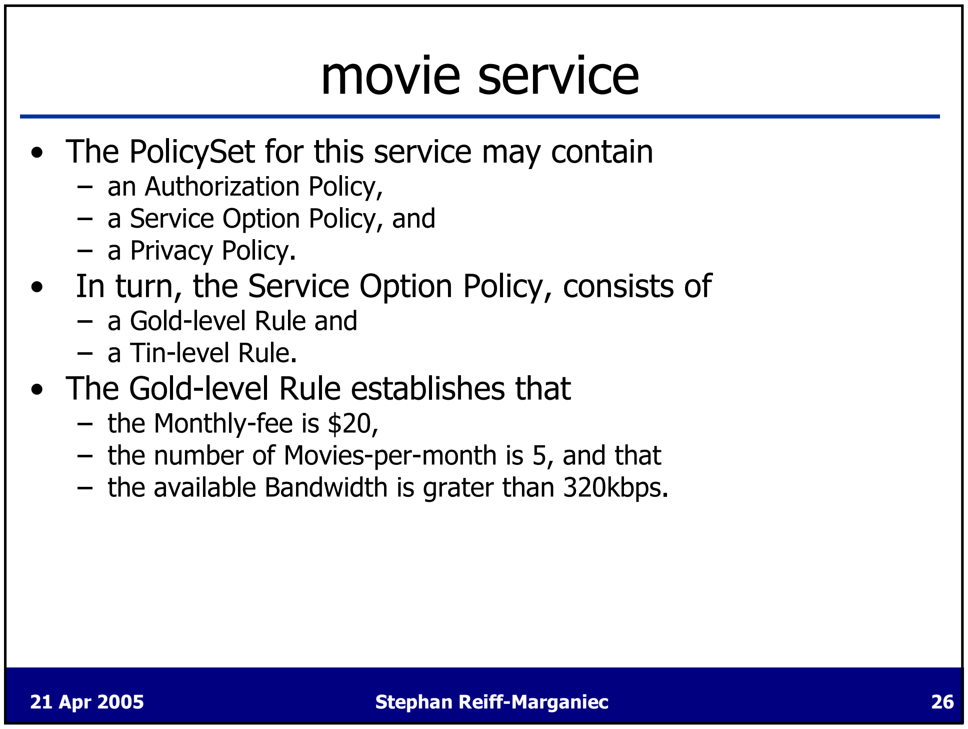
policies...

- ... mean many things to many people:
  - Guiding principles and procedures
  - Management policy,i.e.
    - Event -> Condition -> Action (ECA)
  - Authorization (access control) policy
  - OPI (Obligation, Permission, Interdiction)
    - Deontic logic



for instance

- Web Services Policy Language (WSPL)
- OASIS (**Organization for the Advancement of Structured Information Standards**) a vendor organization including Sun, IBM, Microsoft, ecc.
- Subset of XACML (Access Control)
- Policies are in disjunctive normal form, i.e. a policy is a disjunction of rules, and each rule is a conjunction of predicates.



movie service

- The PolicySet for this service may contain
  - an Authorization Policy,
  - a Service Option Policy, and
  - a Privacy Policy.
- In turn, the Service Option Policy, consists of
  - a Gold-level Rule and
  - a Tin-level Rule.
- The Gold-level Rule establishes that
  - the Monthly-fee is $20,
  - the number of Movies-per-month is 5, and that
  - the available Bandwidth is grater than 320kbps.

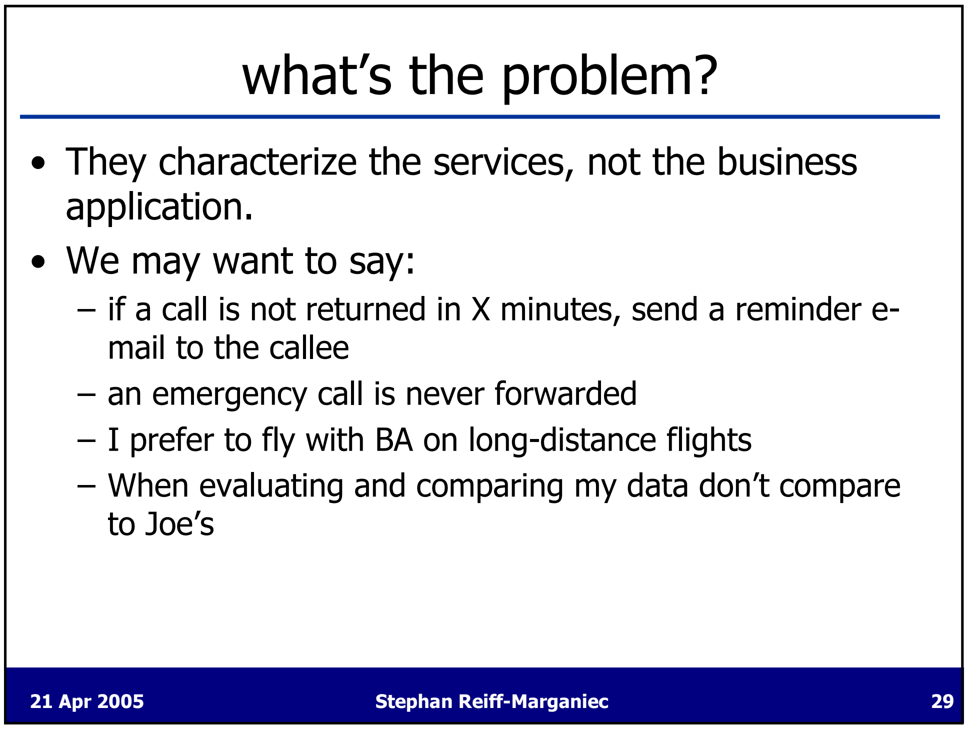
also...

- Web Services Policy Framework (WSPolicy)
- Proposed by a consortium (BEA Systems, IBM, Microsoft, SAP, Sonic Software, and VeriSign)
- Mostly concerned with WS Capabilities
- policy is used to convey conditions on an interaction between two Web service endpoints. Typically, the provider of a Web service exposes a policy to convey conditions under which it provides the service. A requester might use this policy to decide whether or not to use the service.

their example…

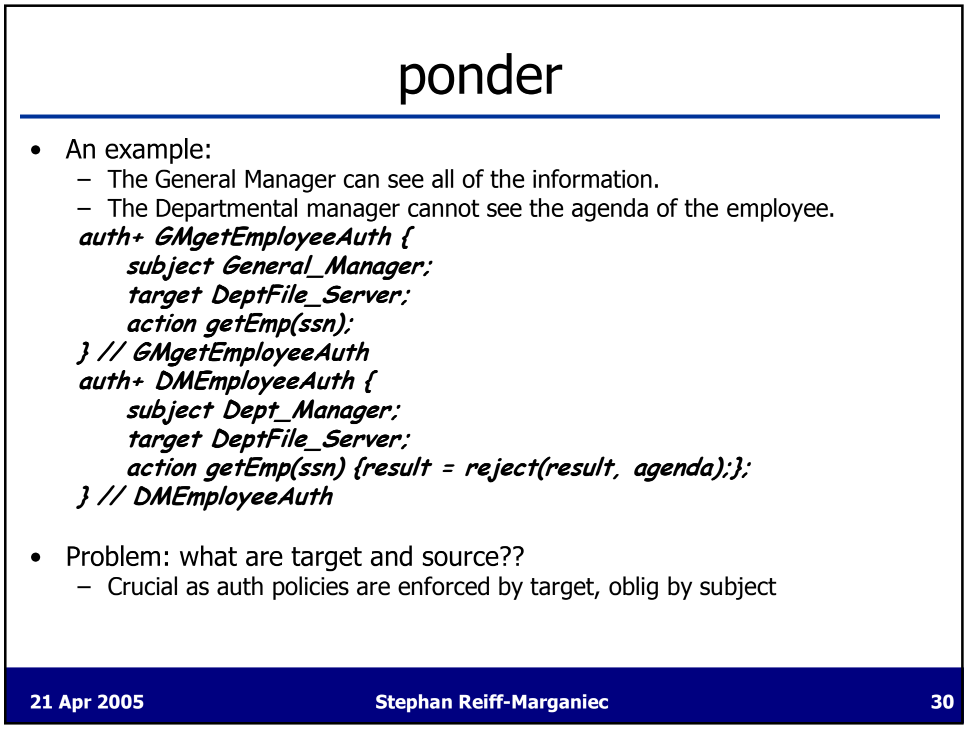
- … of policy is one stating that a Web service uses one out of a list of cryptographics algorithms.

- Both WSPL and WSPolicy have a notion of policy merge (negotiation, intersection)
- Both are XML based



what’s the problem?

- They characterize the services, not the business application.
- We may want to say:
  - if a call is not returned in X minutes, send a reminder e-mail to the callee
  - an emergency call is never forwarded
  - I prefer to fly with BA on long-distance flights
  - When evaluating and comparing my data don’t compare to Joe’s



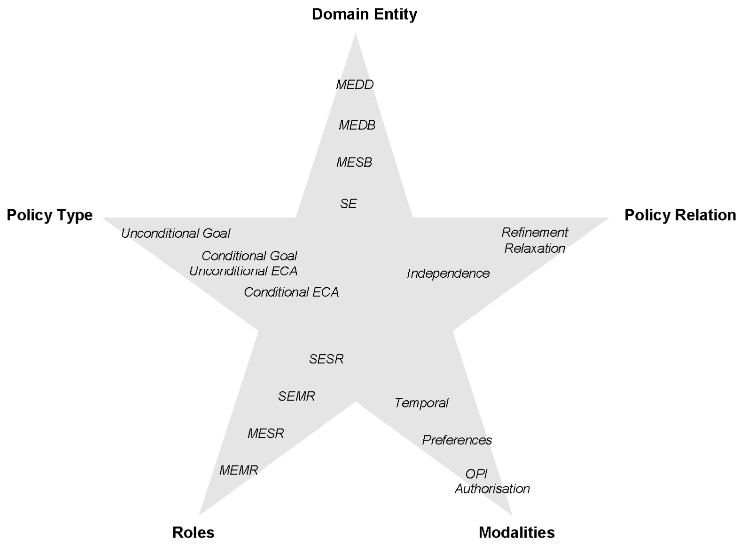
ponder

- An example:
  - The General Manager can see all of the information.
  - The Departmental manager cannot see the agenda of the employee.

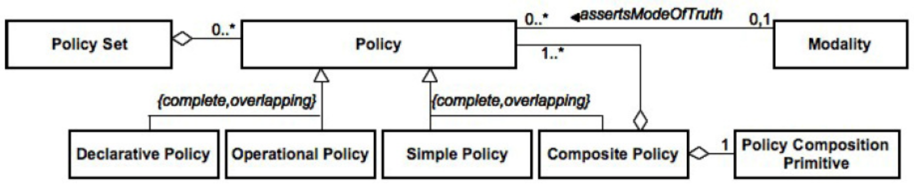
```
auth+ GMgetEmployeeAuth {
    subject General_Manager;
    target DeptFile_Server;
    action getEmp(ssn);
} // GMgetEmployeeAuth
auth+ DMEmployeeAuth {
    subject Dept_Manager;
    target DeptFile_Server;
    action getEmp(ssn) {result = reject(result, agenda);};
} // DMEmployeeAuth
```

- Problem: what are target and source??
  - Crucial as auth policies are enforced by target, oblig by subject

dimensions of policy conflict



a policy model



Policies for end-user services in telecommunications

University of Leicester

motivation

- (tele)communications central to daily activities
- users communication needs
  - enabling communications
- user should be in control
  - always on, mobility,
- merging of communications technologies
  - traditional telecommunications
  - conferencing
  - device control
  - wireless, mobile
  - VoIP…
    - email
    - video
    - home automation
    - ad hoc networking

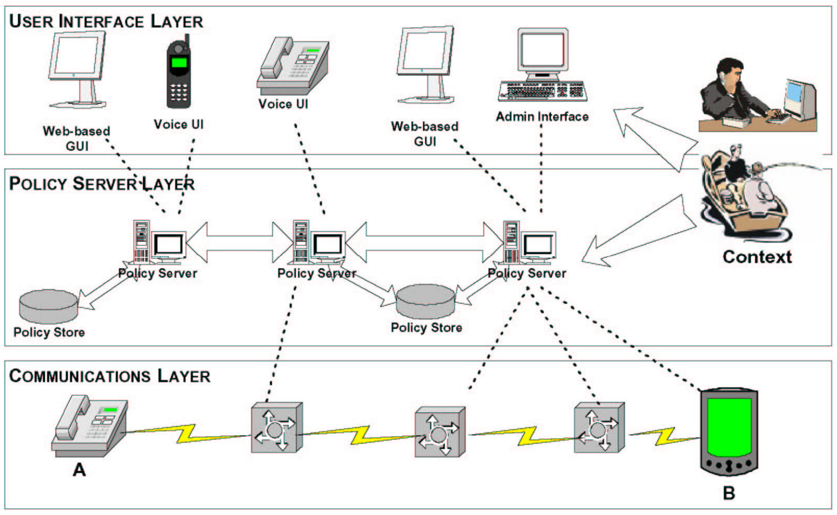
21 Apr 2005 Stephan Reiff-Marganiec 34

policies as features

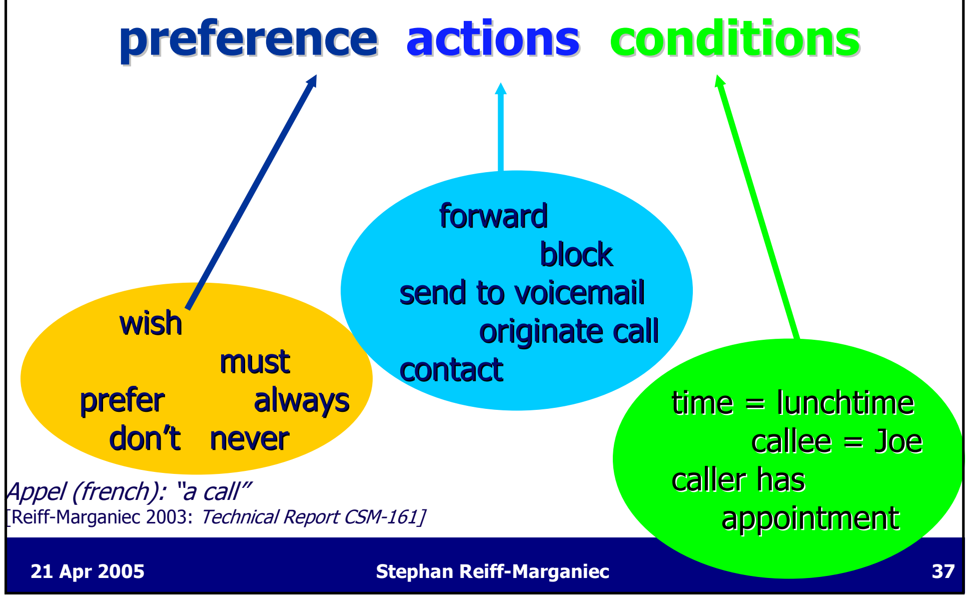
- policies can be used to control calls
- are user definable/editable
- make use of context
- more flexible
- more abstract/higher level

- we could
  - enhance features with new capabilities to do the same
  - but would need to move away from “operator provides” paradigm

enhanced call control architecture

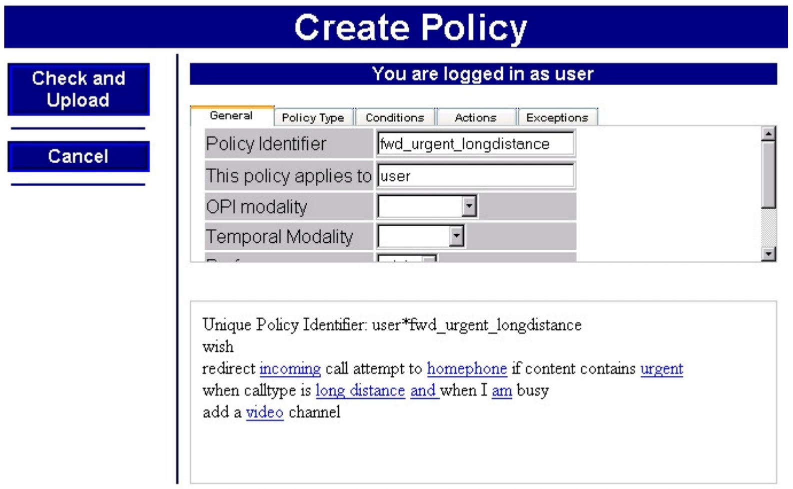


what matter's to users



Appel (french): "a call"
[Reiff-Marganiec 2003: *Technical Report CSM-161]*

policy wizard



Policy Identifier	fwd_urgent_longdistance
This policy applies to	user
OPI modality	
Temporal Modality	

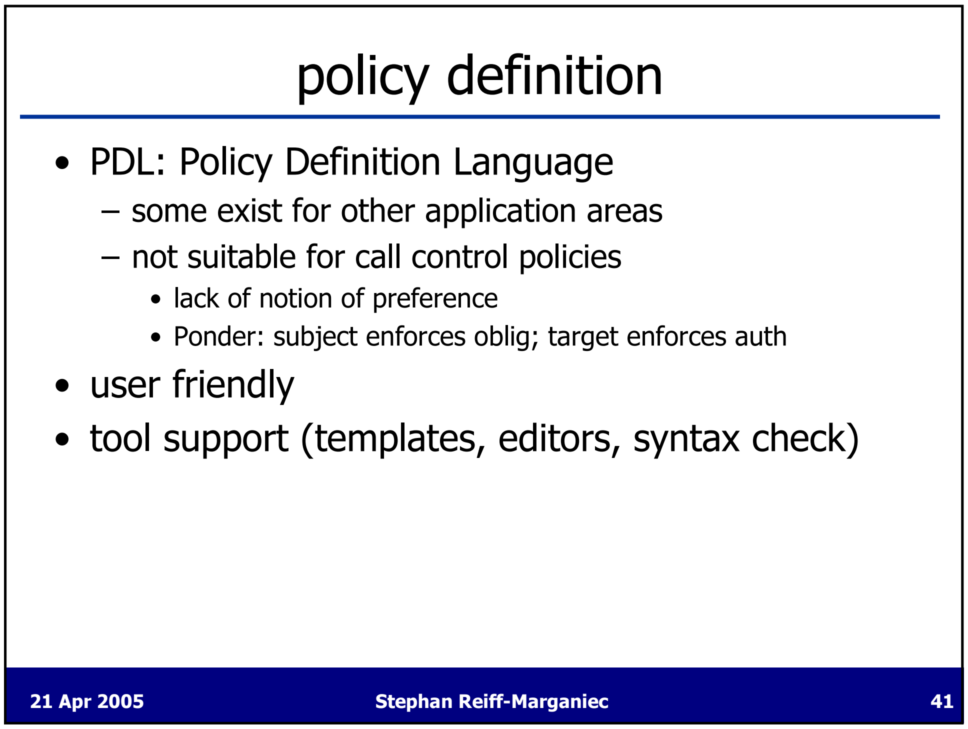
Unique Policy Identifier: user*fwd_urgent_longdistance
wish
redirect incoming call attempt to homephone if content contains urgent
when calltype is long distance and when I am busy
add a video channel

policy lifecycle

- Definition
- Deployment
- Enforcement
- (Decommission/Undeploy/Update)

policy: examples

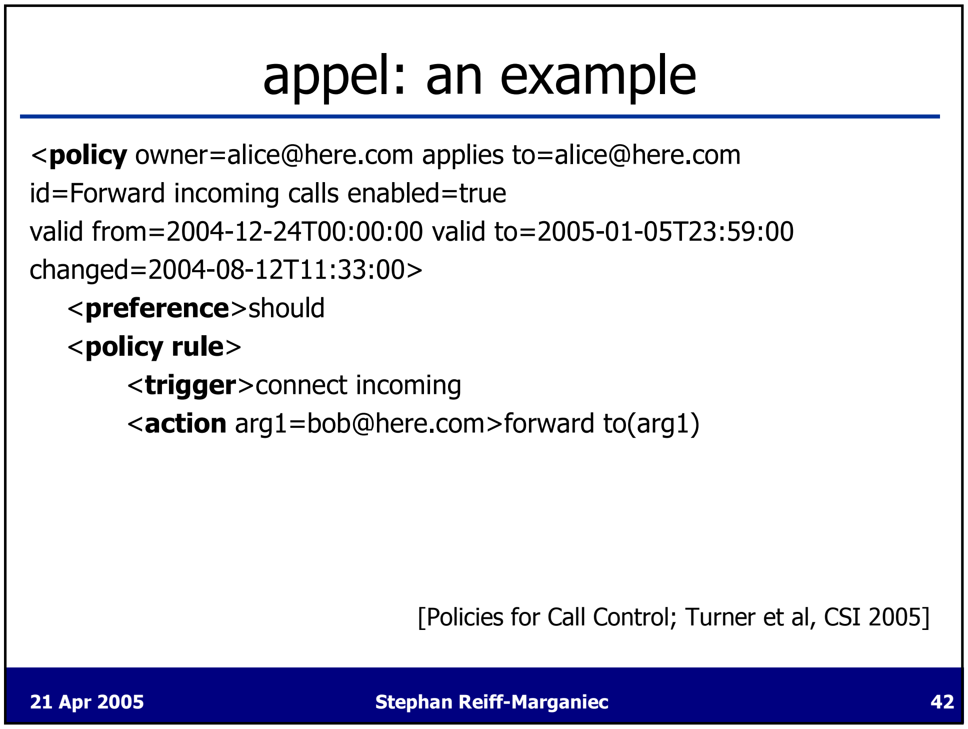
- lecturers are unavailable at lunchtime to discuss any matters with students, except it is an emergency
- I prefer to speak to John or Mary if Paul is busy
- if my call is not returned within one hour, send an email reminder to the callee
- always notify me when an expected visitor arrives at reception
- When the police calls, include the company lawyer in the call
- don't disturb Ken at lunchtime on Fridays



policy definition

- PDL: Policy Definition Language
  - some exist for other application areas
  - not suitable for call control policies
    - lack of notion of preference
    - Ponder: subject enforces oblig; target enforces auth
- user friendly
- tool support (templates, editors, syntax check)

21 Apr 2005 Stephan Reiff-Marganiec 41

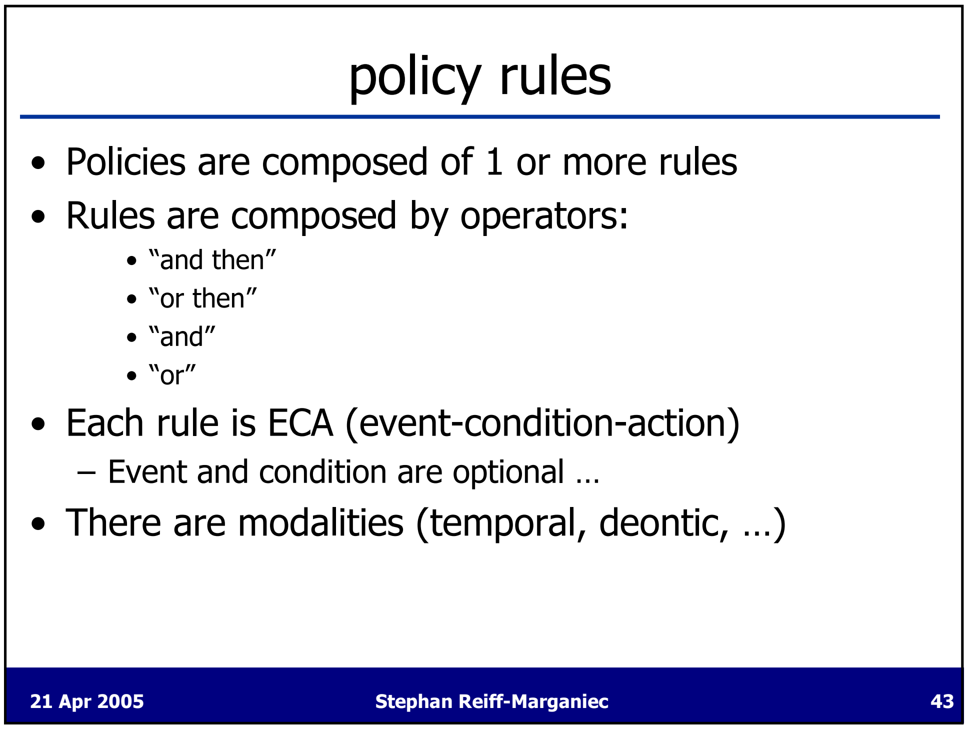


appel: an example

<**policy** owner=alice@here.com applies to=alice@here.com
id=Forward incoming calls enabled=true
valid from=2004-12-24T00:00:00 valid to=2005-01-05T23:59:00
changed=2004-08-12T11:33:00>
    <**preference**>should
    <**policy rule**>
        <**trigger**>connect incoming
        <**action** arg1=bob@here.com>forward to(arg1)

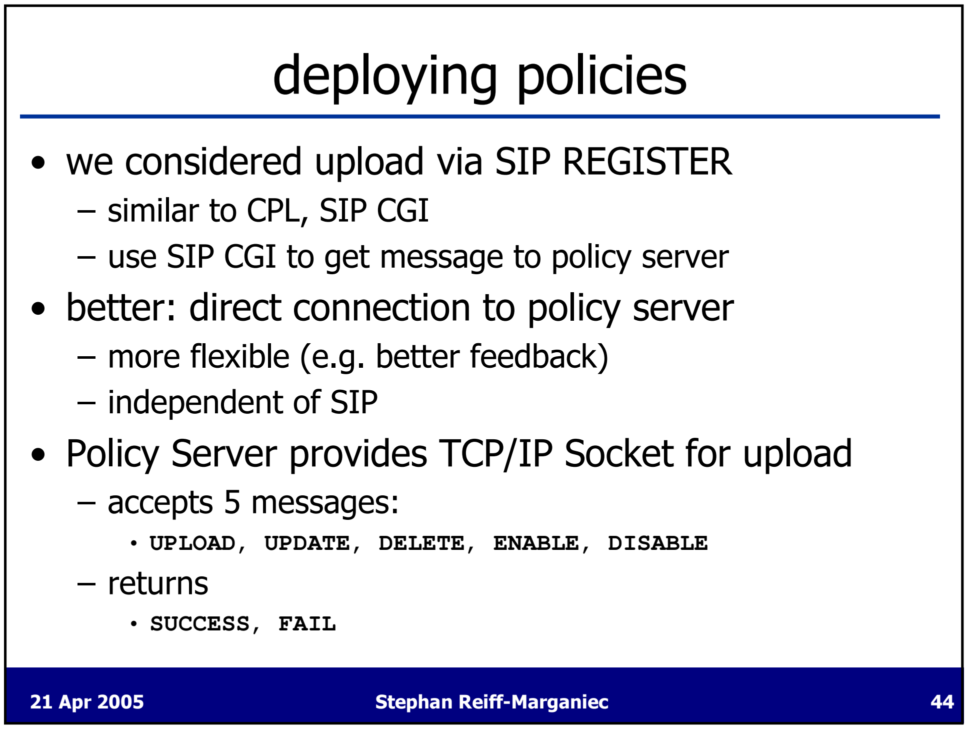
[Policies for Call Control; Turner et al, CSI 2005]

21 Apr 2005 Stephan Reiff-Marganiec 42



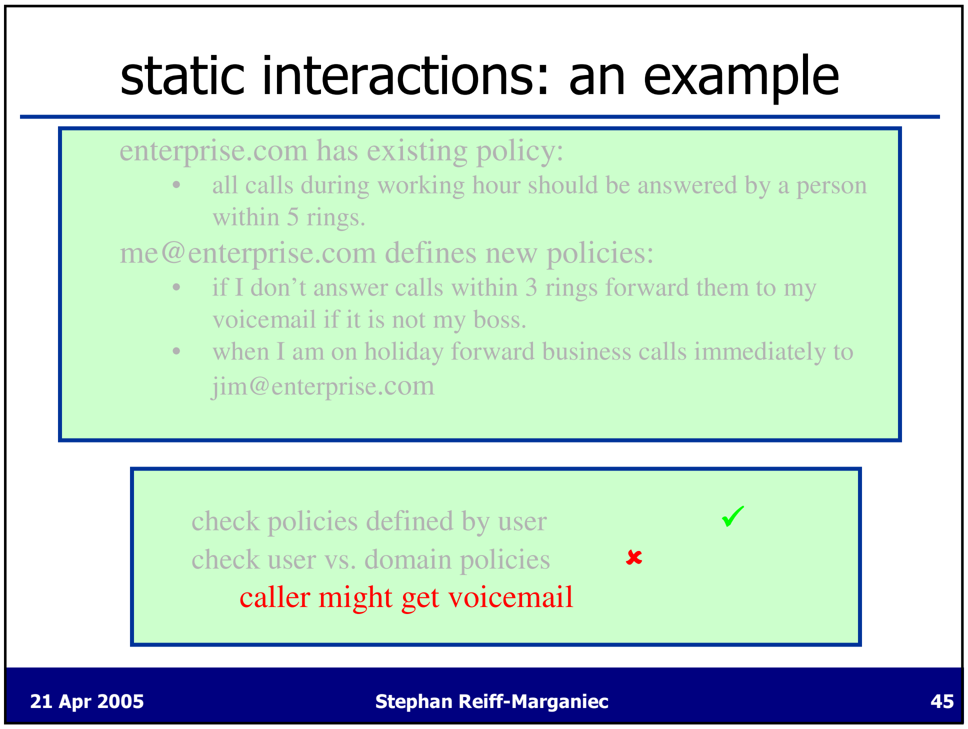
policy rules

- Policies are composed of 1 or more rules
- Rules are composed by operators:
    - "and then"
    - "or then"
    - "and"
    - "or"
- Each rule is ECA (event-condition-action)
    - Event and condition are optional …
- There are modalities (temporal, deontic, …)



deploying policies

- we considered upload via SIP REGISTER
    - similar to CPL, SIP CGI
    - use SIP CGI to get message to policy server
- better: direct connection to policy server
    - more flexible (e.g. better feedback)
    - independent of SIP
- Policy Server provides TCP/IP Socket for upload
    - accepts 5 messages:
        - `UPLOAD, UPDATE, DELETE, ENABLE, DISABLE`
    - returns
        - `SUCCESS, FAIL`



static interactions: an example

enterprise.com has existing policy:

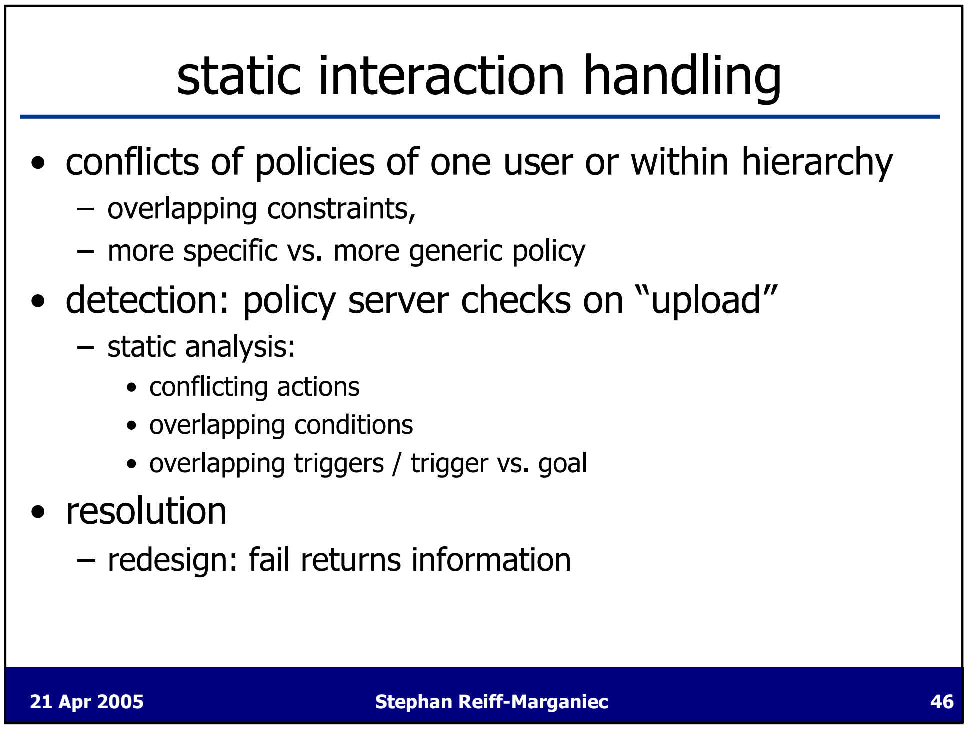
- all calls during working hour should be answered by a person within 5 rings.

me@enterprise.com defines new policies:

- if I don't answer calls within 3 rings forward them to my voicemail if it is not my boss.
- when I am on holiday forward business calls immediately to jim@enterprise.com

check policies defined by user ✓
check user vs. domain policies ✗
caller might get voicemail

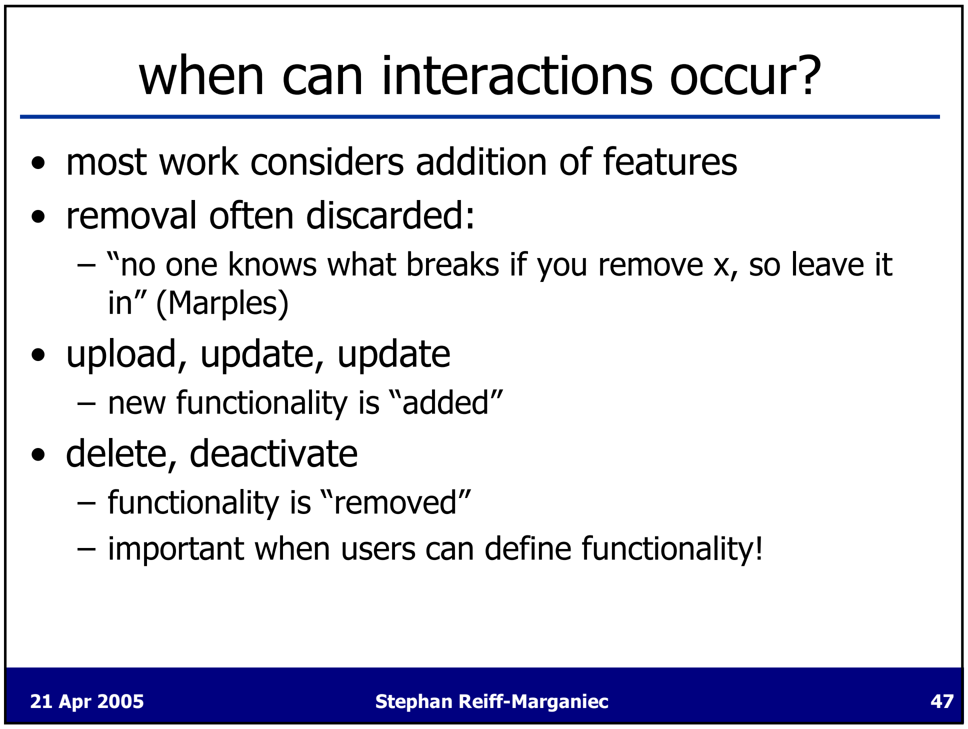
21 Apr 2005 Stephan Reiff-Marganiec 45



static interaction handling

- conflicts of policies of one user or within hierarchy
  - overlapping constraints,
  - more specific vs. more generic policy
- detection: policy server checks on "upload"
  - static analysis:
    - conflicting actions
    - overlapping conditions
    - overlapping triggers / trigger vs. goal
- resolution
  - redesign: fail returns information

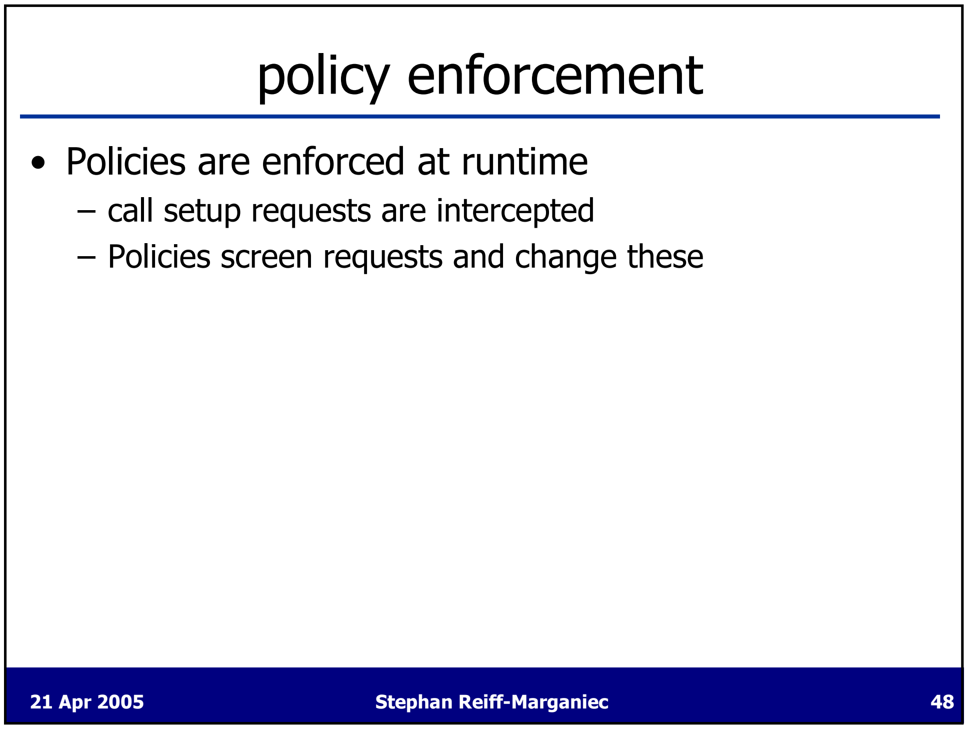
21 Apr 2005 Stephan Reiff-Marganiec 46



when can interactions occur?

- most work considers addition of features
- removal often discarded:
  - "no one knows what breaks if you remove x, so leave it in" (Marples)
- upload, update, update
  - new functionality is "added"
- delete, deactivate
  - functionality is "removed"
  - important when users can define functionality!

21 Apr 2005 Stephan Reiff-Marganiec 47



policy enforcement

- Policies are enforced at runtime
  - call setup requests are intercepted
  - Policies screen requests and change these

21 Apr 2005 Stephan Reiff-Marganiec 48

dynamic interactions: an example

mary@enterprise.com has policy:
- I prefer to speak to John if Paul is busy.

paul@elsewhere.com has policy:
- I expect that my calls are redirected to Joanne when I am busy.

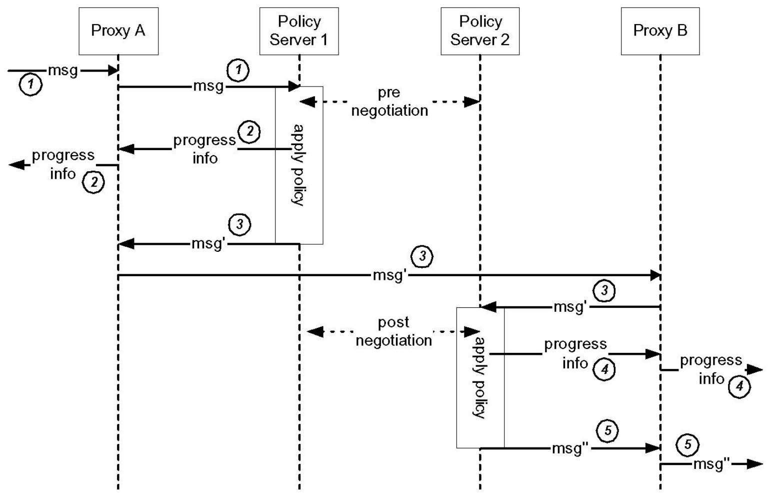
•Mary rings Paul
•Paul is busy

Mary rings Paul; Paul is busy
conflict: forward to Joanne or John??
✓ Joanne: using preference
? could also negotiate ...

dynamic interaction handling

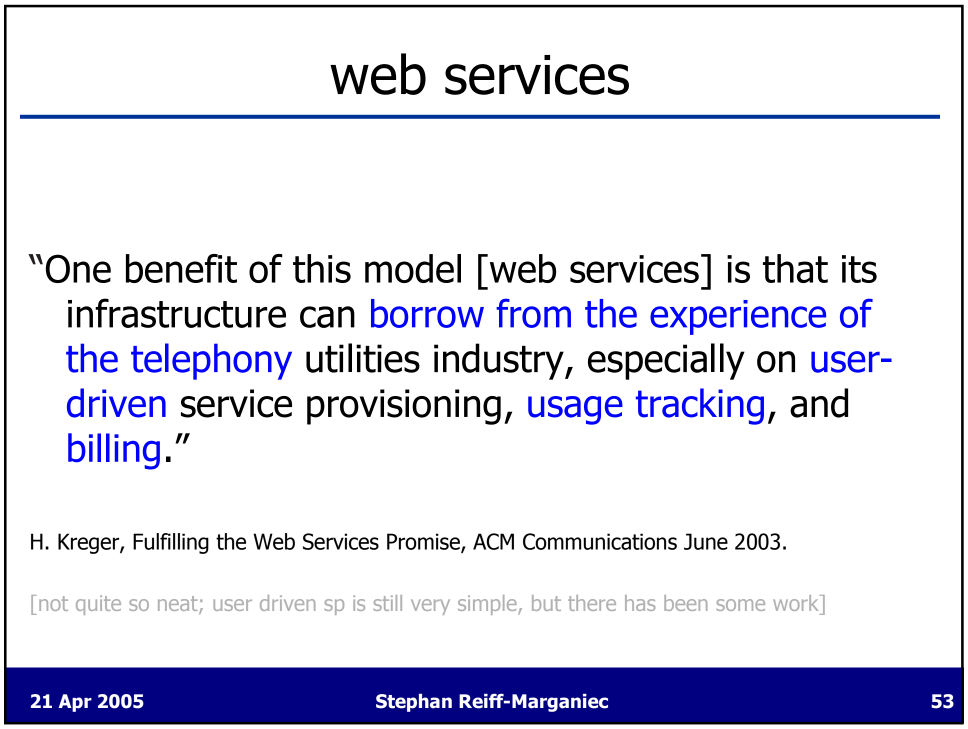
- policies defined by different users (outside hierarchy) might be inconsistent
  - requires dynamic detection and resolution
  - feature manager approaches
    - maybe guided by generic offline analysis
  - negotiation approaches
    - first proposed 1993; never took off …
    - pre negotiation: resolve conflicts before actions are committed to call path (e.g. 3 way conference)
    - post negotiation: resolve conflicts as they are detected
    - this is were agents fit in!
  - we suggest a combination of both

handling policy conflict



End-user services in service oriented architectures

University *of* Leicester



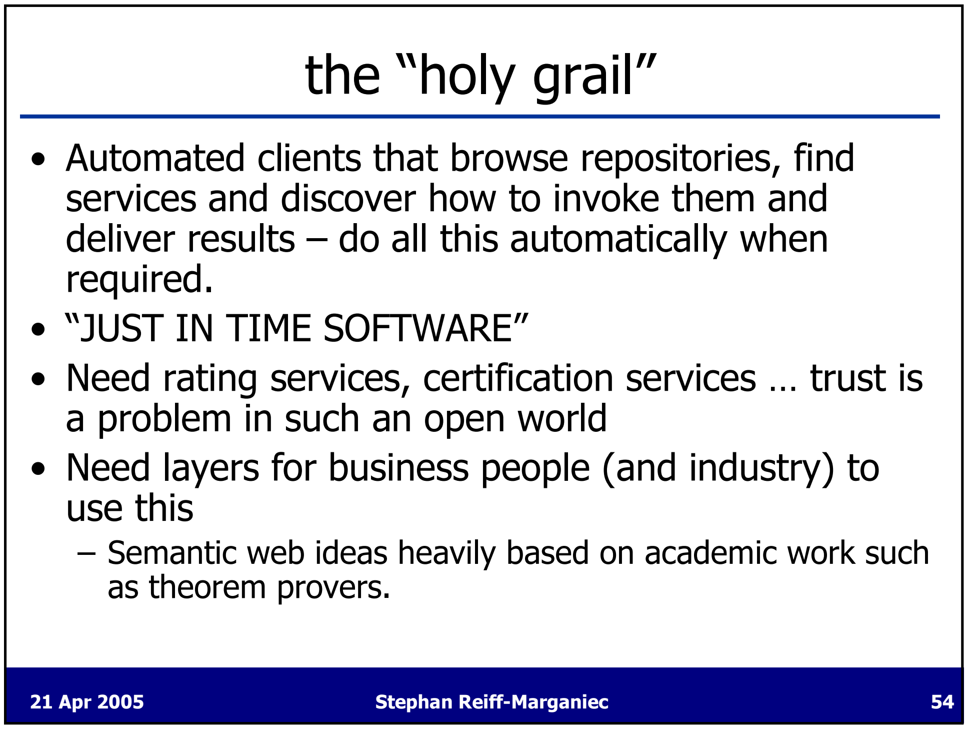
web services

"One benefit of this model [web services] is that its infrastructure can borrow from the experience of the telephony utilities industry, especially on user-driven service provisioning, usage tracking, and billing."

H. Kreger, Fulfilling the Web Services Promise, ACM Communications June 2003.

[not quite so neat; user driven sp is still very simple, but there has been some work]

21 Apr 2005 Stephan Reiff-Marganiec 53

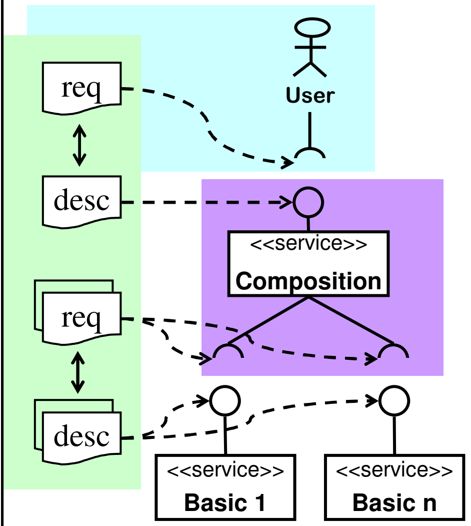


the "holy grail"

- Automated clients that browse repositories, find services and discover how to invoke them and deliver results – do all this automatically when required.
- "JUST IN TIME SOFTWARE"
- Need rating services, certification services … trust is a problem in such an open world
- Need layers for business people (and industry) to use this
  - Semantic web ideas heavily based on academic work such as theorem provers.

21 Apr 2005 Stephan Reiff-Marganiec 54

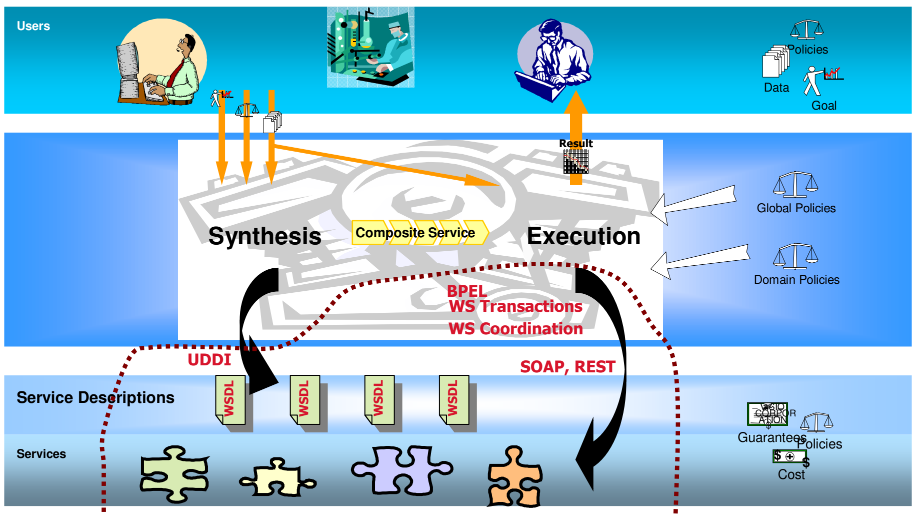
requirements and descriptions



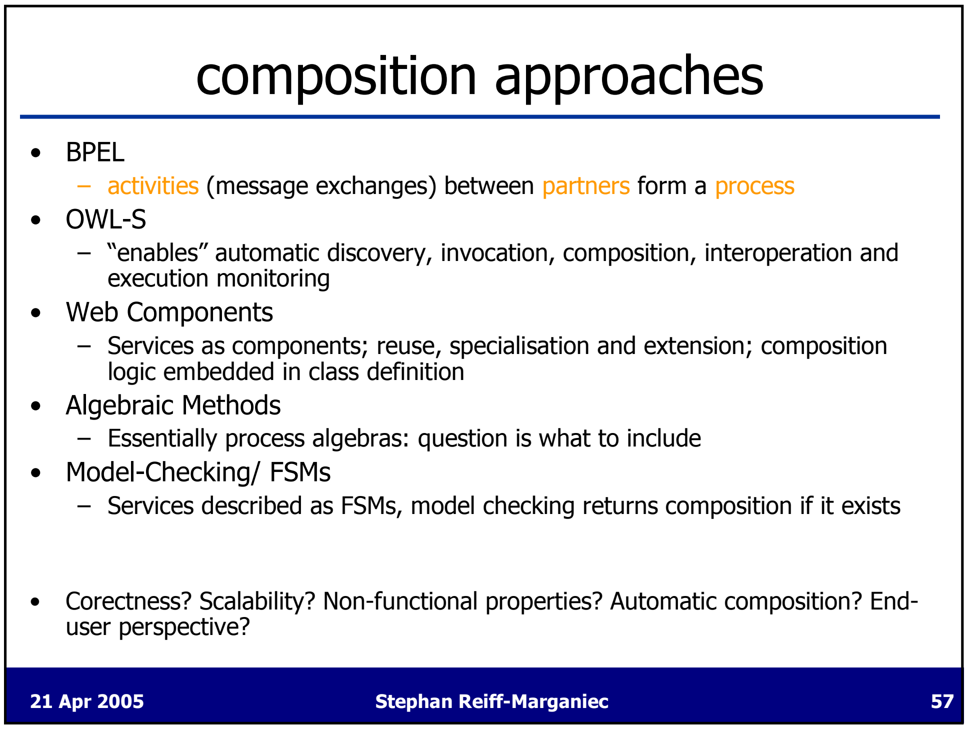
- Service Composition
- Service Description
- Service-Oriented Business Modelling

21 Apr 2005 Stephan Reiff-Marganiec 55

enhanced services architecture



21 Apr 2005 Stephan Reiff-Marganiec 56

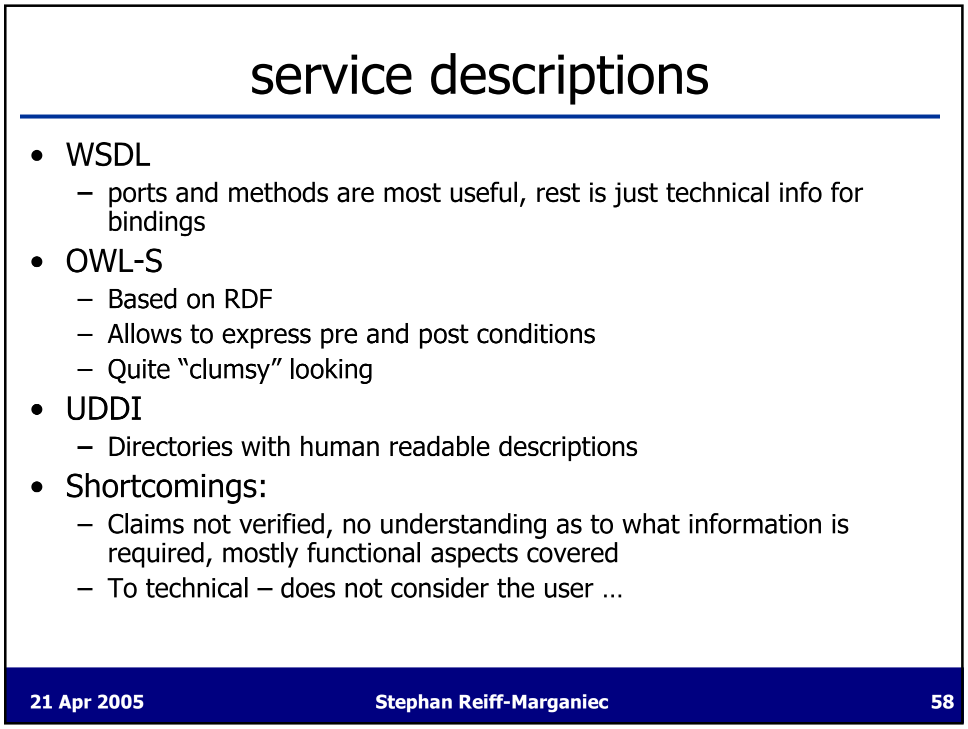


composition approaches

- BPEL
  - activities (message exchanges) between partners form a process
- OWL-S
  - "enables" automatic discovery, invocation, composition, interoperation and execution monitoring
- Web Components
  - Services as components; reuse, specialisation and extension; composition logic embedded in class definition
- Algebraic Methods
  - Essentially process algebras: question is what to include
- Model-Checking/ FSMs
  - Services described as FSMs, model checking returns composition if it exists

- Corectness? Scalability? Non-functional properties? Automatic composition? End-user perspective?

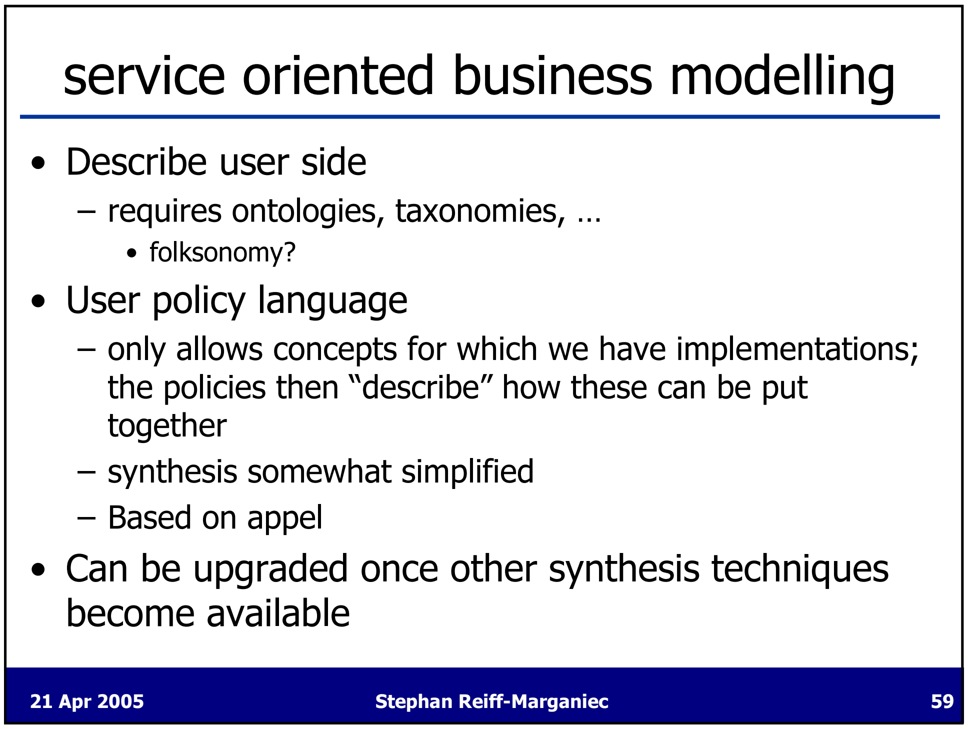
21 Apr 2005 Stephan Reiff-Marganiec 57



service descriptions

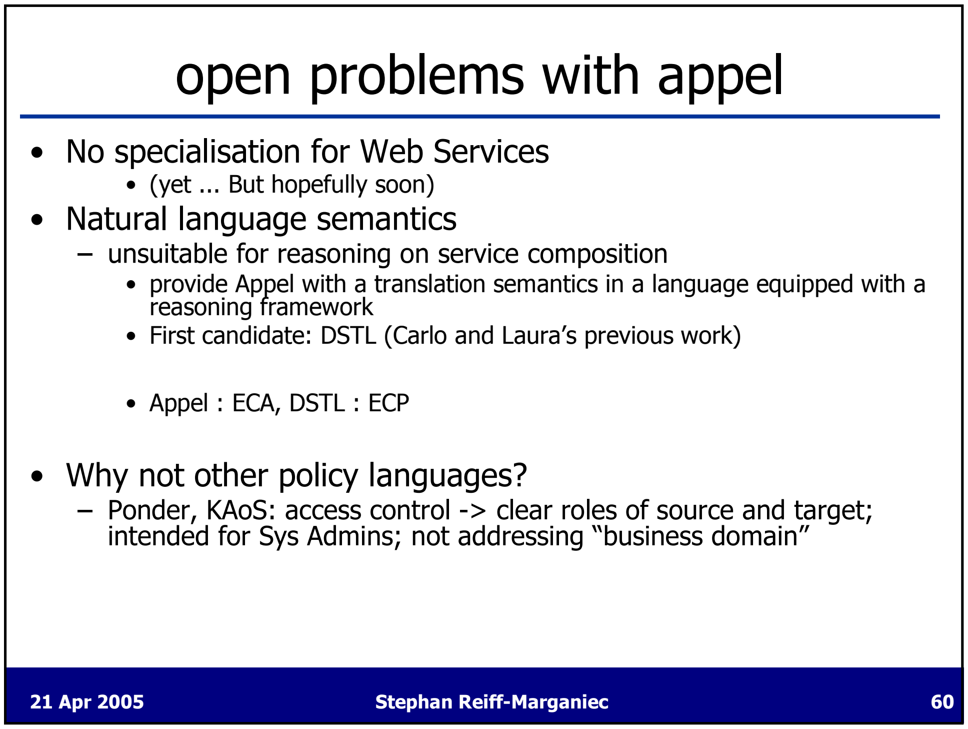
- WSDL
  - ports and methods are most useful, rest is just technical info for bindings
- OWL-S
  - Based on RDF
  - Allows to express pre and post conditions
  - Quite "clumsy" looking
- UDDI
  - Directories with human readable descriptions
- Shortcomings:
  - Claims not verified, no understanding as to what information is required, mostly functional aspects covered
  - To technical – does not consider the user …

21 Apr 2005 Stephan Reiff-Marganiec 58



service oriented business modelling

- Describe user side
  - requires ontologies, taxonomies, …
    - folksonomy?
- User policy language
  - only allows concepts for which we have implementations; the policies then "describe" how these can be put together
  - synthesis somewhat simplified
  - Based on appel
- Can be upgraded once other synthesis techniques become available



open problems with appel

- No specialisation for Web Services
    - (yet ... But hopefully soon)
- Natural language semantics
  - unsuitable for reasoning on service composition
    - provide Appel with a translation semantics in a language equipped with a reasoning framework
    - First candidate: DSTL (Carlo and Laura's previous work)

    - Appel : ECA, DSTL : ECP

- Why not other policy languages?
  - Ponder, KAoS: access control -> clear roles of source and target; intended for Sys Admins; not addressing "business domain"

business modelling

- Business Modelling Languages
  - Mostly workflows: UML activity diagrams, BPML, Stephen's proposal (BPM 2006 paper)
  - Main concept: tasks and the ordering of execution
    - Data and control flows might be separate
  - At business level, but specific for each process
  - We also have overarching constraints that apply to many processes, are possibly more agile

current solutions 1

- Approach 1: Composition as Requirements
  - BPEL:

```
<process name="test">
<partnerLinks>
  <partnerLink name="client"/>
  <partnerLink name="serviceA"/>
  <partnerLink name="serviceB"/>
  <partnerLink name="serviceC"/>
</partnerLinks>
<variables>
  <variable name="processInput"/>
  <variable name="AInput"/>
  <variable name="AOutput"/>
  <variable name="BCInput"/>
  <variable name="BOutput"/>
  <variable name="COutput"/>
  <variable name="processOutput"/>
  <variable name="AError"/>
</variables>
<sequence>
  <receive name="receiveInput"
    variable="input"/>
    <assign>
      <copy>
        <from variable="processInput"/>
        <to variable="AInput"/>
      </copy>
</assign>
<scope>
  <faultHandlers>
    <catch
      faultName="faultA"
      fault-
      Variable="AError"/
      >
  </faultHandlers>
  <sequence>
    <invoke
      name="invokeA"
      partner-
      Link="serviceA"
      inputVariable="AIn
      put" output-
      Variable="AOutput"
      />
  </sequence>
</scope>
<assign>
  <copy>
    <from
    variable="AOutput"/>
      <to
      variable="BCInput"/>
    </copy>
  </assign>
  <flow>
    <sequence>
      <invoke
        name="invokeB"
        partner-
        Link="serviceB"
        inputVariable="BCI
        nput"/>
      <receive
        name="receive_invo
        keB"
        partnerLink="servi
        ceB"
        variable="BOutput"
        />
    </sequence>
    <sequence>
      <invoke
        name="invokeC"
        partner-
        Link="serviceC"
        inputVariable="BCI
        nput"/>
      <receive
        name="receive_invo
        keC"
        partnerLink="servi
        ceC"
        variable="COutput"
        />
    </sequence>
  </flow>
  <switch>
    <case>
      <!—assign value to
      processOutput—>
    </case>
  </switch>
  <invoke name="reply"
    partnerLink="client"
    inputVariable="process
    Output"/>
  </sequence>
</process>
```

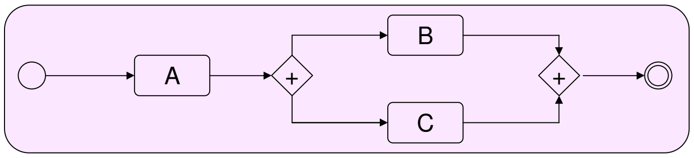
  - DAML-S

```
<daml:Class rdf:ID="test">
 <daml:subClassOf
rdf:resource="Process.CompositeProcess"/>
 <daml:subClassOf>
  <daml:Restriction>
   <daml:onProperty
rdf:resource="Process#composedOf"/>
   <daml:toClass>
    <daml:Class>
     <daml:intersectionOf rdf:parse-
Type="daml:collection">
      <daml:Class
rdf:about="process:Sequence">
       <daml:Restriction>
        <daml:onProperty
rdf:resource="Process#components"/>
        <daml:toClass>
         <daml:Class>
          <process:listOfInstancesOf
               rdf:parseType="daml:col-
lection">
          <daml:Class rdf:about="#ser-
viceA"/>
          <daml:Class
rdf:about="process:Split">
           <daml:Restriction>
            <daml:onProperty
rdf:resource="Process#components"/>
            <daml:toClass>
             <daml:Class>
              <process:listOfInstancesOf
                 rdf:parseType="daml:col-
lection">
                <daml:Class
rdf:about="#serviceB"/>
                <daml:Class
rdf:about="#serviceC"/>
              </process:listOfInstance-
sOf>
             </daml:Class>
         . . .
```

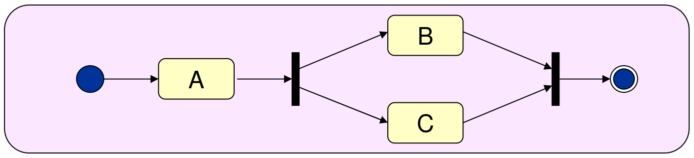
Code snippets taken from Milanovic and Malek: Current Solutions for Web Service Composition. IEEE Internet Computing, Nov/Dec 04

current solutions 2

- Approach 2: Specialised Requirements Language
  - BPMN:



  - UML:



wedding example

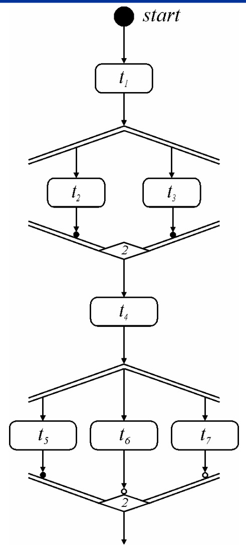
- Business goal *g* = "plan wedding";
- Broken down into objectives (composite tasks):
  - ct_1 = plan pre-wedding celebrations;
  - ct_2 = plan preparations;
  - ct_3 = plan legalities;
  - ct_4 = plan ceremony;
  - ct_5 = plan post-ceremony celebrations;
  - ct_6 = plan honeymoon.
- Tasks are arranged according to result timeline, not according to execution timeline!
  - e.g. ceremony and post-ceremony celebrations often planned in parallel.
- Policies:
  - The entire event should not cost more than £10k;
  - The ceremony and post-ceremony celebrations should be on the same day;
  - The honeymoon should be booked through a known and trusted travel agency.

booking the honeymoon 1

Flows:
- Control runs from start to finish;
- lines indicate control flow routes;
- A task is executed when control reaches it;
- Control proceeds when the task has finished.

Flow Split:
- Control proceeds down each output simultaneously;
- No limit on number of output flows;
- Parallel split workflow pattern

Conditional Merge:
- Forces synchronisation;
- Mandatory and optional flows;
- Specifies minimum number of flows;
- Discriminator workflow pattern.

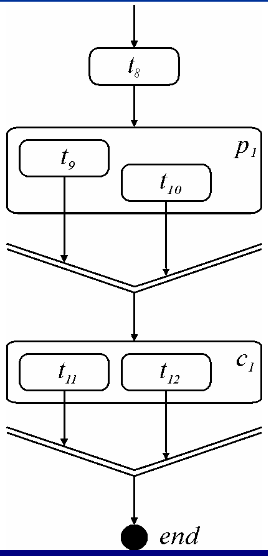


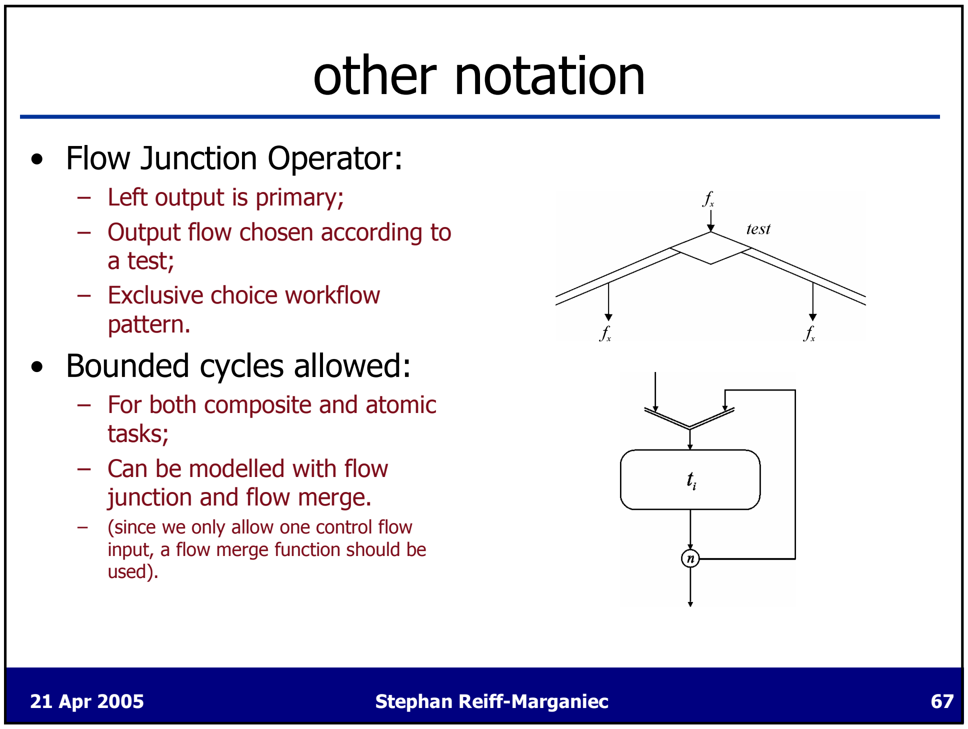
booking the honeymoon 2

Strict Preference:
- New workflow pattern.

Flow Merge:
- Incoming set of control flows contains only one active flow;
- No synchronisation issue;
- (Multiple) Merge workflow pattern.

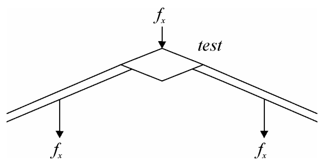
Random Choice:
- All tasks invoked;
- When a first gets to a "commit", all others are cancelled;
- New workflow pattern.

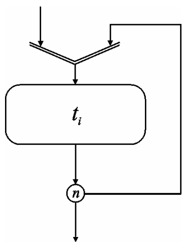


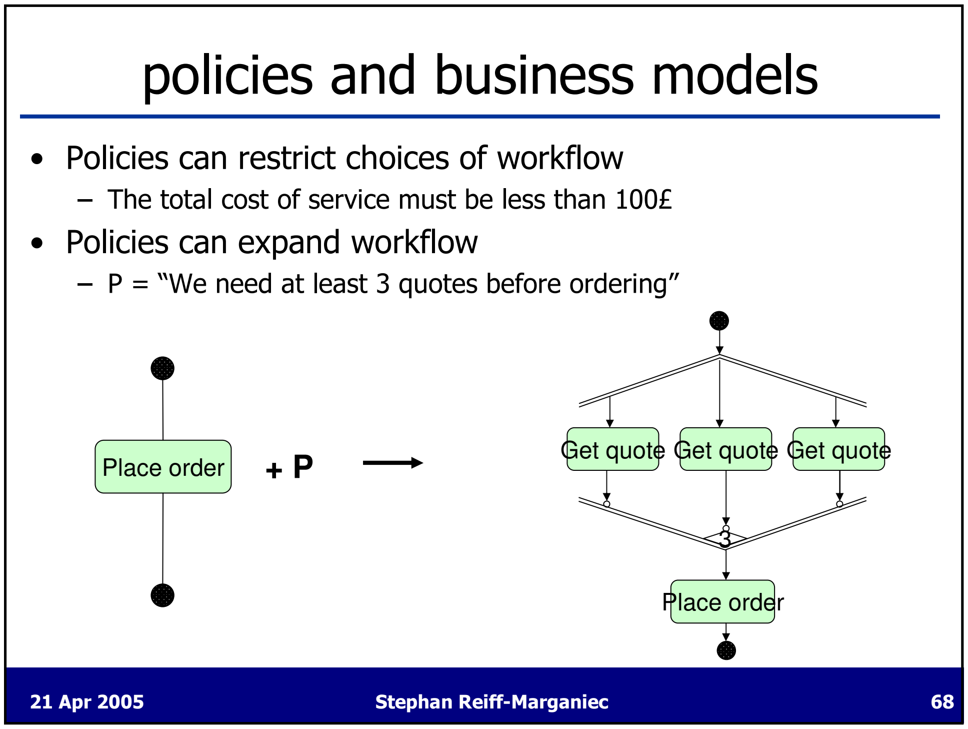


other notation

- Flow Junction Operator:
  - Left output is primary;
  - Output flow chosen according to a test;
  - Exclusive choice workflow pattern.
- Bounded cycles allowed:
  - For both composite and atomic tasks;
  - Can be modelled with flow junction and flow merge.
  - (since we only allow one control flow input, a flow merge function should be used).

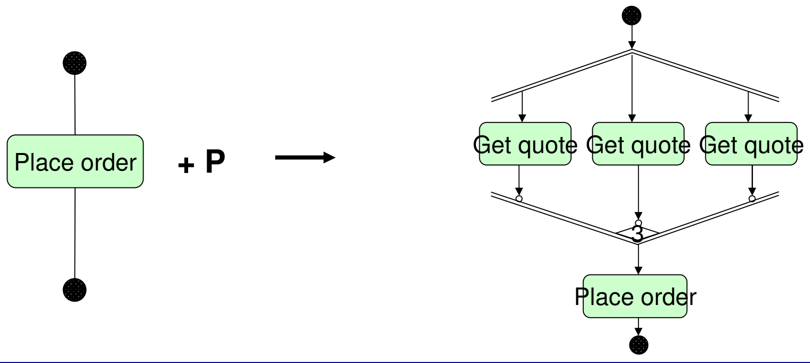


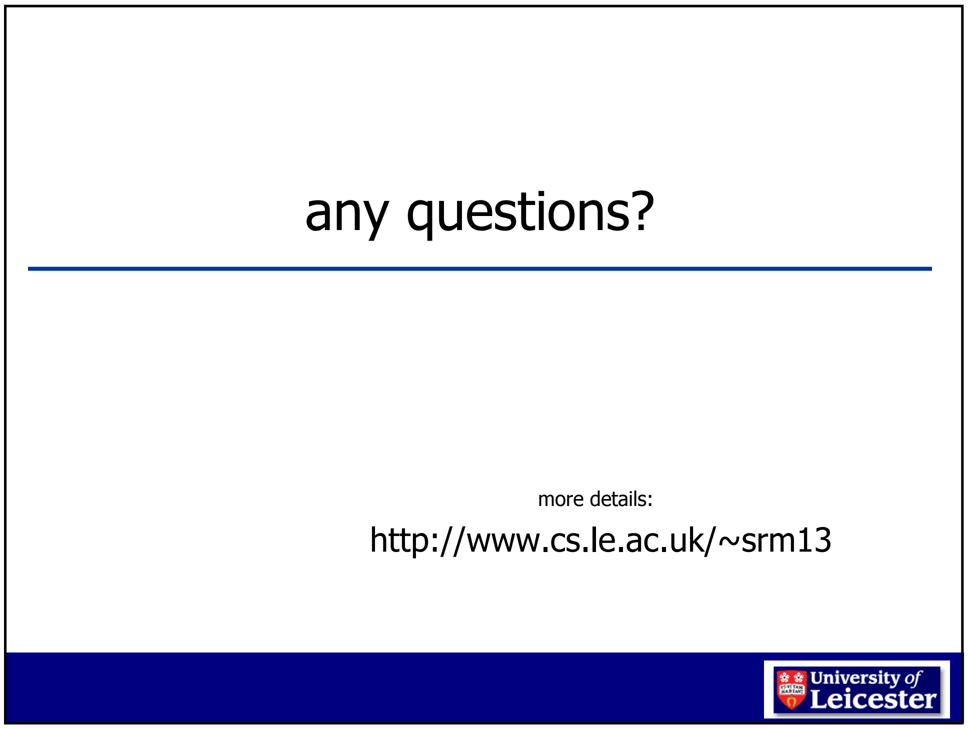




policies and business models

- Policies can restrict choices of workflow
  - The total cost of service must be less than 100£
- Policies can expand workflow
  - P = "We need at least 3 quotes before ordering"





any questions?

more details:

http://www.cs.le.ac.uk/~srm13

University *of* Leicester