



Combining Different Business Rules Technologies: A Rationalization

IBM Research

H. Chan, L. Degenaro, I. Rouvellou

Country Companies

K. Rasmus

MIT Sloan School of Management

B. Grosf

IBM Software Solutions

D. Ehnebuske, B. McKee

Motivation

- Business rules are
 - policies or practices of the business
 - anything that can be put in a "if then"
- Business rules have been embedded in applications
 - Prone to inconsistency
 - Difficult to maintain : "harvesting" is very difficult
- Many technologies have grown around the development, management and execution of business rules including
 - Externalized/componentized rules
 - Knowledge-based inference engines
- Technologies vary in their strengths and weaknesses
- Can we leverage their strengths and minimize their weaknesses by combining them?

Common Rules Overview

- Basic rule representation: Logic programs (LP's).
 - LP's in declarative sense, not prolog.
 - representation = syntax + deep semantic
- Extends rule representation to:
 - Courteous LP's
 - prioritized handling of conflicts
 - Situated (Courteous) LP's
 - procedural attachments
- Courteous Compiler from courteous LP's to ordinary LP's
- XML Interlingua
- Simple Inferencing/Execution Engine
 - forward-chaining situated courteous LP's

ABR Overview

- Domain experts see business processes as "themes and variations"
- ABR allows to structure applications to match built in core behavior with variations specified, managed, and applied externally
 - structuring is done at the object level and fits naturally within it
- Structured exit points from main code (trigger points)
 - are explicitly identified during analysis and design
 - most follow a small set of patterns (directly supported by ABR)
 - assemble runtime info and find the applicable rules (query)
- An ABR rule is a persistent object encapsulating
 - a piece of code (Rule::fire()) which implements the "variation"
 - "context" attributes used to select the applicable rules at runtime

Differences in the 2 rules approaches

- Expected dynamics of the rules
 - CR: rapidly changing rule base (1 per transaction possible)
 - ABR: release oriented rules with start/end dates
- Conflict handling capability
- Discovery process
- Tightness of integration with the rest of the application code
- Level of interaction of rules
 - CR: many rules in a group
 - ABR: 1-10 rules per context
- Complexity of rules
- Complexity of interaction
- Performance dynamic

Combining the two technologies

- Using them together
 - ABR wrapping CR invocation when complex/smart inferencing needed at well defined points in a business process
 - ABR and CR used for different portions of the business process
 - different groups are predisposed to a particular rule representation
 - different level of ambiguity/conflict
- Using them at different point in the application lifecycle
 - start capturing rules with CR when architecture is still "in flight"
 - use CR to discover interaction between rules
 - move from ABR to CR (or vice-versa) if the rate of change of the rules increases (decreases)

Enhancing of the current technologies

- Data mining
 - analyze data to derive rules
 - analyze rules to simplify them
- Customizations for Supporting Specific Rules Patterns
 - specific implementations to handle common rule patterns
 - framework to build customized rules front-ends that accommodate the specific rules patterns of a business and/or application
- Authoring of rules
 - conversational/iterative specification of rules
 - integration with UML like specifications

Conclusion

- Many rules technologies/products are available
- Using two rules technologies illustrating knowledge based inferencing (CR) and externalized/componentized rules (ABR), we tried to articulate a "unifying rule vision"
- Different rules technologies/products have different strengths and weaknesses
- We described ways of combining them to leverage their strengths and minimize their weaknesses
- We also described opportunities to improve both technologies that would enhance their business benefits