Semantic Real Time Intelligent Decision Automation

Bill Guinn
Amdocs Inc., 1104 Investment Boulevard
El Dorado Hills, CA 95762
www.amdocs.com
bill.guinn@amdocs.com

Jans Aasman
Franz Inc., 2201 Broadway, Suite 715
Oakland, CA 94612, USA
www.franz.com
ja@franz.com

Abstract: The Intelligence Community has systems that deal with overwhelming numbers of events that have to be analyzed in real time. Ideally these systems predict malicious events and aberrant human behavior far enough in advance so that appropriate action can be taken. The challenge is that events come from various real time sources and various databases and do not fit together well. The Intelligence Community (IC) has recognized that semantic technology might be of help in a number of ways. Semantic Technology helps in integrating event streams from various sources and it can describe the people, organizations, relationships, situations and threats in a more declarative way so that it is easier to disambiguate people and groups and it is easier to write rules and analytics to generate predictions. There are a number of attempts in the intelligence community to build Semantic Platforms that can do what we just described. There is also one system that does just this in a commercial setting.

Keywords: Event Processing, Bayesian Belief Networks, CRM, Telecommunications, Semantic Web, Rule Based System, Triple Store

1. Introduction

This article describes a Semantic Platform that is used in a Telecommunications setting, which deals with the same magnitude of complexity as the IC and definitely has the same massive scaling challenge: support 10 billion events per day with sub second response time.

2. AIDA - Amdocs Intelligent Decision Automation

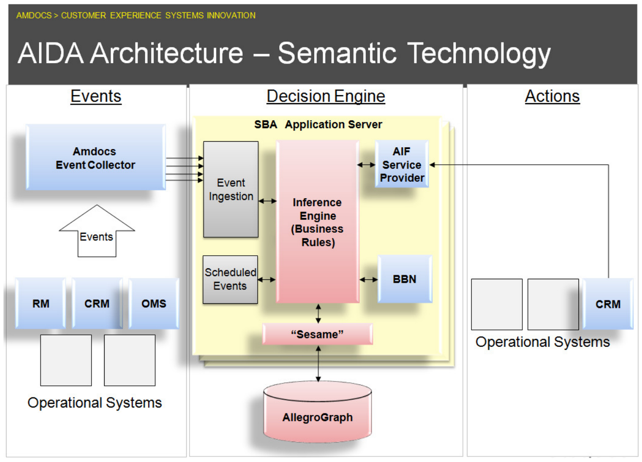
The AIDA Semantic Platform is built by Amdocs in cooperation with partners like Franz Inc. for the AllegroGraph triple store, Norsys Software for a Bayesian Belief Network system, and Gigaspaces Technologies for the Java middleware layer. Amdocs is a publicly traded company (NYSE: DOX) that is the market leader in customer experience systems for Telecommunications and Cable companies including Billing, CRM, Network Planning and Provisioning.

2.1 What is the platform used for?

The first use case that is currently being marketed by Amdocs is for improving customer care. Currently running a Call Center is one of the most costly elements in the total operational budget of any service oriented business. In North America the average cost of servicing a call with a Customer Service Representative (CSR) is $.50/minute. Large organizations which may have tens of thousands of CSRs, can save millions a year simply by shaving a few seconds off of their support calls. Studies showed that on average the CSR will go through 68 screens in one customer interaction. Wouldn't it be perfect that when you call, the CSR already knows what you are calling about and has the solution at hand? Both the Telecom and the customer win.

2.2 What are the elements of the platform?

Figure 1 shows some of the important elements: an Event Collector, a Decision Engine, the AllegroGraph triple store, a Bayesian Belief Network and a workbench for analysts and business people. Combined, this pipeline of technology implements an event-condition-action framework to drive business processing in real time. We'll discuss each of the platform's components in more detail.



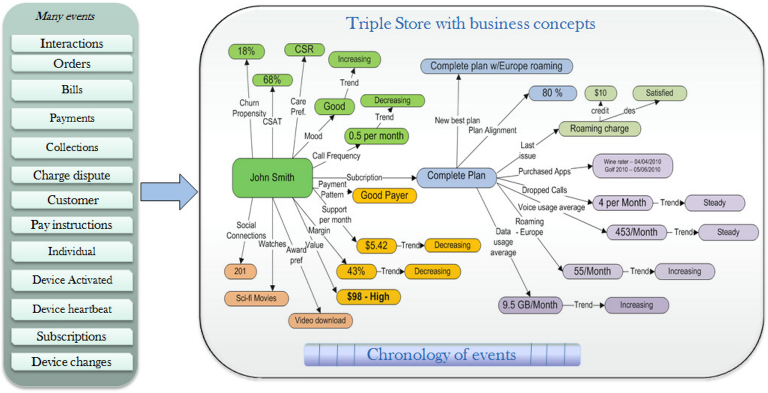
2.2.1 Event Collector

The Event Collector is a tool that is connected to all the systems that are usually involved in running the Telecom business. Examples of the data inflow are Call Detail Records, new and terminating subscriptions, outgoing bills, payments and collections, device heart beats, device changes, and network sensor data. However, there are also new sources of information: location based information, e-commerce events, sending of SMS and email, application downloads on iPhones or Android phones, etc, etc.

2.2.2 Decision Engine

The most important role of this component is [a] to compute over 100 high level business concepts that describe a customer in detail, and [b] to predict a customer behavior – as an example, when and why a customer will contact the Telco. The workflow is roughly as follows. First, the Event Collector receives a new event and encodes it as a set of RDF triples. Second, the Decision Engine then retrieves all the known triples about the subject of the event (for example a particular customer) and combines it with the new event triples and re-computes any affected high level business concepts. Part of this computation is done by invoking a Bayesian Belief Network (BBN). The BBN generates new predictions that are stored as triples. Figure 2 is an illustration of how events lead ultimately to a semantic network describing a customer. Note that figure 2 shows only a few percent of the total knowledge of a customer. Note also that this Decision Engine contains a rule based system that was developed by Amdocs. Due to their non-deterministic performance, RDFS or OWL reasoning was not used. Instead a custom forward and backward chaining rules engine was built to execute hundreds of inferences in milliseconds.

Events from many source systems are transformed into a set of related business concepts



2.2.3 Bayesian Belief Network

An important part of the platform is to predict behavior. Typical inferences are:

- Will the customer make a payment on time?
- Is there a reason (such as usage on a customer's bill) which will motivate the customer to call support?
- When is the customer likely to cancel service?

In other words the system predicts the likelihood of something happening estimated by the frequency of particular configurations of events and attributes. This can apply to a single individual or for a group of individuals that are roughly the same (for some statistical and semantic interpretation of 'same'). Just using ontologies and events represented as RDF is not sufficient to compute these likelihoods. One needs statistical and machine learning techniques combined with the knowledge expressed as triples to generate predictions.

BBNs provide a natural method for representing probabilistic dependencies among a set of variables and events. The BBN used in Amdocs' platform is an integral part of the Decision Engine.

2.2.4 Triple Store (AllegroGraph)

AllegroGraph fulfills a number of requirements that make it useful as an event database. Actually, a number of customers, even within DOD, use AllegroGraph in this capacity. Let us look at a simplified ontology of an event. We see that an event has the following:

- Hierarchy of types - Think of meetings, communications event, financial transactions, visit, attack/truce, an insurance claim, a purchase order, an observation. Working with types requires RDFS++ reasoning
- List of actors - Any of examples from the previous paragraph involve at least two actors, but usually more. In order to establish social networks, distances and strengths of connections between people and to find leaders in groups requires Social Network Analysis (SNA) techniques. AllegroGraph has an extensive SNA library.
- A place - Nearly all of the events above happen somewhere. AllegroGraph provides extremely efficient Geospatial indexing and GeoSpatial operators like proximity search.
- A time - All of the above events have a start time and possibly an end time or duration. AllegroGraph also provides efficient Temporal Indexing and Temporal Reasoning.

AllegroGraph does all of the above and works with billions of triples per machine instance. However, Amdocs takes AllegroGraph to a complete new level in terms of scalability. In Amdocs' AIDA Platform, all customer and device related events and even the predictions are stored as RDF triples in AllegroGraph. The scalability challenges are non-trivial. Consider a Telecom company with 80,000,000 customers. The typical number of events seen throughout the enterprise will total around 8 billion a day. Assuming each event generates on average 20 triples, you have generated a trillion triples in a few days.

Here are some of the new requirements for AllegroGraph in this platform:

- The triple store allows for a constant insert speed at more than 100,000 triples per second. All the triples need to remain indexed at all time and queries are interspersed with additions and deletions.
- The triple is transactional, even at high speed and high scale, and needs to be replicated at all times for scalability and high availability.

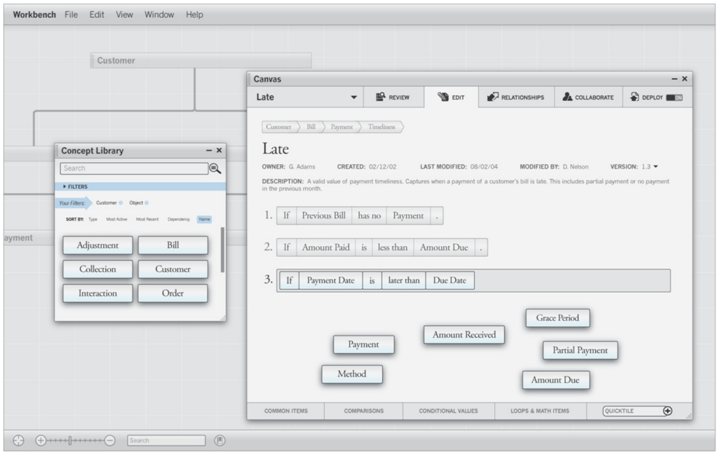
Currently AllegroGraph 4.0 can reach this scale by carefully partitioning the event data over multiple instances. However, in order to make this partitioning easier Franz is working on customizations that will offer High Availability that is completely distributed and can transparently deal with trillions of triples.

2.3 Actions

Once we have seen what is transpiring through the capture of an event, thought about what it means through inferencing and updating over 100 business concepts, we need to take the appropriate action. Using standard SOA integration schemes, the platform invokes the appropriate applications to drive the customer experience, and overall business processes. With this complete event-condition-action capability, a closed loop is created where the effect of our actions are seen in turn as new incoming events. This allows us to create goal oriented self-optimizing processes, and to learn from the actions taken under specific circumstances using backward chaining rules. This inherent feedback loop enables training and refining the BBN for better predictions along with honing the policy model which controls the business.

2.4 Workbench

A critical requirement for the AIDA Platform was that policies, rules, and the declaration of the ontology could be composed by analysts and business people who can't write code, and have no idea how to define an ontology, train a BBN, or write SPARQL. So Amdocs implemented a Workbench on top of their rule and decision engine which controls the declaration and configuration of the entire event-condition-action scheme. Figure 3 demonstrates a business user interface for a sophisticated "Magnetic Poetry", essentially a GUI to create if-then rules and policies. These if-then rules are translated into Java code (Figure 4) and SPARQL. When these rules are executed, new business concepts are generated and stored again as triples in the triple store.

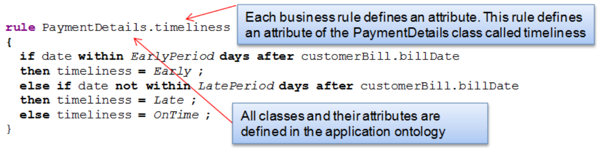


Late

OWNER: G. Adams CREATED: 02/12/02 LAST MODIFIED: 08/02/04 MODIFIED BY: D. Nelson VERSION: 1.3

DESCRIPTION: A valid value of payment timeliness. Captures when a payment of a customer's bill is late. This includes partial payment or no payment in the previous month.

1. If Previous Bill has no Payment .
2. If Amount Paid is less than Amount Due .
3. If Payment Date is later than Due Date



```
rule PaymentDetails.timeliness
{
  if date within EarlyPeriod days after customerBill.billDate
  then timeliness = Early ;
  else if date not within LatePeriod days after customerBill.billDate
  then timeliness = Late ;
  else timeliness = OnTime ;
}
```

Each business rule defines an attribute. This rule defines an attribute of the PaymentDetails class called timeliness

All classes and their attributes are defined in the application ontology

3. Conclusion

This paper provides a brief review of a commercially available Semantic Platform for handling over 10 billion events a day in a very low latency, high availability implementation. By monitoring customer interactions and their network activities, and predicting customer behavior we can take immediate action to ensure an optimized business and customer experience. We believe that there are many similarities with the workflow and processes within the Intelligence Community that should be explored.