Business Rules for E-Commerce:
Interoperability and Conflict Handling

Dr. Benjamin Grosof
IBM T.J. Watson Research Center
Hawthorne, NY, USA
grosof@us.ibm.com http://www.research.ibm.com/people/g/grosof
Overview

• Intro & Background
  – Overall mission, technical approach, industry trends
  – EECOMS project (NIST ATP) on inter-enterprise supply chain integration

• Core Technology: CommonRules prototype based on logic program KR
  – Innovative conflict handling, procedural attachments; sample engine
  – XML Interlingua between heterogeneous rule systems; standards play

• Applications: represent business processes/workflow, policies, products
  – EECOMS supply chain scenarios, Net.Commerce-type B2C scenario

• Directions
  – External alpha release in about 7/99 of core technology
  – Explore more applications
Vision overall

Vision: Rules as an important aspect of coming world of Internet e-business: rule-based business processes for both B2B and B2C.

- represent buyer’s requests, interests, bids
- represent seller’s offerings of products & services, capabilities, bids; map offerings from multiple suppliers to common catalog.
- represent business processes, e.g., sales help, customer help, procurement, authorization, brokering, workflow.
- automatic execution; matchmaking of buyers with sellers
- high level of conceptual abstraction, easier understanding and specification by non-programmers
Rules: Fundamental Technical Approach

- Aim to enable: exchange & update business rules, dynamically.
- Context: key application logic is represented via rules, in many systems. E.g.,
  - rules about terms & conditions associated with a product or service in Internet purchasing.
  - exchange among multiple supply chain players:
    - price vs. quantity vs. delivery date
    - when and how to order or return items, that impact planning.
Rules: Fundamental Technical Approach (continued)

- Declarative approach: provide semantics that is clean and deep.
- Facilitate specification of a given rule set:
  - by multiple authors, cross-enterprise, cross-application
  - by non-technical authors
  - dynamically
  - with abstraction level more easily human-understandable
- Enable conflict handling in multiple rule systems.
- Interoperate between multiple rule systems via common-core interlingua: inter-agent standards in XML.
Applications of Rules

our work to-date: Overview

• Can view generally in terms of business processes, including workflow.
  – rules are good to capture if-then conditionality esp. involving chaining.
• Storefront/catalog-based services: initially, B2C personalization/promotions.
• *Contracts/agreements.
  – represent products or services, or service terms & conditions of product.
  – executable specification
  – partially-specified / template, during process of negotiation.
• *Negotiation, esp. B2B:
  – represent contents of proposals, counter-proposals, RFQ’s, RFP’s.
  – configure auction mechanisms based on contract templates.
• *Security authorization policies: including delegation, certificates.
  – often, are really a part of overall business policy, at application-level.
• * = in EECOMS inter-enterprise supply chain scenarios. EECOMS is a $29Million 3-year NIST ATP consortium effort led by IBM.
Applications of Rules: earlier work on Agent Building Environment

- Can view generally in terms of business processes, including workflow.
  - rules are good to capture if-then conditionality esp. involving chaining.
- Embeddable technology for building rule-based intelligent agent capabilities into applications.
- Class of applications: filtering and routing of info items
  - mail, news, Lotus Notes documents
  - customer service / help desk
  - workflow in manufacturing: design changes, plant floor alerts
**EECOMS Example of Conflicting Rules**

- Vendor’s rules that prescribe how buyer must place or modify an order:
  - A) 14 days ahead if the buyer is a qualified customer.
  - B) 30 days ahead if the ordered item is a minor part.
  - C) 2 days ahead if the ordered item’s item-type is backlogged at the vendor, the order is a modification to reduce the quantity of the item, and the buyer is a qualified customer.

- Suppose more than one of the above applies to the current order? **Conflict!**

- Helpful Approach: **precedence** between the rules. Often only *partial* order of precedence is justified. E.g., C > A.
Courteous LP’s: Example

- `<leadTimeRule1>` orderModificationNotice(?Order,14days)
  - ← preferredCustomerOf(?Buyer,?Seller) ∧
  - purchaseOrder(?Order,?Buyer,?Seller).
- `<leadTimeRule2>` orderModificationNotice(?Order,30days)
  - ← minorPart(?Buyer,?Seller,?Order) ∧
  - purchaseOrder(?Order,?Buyer,?Seller).
- `<leadTimeRule3>` orderModificationNotice(?Order,2days)
  - ← preferredCustomerOf(?Buyer,?Seller) ∧
  - orderModificationType(?Order,reduce) ∧
  - orderItemIsInBacklog(?Order) ∧
  - purchaseOrder(?Order,?Buyer,?Seller).
  - overrides(leadTimeRule3, leadTimeRule1).
- ⊥ ← orderModificationNotice(?Order,?X) ∧
  - orderModificationNotice(?Order,?Y); GIVEN ?X ≠ ?Y.
XML Interlingua for Example

```xml
<clp>
  <erule rulelabel="leadTimeRule1">
    <head>
      <cliteral predicate="orderModificationNotice">
        <variable name="?Order"/>
        <function name="days14"/>
      </cliteral>
    </head>
    <body>
      <and>
        <fcliteral predicate="preferredCustomerOf">
          <variable name="?Buyer"/>
          <variable name="?Seller"/>
        </fcliteral>
        <fcliteral predicate="purchaseOrder">
          <variable name="?Order"/>
          <variable name="?Buyer"/>
          <variable name="?Seller"/>
        </fcliteral>
      </and>
    </body>
  </erule>
  ...
</clp>
```
CommonRules Example: bookstore Web storefront

- B2C personalized promotions:
  - discounting
  - showing targeted ads with incentives
- Rules & facts from:
  - marketing managers: with updates & merges
    - priorities from recency, authority, specificity
  - data mining
  - DB
  - dynamic Web session data
• Running example in CommonRules: includes about 60 rules and facts.

• See IBM Research Report RC 21473 “DIPLOMAT…Demonstration”

• Alternatively, see file bookstoreExampleUnified10-30-98.txt.
CommonRules technology overview

• Java library, V1 prototype running.
  – plan external alpha release 7/30/99 on AlphaWorks
    http://alphaworks.ibm.com
  – piloting in EECOMS $29Million NIST ATP project (IBM, Baan, Boeing, universities, other co.’s)

• Basic rule representation: Logic programs (LP’s).
  – LP’s in declarative sense, not Prolog. E.g., forward or backward chaining.
  – representation = syntax + deep semantics.
    • semantics of rule set = its set of valid conclusions.
CommonRules technology overview (continued)

• Extends rule representation to:
  – **Courteous** LP’s:
    • prioritized handling of **conflicts**, e.g., in updating/merging.
  – **Situated** (Courteous) LP’s:
    • **procedural attachments** to invoke non-reasoning actions or queries, via methods external to inferencing engine.
• **Courteous Compiler** from courteous LP’s to ordinary LP’s.
• **XML Interlingua** and sample translators.
  – interlingua = common rule representation for translation between heterogeneous rule systems. Suitable to become **industry standard**.
• Sample Inferencing/Execution **Engine**:
  – forward-chaining situated courteous LP’s.
Current-version CommonRules

conflict-handling transformer

- courteous representation (Log. Prog.)
- mutex priorities

equiv. equivalent semantically

ordinary/vanilla representation

engine: forward situated LP

interlingua

parsing/translating in & out

Y Rule family

X Rule family

Logic Program family

common cores

deep shared semantics in common representation:

situatd courteous LP’s

XML, KIF, Heterogeneous rule systems

other string formats

app 1

rule sys 1

XSB

app 2

rule sys 2

Smodels

app N

rule sys N

CR.
Flavors of Rules Commercially Most Important in E-Business

- E.g., in OO app’s, DB’s, workflows.

- Logic Programs (in pure knowledge-representation sense): e.g.,
  - Relational databases, SQL.
  - Prolog; Knowledge-based systems.

- Production rules (OPS5 heritage): e.g.,
  - Neuron Data rule-based Java objects.
  - IBM VisualBanker using (Haley).

- Event-Condition-Action rules (loose family), cf.:
  - business process automation / workflow tools.
  - active databases; publish-subscribe.
Overview (repeated)

• Intro & Background
  – Overall mission, technical approach, industry trends
  – EECOMS project (NIST ATP) on inter-enterprise supply chain integration
• Core Technology: CommonRules prototype based on logic program KR
  – Innovative conflict handling, procedural attachments; sample engine
  – XML Interlingua between heterogeneous rule systems; standards play
• Applications: represent business processes/workflow, policies, products
  – EECOMS supply chain scenarios, Net.Commerce-type B2C scenario
• Directions
  – External alpha release in 7/99 of core technology
  – Explore more applications
OUTLINE OF OPTIONALS SLIDES

• Part 1: for Introductory section of talk

• Part 2: about EECOMS

• Part 3: more Technical Details, including logic programs, courteous, situated, applications, etc..
Rules across Applications

Application 1, e.g., seller e-storefront

Business Logic

Rules

Application 2, e.g., buyer shopbot agent

Business Logic

Rules

Interchange
Application Using Rules

- Rule System
  - representation
- rule editor
- rule execution
  - engine

Application
Roles for Rules

• 1st step: Rules as rigorous specification without execution.

• 2nd step (our main focus): Rules as executable specification.
  – Coarse-grain integration with rest of code. E.g.:
    • rules engine in wrapper, agent, workflow, database, extended transaction monitor, ....

• 3rd step: Fine-grain integration with rest of code. E.g.:
  – Object-oriented application development tool’s rules feature.
Rules Authoring Approaches

- Graphical.
- Natural Language (limited).
- Pre-defined templates and vocabulary.
  - domain/industry specific.
  - application/suite specific.
PART-2 OPTIONALS FOLLOW:

about EECOMS
**EECOMS Supply Chain Project: Overview**

- IBM-led consortium, 50%-funded by US government’s NIST Advanced Technology Program. $29M over 3 years, ends 2001.
- Advanced supply chain management. Consortium includes manufacturing software vendors, rules and tool vendors, manufacturer customers.
- Business Focus: improve “agility” of manufacturing. Respond to common but unpredictable events such as late delivery, plant line breakdown, larger than expected order. React quickly, including modify plans, schedules. Integrate: typically multi-application, very often multi-enterprise.
- Technical Focus: **rules and conflict handling**: virtual situation room for human collaborative workflow; attendant tools, agents, and security issues.
- Is follow-on to CIIMPLEX (IBM-led NIST ATP $22M) & challenges it identified. Shares: consortium, scenarios, agent-based approach.
EECOMS Supply Chain Project: *Our Role*

- **Rules** primarily:
  - application Scenarios with customers.
  - *interlingua*.
  - conflict handling.
  - for *security authorization* policies (trust management).

- Multiple, heterogeneous rule systems within consortium: from vendors, universities.
EECOMS: Participants

- IBM is leader of consortium, via its Manufacturing ISU. Runs EECOMS Project Office, EECOMS Integration Center and Tools Dev.
- Berclain, part of Baan: major manufacturing / ERP software vendor.
- Vitria, EnvisionIt, IndX, Scandura: (small) tools/specification vendors.
- TRW supply-chain consulting practice. Acts as customer(s).
EECOMS: Main Collaborators

- Prof.’s Tim Finin and Yannis Labrou of U. Maryland (Baltimore):
  - negotiation protocols between agents; FIPA standards for these
  - XML versions of FIPA and KIF draft standards
  - security authorization policies, delegation
- Prof.’s Bill Chu and Bob Wilhelm of U. N. Carolina (Charlotte)
  - constraint satisfaction and optimization in private negotiation decisions
  - integrating rule-based authorization policies into security services
- Prof.’s Stanley Su and Joachim Hammer of U. Florida (Gainesville)
  - negotiation architecture
  - Event-Condition-Action rule system
- Lynne Thieme of Vitria Technologies
  - business process automation and publish-subscribe
  - Event-Condition-Action rule system
**EECOMS: Virtual Situation Room**

- **Human** business process and workflow: for timely, agile response.
- **Distributed, inter-enterprise.**
- Leads:
  - Concept: Bill Tolone of UNC Charlotte (IBM sub-contractor)
PART-3 OPTIONALS FOLLOW:
about more Technical Details
Logic Programs as basic representation: Definition

- A LP is a set of (premise) rules; semantically, it specifies a set of conclusions.
- Example rule:
  
  `sendPage(?msg,Joe) ← from(?msg,?s) ∧ urgent(?msg) ∧ caresAbout(Joe,?s).`

  where the “?” prefix indicates a logical variable.
- Generally, a rule has the form of
  
  `Head ← Body`:

  `H ← B_1 ∧ ... ∧ B_j ∧ ~B_{j+1} ∧ ... ∧ ~B_m .`

  where `m ≥ 0`; `∧` stands for logical “AND”; `←` stands for logical “IF”; and `H, B_1, ..., B_m` are each an atom with form: `Predicate(Term_1, ..., Term_k)`.
- A predicate = a relation. An atom semantically denotes a boolean.
- `~` stands for negation-as-failure (a.k.a. weak negation, default negation).
  - The negation-as-failure construct is logically non-monotonic.
  - Intuitively, `~p` means p’s truth value is either `false OR unknown`.
Logic Programs: Definition (continued)

- Each argument Term_1, ..., Term_k is a term.
- A term is either a logical constant (e.g., “Joe”) OR a logical variable (e.g., “?msg”) OR a functional expression of the form:
  LogicalFunction(Term_1, ..., Term_k)
- A functional expression semantically essentially denotes a logical constant.
- A term, atom, or rule is called “ground” when it has no logical variables.
- A fact is a ground rule with empty body.
- A primitive conclusion has the form of a ground atom (compound conclusions are built up from these via logical operators such as AND etc.).
- Semantically, a rule or LP stands for the set of all its ground instances.
- (Observe that a rule body can represent an expression in relational algebra cf. relational DB’s (e.g., SQL).)
Logic Programs as basic representation: Advantages

• **Declarative:** semantics is independent of inferencing procedure implementation, e.g., forward vs. backward chaining, sequencing of executing rules or conditions within rules.

• **Expressive:** relational expressions cf. SQL, large fragment of first-order logic, chaining, basic logical non-monotonicity (unlike first-order logic).

• **Efficient:** computationally tractable given two reasonable restrictions:
  – 1. Datalog = no logical functions of non-zero arity.
  – 2. Bounded number $v$ of logical variables per rule.
  – $m = O(n^{(v+1)})$, where $n = ||LP||$, $m = ||\text{ground-instantiated LP}||$.
  – Inferencing time is $O(m)$ for broad (acyclic) case, $O(m^2)$ generally (for well-founded semantics).
  – By contrast, first-order-logic inferencing is NP-hard.
Logic Programs: Advantages (continued)

• Widely deployed and familiar:
  – relational DB’s, SQL
  – Prolog
  – intelligent agents and knowledge-based systems
    • e.g., IBM’s Agent Building Environment
• Common core shared semantically by many rule systems: e.g., production rules, Event-Condition-Action rules, first-order-logic each overlap strongly with LP’s.
Courteous LP’s: the What

- Updating/merging of rules sets: is crucial, often generates conflict.
- **Courteous** LP’s feature prioritized handling of conflicts.
- Specify scope of conflict via a set of [mutual exclusion](#) constraints:
  - E.g., ⊥ ← discount(?product,5%) ∧ discount(?product,10%).
  - E.g., ⊥ ← loyalCustomer(?c,?s) ∧ premiereCustomer(?c,?s).
  - Permit classical-negation of atoms: ¬p means p has truth value false
    - implicitly, ⊥ ← p ∧ ¬p for every atom p.
- **Priorities** between rules: partially-ordered.
  - Represent priorities via reserved predicate that compares rule labels:
    - overrides(rule1,rule2) means rule1 is higher-priority than rule2.
    - Each rule optionally has a rule label whose form is a logical constant.
    - overrides can be reasoned about, just like any other predicate.
Courteous LP’s: Example (repeated)

- `<leadTimeRule1> orderModificationNotice(?Order,14days)
  - ⊥ preferredCustomerOf(?Buyer,?Seller) ∧
  - purchaseOrder(?Order,?Buyer,?Seller) .
- `<leadTimeRule2> orderModificationNotice(?Order,30days)
  - ⊥ minorPart(?Buyer,?Seller,?Order) ∧
  - purchaseOrder(?Order,?Buyer,?Seller) .
- `<leadTimeRule3> orderModificationNotice(?Order,2days)
  - ⊥ preferredCustomerOf(?Buyer,?Seller) ∧
  - orderModificationType(?Order,reduce) ∧
  - orderItemIsInBacklog(?Order) ∧
  - purchaseOrder(?Order,?Buyer,?Seller) .
  - overrides(leadTimeRule3 , leadTimeRule1) .
- ⊥ orderModificationNotice(?Order,?X) ∧
- orderModificationNotice(?Order,?Y); GIVEN  ?X ≠ ?Y.
Priorities are available and useful

• Priority information is naturally available and useful. E.g.,
  – **recency**: higher priority for more recent updates.
  – **specificity**: higher priority for more specific cases (e.g., exceptional cases, sub-cases, inheritance).
  – **authority**: higher priority for more authoritative sources (e.g., legal regulations, organizational imperatives).
  – **reliability**: higher priority for more reliable sources (e.g., security certificates obtained by delegation, assumptions, observational data).
  – **closed world**: lowest priority for catch-cases.

• Many practical rule systems employ priorities of some kind, often implicit, e.g.,
  – rule sequencing in Prolog and production rules.
Courteous LP’s: Advantages

• Facilitate updating and merging.

• Expressive: classical negation, mutual exclusions, partially-ordered prioritization, reasoning to infer prioritization.

• Guarantee consistent, unique set of conclusions.
  – Mutual exclusion is enforced. E.g., never conclude both $p$ & $\neg p$.

• Efficient: low computational overhead beyond ordinary LP’s.
  – Tractable given reasonable restrictions (Datalog, bound $v$ on #var’s/rule):
    • extra cost is equivalent to increasing $v$ to $(v+2)$ in ordinary LP’s.
  – By contrast, more expressive prioritized rule representations (e.g., Prioritized Default Logic) add NP-hard overhead.
Situated LP’s: Overview

- Point of departure: LP’s are pure-belief representation, but most practical rule systems want to invoke external procedures.

- Situated LP’s feature a semantically-clean kind of procedural attachments. I.e., they hook beliefs to drive procedural API’s outside the rule engine.

- Procedural attachments for sensing (queries) when testing an antecedent condition or for effecting (actions) upon concluding a consequent condition. Attached procedure is invoked when testing or concluding in inferencing.

- Sensor or effector link statement specifies an association from a predicate to a procedural call pattern, e.g., a method. A link is specified as part of the representation. I.e., a SLP is a conduct set that includes links as well as rules.
Situated LP’s: Overview (continued)

- phoneNumberOfPredicate ::s:: BoeingBluePagesClass.getPhoneMethod. 
  ex. sensor link
- shouldSendPagePredicate ::e:: ATTPagerClass.goPageMethod. 
  ex. effector link
- Sensor procedure may require some arguments to be ground, i.e., bound; in general it has a specified binding-signature.
- Enable dynamic loading and remote loading of the attached procedures (exploit Java goodness).

- Overall: cleanly separate out the procedural semantics as a declarative extension of the pure-belief declarative semantics. Easily separate chaining from action.
Courteous Compiler

- Transformer compiles a courteous LP into an ordinary LP.
- A radically innovative approach in rules representation.
- “Compiles away” conflict, as modular add-on to rule system X’s
  - inferencing
  - specification
- Enables courteous features to be added to, or implemented in, a variety of rule systems.
Interlingua Concept

• Challenge: heterogeneity of rule systems (within applications) to be integrated.
  – Each has own rule representation. N of them.
  – Representation = syntax + semantics.

• Approach: translate via interlingua rep’n.
  – rep’n A ↔ interlingua ↔ rep’n B.
  – Advantage: O(N) translators instead of O(N^2).

• *Focus: commercially important rep’n families:
  – LP’s, production (OPS5), Event-Condition-Action.
  – E.g., in OO app’s, DB’s, workflows.
Interlingua:

Deep Shared Semantics at Core

- Desire: deep semantics (model-theoretic) to
  - understand and execute **imported** rules.

- Possible only for shared expressive subsets: “cores”.
  - Rest translated with superficial semantics.

- Desideratum: declarativeness of core / rep’n (in sense of knowledge representation theory).
  - Maximize overall advantages of rules:
    - Non-programmers understand & modify.
    - Dynamically (run-time) modify.
Interlingua: Going Beyond KIF

- Point of departure is KIF: Knowledge Interchange Format
  - Intent: general-knowledge interlingua.
  - Emerging standard, in ANSI committee.
  - Main focus: classical logic, esp. first-order.
  - Has major limitations remedied by our Interlingua:
    - logically monotonic; no conflict handling or priorities.
    - pure-belief: no procedural attachments.
- Our Interlingua uses situated courteous LP’s rep’n.
  - Complements KIF. Overlaps on:
    - pure-belief rules without negation-as-failure.
Interlingua: current version

- XML format ↔ Java objects ↔ text syntax for rule system X
- Sample translators to/fro KIF and 3 rule systems in LP family, initially with pure-belief semantics as courteous LP rep’n.
  - backward-direction: XSB (Stonybrook, commercializing).
  - forward-direction: Smodels (Helsinki, academic).
  - CommonRules courteous LP text syntax:
    - courteous expressiveness: mutex’s, priorities.
Interlingua: early standards engagements

- FIPA = Foundation for Intelligent Physical Agents
  - Standards body: main industry locus of action since ‘96 for intelligent agents knowledge-interchange standards work, e.g., for Internet e-commerce negotiation between agents.
  - Current draft standard uses ANSI KIF to represent rules.
  - In collaboration with U. Maryland, we are driving XML version of KIF, in synch with our IBM rules interlingua and EECOMS.
  - In early discussions about going beyond KIF.
- ANSI KIF
  - We have long-running role in committee: non-monotonicity and conflict handling.
Sample Rule Engine: current version

- Sample Inferencing/Execution Engine:
  - forward-chaining situated courteous LP’s.
- Composes courteous compiler with engine for situated ordinary LP’s.
- Intent is proof-of-concept for Interlingua, courteous compiler, situated techniques.
  - Not performance-tuned.
  - Lacks various features found in best-of-breed commercial rule systems.
    - Restricted to acyclic case (no predicate depends thru rules on itself).
- IBM is not in the rule-engine business, we will partner with commercial rule-system vendors to license their technology
  - ... which CommonRules complements and extends.
Contracts/Agreements

• Use Interlingua to represent products or services, or service terms & conditions of product, e.g., in catalog or during negotiation.
  – E.g., business process to return an item for repair, or to deliver an order.
• Executable specification; situated / procedural attachments is esp. useful.
• Partially-specified / template, esp. during process of negotiation.
• Complement XML ontologies already evolving for various domains.
  – Ontology = formally-represented vocabulary / definitions
• Configure automated auction mechanisms, based on contract templates.
  – which goods, which attributes (e.g., price, delivery-date) are at issue.
• In collaboration with: EECOMS and
  – Prof. Mike Wellman and PhD student Dan Reeves of U. Michigan (Ann Arbor), as IBM-funded University Partnership project started 1/99.
    • Mike leads AuctionBot, a major existing automated Internet service.
Negotiation, esp. B2B

- Use Interlingua to represent contents of:
  - Requests For Quotation or Proposal, i.e., statements of buyer interests, that initiate inter-enterprise negotiation.
  - responses to such RFQ’s / RFP’s by seller.
  - proposals and counter-proposals and “side information” exchanged during back-and-forth negotiation / bargaining between buyer and seller.
  - statements of seller/supplier capabilities/interests, e.g., important for source selection as well as bargaining.
- Configure auction mechanisms based on contract templates. (with U. Mich.)
- In collaboration with U. Mich. and:
  - in EECOMS overall effort, negotiation is a main ‘99 focus of demos and scenarios.
Security Authorization Policies

- Use rule-based executable specification of security authorization policies, a.k.a. trust management: including delegation, certificates.
  - We have the first step of an expressive extension of courteous LP’s to handle delegation and certificates.
- Often, authorization policy is really a part of overall business policy, at app-level. This contrasts with authentication.
- Advantages of rule-based approach, esp. from declarative semantics:
  - principled handling of negation and conflict.
  - provable guarantees of behavior of implementation.
  - more human-understandable and easy to modify.
  - easier integration with general business policy.
Security Authorization Policies
(continued)

• In collaboration with:
  – Joan Feigenbaum of ATT Research which has a leading effort on trust management, and her PhD student Ninghui Li of NYU.
  – EECOMS: as a main focus of EECOMS innovation in Security SIG (area of consortium).
overall key Challenges identified

• Exchange & update business rules, dynamically.  E.g.,
  – compose an application from multiple components.
    • Net.Commerce-type B2C: marketing manager, data miner, ...
  – buy/sell or integrate between heterogenous peer applications.
    • B2B inter-enterprise supply chain in manufacturing industry.
      – E.g., when and how to order or return items, that impact planning.
    • Internet purchasing: terms & conditions.

• Facilitate specification, dynamically, of a given rule set by
  – multiple authors.
    • e.g., cross-enterprise or cross-application.
  – non-technical/non-programmer authors.
    • make abstraction level more easily human-understandable.