

WGSigma Systems

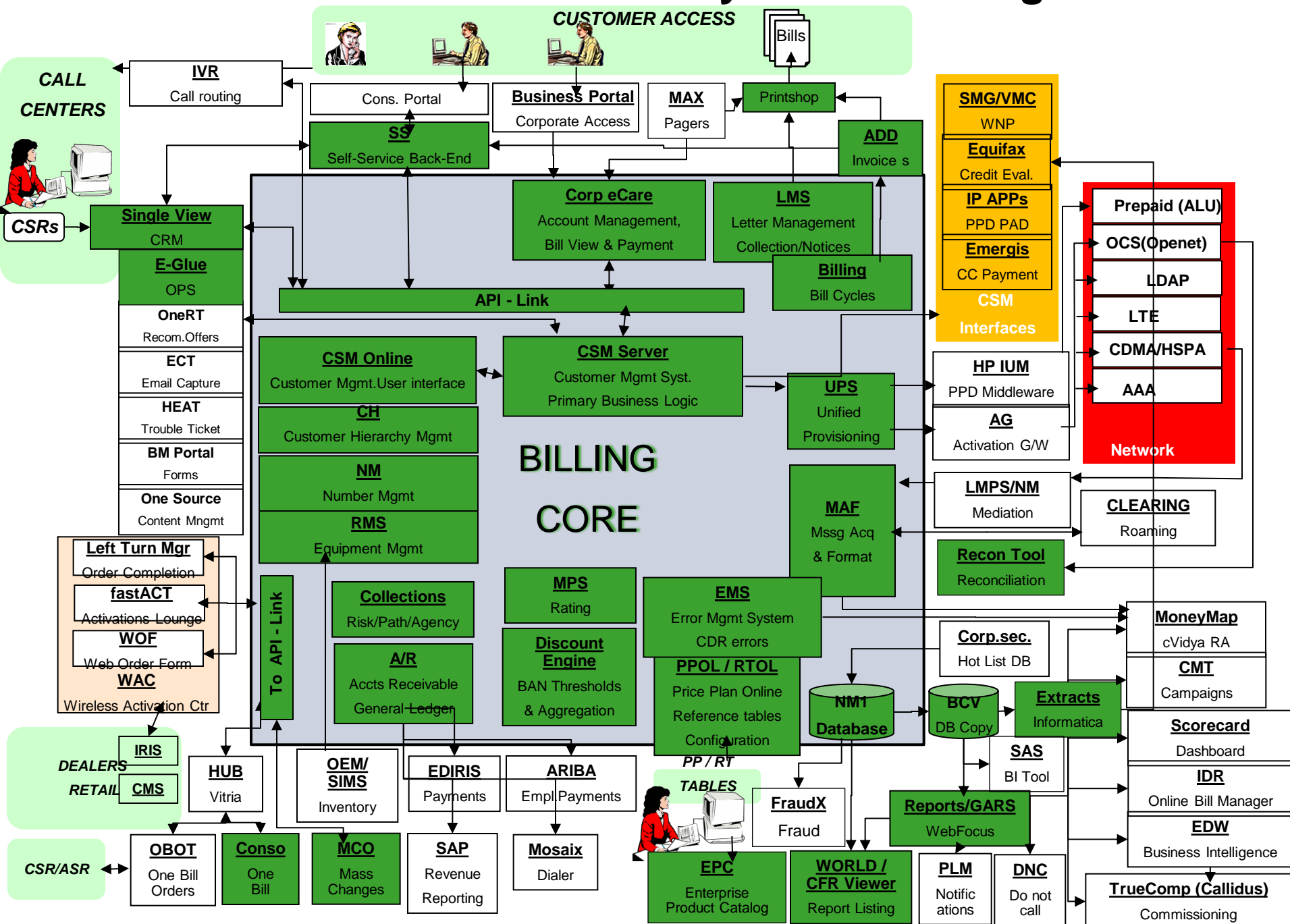
***Architecting intelligent real-time systems processing
billions of events a day***

William Guinn
SVP and CTO
San Jose, CA
william.guinn@wgsigmasytems.com

The Universal Corporate Challenge:

Hundreds or thousands of systems
performing transactions in silos with no
collective intelligence

About 20% of the Customer Systems in a large Telco



The Goal:

Create a collective knowledge of the operation where processing decisions are holistically optimized

....Patients

....Customers

...Subscribers

....Account Holders

Closed Loop Event – Condition – Action

Acquire

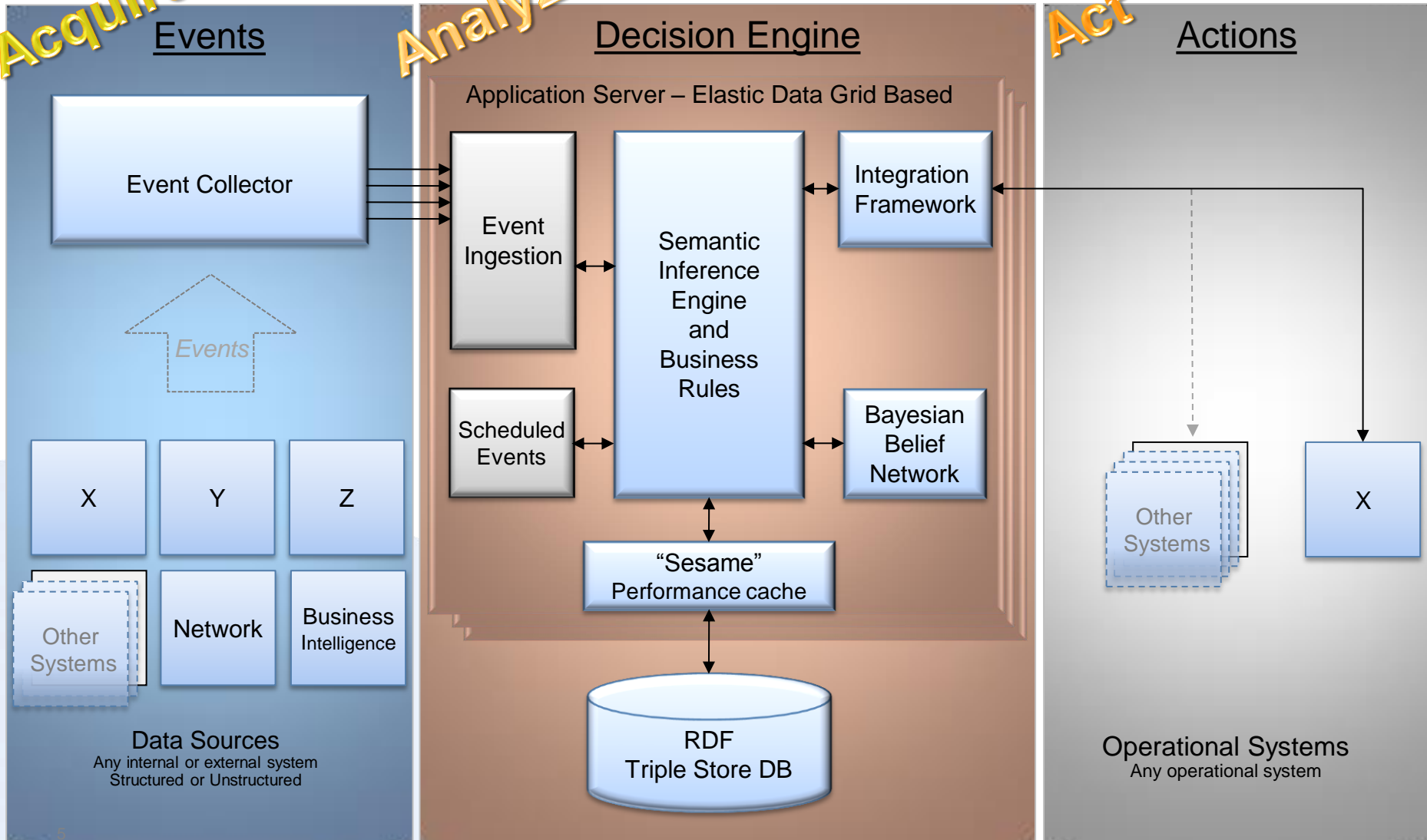
Events

Analyze

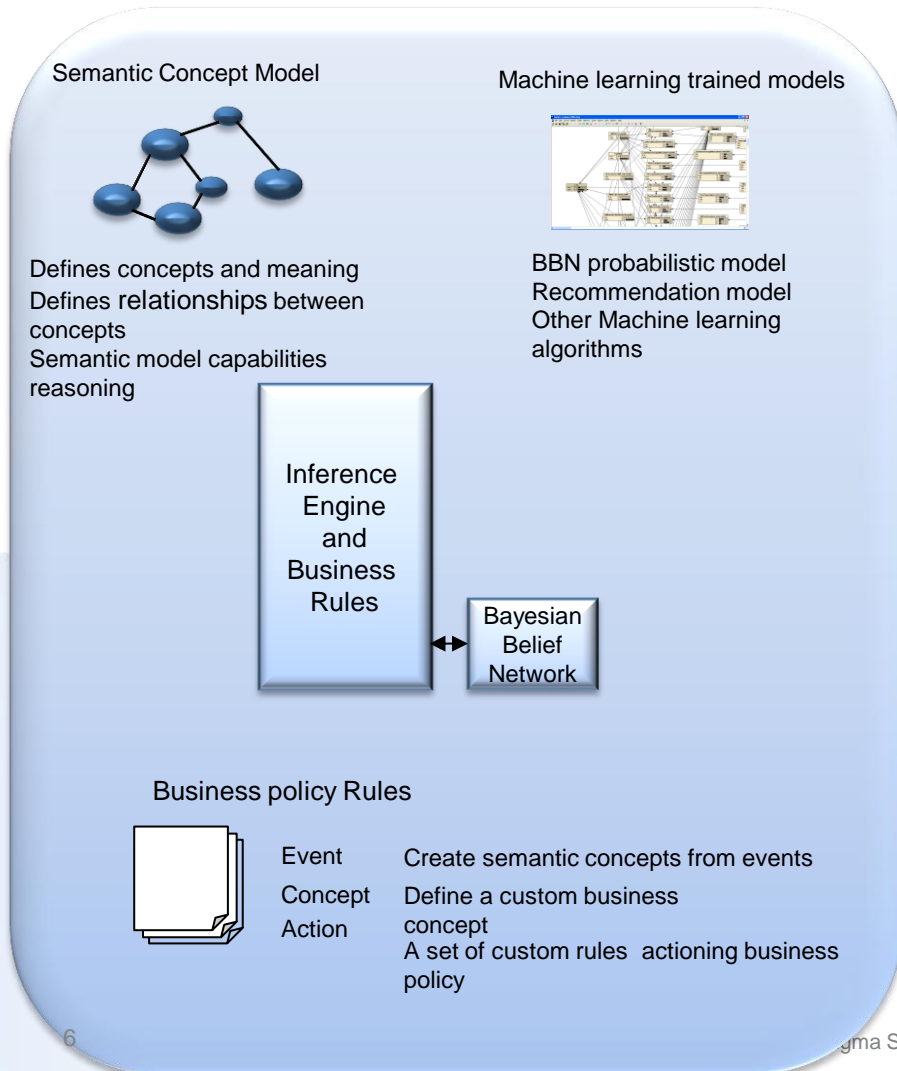
Decision Engine

Act

Actions



Inferencing



Semantic Concept Model

The model defines the concepts including the high level business concepts

The model contains the relationship between concepts including the dependencies

Inference

When an event occurs the event handler rule fires for that event
 Evaluates the event message
 Evaluates the existing ontology
 Determines which semantic instances to create or update

When any data changes, the inference engine fires in a “When - Then” style of computing, updating all “Automatic” concepts. Custom concept rules are fired if necessary. This creates a chain of updates

When a “on demand” concept is needed the inference engine finds and computes all of the dependant concepts

Machine learning

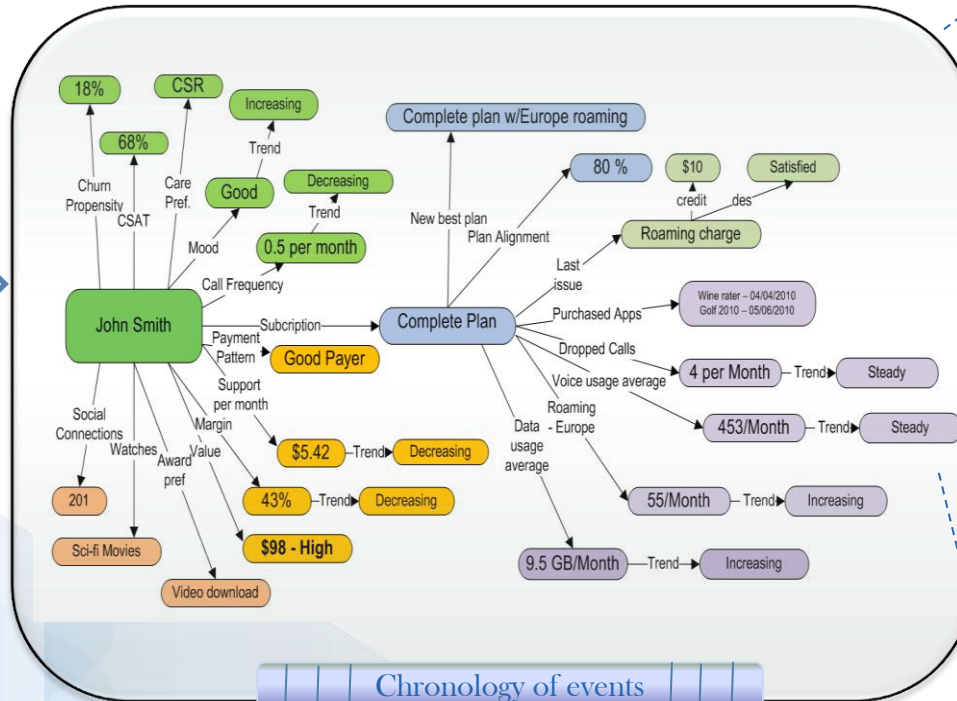
When a concept is dependent on “machine learned” information the inference engine manages the invocation and timing of interfacing

Use Case: Customer Interaction Prediction and Best Action

Events collected in real time

- Interactions
- Orders
- Bills
- Payments
- Collections
- Charge dispute
- Customer
- Pay instructions
- Individual
- Device Activated
- Device heartbeat
- Subscriptions
- Device changes

Transformed into a connected graph of business concepts



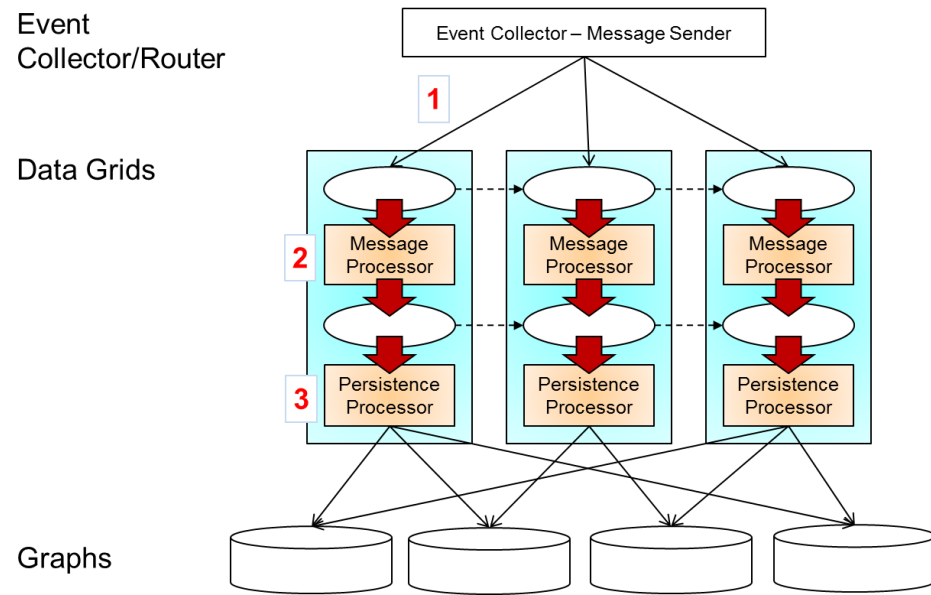
Predictions & Actions

- Probability of call motivation
- Plan Overage
 - Bill greater than last month
 - Prorates
 - Roaming charges
 - Third Party Charges
 - Abnormal fee
 - Rate increases
 - Charge dispute
 - First bill
 - Past due amount
 - Pay bill
 - Customer Cancellation
 - Reactivation
 - Device activation
 - Device education
 - Device exchange
 - Device Lost
 - Device not working
 - Device resume
 - Service data not working
 - Service text not working
 - Service voice not working
 - Subscription cancellation
 - Wrong plan

- | | |
|-----------------------|-------------------------------------|
| Subjective | "good payer" |
| Patterns | "always pays 2 days late" |
| Trends | "improving payer" |
| Geospatial | "within 5 miles of the tower" |
| Time | "within 5 minutes of an outage" |
| Probability | "probably will call about the bill" |
| Absence of occurrence | "missed payment" |
| Relationship between | "friend of a friend" |

Scalability & Resilience

- Use in memory elastic data grid
 - New hardware nodes can be added and removed dynamically
 - Data is replicated to one other node (no single point of failure)
 - If a server fails, the backup becomes the primary seamlessly
- Partition grid across objects
- Graphs are partitioned across multiple triple store instances
 - A single index instance defines the location of each graph
- Historical data is subdivided into graphs by timeframe (e.g. quarters)



Make it Business Friendly

The screenshot displays a business rule engine interface. On the left is a 'Library' pane for 'Composer Demo Application' containing a tree of 'Customer' entities and their attributes. The main workspace shows a 'Customer Payment Pattern' rule with a visual logic flow. The rule is titled 'Customer Payment Pattern' and includes metadata: 'LAST MODIFIED: 02/12/2010', 'MODIFIED BY: Greg Sloan', 'VERSION: 1.3', and 'COMPUTED: Automatically'. The description states: 'Determine the payment pattern for the customer by evaluating payments determining good payer, bad payer, worsening payer or improving payer'. The rule logic consists of five numbered conditions:

- Find all of the Previous Bills Within 6 months
- If all bills have Payment Timeliness equal to early then Payment Pattern is good payer
- otherwise If all bills have Payment Timeliness equal to late then Payment Pattern is bad payer
- otherwise If at least 75% of Earlier Bills are early or on time and later bills are late then Payment Pattern is worsening payer
- If At least 50% of Earlier Bills are Late and all later bills are On time or early then Payment Pattern is Improving payer

At the bottom left, a 'Customer' entity description is provided: 'Description. The role played by an Individual or Organization in a business relationship with the service provider in which they intend to buy, buy, or receive products or services from the service p'

Technology Stack

Technology	Purpose
TopBraid Composer	Ontology Modeling
Java	Event Pipeline, Decision Engine, Integrations
R	Statistical and Predictive Modeling
ETL	Event ingestion - Extract, Transform, Mapping Relational to Graph
Red Hat BPMS & Drools	Forward and backward chaining rules, process definition and control
Red Hat HornetQ	JMS Queue persistent multi-threaded input and output (publish-subscribe)
Gigaspace	Data Grid
Apache UIMA + Solr	Natural Language Processing, Semantic Searches, Content Analytics
Norsys Netica	BBN
Franz Allegrograph	Triple Store (RDFS)
Apache Cassandra	Time series data store