

Hypothesis Representation for Processes and Threats

Kevin B. Pratt
Essex Corporation
1235 Evans Road
Melbourne, Florida 32904, USA

Dorota K. Woodbury
Essex Corporation
1235 Evans Road
Melbourne, Florida 32904, USA

ABSTRACT: *We describe novel methods for highly adaptable representation of hypotheses about the complex temporal patterns typical of processes and threats. These methods permit expression and preservation of hypotheses about normal and abnormal processes of significant subtlety and complexity. The hypotheses can capture institutional domain knowledge and facilitate sharing across the organization. A hypothesis can contain as components anomalies, mixed data types, text, missing, and partial information, and various temporal relations among groups of those components. Each component can be declared as necessary for, or sufficient to enhance or diminish a hypothesis. As new information is received daily in a data warehouse, we search for evidence, including new and previously acquired facts that may partially support a hypothesis about an undesirable situation or process. This allows us to detect yet unconsummated processes in order to provide “first alert” to emerging threats. A production system containing these methods has been delivered to a government customer.*

KEYWORDS: Hypothesis search, knowledge representation, process specification, emerging threat, complex temporal pattern.

1. INTRODUCTION

We designed, built and delivered a multi-part system designed to find emerging patterns in a large data warehouse where data is constantly being added. The first part of the system enables an analyst to express a hypothesis about a complex temporal process. Typically the hypothesis is about how factors and predicate events can or do build up to or presage a “bad” event. The hypothesis is then retained in a repository. The second part of the system uses the hypothesis as a framework for searching for data in the warehouse that would support the hypothesis, fully or partially. The third part of the system alerts the analyst when there is enough supporting evidence that an instance of the hypothesized process is occurring or is predicted to occur and provides a report on that evidence. The fourth part of the system provides a means to manage the system – to easily revise, expire, and share hypotheses, alerts and reports to various groups of users. Graphic user interfaces were provided for ease of use. A production system containing these methods has been delivered to a government customer.

This article focuses on the part of the system that provides the user with the ability to express a hypothesis about complex temporal processes. It also briefly describes how

the system uses the hypothesis to detect emerging patterns so that early alerts can be provided to users. The benefit of an early alert is that it may be possible either to intervene before or prepare for a consummating “bad” event.

The hypotheses of domain experts within an organization constitute an important part of the institutional knowledge of the organization. Sometimes those hypotheses are historically well substantiated, sometimes those are “educated guesses,” and sometimes those are loose intuitions from single anecdotes. A hypothesis may or may not assert “causation.” It seldom fully specifies the process. A hypothesis may embody a poorly understood structure of “suspicious coincidences” or rough conjectures.

It is important to have a way to capture, preserve and share institutional knowledge so that knowledge assets are not lost when personnel retire or transfer. Also greater benefit and lower personnel costs result when an automated system can constantly test whether new evidence (combined with older evidence) may now support a hypothesis about a pattern of interest. An automated system does not become fatigued by tedium, overlook a component of a hypothesis, or forget to share an insight with collaborators.

By “process” we mean a somewhat structured or sequenced set of facts or events, or groups of sets of facts and events that may exist before or around a consummating event. We use “process” to mean a structure of behavior such as that in worldwide logistics for product delivery or such as collaborative human behaviors in constructing a terrorist attack incident. We do not mean “process” narrowly, such as the deterministic procedure followed by a manufacturing machine to make a machined part.

Our contribution is a highly flexible and expressive method for a subject matter expert to state a hypothesis about complex processes. This novel method provides representation of a broad range of complex temporal patterns. Mixed data types, text, anomalies, missing, and partial information can each be necessary components of, or can enhance or diminish a complex temporal pattern. We are unaware of other methods of expression that capture this range of loose intuition, yet support deterministic search in a body of evidence such as a data warehouse.

Similar needs to preserve institutional knowledge and hypotheses about complex processes arise in logistics, biomedical research, social policy making, educational planning, political campaigns, military tactics, and anti-terrorism response.

Example: Consider an international consumer products company. Over many years, various subject matter experts within the company have gained insights into complex processes that are proceeding smoothly versus processes that are moving toward problems or disasters. (We do not mean a problem evidenced by simple trends in indicator values, but rather complex combinations of facts and events that can come together to create problem situations.)

Formal preservation of this knowledge is inhibited by the difficulty of representing these intuitions. The hypotheses used by the experts may reflect loosely organized, partial information, may involve a mix of normal and anomalous events or episodes, and often involve imprecise temporal notions. Fig. 1 provides an example of such hypothesis. Fig. 2 shows a tree representation of the hypothesis from Fig. 1.

When there is some anomaly with the Italian suppliers of motherboards in the spring, and then the German assembly plant shuts down entirely for August holidays and about the same time autumn typhoons interfere with air transport flights out of Southeast Asia, there will be numerous consumer returns of our products delivered to retailers in the US during December unless we adjust production by adding factory employees to the assembly department at the German plant in the period just before December. We also believe there will be fewer returns if we use clear packaging on these products.

Figure 1. An example of a domain expert's intuition (hypothesis) about a complex temporal process. Note that while the hypothesis may or may not be historically supported by a statistically valid set of actual incidents, the fact that the hypothesis is stated by an experienced, knowledgeable and insightful domain expert recommends that its possible occurrence be investigated further.

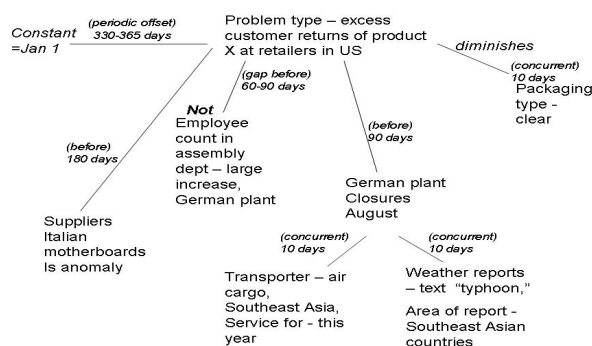


Figure 2. Tree Representation of Complex Temporal Pattern Composed of Simple Patterns

Consider also that the company in the example has built a data warehouse for its operations, containing reports and data collected about its suppliers, manufacturing methods, plants, supplies, quality and production. The warehouse data is updated daily. Although the warehouse is commonly used for reporting on status and events, it has been also planned to

include and accumulate the kind of facts that the domain experts use when stating their hypotheses, and when applying their expertise to anticipate threats to smooth operations.

An advanced data warehouse is often supported by servient components, such as those that extract salient concepts from text, resolve entity references, generate statistical aggregations from the atomic data in the warehouse, extract economic back-ground information from the internet, and identify trends and anomalies in the behaviors of the company's own operations and those outside parties that it relies on. Consider that the company has this type of warehouse and these derived facts are being populated in near real-time as data streams into the warehouse. This means that, for example, a hypothesis can use as a component fact the fact that a particular variety of anomaly has occurred or that customers that were previously thought to be different have now been reconciled as one.

2. PRIOR WORK

Research on how to represent complex processes proceeds in diverse fields. We mention recent work in XML-based ontological models, including Process Specification Language (PSL) and Knowledge Interchange Format (KIF), in representation of institutional memory in Enterprise Resource Planning (ERP) systems such as SAP, in engineering manufacturing processes, in robotic navigation, in symbolic-linguistic representations of natural language conversation processes, and general pattern deformation theory. XML has been extended by PSL to specify processes [3, 9]. PSL utilizes KIF and Unified Modeling Language (UML) representations, and is defined in first-order logic. It is intended to comprehensively define all possible traces of a described process, and employs sub-trees for component processes [3]. PSL uses crisp constraint rules to accept legal branches in a process. A key strength of PSL is its foundation on well understood mathematical structures, and its usefulness in automated reasoning. Three significant PSL limitations are its implicit requirement of an *a priori* omniscient world view that will allow statements of constraints that can legalize possible branches, its orientation to forward directional process trees, and its weakness in representing homogeneous temporal episodes.

Representation of institutional memory has been an important function of ERP systems such as SAP [13, 15]. The focus has been on the representation of "explicit" models of work flows necessary to achieve institutional goals. Intuitions have been hard to capture [11, 13]. Such tacit knowledge is highly personal and hard to formalize, making it difficult to communicate or to share with others without a process of knowledge conversion from tacit to explicit representation [12]. The ERP representations assume that the goal is to describe a comprehensive and detailed sequence leading to an achievement so that the business

process can be reliably repeated. Necessarily, ERP representation methods are not intended to capture incomplete knowledge, abnormalities, and un-ordered events and episodes.

One approach to modeling of partially understood health and chemical processes, specifically trends [14], has relied on representations based on “landmark sequences” and fuzzy logic to adapt to partial information and varying granularities of time and knowledge detail [4, 5]. The technique essentially creates dynamic multivariate symbols that participate in a grammar containing transformation constraints. Temporal episodes containing events of unknown ordering can be accommodated, but the focus is on describing sequence based patterns. While some processes can be represented as a pattern deformation sequence [4, 6], that representation does not accommodate incidental parallelisms, merges of threads, or topological discontinuities.

The need to represent route finding and re-route processes in an unknown environment has been approached with probabilistic, graph theoretic, and Markov methods, including recent hidden semi-Markov models, relational Markov models, and dynamic probabilistic relational models in robotics and assisted cognition systems [8, 10]. One promising approach has focused on environmental state recognition using layers of concepts of varying granularity. To meet real-time performance goals, both *a priori* and acquired intuitions about loosely specified “situations” called *frames* are used to facilitate decision making [8].

Natural language processing confronts difficult issues of representing conversation processes and representation of symbols, such as the incompletely specified concepts internalized by communicators [7]. Extraction of meaning from strings of words using unorganized lexeme sets is limited by the failure to represent structure. However, many syntax and grammar based structures reflect more the way that speech is structured than how the concepts being discussed are organized. Many standard tools of computational linguistics such as concept frequencies, Markov methods, BBNs and graph theoretic methods assume crisp logic and complete *a priori* knowledge of a static concept and constraint ontology applicable to the discussion domain [7].

One of the more difficult issues in process representation is expressive representation of the varieties of appropriate time concepts. Allen [1, 2] has cataloged concepts of time and investigated the relationships among homogeneous episodes, heterogeneous episodes and atomic events, as well as related issues of time granularity. We selected a subset of Allen’s concepts to represent four key concepts needed for describing complex temporal patterns in our problem domain: periodic recurrence, concurrence, before, and gap-before. These are defined below.

3. REPRESENTATION OF HYPOTHESES ABOUT COMPLEX TEMPORAL PATTERNS

Humans are able to describe and recognize complex process patterns. These activities involve high level cognition, symbolic use of unresolved ambiguity and imprecise temporal constructs. In order for a subject matter expert to work effectively with a system that might assist in the representation of hypotheses about complex processes, the system should offer significant expressiveness.

On the other hand, the representations must meet other usability and computability criteria:

- In order to prevent miscommunications between users and within the system, a representation must not be ambiguous,
- The representation must permit deterministic searches for its existence in a data warehouse
- For ease of use, the set of relations among parts of a pattern should be small and intuitive
- Cyclic or otherwise non-computable representations must be illegal.

We satisfied these criteria by using the small number of constraints, discussed below, and by embedding those constraints in the graphic user interface through which the user constructed a hypothesis, that is a representation of a process.

The user builds a complex pattern by defining simple patterns, and then constructing a tree that has simple patterns as nodes, and temporal relations as edges. The root node is the concluding simple pattern. Often, but not necessarily, the concluding simple pattern of a complex pattern is a partially specified multi-variate outcome. The user also assigns importance to patterns. We will discuss how the user constructs patterns in Section 4.

3.1 Expressiveness

In order to provide sufficient expressiveness, we chose to offer a mix of “types of knowledge” representation (Table 1). Crisp and fuzzy representations are likely familiar to readers, so we discuss the remainder.

The representation of context relativity (Table 1) relate to objects populated in the data warehouse by other warehouse components that calculate the degree of normalcy or anomaly of events in a large set of contexts. This allows the user to select, for example, that a pattern includes “a moderate anomaly in the context of supplier A’s scheduling of air transport delivery to the Paris factory on Fridays” or that a pattern includes more generally “any anomaly at the Paris factory in a time frame.”

In evaluation of evidence of a hypothesis, we observe that a particular piece of evidence can be a necessary component of a pattern, but that in other instances a piece of evidence enhances or diminishes our belief in a pattern. For example, if a criminal runs from a crime scene, it increases our belief

Representation Type	Examples
Objects	Discrete-named, collections, value ranges, missing
Crisp (Boolean) operators	Necessity, exclusion (necessarily not), conjunction, disjunction
Context relativity	Normal activity, anomalous situations
Evidentiary significance	Necessary, necessarily not, enhancing, diminishing
Fuzzy/approximate/similar	Fuzzification, range, text similarity, soundex, object similarity
Temporal objects	Event, homogeneous episode
Temporal relations	Before, gap-before, concurrent, periodic offset
Importance	A level of criticality that prioritizes a user's action

that he is guilty, but running away is not a necessary element of crime under the law. We believe that the ability to include in a hypothesis enhancing or diminishing events or episodes significantly increases the expressiveness and usefulness of hypotheses.

In representation of processes, we needed to represent both atomic events and homogeneous episodes. Homogeneous episodes (where any atomic events within the episode are not ordered) were necessary both as a useful abstraction, and to accommodate missing data or attributes, temporal imprecision of reports, or un or partially-ordered collections of atomic events. In the data warehouse, an episode can have an identifier, and facts that have no date can be linked to an episode using its identifier. However, an episode must contain at least one datum with a date stamp. In some data a date stamp is only an estimate.

3.2 Temporal Relations

The common representation of a single temporal relation, sequence, is insufficient when homogeneous episodes are

part of the domain of discourse. Sequence is also insufficient for effective abstraction of periodicity relative to a recurring benchmark, such as the notion of “every summer.” Further, human speech is often ambiguous with respect to relations of episodes.

We concluded that four temporal relations would provide sufficient expressivity for our domain. To resolve ambiguity among users and prevent miscommunication with the data warehouse system, we provided definitions for the temporal notions of *before*, *gap before*, *concurrent*, and *periodic offset*. (Table 2)

In the following discussion, we use “child” exclusively to refer to placement in the temporal relation tree relative to another node. A child does not inherit any attributes of its parent. Also, a child generally *precedes* a parent in time, as is common in backward planning notation.

3.3 Syntactic Constraints on Representations

In order to prevent miscommunication, we applied a small set of constraints to permissible complex expressions (Fig. 3). Because every complex pattern is modeled as a tree, the constraints can be stated easily as rules of tree construction. These are enforced as part of input validation during the user's pattern construction in the graphical user interface.

The first constraint in Fig. 3 enforces the notion that a complex pattern requires some relationship among all its constituent parts. The second rule, of single parentage, has the beneficial effect of preventing cycles. The “NOT” rule improves tractability by disallowing branches below nodes of potentially great breadth. The enhancing/minimizing rule enforces the intuition that normally these are leaf concepts, and prevents quandaries such as, “how is evidence of a parent affected when there is minimizing grand-child of an enhancing child.” The rule that periodic offset applies only to root nodes assures that a complex pattern is internally temporally consistent. Finally, the “OR” rule improves

The temporal objects child <i>C</i> and parent <i>P</i> may be events or episodes. Every event or episode must have at least one date associated with a datum that is a component of that event or episode. A child never follows a parent in time, except possibly in a minimal way in a <i>concurrent</i> relationship.	
Temporal Relation	Definition
<i>Before</i> – example: <i>C</i> before <i>P</i> and <i>C</i> within <i>d</i> of <i>P</i> .	All dates of data in <i>C</i> precede any date of a datum in <i>P</i> and some date of a datum in <i>C</i> is <i>d</i> days before the earliest datum in <i>P</i> . (There is no overlap of dates in <i>C</i> and <i>P</i> .)
<i>Gap before</i> - example: <i>C</i> between <i>n</i> and <i>m</i> days before <i>P</i>	Some date of a datum in <i>C</i> meets the dual constraints that it at least <i>m</i> and not more than <i>n</i> days before the date of some datum in <i>P</i> . (An overlap of dates in <i>C</i> and <i>P</i> is possible; however the user's choice of <i>n</i> and <i>m</i> generally make an overlap unlikely and therefore consistent with the intuition of “gap before.”)
<i>Concurrent</i> – example: <i>C</i> is within +/- <i>n</i> days of <i>P</i> .	Some date of a datum in <i>C</i> meets the dual constraints that it is no more than <i>n</i> days before the earliest and no more than <i>n</i> days after the latest date of a datum in <i>P</i> . (Notion: There is similarity of dates in <i>C</i> and <i>P</i> .)
<i>Periodic offset</i> – example: <i>P</i> is <i>n</i> to <i>n+d</i> days after a recurring calendar benchmark (e.g. January 1)	Some date of a datum in <i>P</i> occurs between <i>n</i> and <i>n+d</i> days after the calendar benchmark.

A complex pattern is represented as a single rooted tree, where each node is a simple pattern and the edge between a child and its parent node is a temporal relation.

1. Each child pattern has a single parent (however a child can be cloned and separately instantiated when needed so that the clone can be attached to a different parent).
2. A “NOT” pattern must be a leaf.
3. An “enhancing” or “minimizing” pattern must be a leaf.
4. Only the Root Pattern may have a periodic offset.
5. If an “OR” relation between siblings is desired, the complex pattern must be cloned and the second sibling substituted for the original.

Figure 3. Constraints on a Complex Pattern Specification

search tractability, as explained below, and forces the user to think through whether the use of OR correctly cascades down the tree consistent with the intended representation.

3.4 Complex Hypothesis Representation as Tree

The complex temporal pattern that composes a hypothesis consists of simple patterns linked by temporal relations, organized as a single rooted – single parent tree structure. The simple patterns are nodes in the tree, and can be considered representations of state, event or episode. Because simple patterns can contain one or many features that have numeric, categorical, binary, date, location, anomaly, text, collection, range, or missing types, a simple pattern is flexible enough to represent the state of an arbitrary concept in the domain of the data warehouse. However, a simple pattern representation need *not* comprehensively specify a state. Many features may be left as un-specified in a simple pattern statement. By using sequences of objects, multi-variate non-linear transitions or transforms across all data types can be represented.

Importantly, the temporal relations allow representation of processes of many steps, but at the same time is robust with respect to missing or unimportant ordering of simple patterns in various sub-trees. In other words, the complex pattern can express parallel threads of temporal objects that eventually relate to a concluding outcome, where some threads may be strongly sequential, and others may not need ordering.

3.5 Formal Representation of a Complex Pattern

A user specifies a complex pattern by writing one or more statements linking a child simple pattern to a parent simple pattern in accord with the formal notation in Fig. 4. The simple example of a use of this statement is to represent that a child occurs before its parent.

The availability of the *not* qualifier on *child pattern* adds capability to the complex pattern definitions. The simplest, and most common use of *not* will be to specify that *parent pattern* occurs but simple *child pattern* does *not* occur within a concurrent time window. However, the use of *not* in complex patterns can be ambiguous to users, and requires enforcement of a precise definition.

< [optional *not* and *period_of_interest_for_not*], *name_of_child_pattern*, *temporal_relation*, *relation_days*, *name_of_parent_pattern*, [optional for leaf child *enhancing* statement], [optional for root pattern *minimum annual offset* and *maximum annual offset*] > where:

not indicates that the child pattern, if it exists at all, must necessarily not have the *temporal relation* with the parent pattern.

period_of_interest_for_not defines the number of days to search when the use of *not* requires a search

name_of_child_pattern refers to a simple pattern

temporal_relation is one of {concurrent, before, gap_before}

relation_days is one of {an integer days for *concurrent*, null for *before*, an integer days pair for *gap_before*}

name_of_parent_pattern refers to a simple pattern

enhancing is either enhancing or minimizing, and indicates how the child pattern and its relation to the parent pattern affects the strength of belief in the parent pattern

minimum annual_offset is the least number of days after a periodic starting date

maximum annual_offset is the greatest number of days after a periodic starting date

Figure 4. Formal Notation for Constructing a Complex Hypothesis

Definition: *not* means there exists no set of data in the warehouse that matches the child simple pattern in the time frame specified by the temporal relation relative to the time frame of the parent simple pattern. With respect to the relation “*not B before A*,” note that the requirement of a default *d* scope parameter in the temporal relation *before* (Table 2) prevents an unbounded search space.

3.6 Representation of Importance

We found the experts ascribe varying importance to their hypotheses, and to the various simple patterns that compose the complex process. Importance is the level of criticality of the process to the domain expert’s job responsibilities. Importance provides a prioritization for situations needing response or intervention. In the delivered system, several diverse experts may be interested in a process, but each may assign different importance, depending on their responsibilities.

4. IMPLEMENTATION OF HYPOTHESIS AND SEARCH

We describe how complex process patterns are constructed, and how incoming data is searched so as to provide early alerts to emerging process patterns.

4.1 Pattern Construction

Process pattern construction proceeds by the expert first defining one or more simple patterns and then organizing those simple patterns into temporal relation trees. A simple pattern is an encapsulated component that can be used in more than one complex pattern. Each simple pattern has an identifying name and an importance.

A graphical user interface screen (Fig. 5) assists the user in building simple patterns by making fields easy to locate, by populating selection boxes where appropriate, and conducting context sensitive lookups where needed.

For example, the “contractors tab” shown in Fig. 5 includes wildcard lookups, templated collections, lists pre-populated from the warehouse, and a choice between crisp or fuzzy (similarity) matches of this information. A conjunction of information from this and other tabs make up the simple pattern. As shown here, a disjunctive list, e.g. of contractors, can participate as a term in the conjunction that makes up the simple pattern.

After simple patterns have been created, they are assembled using a complex pattern screen (Fig. 6).

Validation checks during or after data entry assures compliance with the constraints described in Fig. 4 and any domain substantive constraints. The ADMIN tab on the screen allows the user to assign an importance, list who should receive alerts regarding the pattern, and set an expiration date after which the pattern will be disregarded.

4.2 Detection of Emerging Patterns and Search Optimization

Many new records are constantly added to the data warehouse. As these are added, an automated warehouse procedure searches for evidence that would support each hypothesis. Conceptually, this warehouse procedure determines whether any *simple* pattern that may be contained within a hypothesis is matched. If a simple pattern is matched, then for all complex patterns (hypotheses) containing that simple pattern, a search is begun for the other simple patterns in that complex pattern. This search is limited by the temporal relation constraints.

The result of the search is a report about the extent to which complex patterns are matched. Because there is no requirement that the root-parent simple pattern or any other simple pattern in the tree be matched, it is possible to locate

Figure 5. Simple Pattern Construction Screen

incomplete complex patterns. This has the beneficial result that emerging, but not yet completed complex patterns can be detected.

Because the warehouse is large, and complex patterns can be quite intricate, it is critical to optimize the pattern detection procedure so that the search space is pruned effectively. The sequencing within the search procedure needs a “processing-time to pruning-benefit” analysis. The effective implementation of this search is highly dependent on the data warehouse structure, and in particular temporal stratifications of data. The search invocation must also be synchronized with servient components’ results, such as those that populate anomaly determinations in the warehouse.

Our experience suggests that tuning the warehouse search procedure, using knowledge about the warehouse implementation and the characteristics of its data, can significantly improve performance over a naïve search that would search for all pieces of each simple pattern node sequentially down the tree. Because the user can also set a minimal level of match to a complex pattern, the procedure should also monitor whether meeting that threshold continues to be possible or the search should be abandoned. We constructed an automated query builder engine that supports the pattern detection procedure in view of these factors, utilizing numerous Oracle “hints” and stacking to assure short-circuit pruning and small joins. There is not room in this article to detail the construction.

4.3 Automated Alerts

Alerts are based on importance of a hypothesis, as assigned by the user and the strength of evidence supporting the hypothesis. Because nodes in the pattern tree may contain individual facts that are not crisply determined, for example approximate string matching or when anomaly strengths occur within the node, results need to be fused into an overall value called *fused importance* that is usable for comparing complex patterns. The *fused importance* calculation begins

Figure 6. Complex Pattern Construction Screen

with evaluations of each simple pattern that is a component of the complex pattern. A simple pattern is a conjunction of necessary facts. If all facts are satisfied at some strength, the strength of the simple pattern, s , is the mean strength of the facts. Otherwise, s is zero.

A complex pattern may have enhancing and minimizing nodes (Those nodes are simple patterns.) Each enhancing or minimizing node causes its parent node's strength to be multiplied by $1+f*s$ or $1-f*s$, where f is a heuristic factor, say 0.4, and s is the calculated strength of the enhancing or minimizing node.

After adjusting parents for enhancing or minimizing nodes, the mean strength of all matched necessary nodes is the strength of the complex pattern, and it is then multiplied by the importance of the hypothesis, as assigned by the user, to produce its *fused importance*. It is possible for a less complete complex pattern with high importance to have a higher *fused importance* than a fully complete complex pattern with low importance. While a strong root node match would seem to imply that the opportunity to respond to an alert has passed, the root pattern may in real-life be part of an extended episode where mitigation opportunities still exist. Thus, we have not modified the *fused importance* formula to reduce *fused importance* where the root has or is occurring. The *fused importance* is used to prioritize results so that the user can investigate further those that are most important. When *fused importance* exceeds a user defined threshold for the individual hypothesis, the system automatically issues an alert to users who are interested in the hypothesis. *Fused importance* is also used to prioritize reports about hypothesis occurrences. The alerts and reports mechanism was implemented using COTS data warehouse triggers and a broadcast agent so that users have reports on emerging problems when they arrive in the morning.

A tool containing this hypothesis representation and alert capability has been implemented and only recently delivered to a government customer. Users' comments, and feedback on performance on the large data warehouse are not yet available.

5. CONCLUSION AND FUTURE WORK

We described a highly expressive tool for capturing subject matter experts' hypotheses about processes. Those hypotheses can be stored, and used to find evidence of emerging threats or other processes of interest in a large data warehouse.

Future work could include investigation of how to reason with these loose, complex and expressive representations. Reasoning may require addition of ontological constructs of world knowledge. Many processes mutate over time as technologies, economics, players or other factors change. In a competitive domain, there may be process mutations that occur because of learning by competitors. The ways

hypotheses change over time are themselves processes that a meta-process pattern system might recognize and anticipate.

6. ACKNOWLEDGMENTS

We are grateful to Dianna Whitt, Marguerite Woods and John Schverak for their valuable comments.

7. REFERENCES

- [1] Allen, J. F. Towards a general theory of action and time. *Artificial Intelligence*, 23(2) (1984), 123-154.
- [2] Allen, J. F. Time and Time Again: The Many Ways to Represent Time. *International Journal of Intelligent Systems*, 6(4) (1991).
- [3] Bock, C. and Gruninger, M. PSL: A semantic domain for flow models. *Software and Systems Modeling J.*, (2004).
- [4] Cheung, J. T.-Y. and Stephanopoulos, G. Representation of Process Trends - Part I. A Formal Representation Framework. *Computers and Chemical Engineering*, 14(4-5) (1990), 495-539.
- [5] Felix, P., Barro, S., and Marin, R. Fuzzy constraint networks for signal pattern recognition. *Artificial Intelligence*, 148(1-2) (2003), 103-140.
- [6] Grenander, U. *Elements of Pattern Theory*. Johns Hopkins U. Press, Baltimore, MD, 1996.
- [7] Jurafsky, D. and Martin, J. H. *Speech and Language Processing*. Prentice Hall, Upper Saddle River, NJ, 2000.
- [8] Kautz, H., Etzioni, O., Fox, D., Weld, D., and Shastri, L. *Foundations of Assisted Cognition Systems*. Tech. Rept. CSE-02-AC-01, Dept. of Computer Science and Engineering, U. Washington, 2003.
- [9] Lubell, J. Process Descriptions. In *Professional XML Meta Data*. Wrox Press, 2001.
- [10] Murphy, R. *Introduction to Robotics*. MIT Press, Cambridge, MA, 2000.
- [11] Nonaka, I. A dynamic theory of organizational knowledge creation. *Organization Science*, 5(1) (1994), 14-37.
- [12] Nonaka, I. & Takeuchi, H. *The Knowledge-Creating Company*. Oxford University Press, NY, 1995.
- [13] Van Stijn, E. and Wensley, A. Organizational Memory and the Completeness of Process Modeling in ERP Systems: Some Concerns, Methods and Directions for Future Research. *Business Process Management J.* 7(3) (2001), 181-194.
- [14] Venkatasubramanian, V., Rengaswamy, R., Kavuri, S., and Yin, K. A review of process fault detection and diagnosis Part III: Process history based methods. *Computers and Chemical Engineering* 27 (2003), 327-346.
- [15] Wijnhoven, F. Managing dynamic organizational memories: instruments for knowledge management. Boxwood Press, Pacific Grove, CA, 1999.