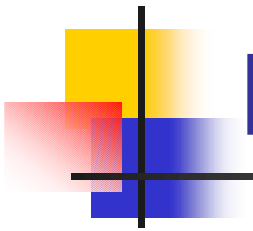
Policy-based SLA negotiation in FORM

Thanassis Tiropanis

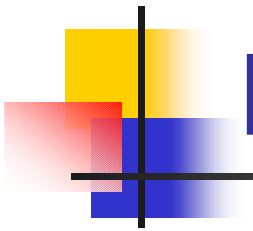
University College London, Computer Science

t.tiropanis@cs.ucl.ac.uk



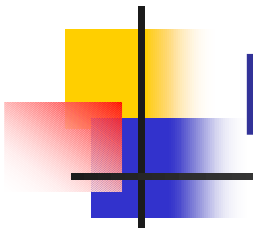
Background

- Imperial College work
  - Ponder language
- DMTF
- IETF
- IST project Tequila
- Other IST projects (CADENUS, AQUILA, etc)



Presentation outline

- Current research topics on Policies
- Use of policies in FORM
- Towards a Generic Policy Model (GPM)
- A Java/XML implementation of a GPM
  - *polML*
- Policy-based SLA negotiation
  - *slaML*
  - SLA negotiation engine
- Further work



Policy issues

- What is hardwired logic and what can be policy based (where is the line?)
- Policy Language, policy meta-model?
- O-O Policies?
- Hierarchical organisation of policies to high-level and low-level policies
- Mapping/transformation of high-level policies to low level ones and vice-versa

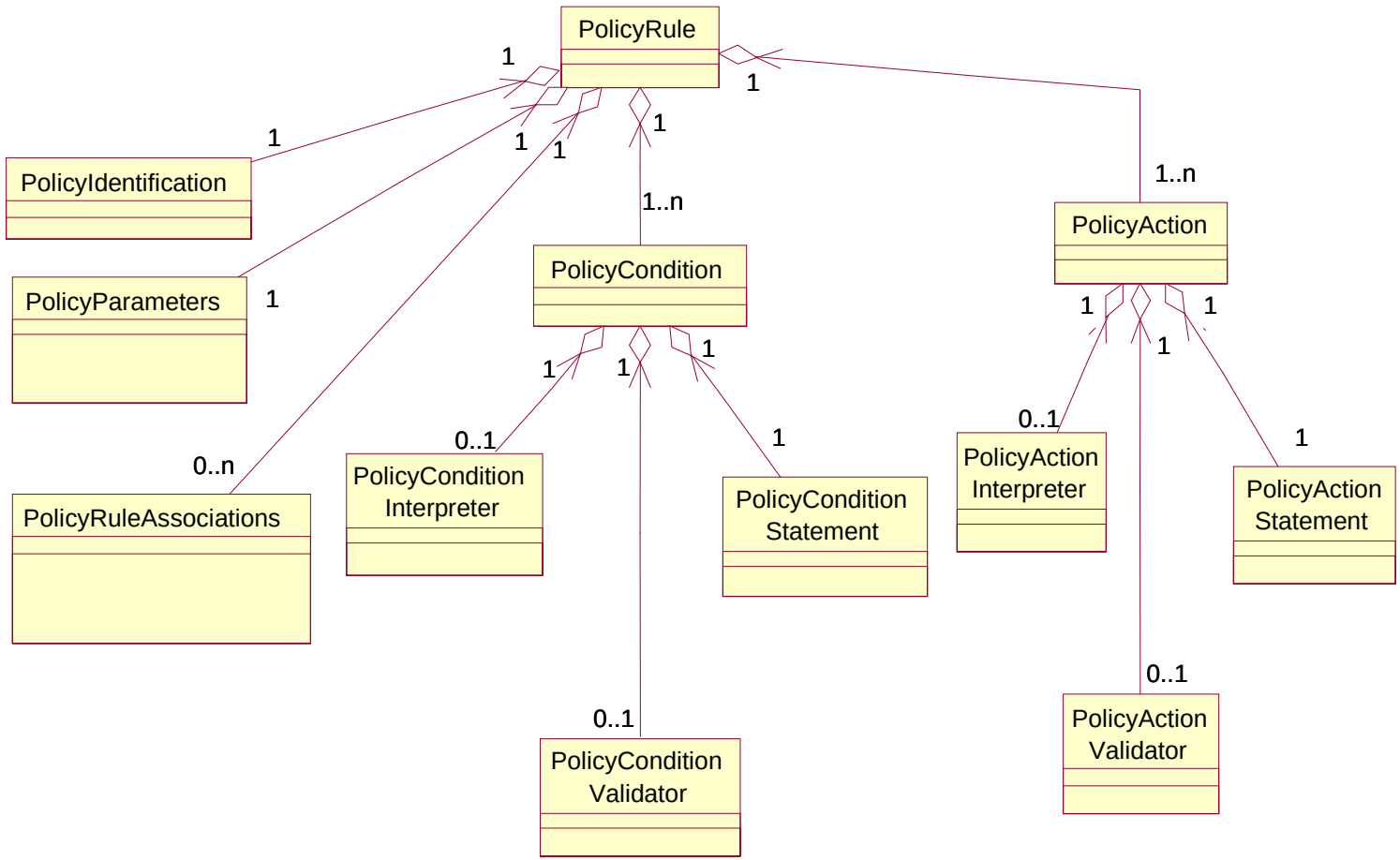
Policy language issues

- There are no generic policy languages at the moment
- There can be different policy languages depending on the application area.
- There can be higher-level policies that translate to lower-level policies which are used in the PEPs (transformation function).
- Without the transformation function policies could be useless.
- Events from lower-level policies can translate to actions on higher policy levels (reverse transformation function)
- Policy meta-language for policy specification

Policy-based SLAs can offer:

- Specification of the SLA negotiation process
- Flexible changes to the logic of the SLA negotiation process
  - When new service templates or service types are added
  - When new discount schemes are introduced
  - When certain combinations of SLA parameters do not work well together, or work better together

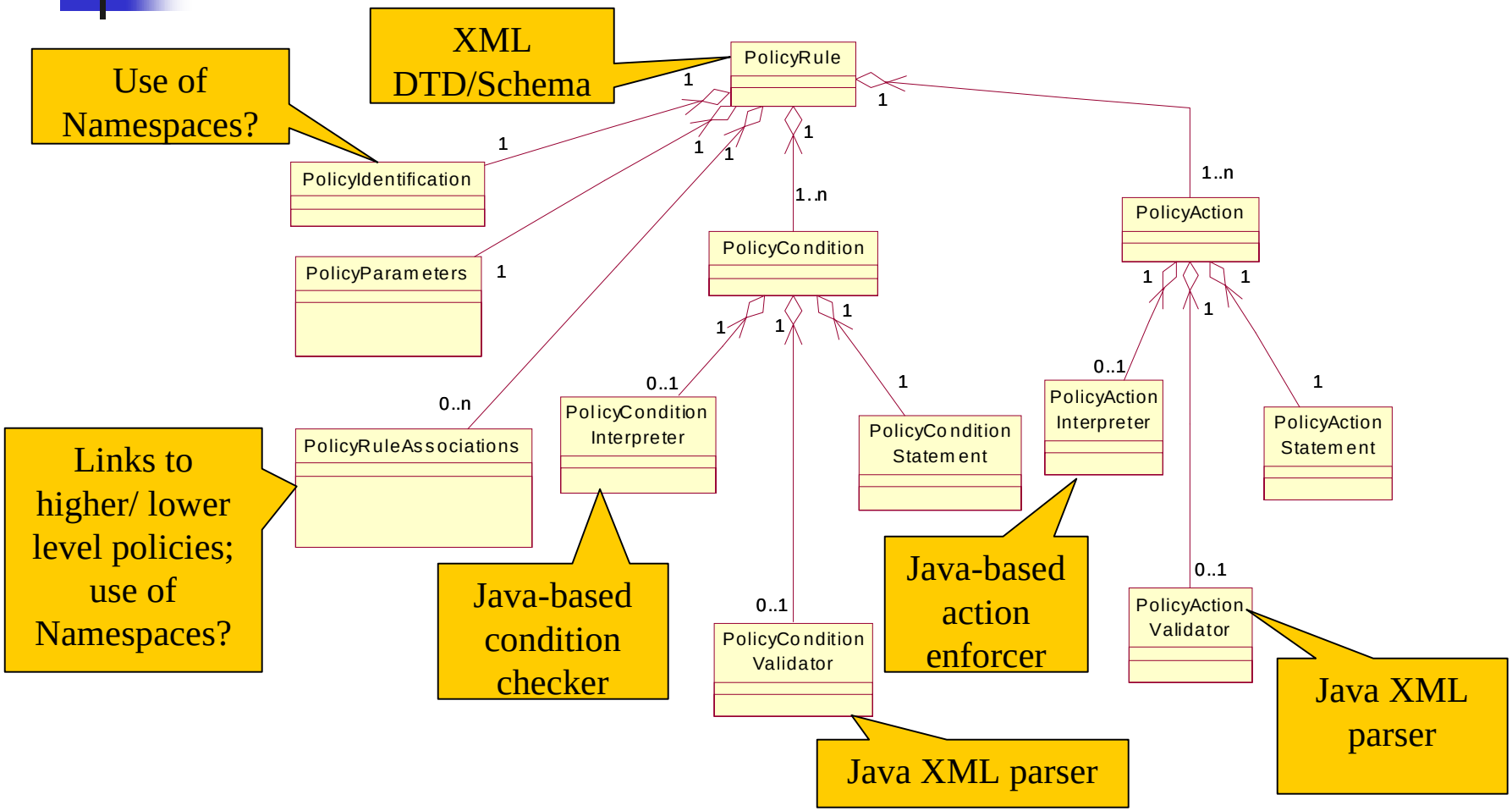
Generic Policy Model (GPM)



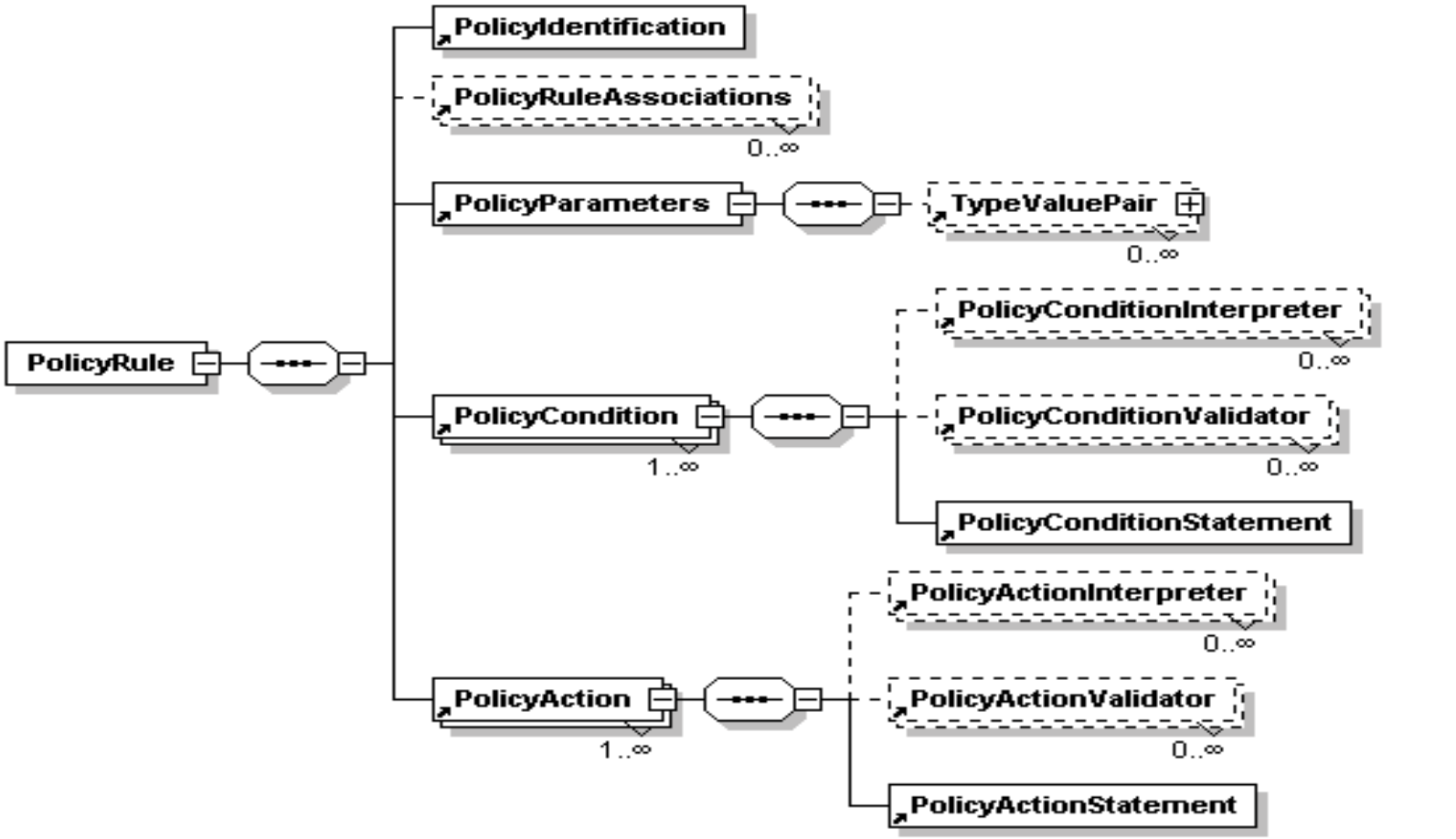
Advantages of the GPM

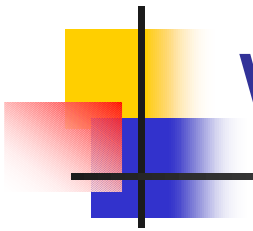
- Generic and policy syntax/language independent
- Can be easily adopted to new application areas
- Can allow for hierarchical policies
- Can assist policy transformation functions
- Can provide for the implementation of policy-independent PDP/PEPs.
- Compatible with IETF/DMTF policy models
- Specified in UML and in XML

A Java/XML implementation



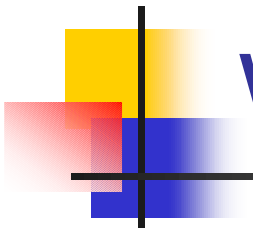
polML: Generic Policy Model DTD





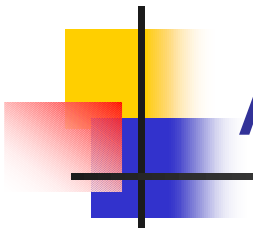
Why use XML

- XML for policies
  - Portable policies
  - Human readable policies
  - Interoperability with applications/databases
  - Definition of meta-policies as DTDs
  - Efficient policy processing, presentation and transformation (XPath, XSLT)
- XML for SLA negotiation
  - Efficient negotiation using a web browser
  - SLA in human-readable form



Why not use XML

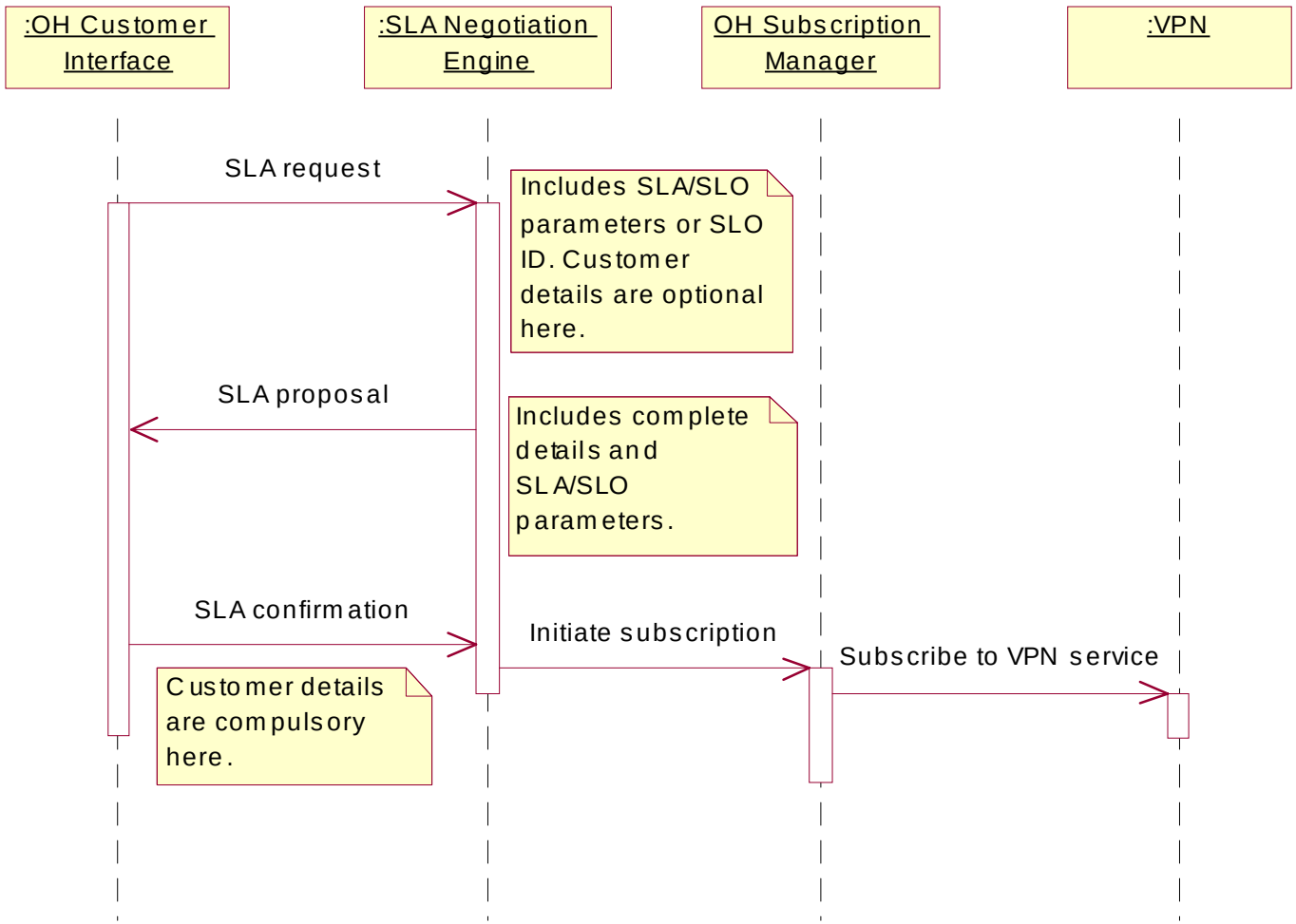
- XML for policies
  - No O-O
  - No built-in infrastructure for hierarchical policy organisation
  - Policy semantics open to interpretation
- XML for SLA negotiation
  - SLA parameter semantics open to interpretation



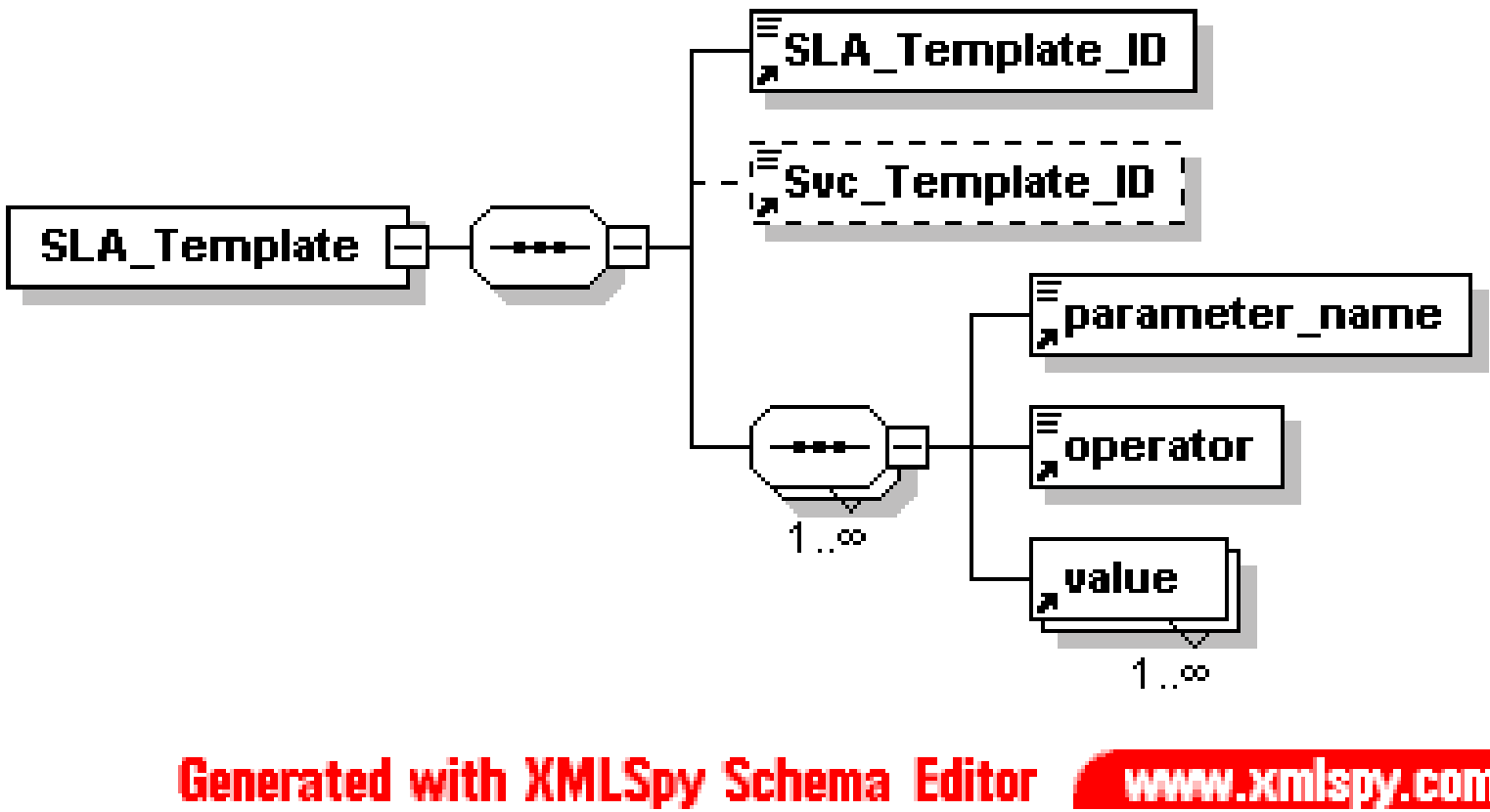
Alternatives to XML use

- For policy definition
  - Use SOX or other facilities to add O-O features?
  - Use another language for policies and offer a transformation function to XML?
- For SLA negotiation
  - SLA parameters are specified in another language; they are then parameterised as XML elements

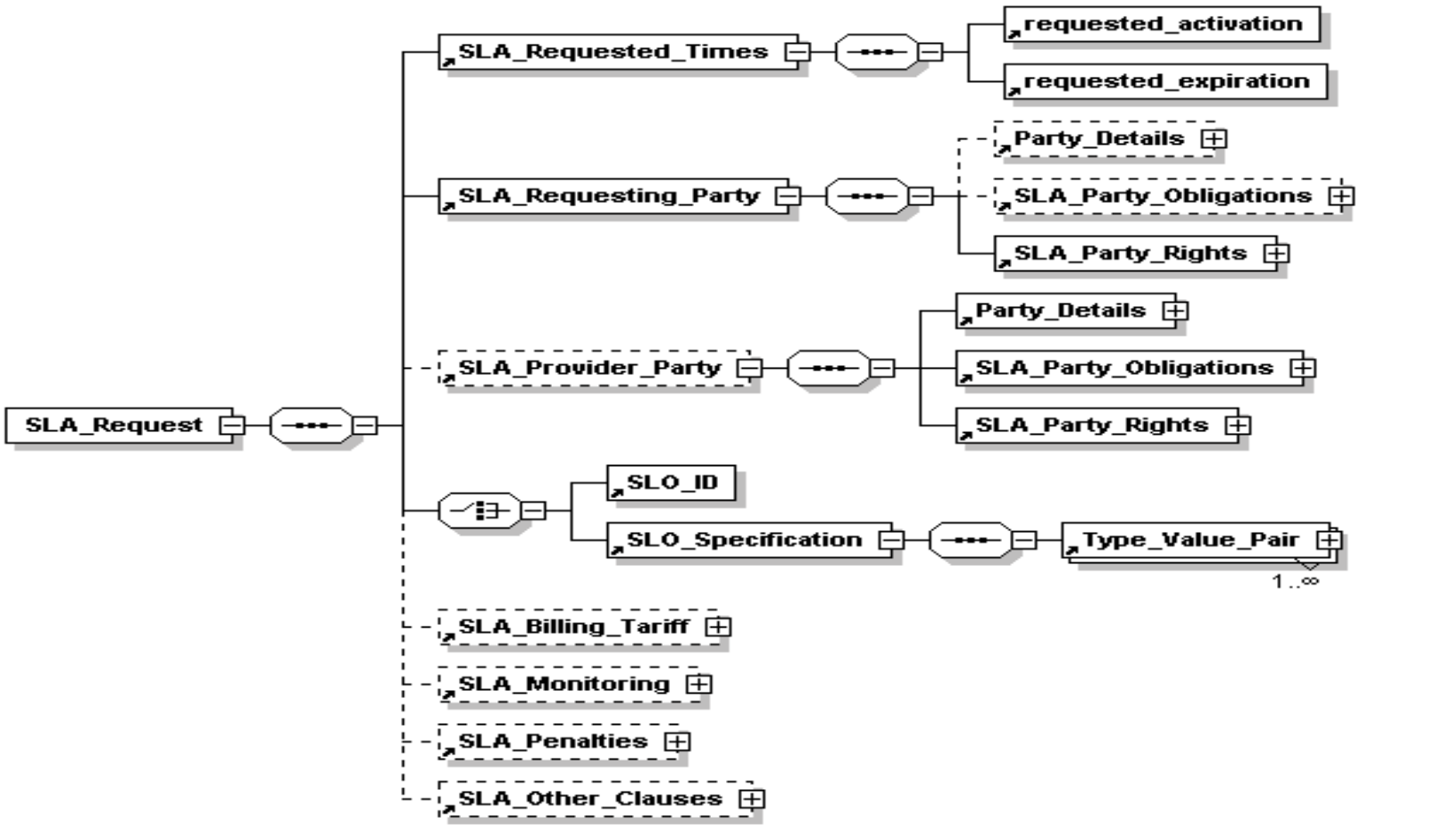
SLA negotiation process



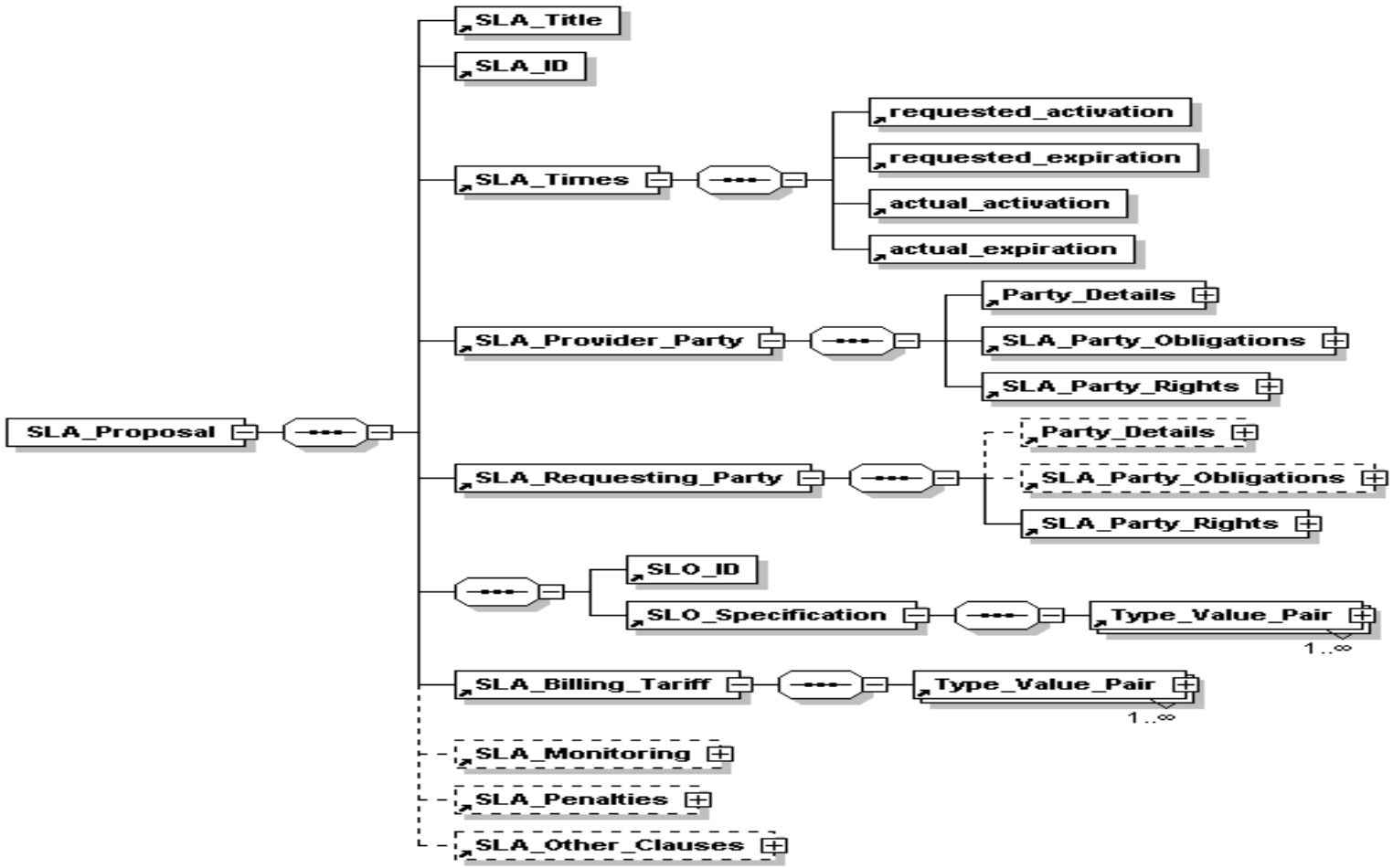
slaML: SLA template



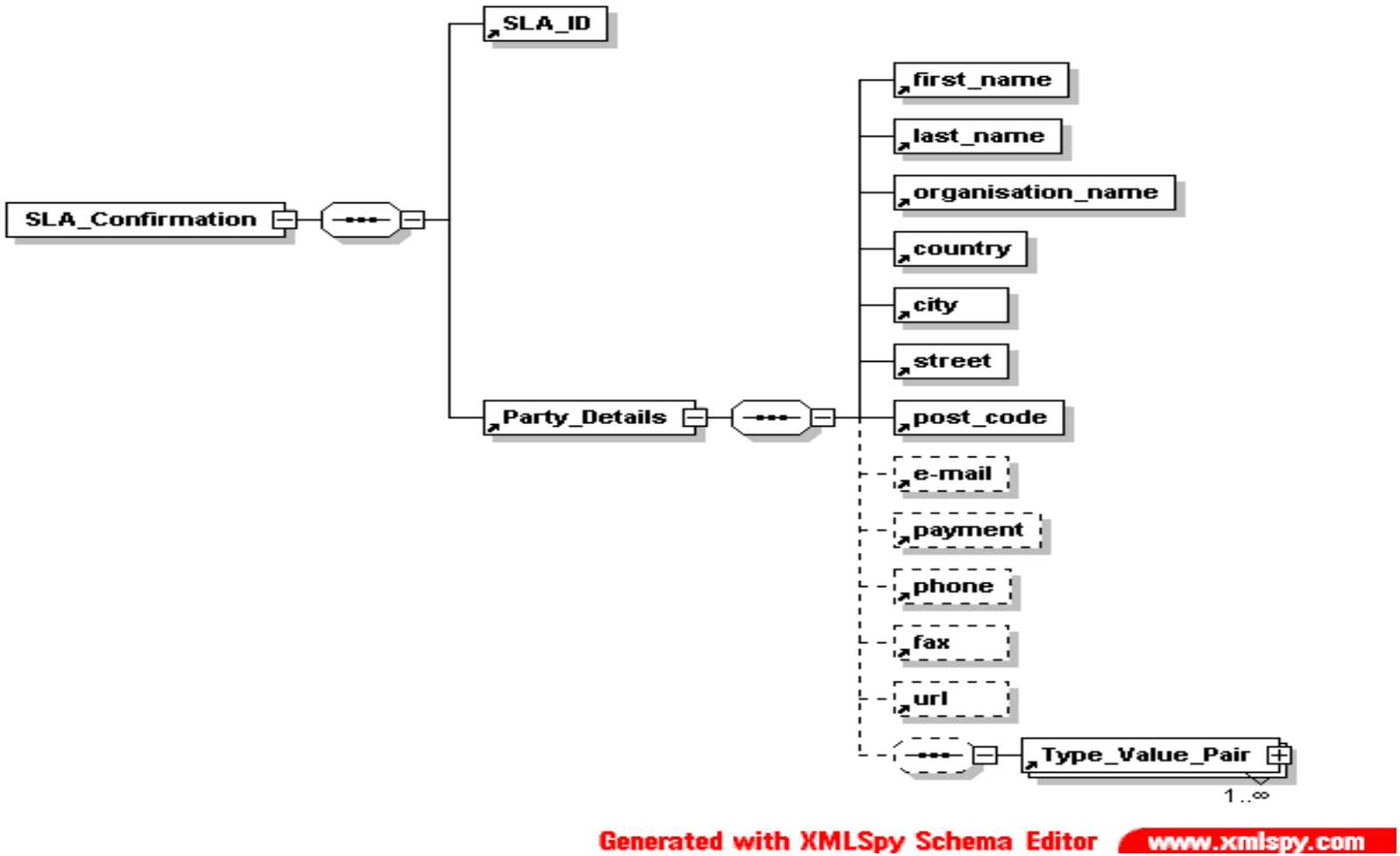
slaML: SLA negotiation request



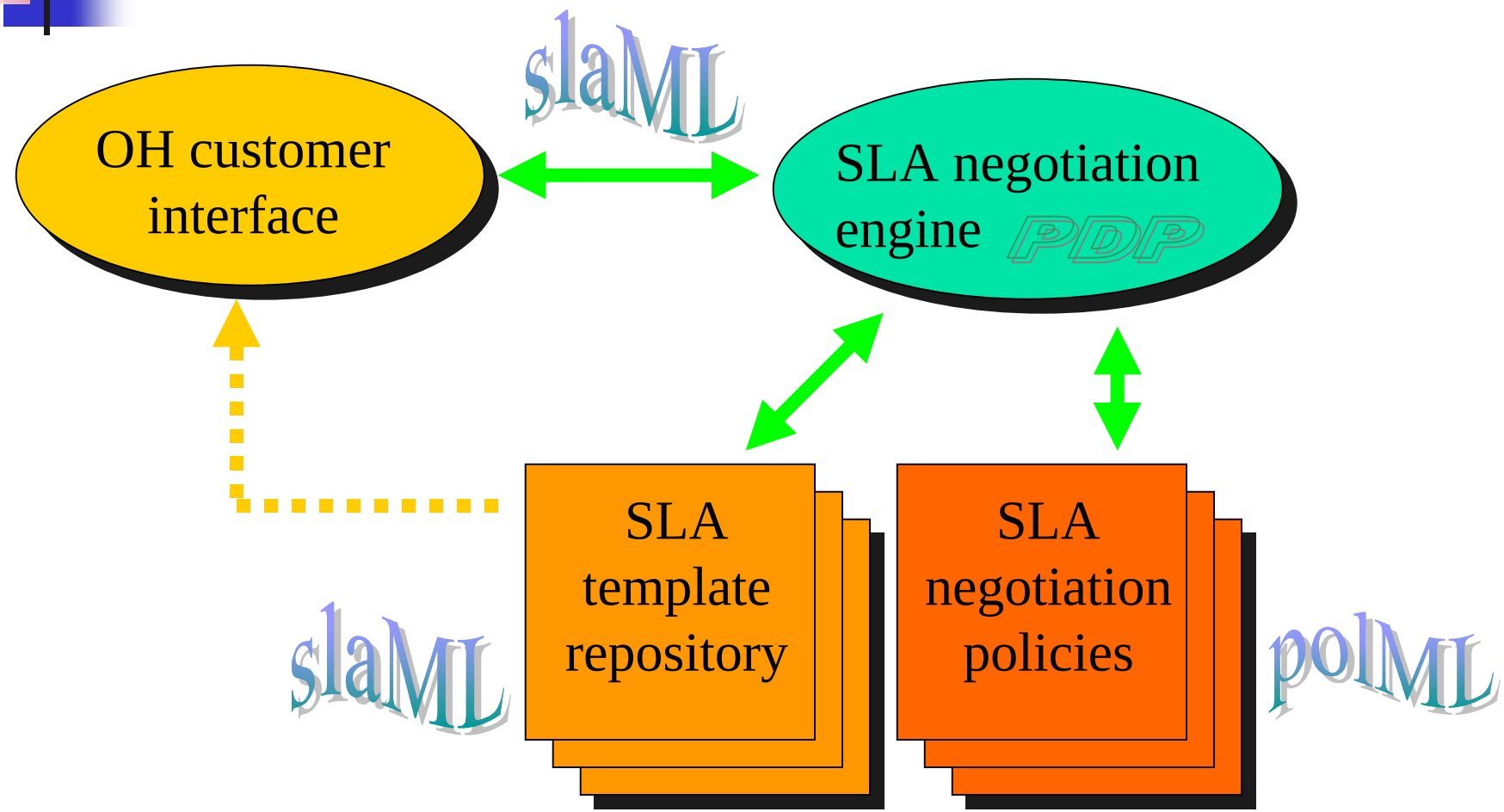
slaML: SLA proposal



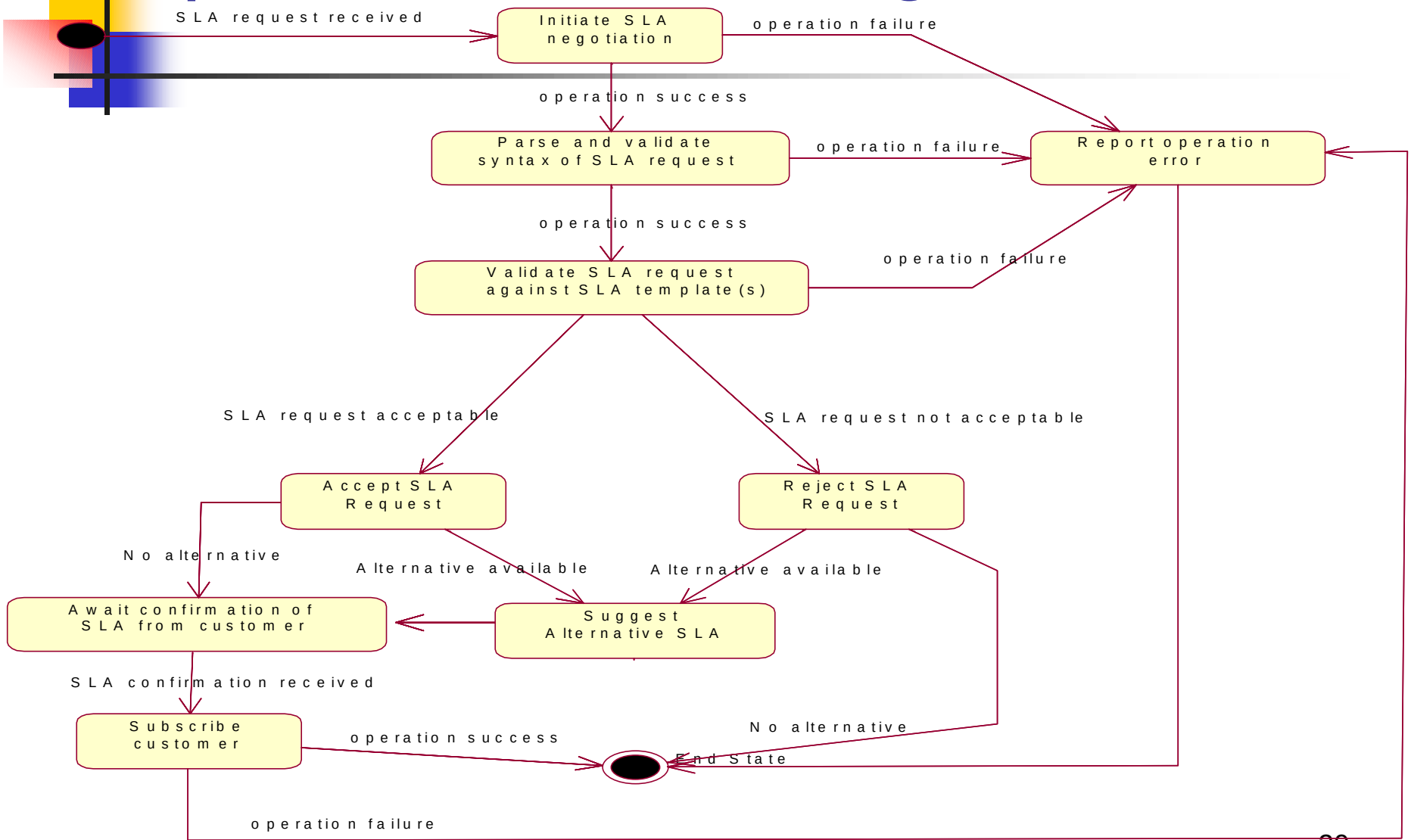
slaML: SLA confirmation

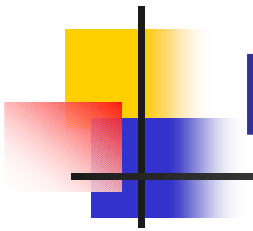


The SLA negotiation engine



Operation of the engine





Further work

- Work on a policy specification language and a policy meta-model
- Develop a SLA policy negotiation engine as a special case of a generic PDP/PEP engine that is based on the generic policy model
- Provide a policy transformation function for specific SLA policies
- Dissemination