Eliminating Information Overload
And Employing Event Reasoning

Established techniques and a proven solution to mine and correlate the knowledge critical to your needs
Eliminating Information Overload and Employing Event Reasoning

Information overload is a serious problem facing many facets of government and industry today, from military command and control systems (C2) to warehousing applications to large-scale ordering and fulfillment systems. A recent study conducted by the University of California, Berkeley (2003) concluded that the worldwide production of information in one year was close to 5.6 Terabytes – a 75% increase from just three years prior†.

In addition to gathering pertinent information, you want to take advantage of event reasoning to produce metadata. By “connecting the dots”, metadata gives you the power to discover hidden insights, provide real-time/near real-time solutions to problems and to forecast future needs. This is the information that keeps you a step ahead of your business’s competitors or your nation’s enemies.

The following sections outline some proven architectural paradigms that combat information overload and assist in the enterprise correlation process.

Targeting and Filtering

In systems with large amounts of data, it is critical to perform as much filtering as possible through the sensors/nodes that gather the initial data. Blindly collecting and passing data to other layers or nodes in the framework bypasses an excellent opportunity to weed out that information that will be of no use to end users. Developing sensors/nodes that have the capability to either:

- follow guidelines as to what data to gather (i.e. rule-based software agents)
- “learn” what data to amass based on an inference engine (i.e. artificial intelligence)

Filtering provides an extremely scalable and robust architecture as data consumption increases in rate and size.

Targeting is the concept of passing a subset of information on to only those users who will somehow benefit from it. Multicasting all knowledge to all users can quickly overwhelm most systems and is unnecessary if there is a means to determine which parties have an interest in the data. Targeting can be a daunting task but there are several methods to implement it:

**Role Assignment**
By assigning a user a role, the system has a set of criteria by which to determine whether or not a user is privileged to certain information. Typical roles are centered on rank (General, Major, Lieutenant, CEO, Senior VP, Manager, etc…). A simple assignment of a role to a user can be accomplished via a mandatory login where the user’s role (and other information) is applied at each login.

**Localities**
Often users are located in the same physical vicinity, such as a SCIF or a regional sales office. These consumers can have the same interest in particular pieces of knowledge if their tasks and goals are the same. This concept can be applied at the network level as well – those users that share an IP subnet or a virtual private network may have the same interest in some set of data regardless of their physical location.

**Publish/Subscribe**
By advertising its abilities and/or knowledge capabilities, sensors/nodes allow users (or other sensors/nodes) to discriminate what data they would like to access. This methodology is described in more detail in the next section.

**Publish/Subscribe**
In the previous section, the publish/subscribe architecture was mentioned in the limited scope of targeting relevant information to a given user(s). However, this concept is so straightforward that its application has a very broad scope.
Eliminating Information Overload and Employing Event Reasoning

The idea is as follows: a component in the system (i.e. producer) broadcasts what data it has access to in a well-defined format. Other components in the system (i.e. consumers) receive this information and decide what, if any, of the data they want. The consumers pass their well-defined request back to the producer. At this point, a stream of data can pass between components. The advantages of this design scheme are extremely beneficial:

- Reduces network congestion
- Promotes scalability
- Enables dynamic management/maintenance
Knowledge Correlation

How data is correlated is highly dependent on the data itself. Listed below are some well-known methods for correlating information.

**Pattern Matching**
Pattern matching is the simplest correlation technique. By finding two (or more) types of data that match (e.g. IP address, proper names, etc…), the system can create a relationship. However, without further rules or guidance, this method can be very resource intensive and ineffective in certain situations.

**Causal Relationships**
Causal relations are based on cause-and-effect associations. For example, a known terrorist is on a flight that arrives in New York City. Less than an hour after the plane’s landing, a hostage situation occurs within a small distance of the airport. It may be appropriate to relate these events by postulating that the terrorist is the cause of the hostage scenario.

**Temporal Relationships**
Temporal relations are based on the premise that the appearance of certain events within a given time period increase the likelihood that they are related to one another. Correlating various IP packet-based attacks to a single DDoS incident is a prime example.

In addition, the correlation engine should store validated correlations (i.e. those relationships which domain experts have asserted are correct) in such a way that the system actively searches for these relationships and quickly notifies the user (via alarms, triggers, etc…) in the future. Thus the system can proactively reduce the time between event occurrence and actions taken.
The Solution: iAgent™

Perceiving the strong demand for information relevance and event reasoning, JBISoft developed Intelligent Agent, or iAgent™. iAgent is a Java-based software application based on a federation of smart agents and a user console.

Smart agents are software components that perform a particular task. Smart agents are usually tasked to monitor and/or retrieve data from any given data source (RDBMS, OODBMS, text files, real-time sensors, web services, XML, legacy systems, etc…). However, agents can perform any task, such as data collection.

Service-Oriented Architectures (SOAs) are frameworks based on a loosely coupled collection of interoperable software services. Since these services are not tightly coupled, services can easily be swapped and overlying processes can quickly change as well.

Dynamic Programming is a set of methods that solve problems by splitting the original problem into sub-problems. It is akin to divide-and-conquer techniques but here only optimal solutions are used in successive iterations.

Smart agents, positioned throughout a network in a distributed fashion, feed their knowledge to all interested console-based users. In addition, agents can communicate with each other, passing metadata that enables them to improve their ability to quickly gather pertinent data (see Figure 2). Using the publish/subscribe paradigm, iAgent adheres to a Service-Oriented Architecture (SOA) giving users a robust, scalable and dynamic framework strong enough to handle even the largest parcels of information.

However, iAgent’s advantages are not limited to how and what data it harvests. Correlating the information into meaningful relationships is one of the core capabilities of iAgent. Using proprietary pattern matching, dynamic programming and other techniques, iAgent automates the effort of enterprise
correlation. This powerful functionality liberates users from hours of manual search-and-associate tasking in addition to allowing decision makers to act on information faster than ever before.

**How iAgent Works**

Smart agents can be either built around custom requirements (e.g. retrieving information from a proprietary satellite imagery database) or redeployed based on extant designs suitable in most applications (e.g. a weather agent that acquires weather statistics from NOAA in real time). Our agents are based on a flexible object-oriented schema allowing rapid development and deployment based on your needs.

The console is a graphical user interface consisting of visualization tools that enable your users to conceptualize, understand and act on knowledge without becoming consumed by it. The console is available as a thick client, an applet or as a portal. Below is a small sampling of our graphical tools:

- **Maps**
  - topological, nautical, aerial
  - ESRI-compliant, lat/long sensitive
- **Video/Audio Feeds**
- **Statistics**
  - Pie/Gantt charts, Histograms
- **IP Geolocation**
- **Chat/Collaboration**
- **Smart Agent Inspector**
- **Ontological Hierarchy**

iAgent’s toolset is based on an adaptable plug-in design. Your users only install those visualization tools that are applicable to your business, keeping processes and workload streamlined. As your business changes, users can install and uninstall plug-ins with a minimum of disruption.
Eliminating Information Overload and Employing Event Reasoning

Figure 3 - Snapshots of iAgent's Toolsets
Eliminating Information Overload and Employing Event Reasoning

Since the entire code base was written in Java, iAgent benefits from all of Java’s advantages: solid security, a network-centric schema and platform independence. Currently, iAgent is compliant with J2SE 5.0 and will install with the JRE if necessary, allowing existing software to run without unwanted side effects. JBISoft is exploring the possibility of porting iAgent to JXTA or JMS as the underlying communication framework to increase compatibility with external systems.

Further Information

JBISoft is a non-traditional, software and service provider specializing in knowledge comprehension, information assurance protection, event reasoning and enterprise correlation. Since 2000, JBISoft has delivered solutions to commercial and governmental entities in a variety of engineering areas.

If you would like to more information on iAgent or any of JBISoft’s products, please visit our website at www.jbisoft.com or email us at sales@jbisoft.com.