Policies in Accountable Contracts:

Self-Enforcing Automatic Rational Contracts between Computers

Brian Shand
Jean Bacon

{Brian.Shand, Jean.Bacon}
@cl.cam.ac.uk
A Tool for Profitable Distributed Services

Contracts
- Negotiated web services, GRID computation, SLA’s

Trust
- Risk of untrustworthy participants

Resources
- Self-funding services and quality guarantees
# Policies in Accountable Contracts

<table>
<thead>
<tr>
<th>Components</th>
<th>Policies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contract architecture</td>
<td>Contract Selection</td>
</tr>
<tr>
<td>Resource-based virtual economy</td>
<td>Contract Performance</td>
</tr>
<tr>
<td>Automatic trustworthiness measurements</td>
<td>Contract Monitoring</td>
</tr>
</tbody>
</table>
Application Scenarios

- Public compute servers (GRID computation)
- Flexible prioritisation in large distributed systems
- Web services
- Publish-subscribe middleware – e.g. Active City
Contracts

• Non-repudiable promises to act

• Contract Definitions
  – A *contract* prescribes actions for each *participant*
  – By *signing* the contract, each participant promises to be bound by it
  – A contract is *agreed* upon when all participants have signed it
  – A contract is *cancelled* when everyone signs a special cancellation message

• But contracts need to be prioritised
Resource Economy

- Homogeneous model
  - money, trust, CPU, network, storage etc.
- Multi-scale representation
- Accounting of all resources

Contracts

{ Non-repudiable promises to act.
  Allow planning.

Resource Model

{ Introduces cost and risk.

Accounting Functions

{ Bind contracts and resources together.
  Can prioritise and monitor contracts.
Contracts and Accounting

• Multi-scale Contract Representation
  1. Server and client identities
  2. Expected resource requirements
  3. Accounting function (payment policy)
  4. Actions to perform

• Accounting Language
  1. Limited language: predictable execution time
  2. High-level description of resource exchange
  3. Allows introspectible contracts

Example: Accounting Functions

```python
def accountEg1(self, atom, imports):
    rate = 1.10*imports[0]
    return [ResourceAtom(
        resources.money,'£',
        rate*atom.quantity)]]

def accountEg2(self, atom, imports):
    if atom.type==resources.cpuTime:
        if imports[0] > 0.10:
            rate = imports[0]
        else: rate = 0.10
    return [ResourceAtom(
        resources.money,'£',
        rate*atom.quantity)]]
```
Accounting Policy

Complex code example:

```python
def processResourceAtom(self, atom, imports):
    if atom.type != resources.cpuTime:
        return [] # Charge for CPU only
    rate = imports[0]
    if self.totalCPU < 10: result = rate+0.01
    else:                  result = rate+0.002
    self.totalCPU += atom.quantity
    return [ResourceAtom(resources.money, '£',
                          result*atom.quantity) ]
```

- Accounting functions allow:
  - Contract policy specification
  - Contract prioritisation
  - Risk estimates with limited resource outlay
Trust Monitoring

- **Subjective trust model** (Jøsang, 2001)
  - Trust, distrust, uncertainty
  - Based on local assessments of trustworthiness
  - Constantly updated with new data

- **Trust delegation certificates**
  - Web of trust, c.f. PGP recommendations
  - **Distributed** trust management; standing surety
  - Subsume reputation agencies
Integrating the Three Strands

Contracts
- Non-repudiable promises to act.
- Allow planning.

Resource Model
- Basis for accounting.

Accounting Functions
- Bind contracts and resources.
- But: Need sanctions for cheats.

Trust Model
- Models expected contract payment.
- Constantly re-assessed.

Accountable Contracts
- Scheduled according to expected profitability.
- Contracts mediated by trust.
Contract Policy Trade-offs

Selection of New Contracts

- Consistency vs Flexibility
- Gambling for Profit

Performance of Existing Contracts

Monitoring Overheads

Profit Incentive
Conclusion

- Accountable contracts support **profit-based distributed services**
- **Separation** of task acceptance and performance
  - introspectible, automatic contracts
  - explicit risk assessments
- Allows **flexible policy specification**
- Further work:
  - Explicit integration of reputation into contracts
  - Theoretical basis for trust model
  - Developing prototype applications