October Rules Fest 2009 Abstracts

Conference Papers
Organized by last name

Dr. Charles Forgy: Production Systems Technology, Founder and Inventor of Rete Algorithm
Closely-Coupled Parallel Rule Bases

One way to use parallelism in rule based systems is to leave the rule-firing semantics unchanged while using parallelism within the engine to speed up execution. An alternative approach is to employ multiple threads of execution at the rule level. (These approaches are of course complementary; a system could be built that used parallel execution where every rule execution thread ran on a parallel engine.) This talk considers one approach to rule-level parallelism: the use of multiple, sequential rule bases which share data at the working-memory level. If a few minor changes are made to the standard production system architecture, such systems can be implemented in a quite straight-forward manner.

Greg Barton: Southwest Airlines, Rules Architect
Early Alert System at Southwest Airlines

Southwest Airlines is venturing into the rules development space with the Early Alert System. EAS enables SWA to have a real-time model of it's entire aircraft fleet, tracking such activities as taxi in, taxi out, and in gate turn. It does this by maintaining a data structure representing physical assets and the activities they perform. Incoming data from those assets update the data structure, and rules react to the changes. We hope to use this paradigm going forward to use rules to monitor other aspects of the enterprise, enabling a more agile and efficient response to the airline's daily operating challenges. Our main points will be the Overview, Feature Review, Design, Other Uses of Rules at present by SWA and the future of rules at SWA.

Carole Ann Berlioz-Matignon: FICO, VP Product Management
Measuring Your Rules' KPIs

Defining and executing business rules may be a challenge in itself but once they are out there, how do you know they are good for your business? Comparing strategies (aka business rules) with higher level tools implies that the appropriate infrastructure collects the relevant data. This presentation will discuss the needs of the business, the instrumentation that supports it as well as best practices, tips and pitfalls. I will also contrast business rules performance monitoring with similar capabilities in BPM solutions.

Daniel Brookshier: No Magic, CTO
Generating Rules from UML

This paper/session will show how to transform UML to generate rules using Velocity templates and Ruby. It will address the question, "Why generate rules?" In this particular case, we are doing it as part of an overall architecture in UML. Using many base rules to seed a rule engine is one way to ensure that these are identified and produced. In addition, we can manage requirements and dependencies in one place. Also, I will discuss some problems and issues with Production Rule Representation (PRR) - a new profile from the Object Management Group (OMG) used to represent rules in UML. Even though there is a bit of a dependency on MagicDraw UML, I will discuss methods without dependency on any one specific UML tool.

Eric Charpentier: Primatec Systems, Founder
Rule Classification First

This presentation will explore a business rules classification that can help facilitate discussions with people of different backgrounds in a project. The word business rules is being used right, left and center in different contexts and there is a need to clarify what is meant by business rule and what categories of business rules exist and how they relate to each other in a project.

The classification is an attempt at identifying different types of business rules, and eventually to document some criteria that will allow a project to decide where the rules should be implemented in an SOA type architecture.

The basic classification is as follows:
1 Rules for data transformation
2 Rules for referential integrity
3 Rules for data validation
4 Rules for security
5 Rules for presentation
6 Rules for workflow or business process
7 Rules for decisions
8 Rules for rating engine

Dr. Jacob Feldman: Open Rules, Inventor
Rule Violations and Over-Constrained Problems

Business rules used by online decision support applications are frequently violated or contradict to each other. It happens not because of a bad rules design but because real-world problems are usually over-constrained and the state of the system is frequently changed. In this talk I will introduce the concepts of hard and soft rules, and will share the experience of measuring, controlling, and minimizing rule violations. The concrete examples of real-world problems will be presented along with the code that explains how to deal with rule violations.

Dr. Gopal Gupta: University of Texas – Dallas, Professor
With Abhilash Tiwari and Siddarth Chitnis
Programming rules using a constraints-embedded spreadsheet interface

Constraint satisfaction problems (CSPs) are ubiquitous in the business world. A large class of CSPs relating to business rules, scheduling, timetabling, product configuration, etc., can be modeled as 2-dimensional tables. We exploit this observation to develop a generic, robust, user-friendly tool called PlanEx that allows non-expert users to solve practical CSPs. PlanEx is implemented as an add-in for Microsoft Excel and exploits constraint logic programming over finite domains (CLP(FD)). PlanEx extends the spreadsheet paradigm by allowing finite domain constraints to be attached to individual cells that are then solved using CLP(FD) to obtain a solution. PlanEx provides a spreadsheet-based user-friendly interface to CLP(FD) by hiding the syntactic and semantic complexity of CLP(FD) from novice users. PlanEx is an innovative tool that is operational. It has been used to solve large, real-life constraint satisfaction problems.

Rolando Hernandez: BIZRULES, Founder

A Model-Driven approach for validating rules, documenting business requirements, and defining system specifications for rules-based systems

BIZRULES and one of their global customers developed a model-driven approach for documenting business requirements (rules, processes, events, etc.) and defining system specifications for rules-based systems. Models and technical documentation that could be understood by both IT and business are sometimes tricky to manage. However, since that is exactly what is needed by most rulebased systems that are implemented in the BRMS space today, this is a critical component.

We needed to create models and technical documentation that could be understood by both IT and business. We needed to validate and verify the rules in the model, before the coding process started, instead of waiting until the system testing process to test the rule logic. We needed to find an alternative to use cases that were not effective for documenting rule-based systems. We needed a solution that relied more on diagrams and visual models rather than textual documentation because our offshore developers did not speak English as their primary language. A picture is worth a thousand words applies very well to business rules. We needed a solution that leveraged enterprise architecture, interaction architecture, rules/process integration, knowledge engineering, rulebase architecture, and business rules methodology. We needed a solution that would work for any rule engine.

This paper will deal with these problems and our solutions: a business rule model that is more than just a pretty picture - it actually "runs" the rules as soon as you document them; a logical system model that describes how the entire system works, how the users interact with the system, what the system does, and how the rules and processes work. This method demonstrates a "real world example" of how we successfully documented rule-based systems, tested the rules before they were coded, and replaced textual use cases with a new visual logical model of the system.

Dr. Rick Hicks: EZ-Xpert, Founder / Texas A&M, Professor
Automated Verification Testing for Propositional Logic Systems

Despite the necessity of verification, it is often overlooked and usually done poorly. It is usually performed manually; most practioners run test cases of a subset of the rulebase to perform verification. Our approach is far different, as we use a tool to constrain the rulebase during construction, so problems are found immediately instead of during testing. By automating verification, we dramatically improve accuracy and lower development time.

This talk will focus on the common verification problems such as completeness, consistency, conciseness, domain constraints, and reachability. We will define these criteria, provide examples, and show how they may be automatically verified in a development environment.

David A. Holz: HRSI, Founder
Rule Patterns and Features in Modern Forward Chaining Engines: Rules Beyond Business Rules

At HRSI, we've long used declarative techniques to generate the software and data used to solve problems for our customers and ourselves, instead of hand-coding most of the source code. In seeking to integrate more AI into our toolkits and runtimes, forward-chaining rule engines have shown themselves to be a very practical approach, effectively converting knowledge and state into behavior.

However, much of the focus of rule systems has been to provide tools for business analysts, as opposed to solving the general programmatic and automation problems that software implementers face. In tackling different problem sets, a number of desired rule features have emerged that are generally not covered by

available production systems. Numeric salience and standard conflict resolution strategies can also become limiting factors as rule-based applications scale up.

We present a set of Rule Patterns (akin to the Design Patterns of OO) addressing these higher-order situations, their underlying rationale and real needs they fulfill, and our testbed custom rule engine where we have implemented some of them as embedded features. It is our hope that these patterns and techniques can be integrated into existing systems to provide greater levels of functionality and applicability of rule engines across multiple domains.

The target audience is programmers desiring to integrate rules inside their programming, and rule engine implementers.

Dr. Leon A. Kappleman: University of North Texas, Professor
John Zachman: Zachman International, Founder
Enterprise Architecture 201: Creating the Information Age Enterprise

Enterprises that are more agile and adaptable are more able to succeed in an Information Age world that demands they do more with less, faster, while traditional boundaries blur, and the rules of engagement change. Succeeding in such a world requires that organizations skillfully manage information about their products, customers, suppliers, markets, assets, and liabilities. Fortunately, most enterprises are skilled in such matters. But succeeding in the world of today, and to a even greater extent in the world of tomorrow, also demands that enterprises master the management all of the knowledge about itself, including details about all of its people and processes, intelligence and knowledge, things and places, timings and motivations, plans and measures, rules and jobs, structures and more. We are in the early stages of developing such skills and capabilities. Enterprise Architecture (EA) is currently the name of this emerging discipline.

Organizations continuously strive to improve, transform and optimize their business processes and practices. Because an organization's information technologies are now the means by which it automates and optimizes processes and practices, IT can either be an enabler or an inhibitor to rapid business transformation. One might conclude that IT is often an inhibitor in light of decades of "alignment of IT with the business" persisting at the top of IT management's key concerns. EA represents a new way of thinking about and managing the enterprise, including its information technologies. Knowingly or not, the "rules community" understands this and implements solutions that treat the rules as an independent variable of the enterprise's business and technology architecture. But rules alone do not make a complete enterprise architecture, and doing EA requires rules too.

EA is all about achieving the vision of bridging the chasm between strategy and implementation, of capturing all the knowledge about the enterprise and making it available in real time for every imaginable management need, and of having a shared "language" of words, graphics, and other depictions to discuss, document, and manage every important aspect of the enterprise. EA is key to being agile, adaptable, interoperable, integrated, lean, secure, responsive, efficient, effective, and thereby more able to succeed in the Information Age.

The Learning Objectives this talk include addressing matters like:

- What is EA and why should you care about it?
- How to get started with EA by building on what you already know and do.
- Who should be involved in EA activities?
- What is the role of rules in EA activities and processes, as well as in the EA itself.
- Why and how our mental models and language about enterprises and IT must evolve.
- How do you do EA and what kinds of skills are needed?
- What to expect and assume on your EA journey?
- How to create value from EA in the short-term, as well as the long?

Dr. Daniel S. Levine: University of Texas – Arlington, Professor
Truth versus Useful Lies

A religious leader in a Kurt Vonnegut novel advocated abandoning the search for truth in favor of a search for useful lies. The switch from truth to useful lies is a good metaphor for what typically occurs in the development of human decision making. As we grow from childhood to adulthood, we gradually make decisions less on the basis of the literal truth of events (what the psychologists Reyna and Brainerd call *verbatim encoding*) and more on the basis of categories of experience (what those psychologists call *gist encoding*). The switch from verbatim to gist makes decisions fit into patterns which are efficient and enable us to live in a complex society. Yet it also leads to some characteristic distortions, such as incorrect encodings of probabilities.

I will sketch some brain processes that may be involved in the "fuzzy traces" that lead to gist encoding and distortion of probability weights. One part of the frontal lobes, the orbitofrontal cortex, plays the major role in making us live by our gists, for good and for ill. But we also need something like the child in *The Emperor's New Clothes* to stand apart from our culture and tell us when we need to restore our verbatim encoding, that is, to seek truth instead of useful lies. Two other parts of the frontal lobes seem to play the roles of challenger (the anterior cingulate cortex) and truth-teller (the dorsolateral prefrontal cortex).

Dr. Hafedh Mili: University of Qubec at Montreal (UQAM), Professor
Title: Agile Business Rule Development: a tautology or an oxymoron?

Abstract: Business rules capture the essence of the decision-making "intelligence" of an organization, and as such, represent an important corporate asset that requires careful management. Careful management entails well-defined processes for managing the lifecycle of business rules, from inception to retirement. At the same time, the implementation of business rules must keep-up with the pace of change of the underlying business operating environment. In this talk, we present the principles behind ABRD, and present some rule governance best practices that combine agility and rigor.

Jason Morris: Morris Technical Solutions, LLC, Founder
Grant Tranter, University of Sydney, Professor
Implementation of a Modified Propose and Revise Problem-Solving Method Using Jess

Information derived from soil property data is very much in demand for making decisions about land and soil use. Unfortunately, there is a dearth of tools for providing this kind of information in academia and in industry. Designed and developed in collaboration with the University of Sydney, Australia as an extension of the work of McBratney et. al., SINFERS is a rule-based expert system that addresses this need by computing biological, chemical, chromal, morphological, and structural soil properties based on an initial soil property input set.

This whitepaper describes SINFERS's inferencing mechanism and architecture, which employs a novel *propose-select-revise* problem-solving strategy to compute new soil properties and to manage meta-level heuristics. Key to the operation of SINFERS is its database of *pedotransfer functions* (PTF)s. A PTF is a multi-variate, statistical-regression function that predicts certain soil properties from other more feasibly obtained properties. Once an initial set of properties has been asserted into SINFERS's working memory, SINFERS uses its rulebase to choose which PTFs to apply to those soil properties, thereby computing a

prediction for their (in most cases) numeric value and uncertainty. As new soil properties are generated, they are added to working memory, which in turn causes new PTFs to be applied. By continuing in this fashion, SINFERS can thus infer the unknown superset of soil properties corresponding to the initial input set. These predictions can be further analyzed to make additional inferences about soil utilization.

James C. Owen: KnowledgeBased Systems Corporation, Founder
Updating Rulebased Forecasting (RBF) to Use Modern Rule Engines

Rulebase Forecasting (RBF) was, at one time, the hope of the future. But since stock trades and normal economic growth are time dependent function and Neural Nets proved to be superior to rules and/or any kind of regression for time series functions, using RBF fell into disuse. However, with the recent downturn in the economy, we are proposing that some of this might have been foreseen had we used the right tools for the right job.

This paper presents the findings that were done previously, the rules, the earlier solutions and solutions proposed for today's RBF systems. The rule logic is isolated out (abstracted out) from the code itself. While the code is still highly complex and technical, these details are hidden from the user with a rulebase that hides the code and thereby makes it readily available for the technical business user to modify if needed.

Mark Proctor: Drools, Inventor
MP1: ***Production Rule Systems - Where do we go from here?***
While I love developing the Drools production rule engine, I have found that I've become increasingly frustrated by the various limitations of expressability and execution flow control. In this talk I plan to cover the current state of play with the Drools rule engine and the research we are currently looking into and ideas we have for the future.

Mark Proctor: Drools, Inventor
MP2: ***Distributed Programming with Rule Engines***
With the increased capabilities in memory, cpu and network throughput users are increasingly looking to exploit distributed systems. Cougar and Jade both offering interesting solutions for distributed and collaborative programming. I've been looking into those products and researching other ideas for Drools, for some time now. In this talk I plan to present the basic foundations that we have put into place for distributed programming in Drools and discuss where we hope to take things.

Gary Riley: CLIPS, Inventor
The CLIPS Implementation of the Rete Pattern Matching Algorithm

Recent changes to the pattern matching algorithm used by CLIPS have dramatically improved performance on the classic Waltz and Miss Manners benchmarks, which are frequently cited as measures of performance for rule-based systems. This white paper will describe the changes made to CLIPS 6.3 that include a restructuring of the Rete network topology, implementation of hashing in the alpha and beta memories, and improved management of the agenda for large numbers of rule activations. In addition, the 'lazy' evaluation of partial matches added to a 'not' conditional element and the asymmetric approach utilized by CLIPS in processing retract operations will be detailed. This paper will focus on the actual details of implementation in the C programming language rather a theoretical discussion of the algorithms used.

Carlos Serrano-Morales: FICO, VP Product Development and Chief Architect
Business Rules in the Cloud

Cloud and Grid computing are emerging as first class citizens in modern enterprise and consumer application architectures. The availability of these architectures presents a number of opportunities for the implementation of decision management components, encompassing business rules, predictive models and optimization. The efficient implementation of computation intensive tasks, parallelization of complex computations and batch jobs in an environment which offers explicit business price management will make the implementation and deployment of the full decision lifecycle in even larger scale enterprise and consumer problems a reality.

Edson Tirelli: Drools, CEP Designer
ET1: Extending General Purpose Engines with Domain Specific Resources

Every business domain has its own specific vocabulary, requirements and solutions. General purpose tools are excellent to solve general problems, but sometimes they fall short or are not easily applied to certain domain problems. One of the major advancements of JBoss Drools 5.0 is its support to domain specific extensions. From Domain Specific Languages that support higher level vocabulary, to pluggable operators and functions that simplify business rules authoring, a whole set of features are available for enterprises to enable easier, faster and less costly business logic management.

During this session attendees will learn how to develop and use: Domain Specific Languages (vocabulary), Operators (constraints), Functions (analysis), Work Items (processes/tasks), among other features. This is a technical presentation targeting software architects, software developers and technical managers on how to improve business application development using domain specific extensions to the JBoss Drools Platform.

Edson Tirelli: Drools, CEP Designer
ET2: Temporal Reasoning: a requirement for CEP

As Complex Event Processing grows in popularity and applicability, the convergence between modeling paradigms demand more and more functional requirements from the available tools. One key requirement for CEP use cases (and standard business scenarios) is the ability to write rules and queries that require some degree of temporal awareness, from simple constraints to actual data inference. More than that, temporal reasoning is a feature on top of which the actual platform can leverage internal optimizations, aiming for resource savings and improved scalability.

This is a deep technical presentation on how temporal reasoning was added to the JBoss Drools platform to enable it for CEP use cases and at the same time how temporal reasoning is used for engine optimizations like match constraining and automatic event garbage collection. Target audience would be researchers, software architects, developers and students looking for insights on the development of temporal reasoning capabilities on top of the Rete core algorithm.

Larry Terrill: EDBC, Founder
Introduction to Inference Engines

For those new to Rete-based inference rule engines, the relationship between the Agenda, Working Memory, and the rule set can be confusing. In this presentation, a simple example of three rules for assigning a flight crew to an aircraft are used to illustrate this relationship and the function of the control strategy of the rule engine, demonstrating refraction, recency, and salience/priority.

First definitions for ?Business Rule? and ?Rule Engine? are presented, following by a high-level description of the execution process used by the engine. Then step-by-step details of populating working memory and rule execution are shown to demonstrate the how changes in working memory cause the scheduling of rule instance that subsequently may or may not be fired as working memory continues to change. The order of the rule instances as they are added is used to demonstrate both salience and recency. The execution of rules instances is used to demonstrate refraction

Paul Vincent: Tibco, Chief Technologist
***What's Different About Rules in CEP*?**

Complex Event Processing can be described as the generic processes and technologies for identifying abstract events. Identifying such events can require several levels of complex event patterns across context, time and sets, and can be defined as a set of "rules" (if <pattern> then <complex event>). Various types of "rule technology" are deployed in CEP, varying from orchestrated SQL-type continuous queries, Event-Condition-Action rules, inference production rules.

At present very few CEP engines embed inferencing capabilities, even though CEP processes are naturally stateful (i.e. recording state information between the arrival of events) and can involve complex chains of deduction (i.e. <event A> leads to <event B> leads to <event C>). In addition, CEP is generally not considered a "business user" domain – analyzing and developing complex event patterns is an expert task not generally amenable to "business manager editing" – although the resultant decisions can of course be treated as "business controlled".

Experienced rule developers will find certain differences in the emphasis and features of CEP development environments, but the differences in rule models are minimal and the other supporting models are often very useful. We will look at a simple example to demonstrate the sorts of additional features the "event processing" domain adds to rule processing.

Luke Voss: Mindviews, Founder

LV1: Building Domain Specific Language Languages for Rulebased Systems

Rule languages have developed into expressive and neat general purpose languages. However, this generality can make it difficult to understand the implementation of many business logic issues within a given context. Building a Domain Specific Language (DSL) on top of a more general rule language can provide powerful abstractions for designing and understanding systems within that domain. This white paper will focus on the practical: when a DSL is worth the cost, the steps needed to build a DSL, and the pitfalls that may derail a DSL development project. We will design and demonstrate a simple math DSL for performing algebraic manipulations and calculations that highlights the issues around DSL development.

Luke Voss: Mindviews, Founder
LV2: Graph Based Knowledge Bases and Rules
One useful abstraction in software systems is the graph consisting of a series of nodes and connecting edges. Rule systems developers can take advantage of this abstraction to build and understand complex relationships between pieces of information and construct pattern matching for structured data. This white paper will show how to use basic graph theory elements in a rule based system to efficiently traverse and pattern match against a knowledge base of facts (nodes) connected by relationships (edges). We will also show an example of a custom rule engine with facts and relationships built in as first-class concepts and show how metadata associated with the nodes and edges can be used to easily define and pattern-match against directed graphs, lists, and other structures.

Luke Voss: Mindviews, Founder
LV2: Designing a System of Rule Based Agents
Complex rule based systems typically use modules for managing separation of concerns and complexity within an application. This white paper will look at using smaller rule engines wrapped as agents as an alternative organizational strategy. The main advantage of using agents is in limiting the scope of any single rule base to a particular domain for better maintainability and modularity. In addition, parallelism and scaling become easier in a decoupled system using agents. The cost for such an arrangement then comes mostly from coordination issues. These issues include the framing of requests from one agent to another, the management of which agents can process which types of requests, and handling multiple asynchronous processes to accomplish a task. We will look at strategies and an example for decoupling a module-based rule system into a collection of interaction rule systems.

Andrew Waterman: El Colegio de la Frontera Sur, San Cristóbal de Las Casas, Professsor
Luis Garcia Barrios: El Colegio de la Frontera Sur, San Cristóbal de Las Casas, Professor
Playing With the Rules: Participatory Modeling and Network Gaming Through a Rules Engine

In this paper, we discuss our implementation of a networked application for the development and execution of multi-player/multi-agent games based upon abstract simulations of conditions present in tropical mountain watersheds with many takeholders; significantly influenced by our work in the Tablón watershed in the Sierra Madre de Chiapas. We explore how the technique of participatory modeling may be used in conjunction with a rules engine to create models whose interaction may be changed independently of their representation in the user interface. We discuss how a rules engine can be further used to construct agents that can participate and play. Our two games, "Manantiales de la Sierra" (Sierra Springs) and "Gente" (People) are discussed in technical detail. We briefly discuss our use of open-standards to handle connectivity, contention and data storage (Glassfish and JPA) and open-source technology (Drools, Flex3) to handle the fundamental logic of our games and to present our rich Internet application (RIA) for player registration, game segregation and play.

Charles Young: Solid Soft, Founder
A Survey of Complex-Event Processing Models

Prof. David Luckham defined an EPN (Event Processing Network) as a network of 'lightweight rules engines' which act as Event Processing Agents (EPAs). He contrasted this with the exploitation of rules-based inference engines as 'heavyweight EPAs'. Complex-event processing (CEP) is inherently rule-based and centres on pattern matching based, in large part, on temporal constraints. CEP, today, is broadly characterised by the use of diverse processing models embodied within different technologies. What are these models? What are their major differentiators, strengths and weaknesses, and how do they compare with Rete engines and other rules processing approaches? Are some models truly more 'lightweight' or 'heavyweight'? What are the underlying differences and similarities and how might each approach best be exploited in building scalable and agile EPNs?

This session provides a technical survey of the 'state of the art' regarding different CEP processing models. It describes, informs and explores, but avoids promoting any single approach or technology. Broadly, most CEP engines use either a relational data query model, often exploited through SQL-like languages, or an approach in which events are composed through the direct application of operators on event streams within different event consumption contexts. These approaches exhibit different and broadly complementary strengths and weaknesses. Rete engines are, in some ways, similar to engines that use data query models. However, they exhibit their own specific characteristics, strengths and weaknesses.

Many of today's CEP technologies promote the implementation of centralised architectures in which engines are connected to the external event sources and sinks through adapters. This contrasts with agile event-driven architectures envisaged by some CEP 'thought leaders' in which networks of EPAs are instantiated dynamically as needed across distributed environments and connected to each other and to sensor networks, external systems and services through the use of containers and service bus patterns. The session will relate different CEP processing models to these broader architectural concerns and approaches.

John Zachman: Zachman International, Founder
Dr. Leon A. Kappleman: University of North Texas, Professor
[See Leon Kappleman above]

October Rules Fest 2009 Tutorial Presentations

Abstracts Organized by Presentation Sequence

8:00 Rolando Hernandez (BizRules)
Manny Gandarillas
Introduction to Rule Harvesting and Knowledge Acquisition

This tutorial will show you a simple methodology for discovering, defining, and documenting business rules. Topics will include tools, tips, and techniques for eliciting rules from experts, documenting rules in your rulebook, designing the rulebases, and designing rule-based systems. Plus a discussion on how rules fit into the enterprise architecture, and how to design simple strategy rules and simple business rules.

9:00 James Owen (KnowledgeBased Systems Corporation)
Introduction to Rulebased Systems

While rulebased systems are considered part of AI, and therefore exotic, they are basically nothing more than another layer of abstraction on the same lines as a C or C++ compiler or an interpreter like Java. However, a rulebase is programmed declaratively rather than procedurally and requires a different thought process. This introduction will cover the history of rulebased systems, talk about some applications and introduce the various syntaxes used for a rulebase language.

10:00 Lawrence Terrill (EBDX.COM)
Introduction to the Rete Algorithm

For those new to Rete-based inference rule engines, the Rete algorithm can seem to be a black box. In this presentation, those new to the inference rule engine get a peek inside that black box to gain a basic understand of how the Rete Algorithm is typically implemented and the effects of changes in Working Memory to this stateful data structure. A small group of rules are introduced (in Drools drl) and a diagram of the resulting Rete data structure presented. The basics of the Discrimination Tree, Alpha memory, Beta memory, Join nodes, and Terminal nodes are discussed with this diagram. The effect of inserting a new object in to work memory is illustrated ith an updated diagram and the effective changes in the network and the agenda discussed.

At the end of the presentation, some practical implications of Rete approach are discussed supported by the previous diagrams. Beginners should take away both an increased appreciation of the Rete algorithm, the basics of its implementation, and some real-world implications on how they can write more efficient rules.

11:00 Greg Barton (Southwest Airlines)
Rule Testing Techniques: White Box, Gray Box, Black Box, and Crazy Box Testing

This tutorial will cover how standard software testing techniques (white, gray, and black box testing) apply to rule systems. It will also introduce a new class of testing, playfully named "crazy box." The idea is this: just as rules can handle intricate and complex problems, the testing of the rules can (and should) be just as complex. Solutions to crazy complex problems require crazy testing to ensure you don't go crazy after deployment.