To appear in The Proceedings of the Fifth IFIP WG 11.10 International Conference on Critical Infrastructure Protection, Dartmouth College, Hanover, New Hampshire, USA, March 23-25, 2011.

Chapter 1

A THREADED EVENT-BASED SIMULATION APPROACH TO ANALYZING THE INTELLIGENCE ON WMD ATTACKS

Qi Fang, Peng Liu, John Yen, Jonathan H. Morgan, Donald R. Shemanski and Frank E. Ritter

Abstract In intelligence analysis, data can be incomplete, ambiguous, and of large volume. Also, the data may be unorganized, which needs a lot of reasoning to analyze covert adversarial activities. The work can be very time-consuming, and the details can be overwhelming. To help analysts understand data of use to predict future events and possible scenarios, we have developed a simulator to provide a framework that enables analysts to find, display and understand data relationships, by connecting the dots of data to create network of information. Also, the simulator can generate alternative storylines, allowing analysts to view all possible outcomes. The simulator can automatically conduct reasoning and detect inconsistent data, which provides more reliable information for analysis. Our simulator can shoulder analysts' work, and they can save a lot of time, allowing them to focus on more detailed analysis. In this paper, we will describe the framework, rationale and applicability of our simulator. Also, we will conduct an experiment using data from the [24 hours] TV show, and we will give experiment results to show how our approach can help with intelligence analysis of WMD attacks against critical infrastructure.

Keywords: Intelligence analysis, threaded event-driven simulation

1. Introduction

Terrorists often target critical infrastructures. The US State Department defines terrorism as "premeditated, politically motivated violence perpetrated against noncombatant targets by sub-national groups or clandestine agents, usually intended to influence the audience" [**?**]. This

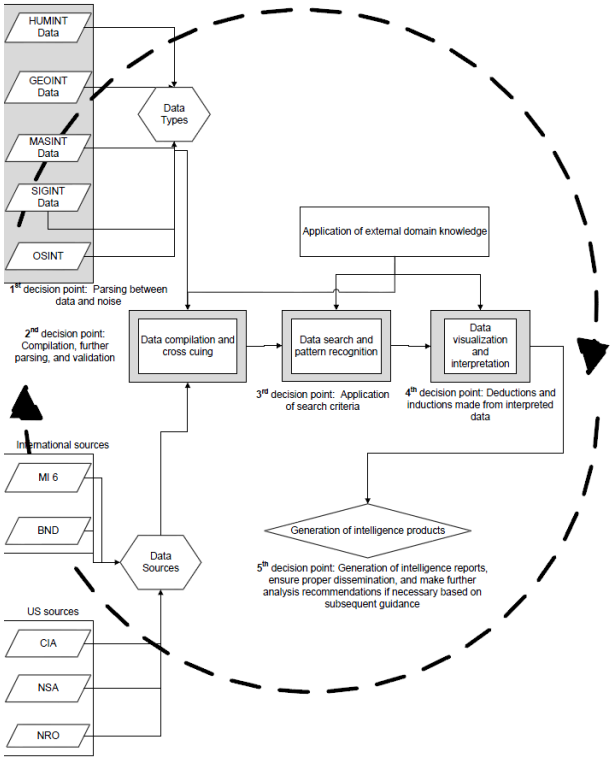

Figure 1. Workflow of the intelligence analysis

paper is on analyzing the intelligence on WMD attacks against critical infrastructures.

There are different types of WMD attacks [**?**], including chemical, biological, radiological, nuclear, and combinations of these. These WMD types differ in the incident numbers, in the processes of plot, threat of possession, attempted acquisition, possession, and use. The general characteristics of terrorists or other clandestine groups who might seek to acquire WMD include [**?**]: cause, commitment, camaraderie, charismatic leaders, cash and resources, and cells (i.e., compartmentalized, cell-based structures). Organizational characteristics include: Command, control and communications, recruitment, weapons procurement, logistics, surveillance, operations, and finance. Organizational complexity, characterized by division of responsibility within the group with respect to the various specific tasks outlined above, generally contributes to the likelihood of a successful, high-yield WMD event, while also entails generating more traceable data.

Presented at: the Fifth IFIP WG 11.10 International Conference on Critical Infrastructure Protection,
Dartmouth College, Hanover, New Hampshire, USA, March 23-25, 2011

To reduce the impact of terrorism in a timely and effective manner, intelligence analysts need to understand continuous incoming information (e.g., important actors, organizations, and events), identify patterns, anomalies, relationships and causal influence, and give alternative explanations (of the same set of intelligence) and possible outcomes for decision making. As noted by George and Bruce [**?**], when given an assignment, analysts search for information, assemble and organize the information in a manner designed to facilitate retrieval and analysis, analyze the information to make an estimative judgment, and write a report. Figure 1 shows the workflow and key decision points in the intelligence gathering and analysis cycle [**?**].

There are four broad challenges in intelligence analysis: data collection, synthesis, validation and interpretation. To make intelligence analysis successful, simulation tools to facilitate the analysis must address these challenges. They must also be operable within time constraints when information may be abundant or incomplete. Also, a lot of reasoning work may be needed in intelligence analysis, which is a time-consuming job when no good tools or automated reasoning aids are provided and analysts have to work on their own.

1.1 Prior Work

A number of techniques have been developed to address the following challenges: better organize information to identify recurring patterns and causal relationships; distinguish relevant information from noise; and infer activities of interest from incomplete data. But they have less emphasis on counter-validation. Also, computer-aided analysis offers intelligence analysts how to think about complex problems when the available information is incomplete or ambiguous, as typically happens in intelligence analysis [**?**]. Modeling and simulation can provide knowledge, understanding, and preparation against future attacks [**?**]. Simulation approaches that have attempted to address these challenges can be broadly grouped into two categories: agent-based and event-based simulations.

Much work has used agent-based simulations. Some use agent-based approaches to provide a powerful approach for modeling emergence and inferring the effects of agent-level decisions or actions on social systems [**?**, **?**, **?**, **?**]. Within this paradigm, agent interactions can be characterized in several different ways: as forms of information diffusion [**?**]; as mechanisms leveraging social influence [**?**]; as trades, contracts, or negotiations [**?**]; or as the consequences of some activity or strategy [**?**]. Agent-based approaches also offer a powerful means of rep-

Presented at: the Fifth IFIP WG 11.10 International Conference on Critical Infrastructure Protection,
Dartmouth College, Hanover, New Hampshire, USA, March 23-25, 2011

resenting agent-level capabilities by representing agent-level constraints, such as the geospatial effects [**?**], psychological limitations [**?**], or socio-cognitive effects [**?**].

Agent-based approaches vary in their portability (the ability to integrate into various simulation environments) and modularity (the ability of the user/analyst to change aspects of the architecture). Agent architectures also differ not only in their capabilities but also with respect to their theoretical entanglements. Powerful complex agents are generally computationally expensive, difficult to integrate into simulation environments, and entail significant theoretical commitments (making them difficult to change). Lighter agents, on the other hand, are often fairly modular and easier to integrate into an existing simulation, but brittle. They generally model a small set of behaviors very well, but require significant modification to model other behaviors of interest. Consequently, agent-based approaches if used exclusively are unlikely to be able to support, without external assistance, a wide range of evolving analysis priorities, or fundamental changes in understanding.

Event-based simulations represent behavior by modeling the causal and temporal preconditions, participants, effects, times, and locations that characterize an event. Unlike an agent's internal state, these characteristics are immediately observable and verifiable. Using hyper-graphs or meta-network representations, analysts can then represent causal, temporal, spatial, and social relationships as ties between events. Event-based simulations can also support both deductive and inferential reasoning. By defining the casual and temporal preconditions associated with an event, a set of potential consequences (or storylines) can be deduced. These storylines can then be compared with reports gathered from human intelligence sources, providing a form of counter-validation. Alternatively, analysts can identify potential group associations or behaviors by highlighting points of co-occurrence or performing other analyses of the network's topology. The notion of "events" has also been used to model the growth of the social networks [**?**].

Figure 2 shows a sparse event network with data collected in Tanzania in 2006 [**?**]. The network has different types of nodes and ties. In the network, red nodes represent agents, orange nodes represent locations, light blue nodes represent resources, and dark blue nodes represent tasks/events. The ties between nodes represent several types of relations such as social relation between agents, spatial relationship between agents and location, ownership relationship between agents and resources, actor relation between agents and tasks, and distance relation between two locations. Even a sparse network could help one see who

Presented at: the Fifth IFIP WG 11.10 International Conference on Critical Infrastructure Protection,
Dartmouth College, Hanover, New Hampshire, USA, March 23-25, 2011

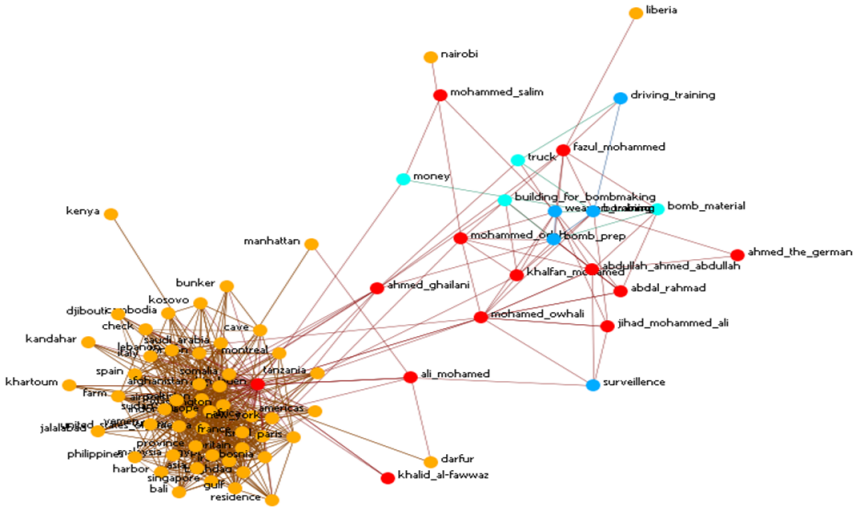

Figure 2. A sparse event network with data collected in Tanzania in 2006 [**?**]

has the highest node centrality, the presence of cliques, and the clique leaders–giving us some idea early on who might be the nodes of interest.

Event-based simulations offer an attractive pragmatic approach to analysis problems. Many of the basic causal and temporal preconditions associated with events can be handled using established techniques [**?**, **?**]. The relative theoretical flexibility of event-based simulations also makes them fairly responsive to user demands. Basic deductions require only rerunning the simulation using a new set of inputs, while implementing new causal or temporal rules can be facilitated through a GUI. On the other hand, event-based simulations generally possess no mechanisms for bringing to bear external knowledge-the scope of their inferences and deductions depends on the completeness of the data available and the expert knowledge encoded into the simulation by simulator designers or past expert users.

1.2 Our Contributions

To address these limitations, we propose a threaded event-based simulation approach for intelligence analysis. Our simulation approach offers analysts a means of identifying causal relationships and patterns in large data sets, detecting missing data, a counter validation approach, and a means of mapping multiple and alternative storylines to better support emerging analysis priorities.

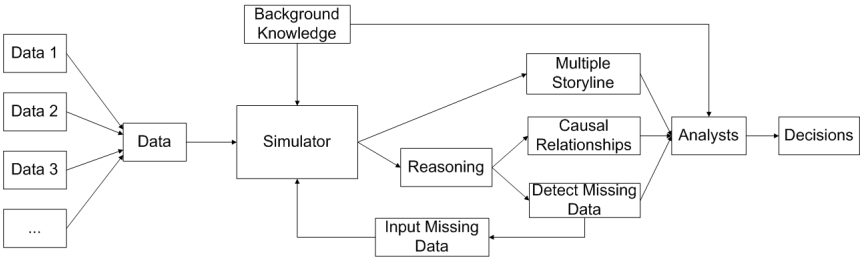

Figure 3. Workflow of the intelligence analysis process with our simulator

Figure 3 depicts an analysis driven workflow using the simulator, supporting decision points 2, 3, and 4 shown in Figure 1. Our simulator provides assisted analysis in the following ways: Information organization; Simulation of event possibilities, represented as storylines; Supports evolving analysis priorities by allowing analysts to examine the effects of different decision points; Discovery of causal relationships; Detection of missing data, and thus a form of counter-validation.

Our simulator decomposes the original data into a set of networks or storylines, by defining three types of nodes: events, actors, and objects. Their associated attributes are also defined. Our simulator is capable of identifying and generating multiple divergent storylines, as well as alternative storylines, within a large dataset. We distinguish between multiple and alternative storylines to indicate whether a decision point generates multiple coexisting possibilities, or if the decision point generates two mutually exclusive possibilities. Figure 4 shows an example of multiple storylines that our simulator generates, and decision nodes are used during simulations. Each storyline is indicated by different colors. Also, the difference between previous approaches and our approach is: in previous approaches, temporal and causal dependencies between events are specified in advance by the knowledge engineer, and if preconditions are not satisfied, the associated event cannot proceed; in our approach, temporal and causal dependencies between events are discovered, and if preconditions fail to be satisfied, this triggers the gathering of missing information, enabling the associated event to proceed after the information becomes available.

To show how well our simulator works, we conduct a case study using data from the popular TV show [24 hours], season 2 as a surface valid example. We parse the whole season TV show into a set of discrete events, along with their attributes. We will show the multiple storylines our simulator generates, with decision nodes and different options provided. We will also show the causal relationships our simulator discovers.

Presented at: the Fifth IFIP WG 11.10 International Conference on Critical Infrastructure Protection,
Dartmouth College, Hanover, New Hampshire, USA, March 23-25, 2011

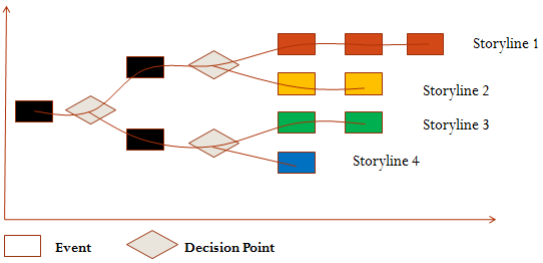

Figure 4. A threaded event-based simulation

To see how the simulator detects missing information, we deliberately delete some events from the original data, and run the simulation again. We will show that our simulator triggers to gather missing information when it detects the missing information.

The rest of this paper is organized as follows: In Section 2, we will introduce the framework of the simulator, including our dataset, key concepts, workflow, software architecture and algorithms. In Section 3, we will introduce the case study, and show the results our simulator generates. In Section 4, we will make conclusions and propose further work.

2. Framework and Simulator Design

The goal of our simulator is to reduce the workload of analysts, so they can focus on core analysis. To achieve this goal, our simulator is designed to provide a framework to connect unorganized data into an information network, generate "multiple storylines" to allow analysts to view different outcomes, and automatically detect causal relationships and missing information.

In this section, we describe our WMD attack dataset extracted from the TV show [24 hours]. Then we define several key concepts used in our model, including "precondition", "effect", and "causal relationship". Finally, we present the simulator's software architecture and algorithms.

2.1 WMD Attack and Dataset

The dataset we use is extracted from the popular TV show [24 hours], season 2. The reason why is because real WMD attack data is very difficult to obtain. This TV show is about terrorists planning to attack Los Angeles (LA) using a nuclear bomb, and the Counter Terrorist Unit (CTU) agents working together to stop the attack. We use episodes 1 to 14, which span a 14 hour time period (of terrorism and anti-terrorism

Presented at: the Fifth IFIP WG 11.10 International Conference on Critical Infrastructure Protection,
Dartmouth College, Hanover, New Hampshire, USA, March 23-25, 2011

activities) in this WMD attack scenario. There are three different kinds of actors: terrorists, agents, and civilians. And there are three story threads in the TV show. The first thread is about how agents find the nuclear bomb and prevent the attack, led by Jack Bauer, a former CTU agent. The second thread is about politics and the divergence in officers' opinions on government policies. The third thread is about the experiences of several civilians during the same time period.

In our case study, we only included scenarios from Thread 1. We extracted 57 discrete and non-overlapping events, 31 actors, and 5 objects. Then, we assigned attributes to all events, actors, and objects. Also, we tracked possible events that could happen and lead to different outcomes, from conversations between actors, and from other possible actions of actors. We set decision nodes between different options, and generated different sets of data, with each set containing all the information in a self-contained storyline.

2.2 Key Concepts

Before we introduce our model, we would first introduce some concepts. In our simulator, *events* indicate that something-of-interest (e.g., attack steps, CTU agents found a bomb) has happened, *actors* indicate participants of events, and *objects* indicate target infrastructures or tools. Events and their relationships generate state changes. Events, actors, objects, and their relationships define *the world*.

We also define *attributes* to capture the properties of events, actors, and objects, and we list them in Appendix Table A1 - Table A3. "Preconditions" and "effects" are two important attributes of events. *Preconditions* describe the state of the world before a change (caused by an event). The simulation treats preconditions as the qualified state(s) of the actors, objects, and relationships, in which the owner event can happen. A state that does not satisfy the preconditions of an event disallows the event to happen in this state. *Effects* describe the state of the world (actors and objects) after a change. The inability to satisfy preconditions results in information gathering (currently in the form of a query to the user). Events can trigger new events. Causal relationships are defined as instances where the effects of an event cause the preconditions of another event to be satisfied. This process can uncover hidden relationships by identifying for the analyst all instances throughout the dataset where this has occurred.

Presented at: the Fifth IFIP WG 11.10 International Conference on Critical Infrastructure Protection,
Dartmouth College, Hanover, New Hampshire, USA, March 23-25, 2011

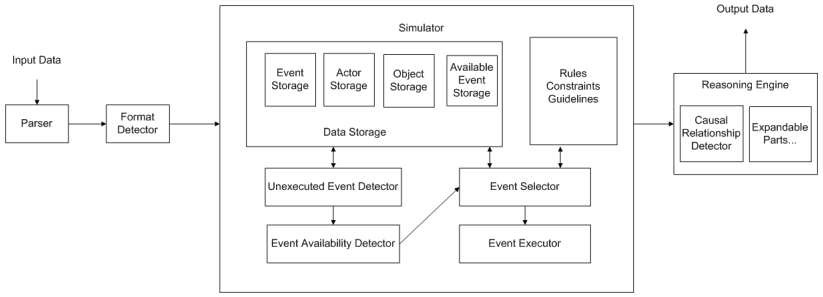

Figure 5. Software architecture of the simulator

2.3 Software Architecture

In this section, we illustrate the software architecture of our simulator. Figure 5 shows the components that support the Reasoning Engine's abilities to detect causal relationships and identify missing data. It also illustrates the selection of events by the simulator; this selection is guided by rules, constraints, and guidelines that can be obtained from or altered by experienced analysts.

The Parser extracts useful information about events and their attributes. The Format Detector is used to detect whether the parsed data is qualified for our simulator to execute. The simulator contains six parts: Data Storage saves all data during the simulation process, and the Available Event Storage holds all events whose preconditions are satisfied; the Rules/constraints/guidelines Component saves all the logic rules used by the simulator; the Unexecuted Event Detector is used to select all unexecuted events; the Event Availability Detector is used to select events with preconditions satisfied, from the output of the Unexecuted Event Detector; the Event Selector is used to select an event with earliest start time, from the output of the Event Availability Detector; the Event Executor executes the selected event, changes according attributes, and marks the event as "executed". After simulation, for each storyline, the simulator gives a chronological sequence of discrete events according to the execution order of events.

The current reasoning engine is designed to detect the causal relationships between events, and detect missing information. It uses the output of the simulator. For each event, the reasoning engine matches its preconditions with effects of all preceding events to check if there is any relationship between the events. The reasoning engine can also detect missing information, when it cannot find the match for preconditions.

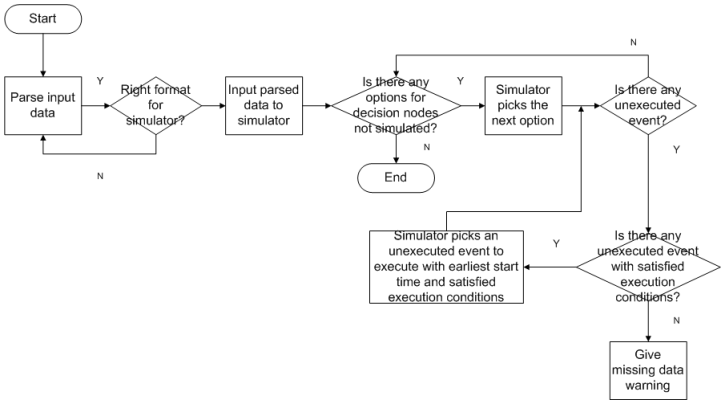

Figure 6. Workflow of the simulator

2.4 The Algorithms

In this section, we will introduce the algorithms of our simulator, including information organization, simulation, generating *"multiple storylines"*, discovery of causal relationships, and missing data. First, we will introduce the workflow of the simulator. This workflow, shown in Figure 6, is as follows:

1) Parse input data into the appropriate XML format.

2) Search if there are any unexecuted options; terminate process if none exist.

3) Select option with the earliest start time and satisfied conditions.

4) Search for subsequent events, and query for addition information if necessary.

5) Generate a network of events based on causal and temporal dependencies.

2.4.1 Information Organization. Our simulator requires input data to contain a set of events with associate attributes in XML format. Event and their attributes are obtained by coding the original data. There are three input files for events, actors, objects and their associate attributes, respectively, in each *"storyline"*. Example data formats are defined in Figure 7.

The Parser of our simulator goes through all input files, parses and extract attributes in each record, and saves objects in event, actor, and object storages, respectively.

2.4.2 Simulation. After parsing the data, the simulator goes through the event storage and checks whether the preconditions of each

Presented at: the Fifth IFIP WG 11.10 International Conference on Critical Infrastructure Protection,
Dartmouth College, Hanover, New Hampshire, USA, March 23-25, 2011


```
<Event>
    <Event ID>1</Event ID>
    <Content>Mamud makes death elusion</Content>
    <Start Time>-2000</Start Time>
    <Location>WestBank@USA</Location>
    <Actors>
        <Actor>Mamud</Actor>
    </Actors>
    <Relationships>
        <Relationship></Relationship>
    </Relationships>
    <Effects>
        <Effect>Mamud .status="dead"</Effect>
    </Effects>
    <Preconditions>
        <Preconditions></Precondition>
    </Preconditions>
</Event>
```

```
<Actor>
    <Actor ID>1</Actor ID>
    <Name>Mamud</Name>
    <Sex>male</Sex>
    <Age>WestBank@USA</Age>
    <Type>terrorist</Type>
    <Status>alive</Status>
    <Level>1</Level>
    <Affiliation>2nd wave</Affiliation>
    <Location>WestBank@USA</Location>
</Actor>
```

```
<Object>
    <Object ID>1</ Object ID>
    <Object Name>Nuclear Bomb</ Object Name>
    < Object Status>"ready"</ Object Status >
    < Object Location>Los Angeles Norton Airport</ Object Location >
</Object>
```

Figure 7. XML formats of input data

event are satisfied. If the event is unexecuted and its preconditions are satisfied, the simulator marks this event "ready". Among the "ready" events, the simulator picks an event with the earliest start time to execute each time. After the event is executed, it may change others events' precondition attributes; we list these attributes as effects. The simulator updates attributes in data storages. After execution, the simulator marks the event "executed". The simulator repeats these steps until all events are executed (this happens when there is no missing data about causal relationships. If there are missing data, we will show how the simulator processes this in 2.4.4). After simulation, the simulator generates a chronological sequence of events according to their execution sequence, each sequence representing a storyline. Algorithm 1 shows how the simulation works.

2.4.3 Discovery of Causal Relationships. The scheme to discover causal relationships is, if one event's effects affect other events' preconditions, we consider there are causal relationships between these events. As shown in Algorithm 2, our simulator matches some event's preconditions with effects of all its preceding events. If these preconditions match, we consider there is a causal relationship between these two events. Because there could be several preconditions for each event, there could be several events that have casual relationships with the current event.

Algorithm 1 Simulation

```
Require: D = {E, A, O, EE}
Require: E, event list; EE, executed event list
Require: A, actor list; O, object list
1: while E.events ≠ 0 do
2:    for event i in E do
3:       if all preconditions of i == true then
4:          set i.status = "ready"
5:       end if
6:    end for
7:    set min_start_time = start_time of event 1 in "ready"
8:    for event j in "ready" events do
9:       if j.start_time < min_start_time then
10:         min_start_time = j.start_time
11:         picked_event = j
12:      end if
13:   end for
14:   execute event j, update A, update O
15:   EE.add(event j)
16:   E.remove(event j)
17: end while
18: output events from EE
```

Algorithm 2 Discovery of Causal Relationships and Missing Data

```
Require: D = {E, EE}
Require: E, event list
Require: EE, executed event list
1: while E.events ≠ NULL do
2:    for event i in E do
3:       for precondition j in i.preconditions do
4:          if j == effect of event m in EE then
5:             output causal relationship m -> i
6:          end if
7:       end for
8:    end for
9:    if no "ready" event in E then
10:      output "missing information"
11:      break;
12:   end if
13: end while
```

2.4.4 Detection of Missing Data. The scheme to detect missing data is based on the causal relationships. The effects of some events will affect the preconditions of other events, and thus trigger those events. If some events are missing, there will be no effects triggering others events to happen. In this way, we detect missing data and require analysts gather relevant data.

As shown in Algorithm 2, our simulator detects whether there are events not executed after simulation. We consider there are missing data if there are events not executed. So the storyline the simulator generates is incomplete. The simulator will give a warning and require relevant data to keep working. In this way, the simulator may help analysts discover useful missing information.

2.4.5 Generating Multiple "Storylines". Our simulator allows analysts to set decision nodes and choose different options at each step to see all possible outcomes. When the simulator encounters a decision node, it simulates one of the options associated with the decision node. After the simulator generates a complete storyline, it goes back to the "decision nodes", and simulates other options until all options are simulated.

Currently, our simulator requires analysts to put different storylines in different files, with each file containing a complete sequence of terrorism events, as defined by the bomb being captured or exploding.

Presented at: the Fifth IFIP WG 11.10 International Conference on Critical Infrastructure Protection,
Dartmouth College, Hanover, New Hampshire, USA, March 23-25, 2011

Table 1. Simulator Output - Part of the Storyline

ID	*Content*	*Time*	*Location*	*Actors*
1	"*Mamud makes death illusion*"	$-20:00$	*WestBank*	*Mamud*
2	"*Get a nuclear bomb*"	$-10:00$	*Norton Airport*	*Mamud*
3	"*Mamud introduces Nina to Joe*"	09 : 01	*LA*	*Mamud, Nina, Joe*
4	"*Joe gets plans of CTU*"	09 : 00	*LA*	*Joe, Nina*
5	"*Marie deals with Ali*"	8 : 50	*Warner*	*Marie, Ali*
...				
57	"*Agent found the bomb*"	13 : 09	*Norton Airport*	*Jack*

3. A Case Study

3.1 Experiment Design

In the experiment, we encoded the data from the TV show [24 hours] using XML. First, we input a complete set of events to let the simulator produce a complete storyline. We checked whether the simulator could detect the causal relationships between events. We also input an incomplete set of events to see whether the simulator can detect inconsistent data. At last, we set a decision node and provide a different event set to show the result of a different storyline.

3.2 Results

3.2.1 Simulation Results. The simulator generated a chronological sequence of events, indicating a complete storyline about how agents collaborate to detect and prevent a WMD attack. Because there was no missing data in this experiment, the simulator output all the 57 events in a sequential order. Table 1 lists part of the simulation output. Figure 8 depicts the sequence of events and decision points, with boxes representing events and diamonds decision points.

3.2.2 Causal Relationships. The simulator also detected the causal relationships between events. We list some causal relationships in Table 2. For example, Event 3 (*"Mamud introduce Nina to Jo"*) triggers the occurrence of Event 4 (*"Joe get plans of CTU@LA from Nina"*). Because the effect of Event 3 is Nina and Joe know each other, and this is the precondition of Event 4, we consider there is a causal relationship between these two events.

3.2.3 Detect Missing Data. In the case study, when we deleted Event 8 (*"Transit the nuclear bomb into USA"*), we observed, as expected, that Event 9 (*"Prepare nuclear bomb"*) and all related events

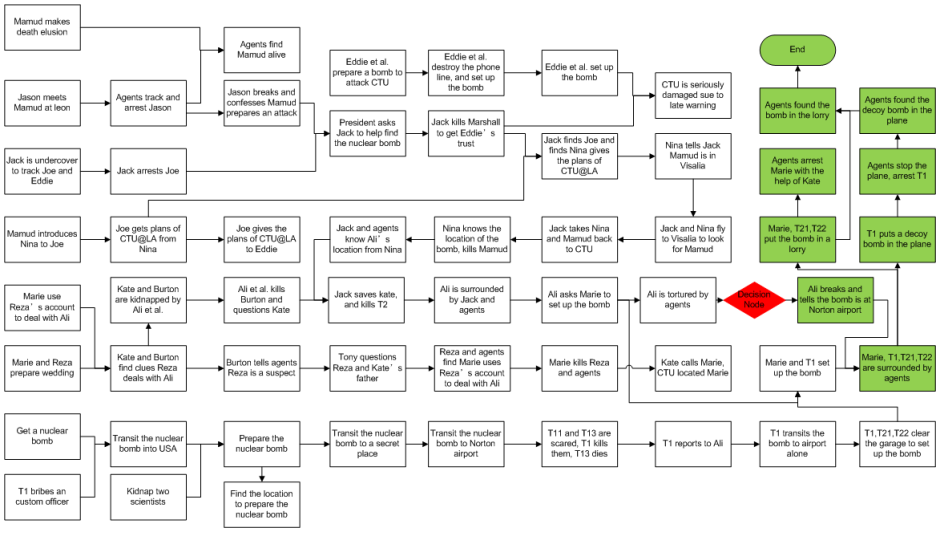

Figure 8. Network of events in [24 hours]

were affected. The simulator provided a warning indicating that some information was missing and queried the user to provide more input, in this case the location information for the device. Besides identifying instances of incomplete information in the available dataset, this mechanism can also identify holes in an account about a set of events, allowing analysts to follow-up on these key details with information sources and collectors.

Table 2. Casual relationships between events

Event	*Preceding Event(s)*
Event 4 : Joe get plans of CTU	*Event 3 : Mamud introduce Nina to Joe*
Event 7 : Joe gives the plans of CTU@LA to Eddie	*Event 4 : Joe get plans of CTU*
Event 8 : Transit the nuclear bomb into USA	*Event2 : Get a nuclear bomb*
Event 11 : Prepare the nuclear bomb	*Event 8 : Transit the nuclear bomb into USA*
...	...

3.2.4 "Multiple Storylines". The TV show only presents one storyline. So we generate different storylines according to other possible decisions in the TV show. We set several decision nodes, generate different options and run the simulator again. Figure 9 shows another storyline, where Ali refuses to confess. Again, events are represented by boxes and decision points by diamonds.

Presented at: the Fifth IFIP WG 11.10 International Conference on Critical Infrastructure Protection,
Dartmouth College, Hanover, New Hampshire, USA, March 23-25, 2011

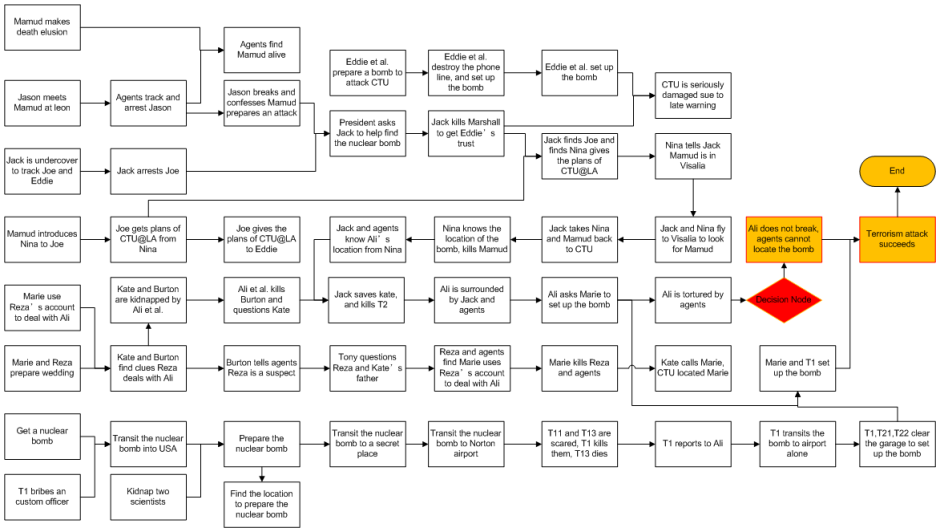

Figure 9. Network events in an alternative storyline

4. Conclusions and Future Work

In this paper, we proposed a threaded event-based simulator to help with intelligence analysis. With this simulator, several functions are provided to shoulder analysts' work and increase reliability: First, the simulator helps better organize a large amount of data by decomposing the original data into a chronological sequence of events, and generating a chronological sequence of events representing the development of a storyline. Second, the simulator automatically detects the causal relationships between events, which are helpful for future analyses such as identifying patterns. Third, the simulator automatically detects some missing information which may be important for intelligence analysis. Fourth, this simulator generates *"multiple storyline"*, which allows analysts to view all possible outcomes given different options for events. To show how our simulator works, we conduct a case study using data extracted from the TV show [24 hours]. We give simulation outputs and *"multiple storyline"*, and we also detect causal relationships and missing information.

Our future work involves the following projects:

Layered Iterative Intelligence Analysis: drawing from diverse information sources, our simulation can initially generate sparse event networks, where the events are defined as instances of co-occurrence (actors, times, locations, activities). These networks can serve to identify ties of inter-

est or hidden relationships that may warrant further scrutiny. With more information, analysts can move from these sparse event networks (similar to the Tanzania data) to richer event networks, encompassing a broader set of events as noted in the [24 hours] example. Further, the ability to examine event-chains individually as storylines allows analysts to not only identify causal chains but also reexamine them periodically in light of new information.

Counter Validation and Hypothesis Testing: the simulation also offers both a basic hypothesis testing capability by simulating multiple and alternative storylines and a validation capability, in that analysts can quickly verify whether an explanation of events (a storyline) is probable.

Multi-dimensional Relation Inference: extending inference capabilities beyond causal and temporal relationships to other types of relations between events, actors, and objects.

Alternative Storylines: extending decision nodes to support the concurrent simulation of multiple alternative storylines, further supporting fault detection. In addition, we have begun to associate probabilities with alternative and multiple storylines.

Hypothesis Reasoning within the Simulation: using agent-based simulations to support hypothesis testing by introducing new information (pertaining specifically to behaviors of interest) to the simulation's dataset, allowing for the possibility of still richer counterfactual scenarios.

5. Acknowledgements

This work was supported by DTRA HDTRA 1-09-1-0054. Baojun Qiu helped us gather data for the case study.

References

[1] S. Raczynski. Simulation of the dynamic interactions between terror and anti-terror organizational structures, *Journal of Artificial Societies and Social Simulation*, 7(2), article 8, 2004.

[2] R. Z. George and J. B. Bruce. Analyzing intelligence: Origins, obstacles, and innovations, Georgetown University Press, Washington, 2008.

[3] R. Smith. Counter terrorism simulation: A new breed of federation, *Proceedings of the Spring 2002 Simulation Interoperability Workshop*, March 2002.

[4] I. C. Moon and K. M. Carley. Modeling and simulating terrorist networks in social and geospatial dimensions, *IEEE Intelligent Systems, Special issue on Social Computing*, 22: 40-49, September/October 2007.

[5] C. T. Morrison, P. R. Cohen, G. W. King, J. Moody and A. Hannon. Simulating terrorist threat in the hats simulator, *Proceedings of the First International Conference on Intelligence Analysis*, 2005.

[6] U. Krause. A discrete nonlinear and non-autonomous model of consensus formation. In S. Elaydi, G. Ladas, J. Popenda and J. Rakowski (Eds.), *Communivations in Difference Equations*, Gordon and Breach Pub., Amsterdam, 227-236, 2000.

Presented at: the Fifth IFIP WG 11.10 International Conference on Critical Infrastructure Protection,
Dartmouth College, Hanover, New Hampshire, USA, March 23-25, 2011

[7] B. Latane and A. Nowak. Self-organizing social systems: necessary and sufficient conditions for the emergence of clustering, consolidation and continuing diversity. In F. J. Barnett and F. J. Boster, *Progress in Communication Sciences*, Ablex Pub., 1-24, 1997.

[8] S. Galam and S. Wonczak. Dictatorship from majority rule voting, *European Physical Journal B*, 18, 183-186, 2000.

[9] M. Taylor and J. Horgan. A conceptual framework for addressing psychological process in the development of the terrorist, *Terrorism and political violence*, 18, 1-17, 2006.

[10] S. R. Haynes, M. A. Cohen and F. E. Ritter. Design patterns of explaining intelligent systems, *International Journal of Human-Computer Studies*, 67(1), 99-110, 2009.

[11] A. Arnold, Y. Liu and N. Abe. Temporal causal modeling with graphical granger methods, *proceedings of the 13th ACM SIGKDD international conference on knowledge discovery and data mining*, 2007.

[12] K. M. Carley. Computational Analysis and Experimentation of Social and Organizational Systems [Slides], *Pittsburgh, PA: Center for Computational Analysis of Social and Oroganizational Systems*, CMU, 2010.

[13] C. Deo, M. E. Kosal and S. Dhongde. Multidisciplinary modeling of socio-economic influence on adversarial intent to acquire, proliferate, and use chemical, biological, radiological, and nuclear weapons, *2009 DTRA Counter-WMD Basic Research Techincal Review*. DTRA: Springfield, VA, 2009.

[14] X. Fan, S. Sun, and J. Yen, On Shared Situation Awareness for Supporting Human Decision-Making Teams, *Proceedings of 2005 AAAI Spring Symposium on AI Technologies for Homeland Security*, pp. 17-24, Stanford, CA, Mar. 2005.

[15] K. Karimi and H. J. Hamilton. Finding Temporal Relations: Causal Bayesian Networks vs. C4.5, *Foundations of Intelligent Systems*, 2009.

[16] J. H. Morgan, G. P. Morgan and F. E. Ritter. A preliminary model of participation for small groups, *Computational and Mathematical Organization Theory*, 16(3), 246-270, 2010.

[17] D. R. Shemanski. Characteristics and attributes of real-world terrorist organizations: Implications for effective network analysis (pp. 1-7), The Pennsylvania State University, College of IST, 2009.

[18] R. Axelrod, and R. A. Hammond. The evolution of ethnocentric behavior, *Paper presented at the Midwest Political Science Convention*, April 3-6, 2003, Chicago, IL.

[19] M. A.Cohen, F. E. Ritter and S. R. Haynes. Using reflective learning to master opponent strategy in a competitive environment, *Proceedings of the 8th International Conference on Cognitive Modeling*, 157-162. Taylor and Francis/Psychology Press: Ann Arbor, MI, 2007.

[20] F. E. Ritter, S. E. Kase, L. C. Klein, J. Bennett and M. Schoelles. Fitting a model to behavior tells us what changes cognitively when under stress and with caffeine, *Proceedings of the Biologically Inspired Cognitive Architectures Symposium at the AAAI Fall Syposium*. Keynote presentation. Technical Report FS-09-01. 109-115. AAAI Press: Menlo Park, CA, 2009.

[21] B. Qiu, K. Ivanova, J. Yen and P. Liu. Behavior Evolution and Event-driven Growth Dynamics in Social Networks, *Proceedings of 2010 IEEE International Conference on Social Computing (SocialCom'10)*, IEEE Press, Minneapolis, MN, 2010.

Appendix

Table A1. Event attributes

Name	*Implication*	*Example*
Event ID	*Identifier of event*	1
Content	*Description of event*	"*Make death illusion*"
Start Time	*The time when event is executed*	$-20:00$
Location	*The place where event happens*	*WestBank@USA*
Actors	*All participants evolved in the event*	*Mamud Faheen*
Relationships	*All relationships shown up in the event*	
Effects	*Effects the event cause after being executed*	*Mamud Faheen.status* $= 2$
Preconditions	*Requirements the event could happen*	*Mamud Faheen.type* $= 1$

Table A2. Actor attributes

Name	*Implication*	*Example*
Actor ID	*Identifier of actor*	1
Name	*Name of actor*	*Mamud*
Sex	$1 - male, 2 - female, 0 - not\ sure$	1
Age	*Positive integer*	49
Type	$1 - terrorist, 2 - anti - terrorist\ agent, 3 - neutral$	1
Status	$0 - dead, 1 - alive, 2 - arrested, 3 - under\ surveillance$	2
Level	*Status level of actor in a task*	10
Affiliation	*Organization actor belongs to*	*2nd wave*
Location	*Location of actor*	*Los Angeles*

Table A3. Object attributes

Name	*Implication*	*Example*
Object ID	*Identifier of object*	1
Object Name	*Name of object*	*Nuclear Bomb*
Object Status	*Status of object*	"*ready*"
Object Location	*Location of object*	*Los Angeles Norton Airport*