14th ICCRTS-2008 “C2 and Agility”

Title: **Socio-technical System Monitoring to Support Enhanced C2 Agility**

Topic 9: **C2 Architectures and Technologies**

Authors:

Georgiy Levchuk, Ph.D., Aptima, Inc.

Darby E. Grande , Ph.D., Aptima, Inc.

Sharnnia Artis, Ph.D., Aptima, Inc.

Correspondence:

Georgiy Levchuk

Aptima Inc.

12 Gill Street, Suite 1400

Woburn, MA 01801

Phone: 781-935-3966 x267

Fax: 781-935-4385

e-mail: georgiy@aptima.com

Abstract

Agile command and control (C2) organizations are increasing the demands for dynamic network resource management. Real-time determination of operator workflow, required to support these processes, currently requires an unacceptably high level of specification by and intervention of the operator. To overcome this burden, we are developing the capability to passively monitor and learn models of individuals’ workflows and team task sequences. In this paper, we describe the hierarchical, context driven, hypothesis testing approach to semi-automated individual and team behavior learning and classification. The aim of the technology, when fully-developed, is to offer a technology to support agile tasking of personnel and organizational resources given the ongoing mission(s), monitor bottlenecks, identify critical coordination points, and assess workflow consistency for training and/or performance improvement.

Introduction

Agile Operations in Large-scale Multi-functional Teams

To enable agile operations, organizations need to adjust both their structure and processes. Large-scale organizations enable the responsiveness, flexibility, and adaptation attributes of agility by employing dynamic retasking and resource reallocation (Alberts and Hayes, 2003). This functionality works well when controlled resources are services, technical systems, or physical assets, and several models exist for centralized and distributed resource allocation and planning. Recently, human resource management became one of the central issues in enable agile operations in socio-technical systems. However, human resources – members of the organization who are responsible for different functions and perform varied tasks – are much harder to control in dynamic organizations, because knowledge of their current workload, much less the true tasks that human team members are working on, is not readily available.

While the workload and overall performance profile could potentially be measured (using cues such as heart rate, perspiration level, eye tracking, etc.), the granularity of this data is not enough to enable true organizational agility. To implement efficient structural adaptation, retasking or to improve information flow in the agile organization, the knowledge is needed about what tasks individuals are currently executing, which tasks they will be working on next, and whether their behavior falls into a specific category. The challenge is that neither this knowledge nor the knowledge of the possible individual and group behavior categories is available from the data about human operators.

Behavior categories can be defined as multiple tasks that an individual or a group will be performing. A behavior category for an individual may represent the expertise level of a human team member and his/her role. A behavior category for a group may represent the overall mission that a team will perform. The structural relationships among the tasks defining the behavior categories form the models of behavior, which in organizational engineering domain are often called *individual* or *team workflow*. Thus, learning, tracking, and managing these workflows become essential for agile organizations. In this paper, we describe the semi-automated models that can enable such processes.

Motivating Example: Air Operation Center and its Cells

Consider the Dynamic Targeting Cell (DTC) within the Air Operations Center (AOC), which is responsible for directing the prosecution of time sensitive, dynamic, and emerging targets while managing and coordinating with other AOC teams to ensure effective execution of the current Air Tasking Order (ATO). A distributed, interdependent C2 organization such as the AOC, working in a dynamic environment, must be able to respond with agility to frequent, sudden external and internal changes. To achieve this objective in the AOC, there must be timely communication, tasking, and information exchange among offensive operations, tasked units, and other Theater Air Control Systems. This high level of coordination between the people and information systems in the socio-technical system that makes up the AOC demands widespread situation awareness regarding the state of tasks and the performance of the teams and individual operators that execute them. More generally, a well-communicated awareness of the state and health of the other nodes in one’s network is essential for supporting the ability of humans and automated systems to respond to local failures in a way that prevents widespread, cascading loss of capability.

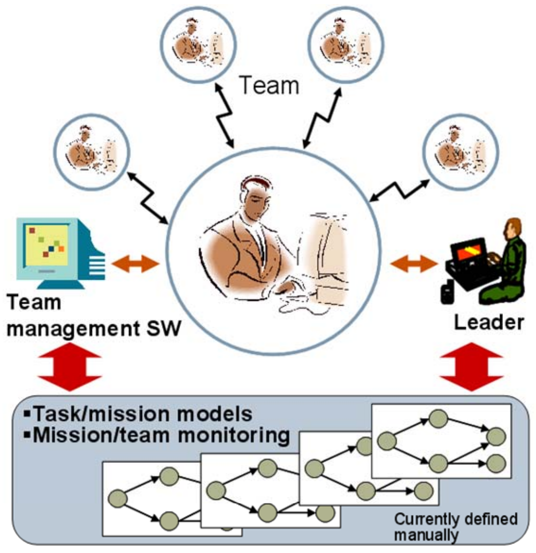
One way to maintain this awareness is to require each node to continually report its status, either to a centralized point of authority, or as a broadcast to the whole networked system. The system then has the challenge of determining which resources need to be notified or re-tasked based on fluctuations in the state of the other resources (nodes) in its critical task path(s). The ability to quickly understand and preemptively act upon local problems is a critical requirement for agile, responsive C2.

To focus the discussion, consider a team of humans working in a fast-paced, workstation-based environment (see Figure 1a). Although they are accomplishing objectives as a team, each individual has an assigned role based on his capabilities. Different types of tasks the teams work on may require different contributions from the many roles, governed by some precedence constraints. In order to ensure that the team is able to respond quickly and efficiently to new external factors, internal resource availability problems, or schedule and priority changes, a system is needed that can passively learn and automatically model the workflow of individual operators and teams from very large data sets of operator actions. The models and monitoring of this system can then be used to make real-time resource re-allocation or re-planning decisions in a dynamic environment. This automated system must also enable expert analysts to explore and modify the workflow models.

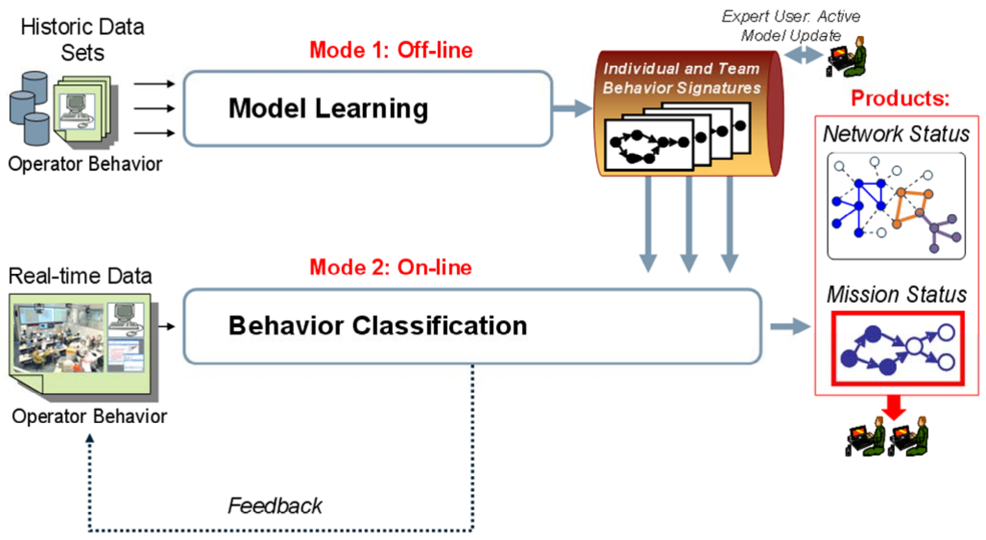
Currently, there are few if any fully automated methods of accomplishing this. Most existing systems analyze the actions of a single operator and use only a single level of data (such as keystrokes) to do so. We approach this problem with determination to (1) leverage different levels of information – from keystrokes to mission models – to constrain inferences concerning tasks (i.e., reduce complexity) and to enable efficient integration of behavior recognition models from different levels; and (2) focus on the behaviors of teams to enable learning and tracking workflow patterns among interdependent and parallel activities.

To this end, we are developing a system called **MIMOD – Monitoring MIssion MODels for Increased Mission Understanding –** to monitor and model the observable behaviors of users at their work stations. As inputs, MIMOD will take user-to-technology interactions and the event stream from user-to-user. These data may include keystrokes and commands, commonly-used networked applications, text communications with other users, geo-spatial areas where users spend the most time, etc. The MIMOD concept has two modes (Figure 1(b)):

- In Mode 1 (offline), MIMOD learns the models of behaviors of the individual operators and the mission of the team based on the ongoing or historic corpus of captured behavior data. With a collection of these observed patterns, analysts and subject-matter experts will be able to explore and tune the operator behavior models as desired.
- In Mode 2 (online), MIMOD recognizes what behavior model is currently active – that is, identifies the structure (often referred to as a *pattern*) of the team's mission workflow (which can be specific to context of the environment and/or team characteristics such as expertise, cohesion, fatigue, etc.), activities and tasks operators are performing and will perform in the future (the state of the mission). By automatically identifying who does what tasks and when, MIMOD allows real-time communication of dependencies within the team members and the mission requirements.



(a) MIMOD Challenge



(b) MIMOD System Overview

Figure 1: The MIMOD Challenge and Overview

MIMOD is intended to support mission-driven network management systems that provide team workflow decision support to warfighters. Such systems could allow the users to: (1) prioritize network resources for ongoing missions; (2) monitor impaired resources and work around them; (3) coordinate interdependent tasks; (4) assess workflow over time, in order to refine it and improve training on challenging tasks; and (5) design future systems that proactively push information to operators to improve task execution and team coordination. Enabled by MIMOD, the mission-driven network management systems will enable agile operations by supporting faster and more efficient retasking, better information flow, and improved collaboration in the context of team- and self-synchronization.

Technical Approach

MIMOD’s workflow learning and tracking technology is based on an information hierarchy that describes operator and team behavior concepts at different levels of granularity. In a feasibility study, we applied the behavior hierarchy to the use-case of a DTC operations scenario, and prototyped and tested several algorithms for different layers of the information hierarchy. The DTC scenario included the observation event data that is similar to the real AOC settings, where constraints on the data collection prevent the systems and human operators from knowing true activities of the users. The activity recognition in our feasibility study was therefore as difficult as it would be in real operational setting.

The information hierarchy. MIMOD’s information decomposition has been developed to quantitatively capture four different layers of information in the behavior representation. Decoupling the behavior models at each layer allows reduction of the complexity and better tailoring of the models to the features of behavior at each layer. Our only requirement is that outputs of lower-level models can be mapped to inputs of higher-level models.

Table 1: Information Decomposition for Operator and Team Behavior Modeling

Concept	Description	Example 1	Example 2
Role / Mission	Responsibility based on tasks performed	Execute plans and follow ATO	Determine weaponeering options
Task	Mission- & process-level engagement	Attack Target	Assign assets + weapons
Activity	Decisions & actions	Move assets in position	Communicate strike package options
Primitive Action/Event	Keystrokes, comms, used applications	Initiate asset movement Ask state of target	Open comm with DTC

The information decomposition defined in this proof-of-concept effort has the following four layers (Table 1):

- **Layer 1: Primitive Actions/Events.** These are data elements that are directly captured from the data and usually cannot be further decomposed. An element could be a mouse click, an application used, a word communicated, etc. Primitive events form activities.
- **Layer 2: Activities.** These consist of multiple events, and they represent purposeful action or decision performed by a single operator. The structure among primitive events as they

come together to form an activity is usually simple. For reasons of simplicity, we assume that such structure has limited temporal constraints. Activities form tasks.

- **Layer 3: Tasks.** These are mission- or process-specific operations that must be conducted by operators in the team. To successfully accomplish a task, an operator must perform a set of activities. A structure of activities would have geo-spatial and temporal constraints. Examples of the task include attacking specific (possibly multiple) targets, making decisions and plans, etc. Tasks together form a mission of the team. For simplicity, we currently assume that each task can be performed by a single operator.
- **Layer 4: Role & Mission.** *Role* is a latent variable that defines the responsibility of the operator in the team. It can be found by observing the profile of tasks performed by the operator(s). For example, if the role of an operator were to determine weaponeering options, then we would see the activity of the operator related to resource allocation. *Mission* is the structured set of tasks that must be executed by (possibly different) team operators. The task structure defines the mission constraints, including task precedence and information flow.

The use of this hierarchy of information enables MIMOD to conduct behavior model learning and classification performed top-down, bottom-up, or a hybrid of these two.

MIMOD use-case: types of events, activities, and tasks. The initial MIMOD use case was built for the DTC, a component of the Offensive Operations Team within the Combat Operations Organization of the AOC. The Offensive Operations Team is responsible for executing the Air Tasking Order (ATO) and makes command and control decisions to ensure the theater offensive air campaign are executed for all offensive missions. The DTC is incorporated within the Offensive Operations Team to identify targets with sufficient value for prosecution within the current ATO. The DTC is capable of nominating or receiving dynamic or emerging targets from all stages of the mission.

To complete the stages of the mission, the DTC follows a Time Sensitive Target (TST) Process. The TST Process, also known as the kill chain, presents information in order to provide a way to identify and group the states of potential targets. The key functions of the kill chain are to find, fix, track, target, engage, and assess the potential targets that emerge in the course of a theater offensive air campaign. To test this initial use case, Aptima has streamlined the process to fit the phases of data already captured and evaluated in Aptima’s Distributed, Dynamic Decision-making (DDD) simulation test bed.

The DTC roles captured in the MIMOD use case are the following:

- **Representative of the Intelligence, Surveillance, and Reconnaissance Cell.** The representative of the Intelligence, Surveillance, and Reconnaissance (ISR) cell is responsible for detecting and identifying Dynamic Targets, verifying Dynamic Targets from the TST nomination list, and performing Battle Damage Assessment (BDA). The ISR ensures that ISR information and support is communicated to the DTC Chief and Ground Track Officer (GTC).
- **Ground Track Coordinator.** The Ground Track Coordinator (GTC) is responsible for monitoring and managing the emerging Dynamic Targets. The GTC is the point of contact for the generating and target accuracy of the ground tracks generated from the Operations floor. The GTC ensures the deconfliction of target information from traditional ISR platforms and non-

traditional ISR platforms and communicates this target information to the DTC Chief, Attack Coordinator (AC), and ISR.

- **Attack Coordinator.** The Attack Coordinator (AC) represents the person who is responsible for determining the best weaponeering option available given targeting constraints. The AC is paired with a Target Duty Officer (TDO). The AC must maintain high situational awareness of the battle-space conditions for their area of responsibility. This includes air and ground threats, weather, terrain, and available assets. The AC is responsible for building a targeting package and presenting it to the DTC Chief for approval.
- **Target Duty Officer.** The Target Duty Officer (TDO) is responsible for all knowledge that relates to weapon effects and ensuring that the package is compliant with collateral damage and positive identification requirements. The TDO must also deconflict the target with no-strike and restricted target lists and communicate this target information to the DTC Chief, AC, and GTC. The TDO is the focal point for all ISR information and support.
- **DTC Chief.** The DTC Chief represents the person who supervises all the targeting functions and coordination of cross-component fires for prosecution of TSTs. This person provides inputs to the Senior Offensive Duty Officer (SODO), approves nominations for TST prosecution from the Senior Intelligence Duty Officer (SIDO) and approves target packages developed by members of the DTC cell.
- **Chief of Combat Operations and Senior Offensive Duty Officer**. The Chief of Combat Operations (CCO) and Senior Offensive Duty Officer (SODO) execute the plans and follow the Air Tasking Order.

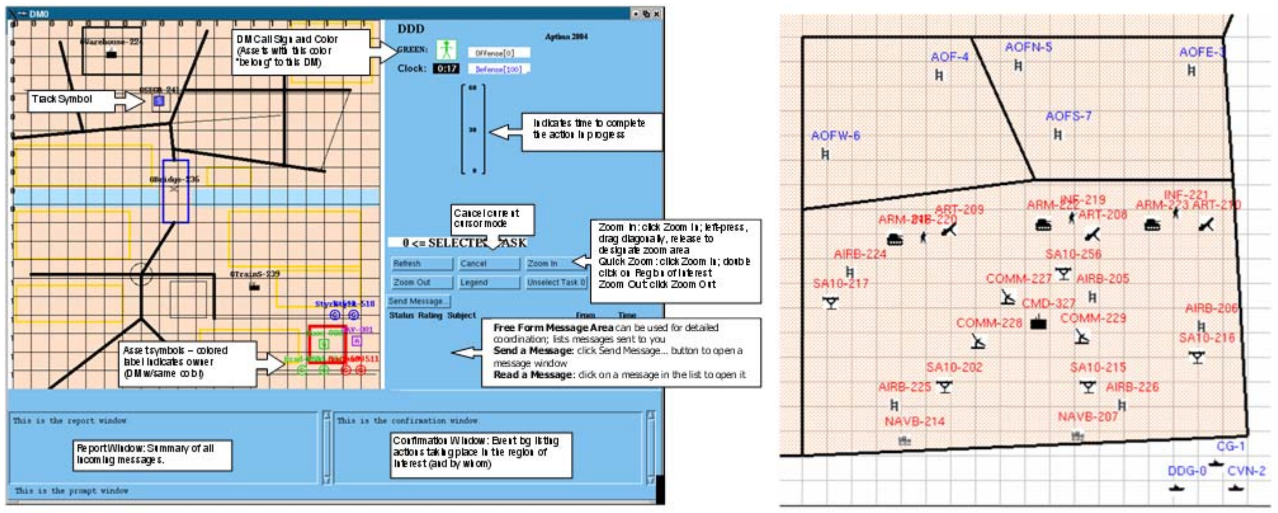
DTC dataset and corresponding human-in-loop experiments. The DTC scenario included the observation event data that is similar to the real AOC settings, where constraints on the data collection prevent the systems and human operators from knowing true activities of the users. The activity recognition in our feasibility study was therefore as difficult as it would be in real operational setting. The dataset was created using the human-in-loop experiments that Aptima and our partners from Write State University have conducted over the years to study organizational reconfiguration, adaptation, and human decision making in collaborative environments.

Our data set included log files from 28 human-in-the-loop experiments conducted using Dynamic Distributed Decision making (DDD) simulation environment. Each experiment represented a virtual battle between the BLUE (friendly forces represented as U.S. Joint Taskforce) and RED (adversaries) assets. The behaviors of RED forces have been scripted, while BLUE command roles representing the DTC and coordinating commanders have been played by seven human participants as follows:

- DTC team: GTC, AC, TDO, Chief (four participants)
- ISR team: ISR (one participant)
- Offensive operations: CCO, SODO (2 participants).

For each scenario, the participants were provided with Air Tasking Order (ATO) and Time Sensitivity Target (TST) guidance. The specific ATO given to the team varied in content by scenario. In general, however, it was a list of approximately 25 targets to be prosecuted, including the weaponry required for successful attack, the time window in which the attack must be accomplished, and an indication of whether Air Support would be required simultaneously. These tasks were then carried out by the SODO and the CCO with their assets (generally 2 F-15E, 3 F-15C, 3 F-16CJ, 2 AV8B, 2 KC-135, and 2 F-18E with suitable combinations of ammunition) during the simulation. Similarly, the TST list (unknown to the simulation participants in advance) would include some combination of the possible TSTs that had been selected for the scenario. The operators would have a list describing the possible types of targets and the appropriate response (priority, weaponry requirements, and time window restrictions). These TSTs included vehicles carrying bioweapons, nuclear weapons, or chemicals; ballistic missile launchers or facilities; scud launchers; adversarial submarines of different kinds; special operations on boats; leadership convoys; and enemy destroyers or aircraft carriers.

The participants controlled different BLUE assets (air and ground) in the game, able to geo-spatially maneuver them and engage in the DDD’s virtual battlefield against RED assets. Each operator worked from a screen such as the one shown below in Figure 2a. In Figure 2b, we show the initial battlefield and terrain layout. The friendly forces control the five airbases in the north. The BLUE forces also had a carrier (CVN-2), a cruiser (CG-1), and a destroyer (DDG-0) positioned south of the hostile nation. The enemy had a command center (CMD) and three communication nodes (COMM) in the center of the country; five air bases and two navy bases; multiple SA-10, ground-launched cruise missile and theatre ballistic missile sites; nuclear-tipped missiles in reinforced concrete silos; three armored and artillery battalions; three infantry brigades; and a bio-weapons facility. Typical engagements included the attacks against RED and BLUE assets using varied asset packages and munitions.


DDD
DM Call Sign and Color (Assets with this color "belong" to this DM)
Track Symbol
Indicates time to complete the action in progress
Cancel current cursor mode
Zoom In: click Zoom In; left-press, drag diagonally, release to designate zoom area
Quick Zoom: click Zoom In; double click on Region of Interest
Zoom Out: click Zoom Out
Free Form Message Area can be used for detailed coordination; lists messages sent to you
Send a Message: click Send Message... button to open a message window
Read a Message: click on a message in the list to open it
Asset symbols – colored label indicates owner (DM w/same color)
Report Window: Summary of all incoming messages.
Confirmation Window: Event log listing actions taking place in the region of interest (and by whom)
AOF-4
AOFN-5
AOFE-3
AOFS-7
AOFW-6
ARM-218
ART-209
ARM-222
INF-219
ART-208
ARM-223
INF-221
ART-210
AIRB-224
SA10-256
SA10-217
COMM-227
AIRB-205
CMD-327
COMM-228
COMM-229
AIRB-206
SA10-216
SA10-202
SA10-215
AIRB-225
AIRB-226
NAVB-214
NAVB-207
CG-1
DDG-0
CVN-2

(a) Example DDD Operator Screen (b) Initial Battlefield Layout

Figure 2: DTC Experiment Layout in DDD Simulator

Each experiment in DDD can capture various events, including:

- information about asset maneuvers (as the players moved their assets in the virtual space), ground and air attacks (BLUE engaging RED assets),
- RED attacks on BLUE targets,
- communications among players (communication was done over the chat function in DDD) and its partial content (including communicating target detection and identification as coordinates and types of the target, proposing or ordering asset packages, providing

information about asset status, informing about intent to prosecute, presenting measurements and assessments of the operator, requesting authorization, providing approvals, etc.), and

- information about battlefield exploitation (in the form of zooming in the terrain, querying the data about targets and assets, and allocating assets to the targets and engagements).

The particulars of the experiment and types of data captured from the player’s interaction with the DDD game were mimicked after the real AOC settings, where information about communications, uses of applications and services, data queries, information consumption, and product generation can be captured, while the true activities and tasks of operators is not available. Recognizing the activities and tasks of operators was the first problem that MIMOD’s system had to solve. For more information about the DTC experiments, see (Shebilske et al., 2007, 2008).

The DTC data was configured and stored as a collection of **data entries**. Entries were extracted from the user-to-computer interactions during the experiments. All entries had the following *content fields*:

- *`timeRecorded`* specifies the time during the mission that the entry occurs and is recorded (note: to simplify the data processing, in current phase of the project we ignore the situations in which entries are recorded some time after they occur);
- `locRecorded` specifies the geo-spatial location of the data entry in the playfield;
- *`ownerID`* specifies the operator who generated the entry;
- `receiverID` specifies the operator who received the entry;
- `targetID` specifies the target (or mission task) that is associated with the data entry;
- *`eventID`* specifies the event associated with the data entry (this information is known);
- **`mappedActivityID`** specifies the activity that the model associates the data entry;
- **`mappedTaskID`** specifies the task that the model associates the data entry;

Similar to “mapped” activity and task, we have generated the **ground truth** of event-to-activity-task for all data entries by semi-manually labeling the data stream.

The content fields in italics above (e.g., *`timeRecorded, ownerID, eventID`*) have values for all data entries. The content fields in bold (**`mappedActivityID, mappedTaskID`**) are determined by the model. Other content fields can contain “null” information, due to absence of the meaningful information for the data entry (e.g., the communication does not have playfield locations), or due to information that was unknown at the time of recording the data entry (e.g., for “asset move” events the target ID may not be known at the time of entry recording).

For our use case, we identified 9 classes of events, 8 types of activities, and 7 types of tasks. We then constrained what events could correspond to which activities, and which activities could correspond to what tasks. Given these constraints, we extracted a training data from DTC human-in-loop experiments. Experiments were performed by different human players performing same seven roles of DTC operators. The DTC was faced with the scenario defined through the ATO and the frequencies and types of unknown targets. Some experiments had the same scenario, but without loss of generality we assume that all experiments contained varied but related scenario situations. Each experiment was documented and the data saved into a *log file*.

Hierarchical Pattern-matching Algorithms

In this section, we describe the general challenges of behavior and activity recognition. There is no single model that can best represent and categorize all types of behavior. Behaviors differ based on context (e.g., mission type and tempo), the level of information (e.g., mission-level tasks versus primitive activities), the source of behavior (e.g., team versus a single operator), etc. In addition, the data that can be observed about behavior have several limitations: even though we are dealing with persistent data collection, the information that could be used for behavior recognition will naturally be imperfect due to missing context and ambiguous observations. As a result, we have begun with analysis of the features of the behaviors that must be captured and the constraints of feasible observation. This list appears below:

- **Effects of latent (unknown) variables on operators’ behavior:** The observed behavior can be affected by goals and expertise of the operator, as well as other variables not known to the model but that must be discovered during the training process.
- **Temporal constraints:** Some primitive elements that make up the behavior models (e.g., tasks composed of activities) will have temporal logic ordering, where one activity must be completed before another could start.
- **Duration:** Some primitive activities occur over time and span different time intervals; therefore modeling their duration is important.
- **Variability:** The models must capture different choices that are made by operators. Due to natural variability of people’s behaviors, many actions are neither necessary nor sufficient and therefore they may or may not be present in a particular instance of an activity. As a result, the models must capture variation in the signals from the activities of the same type – that is, many different lower-level actions in different behavior instances for the same higher-level event.
- **Errors in observed event stream:** Lower-level models might make mistakes in recognizing the situation and therefore provide erroneous outputs to higher levels;
- **Irrelevant observed data:** Many irrelevant activities (clutter) exist in collected data that are not caused by the goals of the operator;
- **Ambiguity of observed data:** The same event at lower level may have been caused by many different events at higher levels, and thus no single decision can be made due to a repeated lower-level event.

For the first proof-of-concept analysis, we designed and used the following process of operator behavior analysis. First, we noted that the DTC use case contained two main types of targets: planned ATO targets, for which the time window and asset packages have been predetermined, and unknown targets, which must be found, fixed, tracked, etc. For non-ATO targets, we therefore have a sequence of tasks that must be performed by different operators of the team, which we call a “mission”. Therefore, multiple such missions are active at the same time, and the number of missions depends on the number of targets.

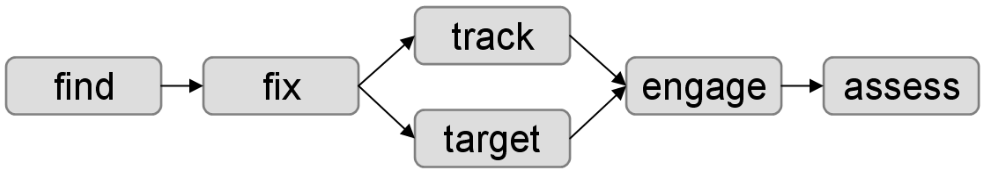


Figure 3: Example of the “Mission” Structure for a non-ATO Target

Figure 3 shows an example of the structure of a mission for a single target. Each node in the graph represents a task that is executed by the DTC operator. Each task involves (possibly multiple) activities, which are represented in the observable data by events. The links in Figure 3 represent the precedence constraints among the tasks and possibly indicate information flow that will occur among operators performing the tasks.

Different kinds of targets may result in different precedence/flow structures, and each task for a corresponding target’s mission may have a different activity model structure. MIMOD is then used in two modes to discover:

- *Mode 1 (offline – “learning”):*
    - the task-based structure of mission models;
    - the activity-based structure of task models for each mission (target class);
    - the event-based structure of activity models for each mission (target class);
- *Mode 2 (online – “classification”):*
    - what “missions” are active;
    - what is the current state of each active mission (what tasks have been accomplished, what tasks are ongoing, and what tasks will be done next);
    - what is a state of each task in the mission (which activities have been completed, what are next activities to be executed);
    - which operators are involved in which tasks.

As we have studied different classes of behavior models that would satisfy the modeling requirements listed in the previous section, we realized that two more requirements must be considered:

- **<u>Complexity of learning must be low:</u>** Learning complexity includes the algorithm complexity, data and memory requirements for learning model patterns from historic data. To enable efficient MIMOD utilization, the system should reside on a single PC-type computer, with algorithm run-time requirements much lower than the real-world clock of the analyzed behaviors (e.g., the month-worth of team data should be processes in minutes-to-hours).
- **<u>Complexity of classification must be low:</u>** Classification complexity includes the algorithm complexity, data and memory requirements for recognizing the behavior workflow structure/pattern and its state. The behavior classification components must be able to run in real time to provide recognition of true tasks and activities performed by operators in timely fashion for instructive or corrective decision analysis.

Table 2 lists the algorithms (in rows) we studied as candidates to be applied at different levels of behavior modeling. Below is a summary of benefits and challenges of each algorithm broken down by its behavior modeling requirements and complexity (in columns).

Table 2: Features of Analyzed Behavior Models (Coloring indicates the capability to address the feature requirements; Green = best, Yellow = moderate, Red = poor)

Model	Latent Effects	Temporal constraints	Duration	Variability	Data Errors	Irrelevant Data	Data Ambiguity	Complexity of learning	Complexity of Classification
LVA	🟩	🟥	🟥	🟨	🟩	🟩	🟨	🟩	🟩
nGrams	🟨	🟨	🟥	🟨	🟨	🟨	🟨	🟩	🟩
HMMs	🟩	🟩	🟨	🟨	🟩	🟨	🟩	🟨	🟩
SCFG	🟨	🟩	🟨	🟩	🟥	🟥	🟩	🟥	🟨
CRFs	🟩	🟩	🟨	🟩	🟩	🟩	🟩	🟨	🟨
PNetR	🟩	🟩	🟩	🟨	🟩	🟨	🟨	🟥	🟨

- **Latent Variable Analysis (LVA):** this is a set of models that represent the events occurring in an unordered manner depending on the latent (unknown) variables that define higher-level concepts. LVA algorithms can learn the behavior models in unsupervised manner. We propose to use these models for activity recognition.
- **nGrams:** these are models that capture simple temporal occurrences of events, and can be efficiently used for behavior structure learning in unsupervised manner and develop (using motifs-based algorithms) the natural descriptions of the learned models to be presented to the analysts. We propose to use these models for activity recognition.
- **Hidden Markov Models (HMM):** these are stochastic models that represent hidden evolution of the true signal and allow temporal modeling. The models can be efficiently learned, and the classification allows pattern and state learning. We propose to use these models for task recognition.
- **Stochastic Context-free Grammars (SCFG):** these models extend HMMs to model more complex signals, but at the cost of increased learning and classification complexity. These models can be used for task recognition.
- **Conditional Random Fields (CRF):** these models extend the modeling power of HMMs while simplifying some of the requirements of SCFGs. We will use these models for task recognition.
- **Plan Network Recognition (PNetR):** these models capture parallel activities and map actions to operators to recognize who is doing what. We propose to use this model for team mission and role recognition.

Following this analysis of various algorithmic approaches to learning and classification, we chose HMMs and PNetR for prototype implementation. Details of each are given below.

Task Behavior Analysis using Hidden Markov Models

Tasks are high-level operations performed by the members of the team. They are either planned, or conducted to respond to specific events. We can conceptualize the tasks as operations that need to be accomplished to achieve some objectives/goals. Each task then consists of activities performed by the operator. The structure of the activities, and the types of activities, then distinguish one task from the other. The activity-based structure of the tasks must capture the temporal evolution of the activities comprising the tasks and uncertainty of observations about the activities, especially ambiguity of inferring the true task from the observed activities. Such challenges are addressed efficiently using Hidden Markov Models (HMMs) to represent the task structure over time. HMMs are stochastic models that represent hidden evolution of the true signal and allow temporal modeling. The models can be efficiently learned, and the classification allows pattern and state learning.

Each HMM corresponds to a single task; it represents the task dynamics as the evolution of operator activities. HMMs constitute a principal method for modeling partially observed stochastic processes and behaviors – processes that have structure in time. An HMM can sequentially process new information (a window of data) each time an observed transaction (in our case an activity recognition at the previous level) occurs. The window of observations could contain a single or a batch of observations and activities to improve the efficiency of a solution. The premise behind HMMs is that the true underlying process (defined as a Markov chain representing the evolution of the activities as a function of time, an example of which is shown in Figure 4) is not directly observable (hidden), but it can be probabilistically inferred through another set of stochastic processes (observed inferences of activities performed at activity recognition layer). HMMs can be constructed by SMEs or learned from historic data defining the task models in the form of underlying activity structures.

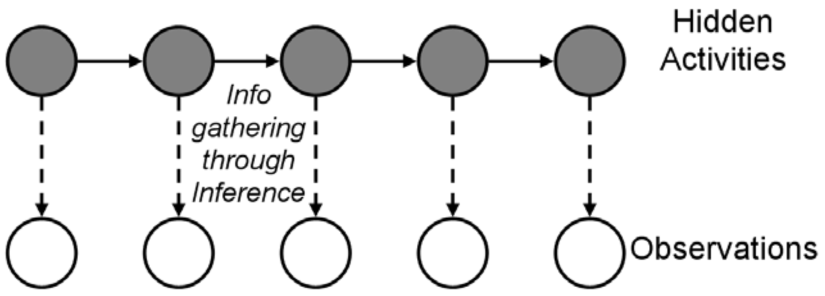


Figure 4: HMM Representation

HMMs can classify hidden task models based upon a set of activity observations because hidden states are statistically related to a noisy observation process. Each state of our HMMs will consists of single or multiple activities, while each observation consists of inferred activities. Inference is made at the activity recognition layer.

Behavior classification using HMMs: HMMs can be used to identify which task is performed at current time, find what is the state of task execution (which activities have been executed so far), and forecast the activities that might be performed next.

Identifying the task being performed by the operator. Every task model is represented with the HMM $\mu = \{A, B, \pi\}$. To achieve this capability, we find the model that has most likely generated the sequence of observations $Y = \{y_1, y_2, ..., y_T\}$, i.e. the model that maximizes the likelihood function $\mu^* = \arg\max_{\mu} P\{Y \mid \mu\}$. We perform computations of the likelihood score $p\{Y \mid \mu\}$ for each model μ using **forward algorithm** with iterations $\alpha_{t+1}(j) = \sum_i \alpha_t(i) a_{ij} b_j(y_{t+1})$. We obtain $p\{Y \mid \mu\} = \sum_i \alpha_T(i)$, and then select the straightforward maximum of the likelihood scores.

Finding the true sequence of activities performed by the operator over time. Given a sequence of observations $Y = \{y_1, y_2, ..., y_T\}$, we are interested in finding the most probable sequence of states $X = \{x_1, x_2, ..., x_T\} \subset \Omega$ that generated these observations. We find X by maximizing the joint probability $P\{X, Y \mid \mu\}$ via Viterbi algorithm. We define the quantity $\delta_t(i) = \max_{x_1,...,x_{t-1}} P\{x_1, ..., x_{t-1}, x_t = \xi_i, y_1, ..., y_t \mid \mu\}$ equal to the highest probability along a single path at time t, which accounts for the first t observations and ends in state ξ_i. This quantity can be computed by induction as $\delta_1(j) = b_i(y_1)\pi_i, \delta_{t+1}(j) = b_j(y_{t+1}) \max_i \delta_t(i) a_{ij}$. The path $\{x_1, x_2, ..., x_T\}$ is then reconstructed backwards as $x_T = \arg\max_i \delta_T(i)$, $x_t = \arg\max_i \delta_t(i) a_{i,x_{t+1}}, t = T-1, ..., 1$.

Forecasting future activities of the operator. Given the sequence of observed activities $\{y_1, y_2, ..., y_t\}$, we need to find what activities we might expect next. One example of the solution is to find the probability that next activity will be ξ_i.This can be achieved by finding the probability $P\{x_{t+1} = \xi_j, y_1, ..., y_t \mid \mu\} = \sum_i \alpha_t(i) a_{ij}$.

Similarly, we can use dynamic programming recursion to discover the highest-probability sequence of activities to follow $\{x_t, x_{t+1}, ..., x_T\}$:

- First, compute iteratively $\omega_t(i) = \alpha_t(i), \omega_{t+1}(j) = \max_i \omega_t(i) a_{ij}$
- Second, reconstruct the path as $x_T = \arg\max_i \omega_T(i)$,

  $x_\tau = \arg\max_i \omega_\tau(i) a_{i,x_{t+1}}, \tau = T-1, ..., t+1$

For every activity, we can compute a probability of its occurrence between time t and T as follows:

- First, use forward computations to find $\psi_t(i) = \alpha_t(i), \psi_{\tau+1}(j) = \sum_i \psi_\tau(i) a_{ij}$
- Second, compute aggregated score $\rho(i) = \sum_{\tau=t+1}^{T} \psi_\tau(i) \theta^{t+1-\tau}$, where $\theta \in [0,1]$ is the time decay.

The function $\rho(i)$ scores a reachability of state ξ_i weighted by time criticality.

Team and Mission Behavior Analysis using Probabilistic Network Recognition Models

The targets are executed by different operators in coordinated manner as they perform different tasks needed for successful target prosecution. If we only model a single operator, we would not be able to understand its dependencies on other team members. Therefore, the model that determines “who does what” and “who depends on whom” must map the tasks in the target-based mission to the operators who perform them. The task-based structure of the mission must capture temporal dependencies among tasks, the possibility of parallel task execution, and the uncertainty and ambiguity of the observed data.

Plan network recognition (PNetR) models the mission as a planned set of tasks. The structure of the mission is defined as a graph with nodes representing the tasks and links representing the relationships among tasks (Figure 5(a)). The tasks have states, which can be defined based on the task execution status (progress of completing activities composing the task). The example in Figure 5(b) shows the dark circles indicated completed tasks and white circles representing the tasks that have not been started. Mission states can be feasible (Figure 5(b)) and infeasible (Figure 5(c)) – where the task status does not satisfy task precedence constraints.

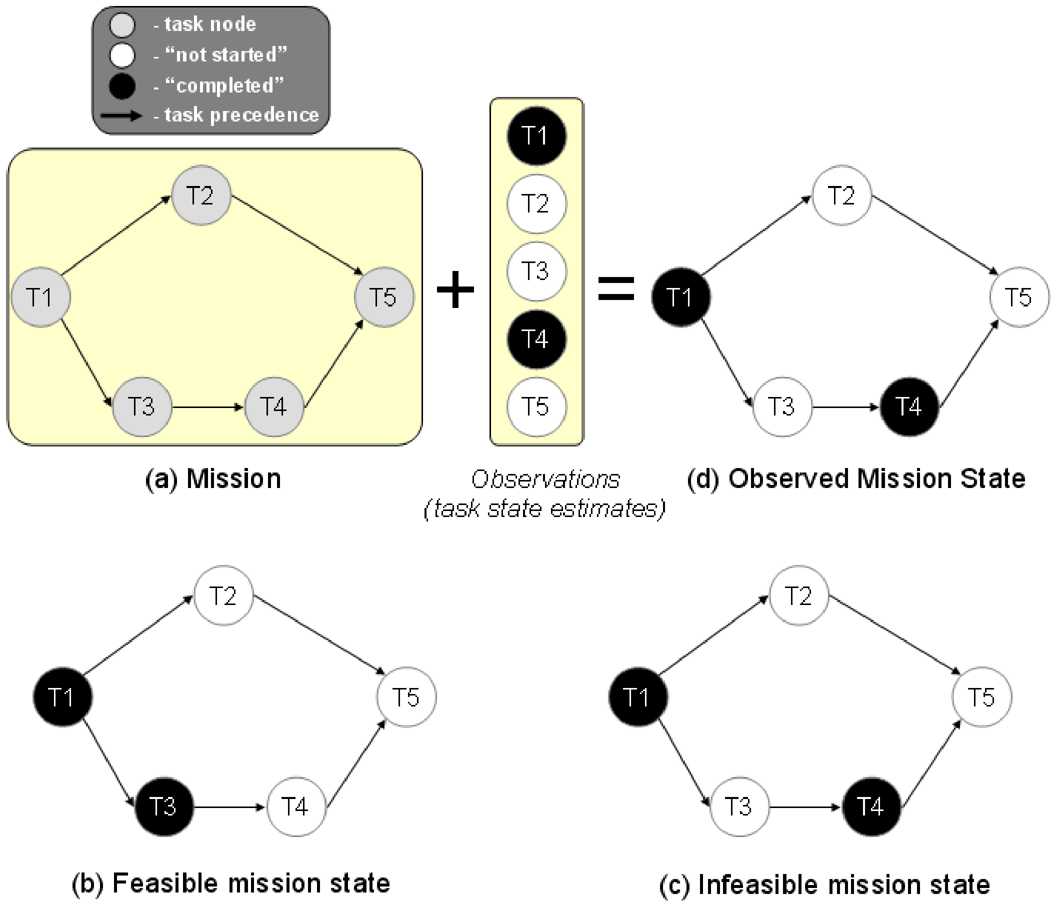

- task node
- “not started”
- “completed”
- task precedence
T1
T2
T3
T4
T5
+
=
(a) Mission
Observations (task state estimates)
(d) Observed Mission State
(b) Feasible mission state
(c) Infeasible mission state

Figure 5: PNetR model --- Example of mission and its state

The hidden state of the mission – true tasks and their states – is not known. Instead, we may assume that we can obtain the estimates of states of the tasks and accordingly construct observed state of the mission (Figure 5(d)). Then, the true hidden mission state need to be inferred based on the observed state. The complication of task-based mission modeling in MIMOD is that in cases the operators have overlapping expertise (roles), we do not know what operator completes what task, and as the result we do not have estimates of the task state.

Mapping of task to operators is obtained to maximize the likelihood probability that team observations have been generated by the mission and corresponding task-to-operator mapping: $P\{Y(i),Y(i,j);i,j=1,...,N \mid M,S\}=P\{A^D \mid A^M,S\}$, where

- $Y(i)=\{y_1(i),...,y_T(i)\}$ are activity observations at operator i;
- $Y(i,j)=\{y_1(i,j),...,y_T(i,j)\}$ are activity observations between operators i and j (e.g., from their communications);
- $A^D=\| a_{ij}^D \|$ with $a_{ii}^D=Y(i)$, $a_{ij}^D=Y(i,j)$ define observations;

- $A^M = \| a^M_{km} \|$ define the mission model, where a^M_{kk} is activity-based model of the task k (e.g., defined using corresponding HMM), and a^M_{km} is model of the flow between tasks k and m.

We then have: $P\{A^D \mid A^M, S\} \approx \prod_{km,k\neq m} P\left(a^D_{ij} \mid a^M_{km} : s_{ki} = s_{mj} = 1\right) \prod_k P\left(a^D_{ii} \mid a^M_{kk} : s_{ki} = 1\right)$. We then can write a negative log-likelihood function as $-\log P\{A^D \mid A^M, S\} \approx -\sum_{ki} s_{ki} \log P\left(a^D_{ii} \mid a^M_{kk}\right) - \sum_{kmij} s_{ki} s_{mj} \log P\left(a^D_{ij} \mid a^M_{km}\right)$.

We now need to account for the knowledge of the task state. Essentially, if the task state is = 0 (task has not been executed), $P\left(a^D_{ii} \mid a^M_{kk}\right)$ is equal to the probability of receiving randomly observations a^D_{ii}. If task state = 1 (task has been executed), we can use the outputs of task recognition layer, as we have obtained from the activity recognition and HMM calculations the likelihood functions $P\left(a^D_{ii} \mid a^M_{kk}\right) = P\{Y(i) \mid k\}$. We can similarly profile the activities between operators.

As the result, we find the mapping by maximizing the quadratic function $Q(s) = \frac{1}{2} \sum_{kimj} c_{km;ij} s_{ki} s_{mj} + \sum_{ki} c_{ki} s_{ki}$, where

- $c_{ki} = -\log P\left(a^D_{ii} \mid a^M_{kk}\right)$, and can be interpreted as a mismatch between model (activity profile) of task k and observed activities for operator i, and
- $c_{ik;mj} = -2\log P\left(a^D_{ij} \mid a^M_{km}\right)$, and can be interpreted as a mismatch between model of flow (activity profile) between tasks k and m versus observed activities between operators i and j.

We force $c_{ik;mj} = 0$ and $c_{ki} = 0$ if state of task k is = 0.

PNetR parameters:

Formally, according to above, we represent a hypothesized mission model as a graph $G_M = (V_M, E_M, A_M)$, – a *model network* where V_M is a set of task nodes, E_M is a set of edges among them, and $A_M = \| a^M_{ij} \|$ is a matrix of model attributes on nodes and links (a^M_{ii} is attributes vector for task node i and a^M_{ij} is attribute vector for link between nodes i and j). Essentially, G_M is a structure of the mission with attributes defining what expertise requirements are for operators and what observations could be collected if the task is executed.

We observed the data about task executions by operators – coming from task recognition layer. This observed data is aggregated to a *data network* – a graph $G_D = (V_D, E_D, A_D)$ with V_D defining a set of observed operator nodes, E_D is a set of observed relationships among them, and $A_D = \| a^D_{ij} \|$ is a matrix of data attributes (task execution/activity profiles) on nodes and links.

Mission model networks are defined using the set of parameters $\theta = \{A_M, P(G_M)\}$, where A_M is the mission network attributes tensor and $P(G_M)$ specifies unknown prior probability for the mission behavior models. These parameters can be learned iteratively in unsupervised manner from a set of historic observations of team behavior represented as network states G_D^t (defined via attribute matrix $\| a_{ij}^D(t) \|$) for times $t = 1,...,T$.

The Expectation Maximization (EM) algorithm can be used to find the model parameters θ. The EM iteratively improves initial estimate θ_0 and generates estimates $\theta_{n+1} = \arg\max_{\theta} Q_n(\theta)$,

where $Q_n(\theta) = E_M \left[\ln \prod_t P\left(G_D^t \mid G_M, S, \theta\right) \middle| G_D^t \right]$.

Behavior classification using PNetR:

Each missions hypothesis is scored using a likelihood function $\mathrm{P}(G_D \mid G_M, S)$ – a probability that observations were generated by the team’s mission G_M if the operators performed tasks based on mapping S.

We find mapping S by solving quadratic assignment problem

$$S^* = \arg\min_S Q(s) = \frac{1}{2} \sum_{kimj} c_{km;ij} s_{ki} s_{mj} + \sum_{ki} c_{ki} s_{ki}$$
$$s.t. \begin{cases} \sum_i s_{ki} = 1 \\ s_{ki} \in \{0,1\} \end{cases}.$$

Such a quadratic assignment problem can be efficiently solved using *graduated assignment algorithm* (Rangarajan, Yuille and Mjolsness, 1999) that iteratively finds a continuous approximation matrix.

Prototype instantiation and results

We developed a MIMOD software prototype to demonstrate the functionality of behavior analysis models. We have implemented a simulation replay capability (see Figure 6): the log files stored in data base containing events from the previous DTC human-in-loop empirical studies are fed into a simulation model that visualizes the geo-spatial terrain, replays all experiment events (asset movements, engagements, communications), and can be used to compare the behavior predictions against true activities. The scenarios played out in the simulations have the six roles described above (controlled by human operators) working both to execute tasks on the pre-existing Air Tasking Order (ATO) and to process dynamic, Time Sensitive Targets (TSTs) as they come in.

To exercise our prototype algorithms implementations, we have conducted several simulation studies to assess the accuracy of behavior recognition. Our data set included log files from 28 human-in-the-loop DTC experiments.

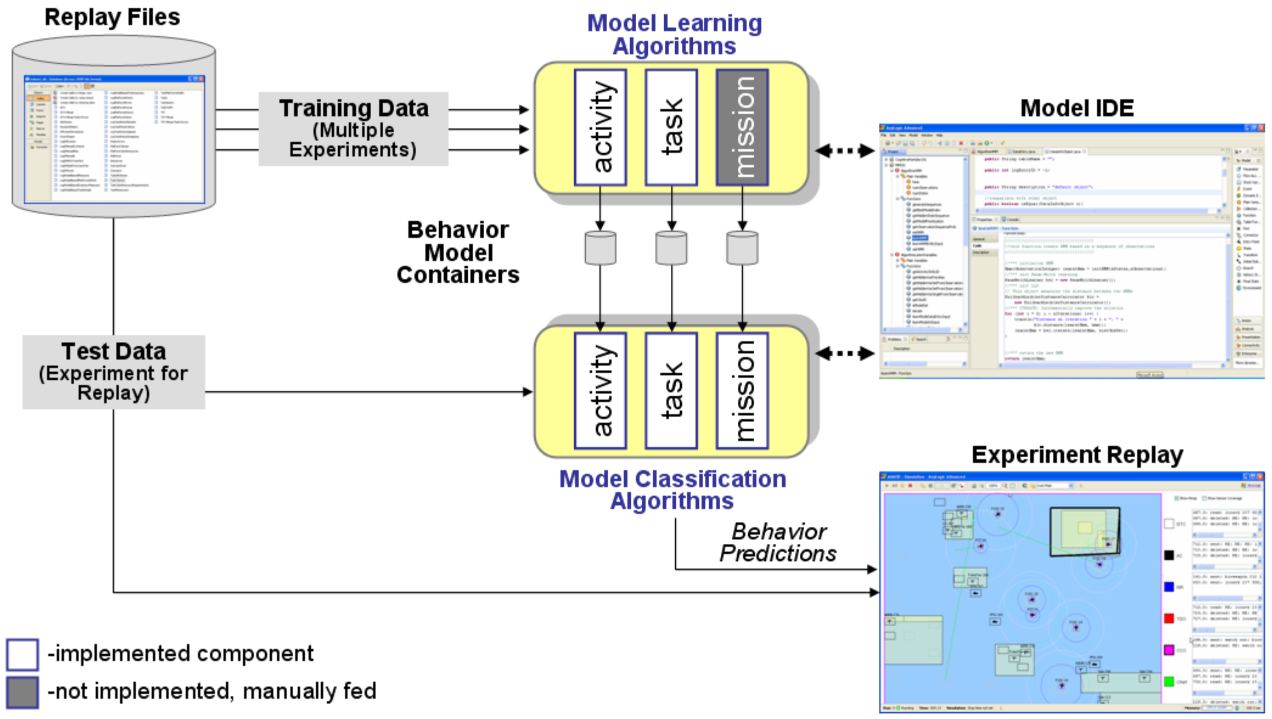

Replay Files
Training Data (Multiple Experiments)
Model Learning Algorithms
activity
task
mission
Model IDE
Behavior Model Containers
Test Data (Experiment for Replay)
activity
task
mission
Model Classification Algorithms
Experiment Replay
Behavior Predictions
-implemented component
-not implemented, manually fed

Figure 6: MIMOD SW Prototype Architecture

Each experiment’s mission contained on average 40-53 target classes. To learn the models of activities, tasks, and missions, we extracted the historical sequences for each *operator* and *target class*. We assumed that behavior patterns for distinct target classes might be different, while behaviors for multiple target instances of the same class were similar and thus belong to the same model. As the result, we obtained approximately 7 (number of operators) X 53 (maximum number of target classes in a scenario) = 371 historic data logs, each containing from several to a hundred time-stamped sequences of data entries. Figure 7 shows conceptually how the data set for training/testing the model was extracted from the full data set for a single experiment. The data in a log consisted of blocks of event sets – one for each experiment. The block contained events for the mission tasks that operators were executing, with each such set containing sequences of events identified with a unique target (of the log’s class). The testing/training data were extracted for all experiments, with a data sequence identified with log ID and target. For each of 371 logs, we learned the models of 8 activities and 7 tasks, for the total maximum number of 2968 activity models and 2597 task models. Note that some of these models would not contain any data, since some operators never performed certain activities and tasks. We assumed that each target generates a mission, so there were 53 classes of missions.

First, we needed to create ground truth. We did this with help of the algorithms and manual relabeling. Each event in the data was assigned a true activity and task that it corresponded to. We did the labeling of activities based on output of the pLSA algorithm and then manually relabeled for correctness. Due to this setup, since our manual labeling was not rigorous, we decided not to estimate the accuracy of pLSA, as results would be biased. We then conducted three tests. First, we attempted to recognize both the activities and tasks performed by operators using HMMs. Results of this analysis are shown in Figure 8, marked “ActivityAccuracy (layer 3)” and “TaskAccuracy (layer 3)” respectively.

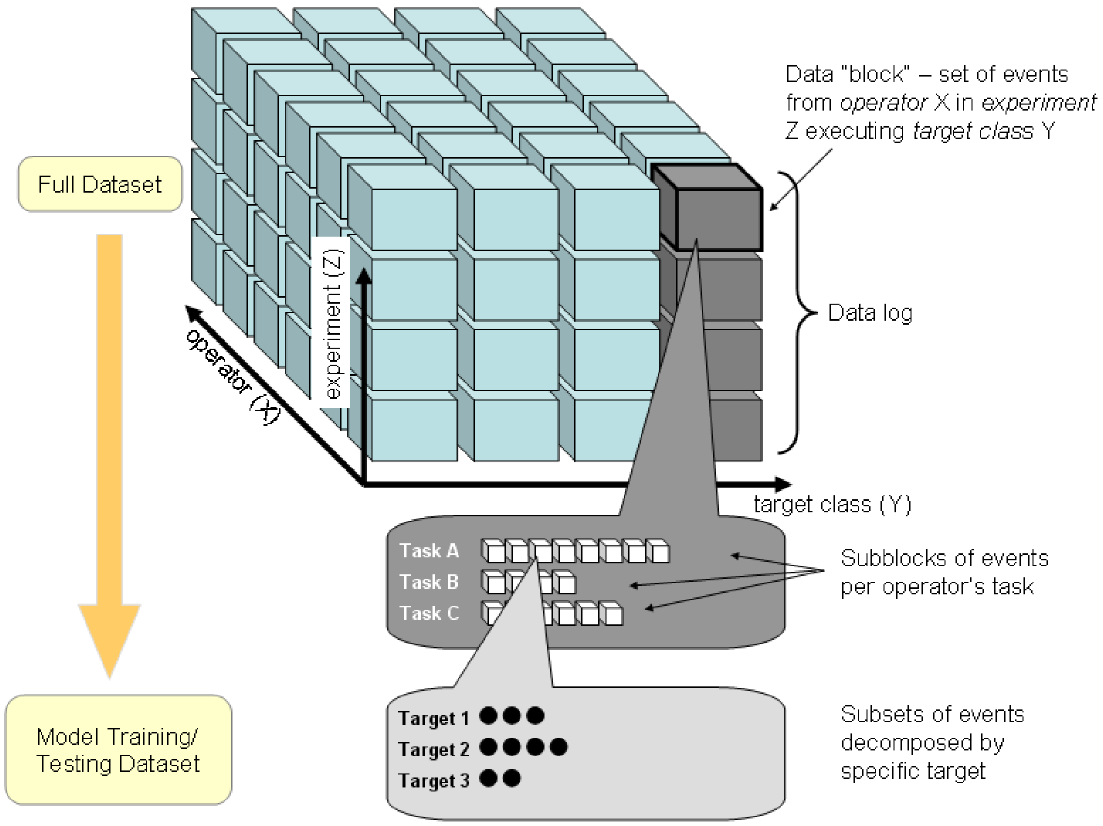

Data "block" – set of events from *operator* X in *experiment* Z executing *target class* Y
Full Dataset
experiment (Z)
operator (X)
Data log
target class (Y)
Task A
Task B
Task C
Subblocks of events per operator's task
Target 1 ●●●
Target 2 ●●●●
Target 3 ●●
Subsets of events decomposed by specific target
Model Training/ Testing Dataset

Figure 7: MIMOD Experiment Data Setup

Following this analysis, we investigated the effect of ambiguity of task recognition at activity layer and how to reduce the corresponding error at mission level (layer 4). That is, the mission pattern matching that maps multiple tasks to team’s operators can correct the errors in task identification obtained at task recognition layer. These results from this analysis are shown as “TaskAccuracy (layer 4)” in Figure 8.

As we reviewed the data for different task classes, we found that the activity and task recognition accuracy differed when we varied the true task that needed to be identified (and accordingly the test data set given to the model). This is due to the fact that different tasks have different behavior patterns, and some tasks had very similar model patterns to others in the set, causing the ambiguity and correspondingly identification accuracy of the true task to be low while identification accuracy of the true constituent activities was high (e.g., the task “Assess” was very similar to two other tasks – “Fix” and “Track”, since all three tasks contain primitive events of type “Send Communication” and “Read Communication”, and no other distinction could be made unless more detailed communication analysis could be performed or the human expert could provide the context). On the positive side, however, the activities of these tasks were similar and the accuracy of their identification was high). In these situations, there are two possible solutions for overcoming this difficulty in task distinction: (1) Intervention by an expert analyst to use MIMOD in the “active” manner – by specifying the context and correcting the model’s outputs; and (2) conducting the mission-level recognition, which maps all tasks in the mission to the team of operators and accounts for the temporal task execution and coordination constraints. The mission mapping that we implemented and tested in Phase I “corrected” the ambiguities of the task level recognition to increase the accuracy of the mission task identification as seen in Figure 8

(“TaskAccuracy (layer 4)”). These initial results indicated that improvement on average of over 30% in the accuracy of task identification can be achieved using mission-to-team mapping, especially in cases of ambiguous tasks when the task recognition accuracy would be low at task model layer 3.

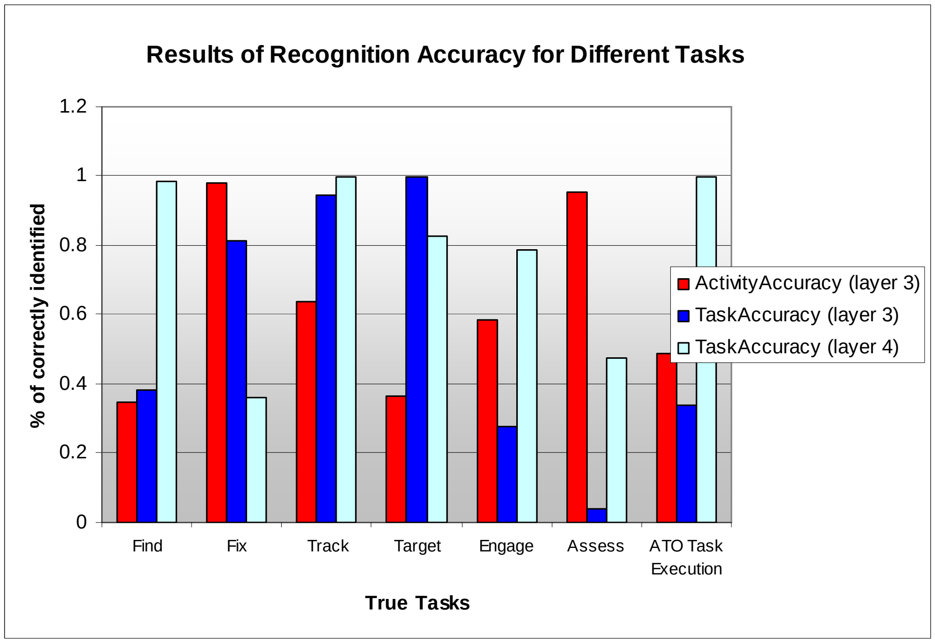


Figure 8: Accuracy of Task and Activity Recognition

Conclusion

Agile command and control (C2) organizations are increasing the demands for dynamic network resource management. Currently, most of the resource management systems require manual input of the behavior patterns. These systems need correctly identified activities and tasks of individuals to plan their employment in future tasks and provide information to them. MIMOD’s behavior pattern learning and tracking will support what is currently a manual and error-prone process. Enabled by MIMOD, the mission-driven network management systems will enable agile operations by supporting faster and more efficient retasking, better information flow, and improved collaboration in the context of team- and self-synchronization.

In this paper we presented a multi-level behavior recognition model based on the information hierarchy of data stream and behavior concepts. The model can learn and track individual and team behavior classes at different levels of granularity, feasibility of which we have demonstrated in a DTC use case with a data structure and quality similar to the operational setting such as real-world AOC and its constituent cells. Using the DTC dataset that contained the ground truth about true activities and tasks of the operators and the overall team, we were able to assess the accuracy of task and workflow recognition algorithms. In our computational sensitivity study we found that recognition accuracy for activities and tasks varied by task type.

Our preliminary findings are that activity and recognition accuracy was high enough to encourage further exploration and algorithm implementation for testing. In particular, we will undertake analysis of the relationship between the task type and the recognition accuracy to determine how to better exploit the areas of successful recognition found in this initial prototype work. Currently, we are working on implementing algorithms for learning the temporal and task content structures of team mission workflow from historic team behavior samples, which will provide the input hypotheses at the top level of behavior hierarchy. Learning team mission patterns is the final objective of the behavior learning and classification in large-scale organizations and will provide the top-down processing that can improve behavior classification at the lower levels of behavior hierarchy. Finally, we plan on developing algorithms to update and modify the behavior patterns iteratively with incoming event data.

References

Alberts, D.S., and Hayes, R.E. *Power to the Edge. Command, Control in the Information Age*. CCRP Publications Series, June 2003.

Bobick, A. and J. Davis, “The recognition of human movement using temporal templates,” IEEE Transaction on Pattern Analysis & Machine Intelligence, 23(3), March 2001.

Bobick, A. and Y. Ivanov, “Action Recognition Using Probabilistic Parsing”, Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, , Santa Barbara, CA, pp. 196-202, June 1998.

Grande, D., G. Levchuk, W. Stacy, and M. Kruger, “Identification of Adversarial Activities: Profiling Latent Uses of Facilities from Structural Data and Real-time Intelligence”, to appear in *Proceedings of the 13th International Command and Control Research and Technology Symposium*, 2008.

Hamid, R., Maddi, S., Johnson, A., Bobick, A., Essa, I. and C. Isbell. “Discovery and Characterization of Activities from Event-Streams”, Proceedings of Uncertainty in Artificial Intelligence (UAI 2005) August 2005.

Hoffman, T. “Unsupervised Learning by Probabilistic Latent Semantic Analysis”, *Machine Learning*, Vol. 42(1-2), pp. 177-196, 2001.

Intille, S. and A.F. Bobick, “Recognizing planned, multi-person action”, Computer Vision and Image Understanding 81, 414-445, 2001.

Ivanov, Y. and A. F. Bobick, “Recognition of Multi-agent Interaction in Video Surveillance”, Proc. of Intl Conference on Computer Vision, Corfu, Greece, pg. 169-176, September 1999.

Ivanov, Y., C. Stauffer, A. Bobick, W.E.L. Grimson “Video Surveillance of Interactions”, Second IEEE Workshop on Visual Surveillance, pp. 82-89, 1999.

Levchuk, G., and K. Chopra, “NetSTAR: Identification of Network Structure, Tasks, Activities, and Roles from Communications”, *Proceedings of the 10th International Command and Control Research and Technology Symposium*, McLean, VA, June, 2005.

Levchuk., G., Yu, F., Tu, H., Pattipati, K., Levchuk, Y. & Entin, E. “Identifying the Enemy – Part I: Automated Network Identification Model”, *Proceedings of the Command and Control Research and Technology Symposium*, Newport RI, 2007.

Levchuk, G., Yu, F., Meirina, C., Singh, S., Levchuk, Y., Pattipati, K., Willett, P., & Kelton, K. “Learning from the Enemy: Approaches to Identifying and Modeling the Hidden Enemy Organization”. In A. Kott (Ed.). *Information warfare and organizational decision-making.* Norwood MA: Artech House, 2007.

Levchuk, G., D. Lea, and K. Pattipati, “Recognition of Coordinated Adversarial Behaviors from Multi-Source Information”, *Proceedings of SPIE Defense and Security Symposium*, Volume 6943, Orlando, FL, 2008.

Levchuk, G., D. Grande, K. Pattipati, Y. Levchuk, A. Kott “Mission Plan Recognition: Developing Smart Automated Opposing Forces for Battlefield Simulations and Intelligence Analyses”, to appear in *Proceedings of the 13th International Command and Control Research and Technology Symposium*, 2008.

Shebilske, W., Gildea, K., Freeman, J., & Levchuk, G. “Training Experienced Teams for New Experiences”, Presented at the 51st annual meeting of the Human Factors, Society, Baltimore, MA, 2007.

Shebilske, W., Gildea, K., Freeman, J., Levchuk, G. Optimizing Instructional Strategies: A Benchmarked Experiential System for Training (BEST). Theoretical Issues in Ergonomic Science. Special Issue on Optimizing Virtual Training Systems, 2008

Shi, Y., Y. Huang, D. Minnen, A. Bobick, and I. Essa, “Propagation Networks for recognition of partially ordered sequential action”, Proceedings of IEEE Computer Vision and Pattern Recognition, Washington, DC, June, 2004.

Sutton, C., K. Rohanimanesh, and A. McCallum. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. In Proceedings of the Twenty-First International Conference on Machine Learning (ICML), 2004.

Yang, Z., and A. Bobick, Visual integration from multiple cameras, IEEE Workshop on applications for computer vision, Denver, January 2005.

Yedidia, J.S., W.T. Freeman, and Y.Weiss. Constructing free energy approximations and generalized belief propagation algorithms. Technical Report TR2004-040, Mitsubishi Electric Research Laboratories, 2004.