SweetDeal: Representing Agent Contracts with Exceptions using XML Rules, Ontologies, and Process Descriptions

Presentation of Paper at the 12th International Conference on the World Wide Web (WWW-2003), Budapest, Hungary, May 22, 2003 (http://www.www2003.org)

Presentation by

Benjamin Grosof

MIT Sloan School of Management

bgrosof@mit.edu http://ebusiness.mit.edu/bgrosof

Joint work with Terrence Poon*

Oracle Corp. tpoon@alum.mit.edu

*Work done while a student at MIT

by Benjamin Grosof copyrights reserved

Outline

- SweetDeal approach of XML rule-based e-contracting ; RuleML
- Overview of New Aspects here:
  - Exception handling in contract → invoke business processes
  - Combine rules with ontologies, cf. Semantic Web vision
  - New prototype; with application Scenario: late delivery in SCM
  - Process ontologies in DAML+OIL formalize MIT Process Handbook content
- Example Scenario: A Whirlwind Tour
  - Rules specify how to detect, avoid, or resolve an exception
  - Modular updating for modifications during negotiation, via prioritized defaults
  - E.g.: late-delivery provisions, counter-proposing
- Conclusions incl. More new Contributions
- Current Work and Future Directions
  - Semantic Web Services

by Benjamin Grosof copyrights reserved

Overall SweetDeal Approach to E-Contracting

- Represent contracts using interoperable rules in XML
  - Declarative logic programs (LP) as knowledge representation (KR)
    - Spawned RuleML: via predecessor BRML in IBM CommonRules
    - Underlying requirements analysis for rule KR and syntax
- Contracts may be: partial or complete, proposed or final
- Rules represent part or all of the contract ; named importable rule modules
- Modular modification during negotiation or completion
    - E.g., add new contract provisions in a counter-proposal
    - E.g., add price, quantity, buyer after auction is completed
- …Using prioritized conflict handling in the rules representation
    - Courteous LP – extension, tractably compileable into Ordinary LP
- Use of procedural attachments for built-ins, external actions or queries
    - Situated LP – extension, provides declarative abstraction
    - E.g., invoke surrounding external procedural business processes
- Thus, overall use the Situated Courteous LP (SCLP) KR in RuleML
- *(Also need "solo" decision making/support by each agent)*

by Benjamin Grosof copyrights reserved

Previous SweetDeal Approach (continued)

- Previous papers:
  - [ACM E-Commerce Conf. 1999]
    - general approach; rules KR, IBM CommonRules; EECOMS
  - [Computational Intelligence 2003]
    - auctions negotiation and configuration, ContractBot
- Previous prototypes with application pilots:
  - EECOMS manufacturing supply chain collaboration: negotiation
    - $29Million NIST ATP
      - by industry consortium (IBM, Boeing, TRW, Baan, Vitria, others)
    - First pilot of IBM CommonRules
  - ContractBot extending U. Michigan's AuctionBot auction server

by Benjamin Grosof copyrights reserved

Advantages of SweetDeal's Rule-based approach, based on Situated Courteous RuleML, to E-Contracting

- Rules (vs. general code) provide high level of conceptual abstraction
  - easier for non-programmers to understand or to specify
- Esp. good for specifying contingent provisions
- Reason about the contract/proposal – tractably, automatically infer
  - hypotheticals ("what-if"), testing, evaluating value
- Communicate: with deep shared semantics (cf. "Semantic Web" vision)
  - via RuleML, inter-operable with same sanctioned inferences
  - ⇔ heterogeneous rule/DB systems / rule-based applications ("agents")
- Executable: can actually execute the contract provisions:
  - infer; ebiz actions via Situated procedural attachments
- Modularly modifiable: can modify easily the contract provisions
  - prioritized defaults and rule modules via Courteous overriding
- Reusability: from KR declarativeness, modularity, interoperability

by Benjamin Grosof copyrights reserved

Examples of Contract Provisions Well-Represented by Rules in Automated Deal Making

- Product descriptions
  - Product catalogs: properties, conditional on other properties.
- Pricing dependent upon: delivery-date, quantity, group memberships, umbrella contract provisions
- Terms & conditions: refund/cancellation timelines/deposits, lateness/quality penalties, ordering lead time, shipping, creditworthiness, biz-partner qualification, service provisions
- Trust
  - Creditworthiness, authorization, required signatures
- *Buyer Requirements (RFQ, RFP) wrt the above*
- *Seller Capabilities (Sourcing, Qualification) wrt the above*

by Benjamin Grosof copyrights reserved

New Extensions to General Approach of SweetDeal

1. Rules represent exception handling contract provisions, that
   - can reference or invoke: associated business processes

2. Combine rules with ontologies
   - Rules can reference Semantic Web ontologies in Description Logic (DL) KR
     - … in DAML+OIL currently (work began when OWL was immature)
   - cf. Semantic Web vision of rules "on top of" ontologies

3. ... E.g., where those ontologies are about the business processes

4. These business processes may themselves be partly or completely specified via rules (executably)

by Benjamin Grosof copyrights reserved

New Prototype and Scenario for SweetDeal

1. New Prototype: SweetDeal (at MIT Sloan) [briefly described in paper]
   - contract authoring, communication, inferencing/execution
   - uses SweetRules toolset for Situated Courteous RuleML
   - queryable repositories for:
     - contract rule modules
     - process ontologies
   - market agents (communicate, negotiate, evaluate, etc. the contracts)

2. New Pilot Example Application Scenario (i.e., Use Case)
   - Exceptions in manufacturing SCM (SCM = Supply Chain Management)
   - Business process ontologies content partly drawn from MIT Process Handbook

by Benjamin Grosof copyrights reserved

Exception Handling ; Services angle

- Exception: something that doesn't go as is normal or expected/usual
  - One important category is: violation of a (contract) commitment
- Exception handler:
  - a business (sub-)process
  - (here:) specified as part of a contract

- Most of the volume of many existing contracts and business process specifications is devoted to exception handling
  - "Murphy's Law", "Stuff Happens", "The course of true love never did run smooth"

- Complex behavior that must be represented
  - Particularly vital for contracts about services
  - E.g., deals about Web Services

by Benjamin Grosof copyrights reserved

Using: MIT Process Handbook

- Our example scenario's process ontologies are drawn partly from the MIT Process Handbook (PH) …
- A previously-existing repository of business process descriptions
- Uniquely large & well-used (by industry biz process designers) [Malone *et al* '95-]
- Includes taxonomic/hierarchical aspects, as a fraction
- Includes exception handling ontology [Klein *et al* 2002]

- New here:
  - formalize PH knowledge in XML Description Logic: DAML+OIL
    - (only a small fraction of its content, so far)
  - enables practical deep inferencing with the PH knowledge
    - … using Semantic Web tools (RuleML/LP and DAML+OIL/OWL/DL)

- Previously PH content was only shallowly automated for inferencing
  - Was NOT represented in Description Logic KR nor in XML (not Webized)
  - (there was a partial PIF encoding, mapping to KIF)

by Benjamin Grosof copyrights reserved

Summary of the Example/Scenario in the Paper

- Buyer goes shopping (a.k.a. procurement)
- Seller sends a proposed contract
- Buyer adds exception handling provision for late delivery penalty
- Seller adds replacement provision for late delivery risk premium
  - Illustrates: Exception handling can itself be subject of negotiation
- RuleML rules refer to DAML+OIL process ontologies, partly PH content

- Can do "what-if" inferencing: e.g., in buyer's version of contract:
- add facts about hypothetical delivery date in contract result
- $\Rightarrow$ delivery is late,
- $\Rightarrow$ which is an exception,
- $\Rightarrow$ a late delivery penalty payment is owed

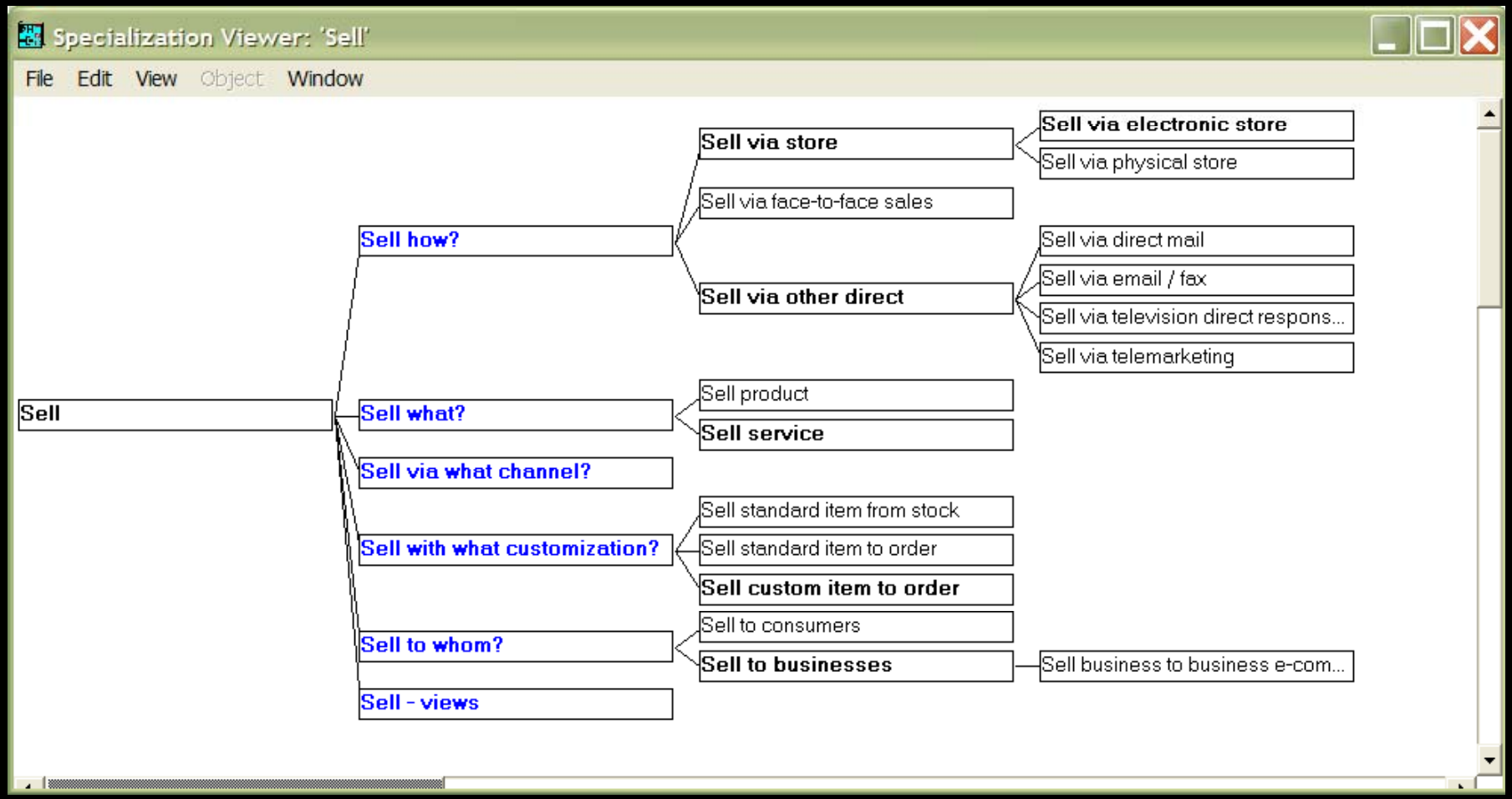
- Can execute aspects of the contract via inferencing
  - E.g., from facts of actual contract result $\Rightarrow$ determine net payment owed

by Benjamin Grosof copyrights reserved

NEXT…A WHIRLWIND TOUR of the Application Scenario described in the Paper

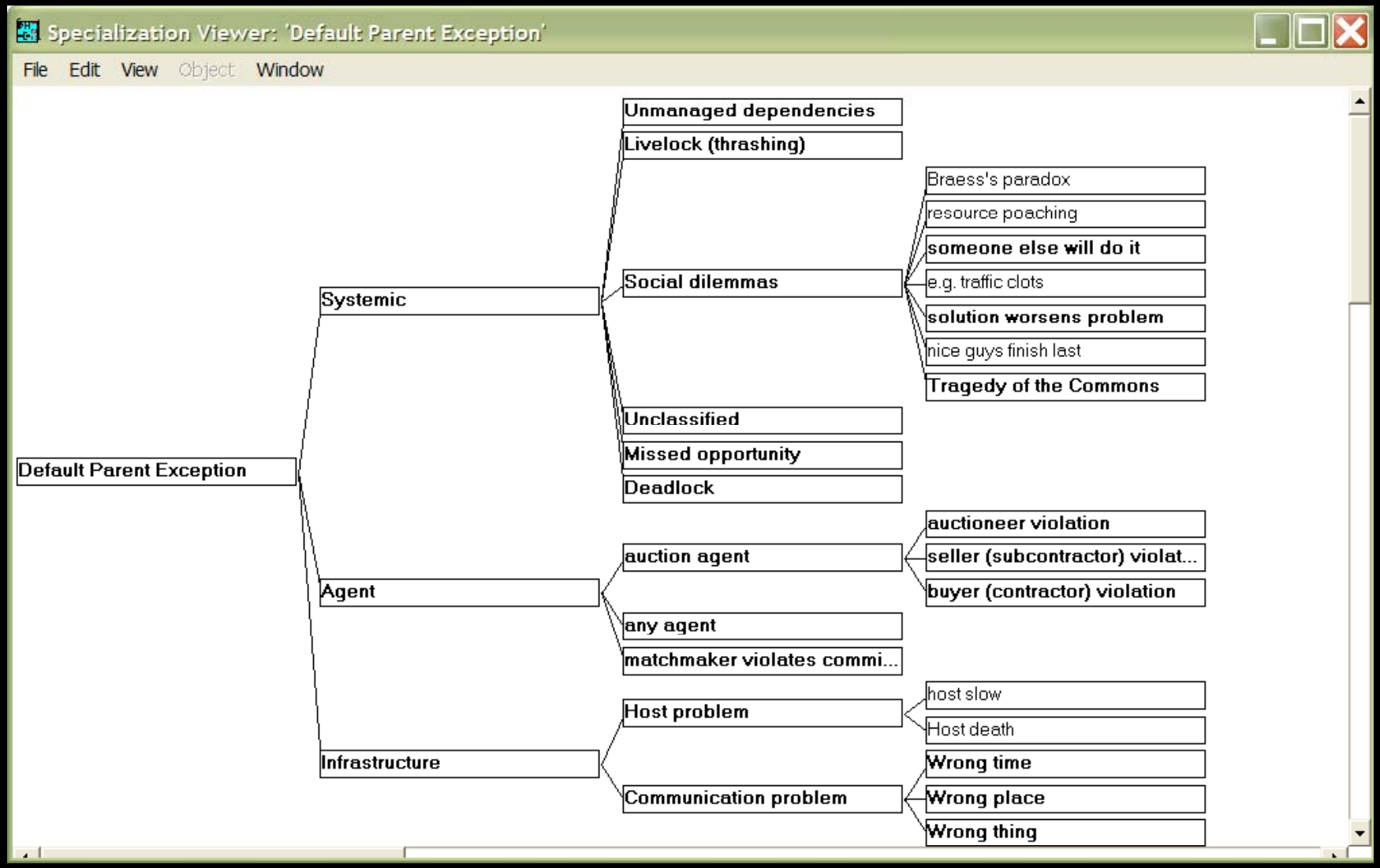
by Benjamin Grosof copyrights reserved

Some Specializations of "Sell" in the MIT Process Handbook (PH)



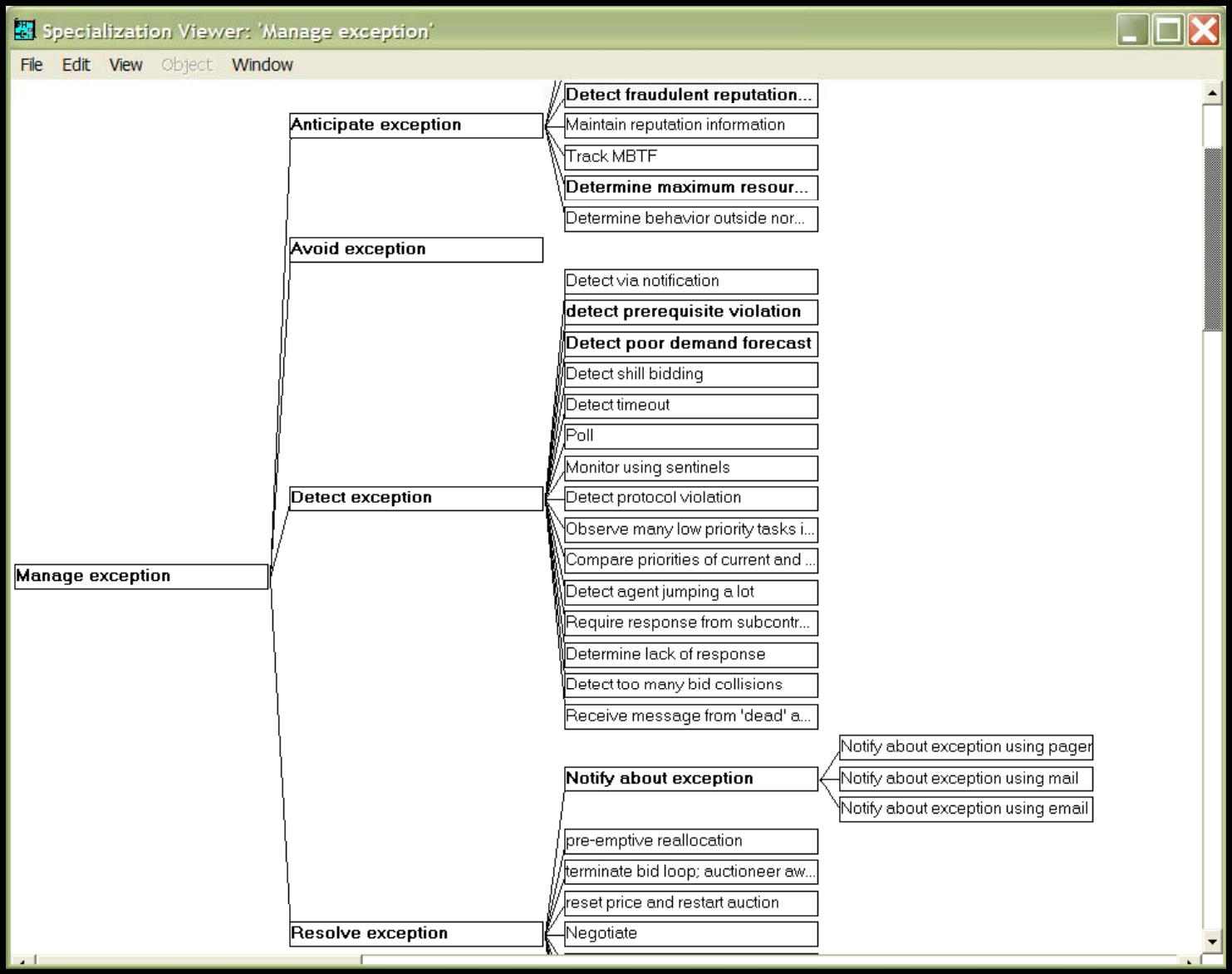
by Benjamin Grosof copyrights reserved

Some Exceptions in the MIT Process Handbook



by Benjamin Grosof copyrights reserved

Some exception handlers in the MIT Process Handbook



by Benjamin Grosof copyrights reserved

Representing PH Process Ontology in DAML+OIL: Some Main Concepts

```
<daml:Class rdf:ID="Process">
  <rdfs:comment>A process</rdfs:comment>
</daml:Class>

<daml:Class rdf:ID="CoordinationMechanism">
  <rdfs:comment>A process that manages activities between multiple
agents</rdfs:comment>
</daml:Class>

<daml:Class rdf:ID="Exception">
  <rdfs:comment>A violation of an inter-agent commitment</rdfs:comment>
</daml:Class>

<daml:Class rdf:ID="ExceptionHandler">
  <rdfs:subClassOf rdf:resource="#Process"/>
  <rdfs:comment>A process that helps to manage a particular
exception</rdfs:comment>
</daml:Class>
```

Define pr.daml

by Benjamin Grosof copyrights reserved

Representing PH Process Ontology in DAML+OIL: More

```
<daml:ObjectProperty rdf:ID="hasException">
  <rdfs:comment>Has a potential exception</rdfs:comment>
  <rdfs:domain rdf:resource="#Process" />
  <rdfs:range rdf:resource="#Exception" />
</daml:ObjectProperty>

<daml:ObjectProperty rdf:ID="isHandledBy">
  <rdfs:comment>Can potentially be handled by, in some way </rdfs:comment>
  <rdfs:domain rdf:resource="#Exception" />
  <rdfs:range rdf:resource="#ExceptionHandler" />
</daml:ObjectProperty>

...
<daml:Class rdf:ID="ContractorDoesNotPay">
  <rdfs:subClassOf rdf:resource="#ContractorViolation"/>
  <rdfs:subClassOf>
   <daml:Restriction>
      <daml:onProperty rdf:resource="#isHandledBy"/>
      <daml:hasClass rdf:resource="#ProvideSafeExchangeProtocols"/>
    </daml:Restriction>
  </rdfs:subClassOf>
</daml:Class>
```


by Benjamin Grosof copyrights reserved

Representing New Contract Ontology in DAML+OIL

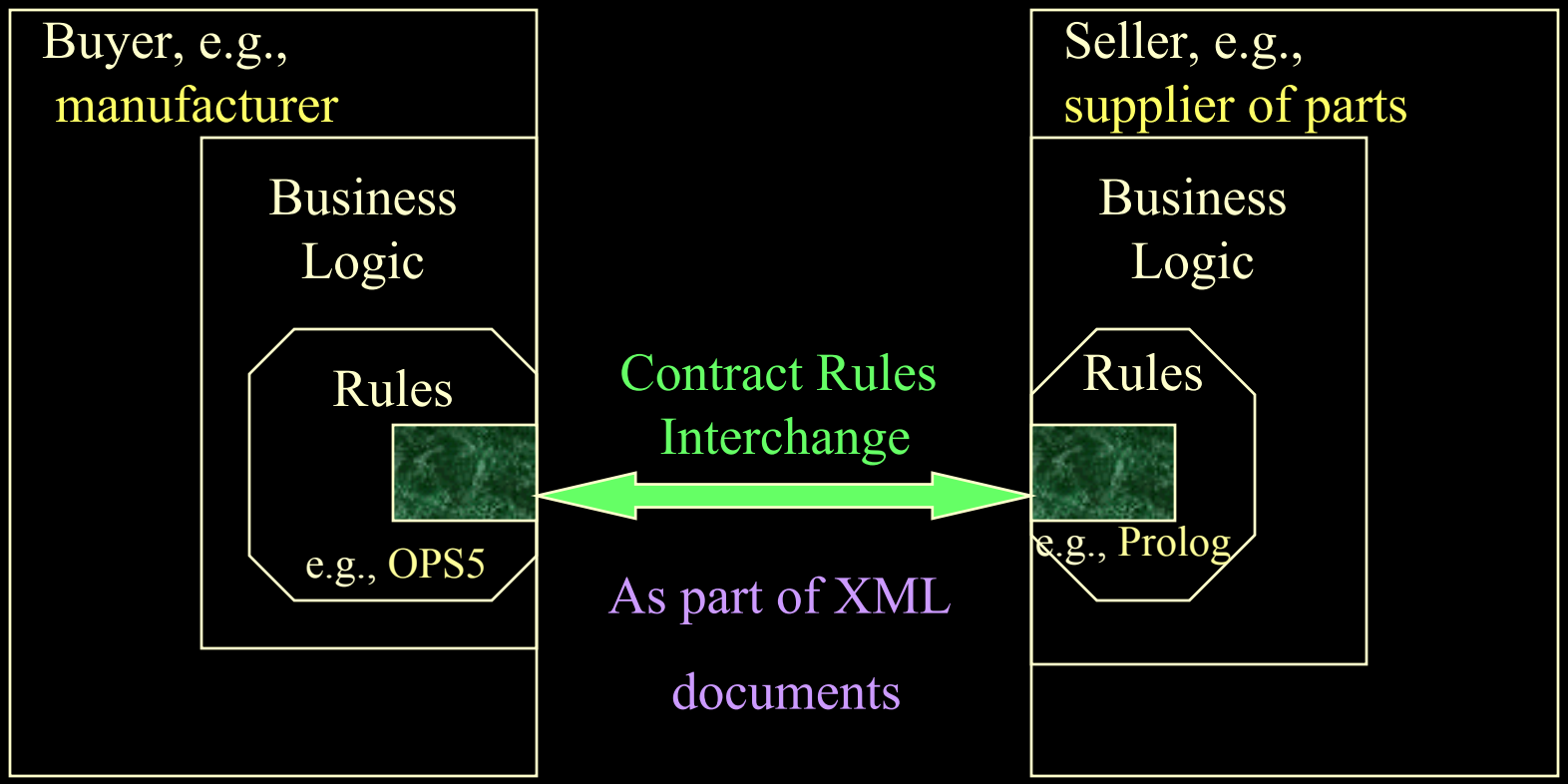
```
<daml:Class rdf:ID="Contract">
  <rdfs:subClassOf>
    <daml:Restriction daml:minCardinality="1">
      <daml:onProperty rdf:resource="#specFor"/>
    </daml:Restriction>
  </rdfs:subClassOf>
</daml:Class>

<daml:ObjectProperty rdf:ID="specFor">
  <rdfs:domain rdf:resource="#Contract" />
  <rdfs:range rdf:resource="http://xmlcontracting.org/pr.daml#Process"/>
</daml:ObjectProperty>

<daml:Class rdf:ID="ContractResult"/>

<daml:ObjectProperty rdf:ID="result">
  <rdfs:domain rdf:resource="#Contract" />
  <rdfs:range rdf:resource="#ContractResult" />
</daml:ObjectProperty>
```

Define sd.daml

(imports pr.daml)

by Benjamin Grosof copyrights reserved

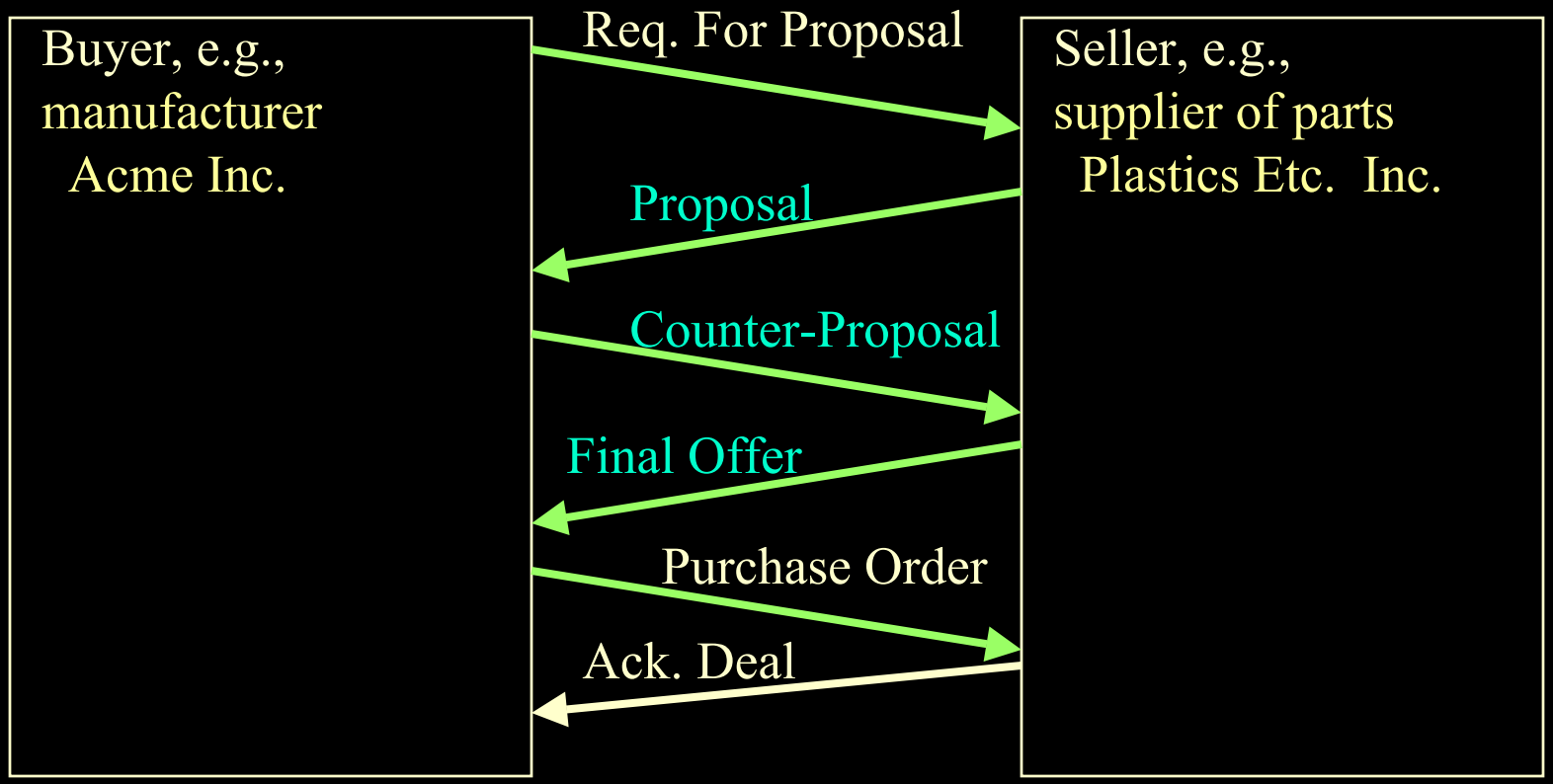
Contract Rules during Negotiation



Contracting parties NEGOTIATE via shared rules.

by Benjamin Grosof copyrights reserved

Exchange of Rules Content during Negotiation: example



by Benjamin Grosof copyrights reserved

Example Contract Payment Rule in RuleML, referring to DAML+OIL ontology

```
<imp>
  <_head> <atom> <_opr>
    <rel> payment </rel> <_opr>
      <tup> <var> R </var> <ind> base </ind> <var> Amount </var> </tup>
  </atom> </_head>
  <_body> <andb> <atom> <_opr>
    <rel href= "http://xmlcontracting.org/sd.daml#result"/>
        </_opr>
          <tup> <ind> co123 </ind> <var> R </var>
         </tup> </atom>

          ...

   </andb>  </_body>  </imp>
```

SCLP TextFile Format for RuleML

```
// comment:  base payment = price * quantity
payment(?R,base,?Amount) <-
http://xmlcontracting.org/sd.daml#result(co123,?R) AND
price(co123,?P) AND quantity(co123,?Q) AND
multiply(?P,?Q,?Amount) ;
```


by Benjamin Grosof copyrights reserved

Example Contract Proposal with Exception Handling

Represented using RuleML & DAML+OIL, Process Descriptions

Using concise text syntax

(SCLP textfile format)

for human reading

```
buyer(co123,acme);
seller(co123,plastics_etc);
product(co123,plastic425);
price(co123,50);
quantity(co123,100);
http://xmlcontracting.org/sd.daml#Contract(co123);
http://xmlcontracting.org/sd.daml#specFor(co123,co123_process);
http://xmlcontracting.org/sd.daml#BuyWithBilateralNegotiation(co123_process);
http://xmlcontracting.org/sd.daml#result(co123,co123_res);
shippingDate(co123,3); // i.e. 3 days after order placed
// base payment = price * quantity
payment(?R,base,?Amount) <-
  http://xmlcontracting.org/sd.daml#result(co123,?R) AND
  price(co123,?P) AND quantity(co123,?Q) AND
  multiply(?P,?Q,?Amount) ;
```


by Benjamin Grosof copyrights reserved

Example Contract Proposal, Continued

- Buyer adds rule modules to the contract proposal to specify:
  - 1. detection of an exception
    - LateDelivery as a potential exception of the contract's process
    - detectLateDelivery as exception handler: recognize occurrence
  - 2. avoidance of an exception (and perhaps also resolution of the exception)
    - lateDeliveryPenalty as exception handler: penalize per day

- Rule module = a nameable ruleset → a subset of overall rulebase
  - can be included directly and/or imported via link; nestable
    - similar to legal contracts' "incorporation by reference"
  - an extension to RuleML; in spirit of "Webizing"

by Benjamin Grosof copyrights reserved

Example Contract Proposal, Continued: lateDeliveryPenalty exception handler module

```
lateDeliveryPenalty_module {
// lateDeliveryPenalty is an instance of PenalizeForContingency
//   (and thus of AvoidException, ExceptionHandler, and Process)
http://xmlcontracting.org/pr.daml#PenalizeForContingency(lateDeliveryPenalty) ;
// lateDeliveryPenalty is intended to avoid exceptions of class
// LateDelivery.
http://xmlcontracting.org/sd.daml#avoidsException(lateDeliveryPenalty,
  http://xmlcontracting.org/pr.daml#LateDelivery);
// penalty = - overdueDays * 200 ; (negative payment by buyer)
<lateDeliveryPenalty_def> payment(?R, contingentPenalty, ?Penalty) <-
  http://xmlcontracting.org/sd.daml#specFor(?CO,?PI) AND
  http://xmlcontracting.org/pr.daml#hasException(?PI,?EI) AND
  http://xmlcontracting.org/pr.daml#isHandledBy(?EI,lateDeliveryPenalty) AND
  http://xmlcontracting.org/sd.daml#result(?CO,?R) AND
  http://xmlcontracting.org/sd.daml#exceptionOccurred(?R,?EI) AND
  shippingDate(?CO,?CODate) AND shippingDate(?R,?RDate) AND
  subtract(?RDate,?CODate,?OverdueDays) AND
  multiply(?OverdueDays, 200, ?Res1) AND multiply(?Res1, -1, ?Penalty) ;
}
<lateDeliveryPenaltyHandlesIt(e1)> // specify lateDeliveryPenalty as a handler for e1
 http://xmlcontracting.org/pr.daml#isHandledBy(e1,lateDeliveryPenalty);
```


by Benjamin Grosof copyrights reserved

Example, Continued: Counter-Proposal

- Seller modifies the draft contract (it's a *negotiation*!)
- Simply adds* another rule module to specify:
  - lateDeliveryRiskPremium as exception handler
    - lump-sum in advance, based on average lateness
      - instead of proportional to actual lateness
  - higher-priority for that module than for the previous proposal, e.g., higher than lateDeliveryPenalty's rule module

- Courteous LP's prioritized conflict handling feature is used
- *NO *change* to previous proposal's rules needed!
  - similar to legal contracts' accumulation of provisions

by Benjamin Grosof copyrights reserved

Example Counter-Proposal's ruleset's prioritized conflict handling

```
// priority specified via syntactically reserved "overrides" predicate

overrides(lateDeliveryRiskPaymentHandlesIt(e1),
                    lateDeliveryPenaltyHandlesIt(e1) ) ;


// There is at most one avoid handler for a given exception instance.
// Consistency is enforced wrt this "mutex" integrity constraint.

  MUTEX
    http://xmlcontracting.org/pr.daml#isHandledBy(?EI, ?EHandler1) AND
    http://xmlcontracting.org/pr.daml#isHandledBy(?EI, ?Ehandler2)
   GIVEN
    http://xmlcontracting.org/sd.daml#AvoidException(?Ehandler1) AND
    http://xmlcontracting.org/sd.daml#AvoidException(?Ehandler2) AND
    notEquals(?Ehandler1, ?Ehandler2);
```


by Benjamin Grosof copyrights reserved

END OF… A WHIRLWIND TOUR of the Application Scenario described in the Paper

by Benjamin Grosof copyrights reserved

Summary of the Example/Scenario in the Paper

- Buyer goes shopping (a.k.a. procurement)
- Seller sends a proposed contract
- Buyer adds exception handling provision for late delivery penalty
- Seller adds replacement provision for late delivery risk premium
  - Illustrates: Exception handling can itself be subject of negotiation
- RuleML rules refer to DAML+OIL process ontologies, partly PH content

- Can do "what-if" inferencing: e.g., in buyer's version of contract:
- add facts about hypothetical delivery date in contract result
- $\Rightarrow$ delivery is late,
- $\Rightarrow$ which is an exception,
- $\Rightarrow$ a late delivery penalty payment is owed

- Can execute aspects of the contract via inferencing
  - E.g., from facts of actual contract result $\Rightarrow$ determine net payment owed

by Benjamin Grosof copyrights reserved

Summary: New Contributions (I)

- Extend general SweetDeal approach to rule-based e-contracting:
  - exception handling
  - rules reference ontologies about, or invoke: business (sub-)processes
    - (rules can specify the business processes too)
- New prototype ("SweetDeal")
- New application scenario: late delivery in SCM; using PH content
- 1st approach to automate MIT Process Handbook using XML or Description Logic KR (bring it to the Semantic Web)

by Benjamin Grosof copyrights reserved

More New Contributions (II)

- A. 1st to Combine RuleML with DAML+OIL/OWL (by reference)
  - Shows how and why to do as representational style (KR, syntax)
    - DAML+OIL class or property used as predicate in RuleML
      - heavily exploit feature of RuleML that predicate can be a URI
    - Semantic Web LP rules "on top of" Semantic Web DL ontologies
    - Synergize the emerging standards for SW rules and SW ontologies
  - 1st combo of nonmonotonic RuleML / SCLP with DL
  - 1st combo of nonmon rules + DL (also [Antoniou], independently)

- B. 1st to Combine further with process descriptions

- 1st substantial e-business application scenario for (A.) and for (B.)

- Use case and requirements for: deeper semantics of combining LP+DL

- Early point of convergence between Semantic Web and Web Services

by Benjamin Grosof copyrights reserved

Current Work / Future Directions

- Theory on combining LP + DL: New Fundamental KR, techniques
  - Description Logic Programs (DLP) [Grosof *et al* WWW-2003]
    - Intersection KR as bridge, map DL ↔ LP for DLP case
    - Enables: completeness/consistency; efficiency in specification & inference
    - SweetOnto prototype soon publicly available

- Our Larger Projects on Semantic Web Services (SWS):
  - Rule KR Technologies for SWS:
    - services use rules+ontologies, rules use services
  - Business Applications and Implications of SWS:
    - Deal layer of SWS: contracts, incl. about SWS
    - Policies, supply chain, finance, travel

  - Participating in SWS Initiative (Language; Co-Chair, Industrial)

by Benjamin Grosof copyrights reserved

More Current Work & Future Directions

- Other SweetDeal prototype architecture:
  - Contract fragments, with queryable repository
    - modules inclusion & naming: new technical aspects for RuleML
  - Agent aspects
  - Invoke business processes via Situated LP procedural attachments
  - Implementation: running, soon publicly available

- Relation to Legal aspects of Contracting ; Legal XML

- Representing Default Inheritance in Ontologies
  - (NB: PH uses default inheritance, although sparingly)

- Relating to emerging WS and SWS standards/pieces:
  - SWSI, DAML-S, WSMF; SOAP, WSDL, UDDI, BPEL4WS
  - E-Business/Agent Messaging, e.g., ebXML, UBL

by Benjamin Grosof copyrights reserved