Framework for behavioral analytics in Generative Policy Management Systems

Maroun Touma*[a], Elisa Bertino[b], Brian Rivera[c], Dinesh Verma[a], Seraphin Calo[a]
[a]IBM TJ Watson Research Center, 1110 Kitchawan Road, Yorktown Heights, NY, USA 10598,
[b]Purdue University, West Lafayette, IN, USA 47907
[c]Army Research Laboratory, 2800 Powder Mill Road, Adelphi, MD, USA 20873

ABSTRACT

Behavioral Analytics (BA) relies on digital breadcrumbs to build user profiles and create clusters of entities that exhibit a large degree of similarity. The prevailing assumption is that an entity will assimilate the group behavior of the cluster it belongs to. Our understanding of BA and its application in different domains continues to evolve and is a direct result of the growing interest in Machine Learning research. When trying to detect security threats, we use BA techniques to identify anomalies, defined in this paper as deviation from the group behavior. Early research papers in this field reveal a high number of false positives where a security alert is triggered based on deviation from the cluster learned behavior but still within the norm of what the system defines as an acceptable behavior. Further, domain specific security policies tend to be narrow and inadequately represent what an entity can do. Hence, they: a) limit the amount of useful data during the learning phase; and, b) lead to violation of policy during the execution phase. In this paper, we propose a framework for future research on the role of policies and behavior security in a coalition setting with emphasis on anomaly detection and individual's deviation from group activities.

Keywords: policy, generative models, behavioral analytics, coalition operations

1. INTRODUCTION

Future coalition operations are likely to involve several autonomous devices, which can range from drones, automated mules, smart cognitive assistants and other assistive devices that have not yet been invented. In all likelihood, a single human would have the assistance of dozens of such devices to attain the goals of these missions. Efficiency in mission effectiveness means that each device operating in such a group be capable of a high degree of autonomy, while still staying in compliance with the higher level directives and policies provided by the human. In this context, policies are used as a tool to constraint the behavior of the asset and avoid extreme situations that could negatively impact the mission objectives. That leads to the interesting challenge of defining a framework which would allow a device to function relatively autonomously, but still be under the control of a human being.

In this paper, we introduce a framework by which the above goal can be obtained. In this framework, we assume that the devices that are under the control of a single human are used in an assistive model, i.e., they are used to make the human more efficient. In most cases, such assistance will be used by a human being to perform tasks that are repetitive and dull. For these category of tasks, we can use the concept of behavioral analytics to improve the autonomy available to a device. The framework within this paper provides an approach for such repetitive tasks, whether performed by one single device or several devices, and allows for a general way for a human being to maintain control. The framework conforms to a model for generative policies, in which devices are allowed to generate their own policies in order to meet a desired goal.

In particular, we focus on activities that are routinely performed by the device when assigned a given task and apply behavioral analytics [9,10,11] techniques to identify anomalies in the device behavior. We further bound our analysis to those anomalies that occur as a result the device interaction with other devices in a narrow operational context. Once an anomaly is detected, it is then correlated to a set of parameters in the device environment and encoded in the form of a generative policy that defines the future behavior of the device when the same set of parameters are encountered [12]. The newly generated policy defines a constraint based on the device's own parameters in order to reduce, or ideally eliminate potential conflicts in the device operating environment. It serves as a long term memory allowing each device, and by extension all devices within similar classification, to optimize its performance under similar conditions. As such, generative policies play a key role in building autonomous systems where the knowledge is extracted from a narrow operational context and encoded as a constraint that can apply to more general situations. Although they are not meant to

directly coordinate the activities of various systems involved in a mission, a similar result can be achieved by allowing each system to self-impose its own constraints based on its interactions with other systems and its environment.

We begin this paper with a brief recap of the generative policy architecture, and discuss the specific context under which we are applying that generative model of policies. We then provide a scenario which explains and motivates the need for the framework we are proposing. This is followed by a presentation of the actual framework.

2. GENERATIVE POLICY ARCHITECTURE

Policy based management is a proven technique for establishing and creating self-organizing autonomous systems [1] and has been applied in many domains including but not limited to industrial systems [2], computer network management [3], enterprise access control [4], distributed systems management [5], security and identity management [6], storage area networks [7] and military sensor networks [8]. A policy provides a constraint limiting the set of configurations that a system is able to take. To a large extent, existing policy management systems have followed an approach by which a human being provides high level policies to a policy based management system, which are then changed through a process of policy refinement to a set of lower level policies that can be distributed to the devices being managed. The devices that are managed implement a policy enforcement point (PEP), which can consult a policy decision point (PDP) to determine how a situation ought to be handled. This allows a significant reduction in the situations where human system administrators need to be involved, but the autonomy available to a device is still very limited. Devices are not able to determine their own policies, but are subject to the policies that are defined by the human administrator.

In order to permit more autonomy, recent policy based management work has explored the concept of generative policies. In the generative policy architecture, the policy refinement process is partially done within the device itself, resulting in the ability of the device to determine and generate its own policies. The difference between the standard policy management model and the generative policy management model is shown in Figure 1. The PRFD component shows the policy refinement done within the device while PRFM shows the policy refinement done within the management system.

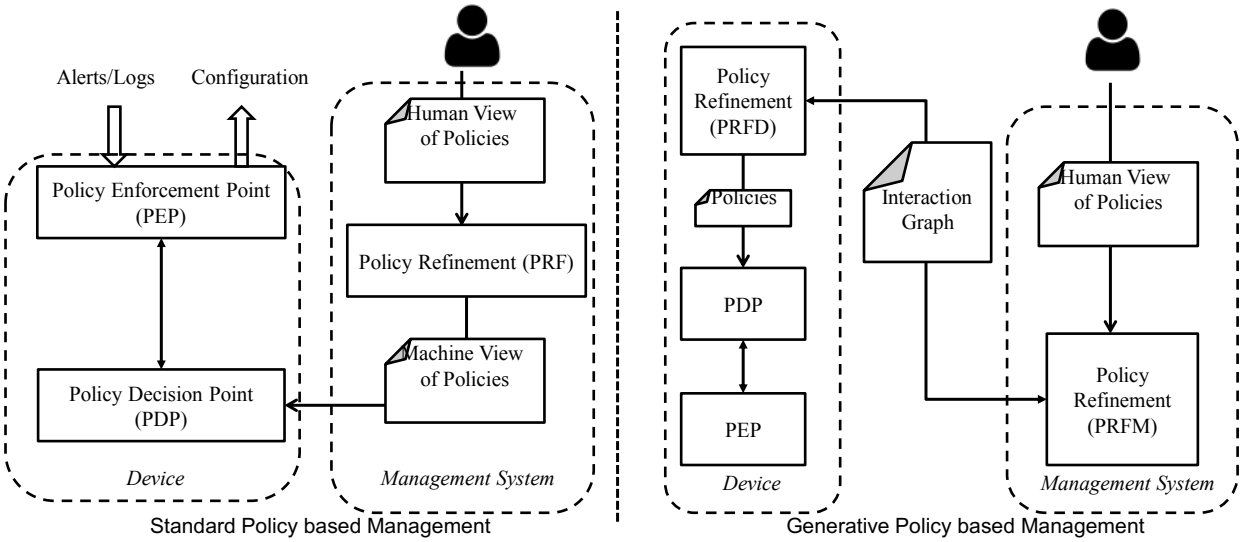

Figure 1. The Standard Policy based Management model compared to the Generative Policy Management model

The management system in the generative policy based management model provides an interaction graph to the devices. The interaction graph provides information to the managed device about the other type of devices that it may need to interact with, and how it ought to work with those other devices.

The generative policy model focuses on interaction graphs that capture information about device roles and how the relationships between the devices with the same roles can provide data to each other that can be used to derive device-specific policies. However, it does not address the question as to how the policies can be modified over time based on the

behavior of different devices. In this paper, we extend the model for generative policies so that it can benefit from past behavior and experiences of different devices in the same role. In this sense, the framework we propose here can be viewed as adding a specialization capability to the generative policy model. Specifically, we look at the situation where devices that have the same role, or perform repetitive actions are able to generate their policies to deal with anomalous situations.

While Generative Policies address a broad range of applications, in this paper, we limit our discussion to generative policies that are meant to bound system operations. Such operational control functions are themselves defined by the characteristics of the device, its assigned role, and the interaction graph. They are affected by the policies that are generated in accordance with the information received from the management system. It is these policies that we propose to modify based upon the past experiences of the device. The PRFD for each system encodes a predefined refinement algorithm that allows it to derive the relevant policies for the device. In our proposed approach, the refinement algorithm is triggered whenever an anomaly is detected and the resulting effect upon the operational control functions is then memorized in the form of a new policy that defines the system's future behavior when it again encounters similar conditions. As such, the policy remains available even when the specific stimulus to the generative algorithm is no longer available. In fact, the main objective of the policy's effect on the operational control functions is to prevent the anomaly from re-occurring. This allows the system to progressively improve such that the correct reaction to a specific interaction is calculated once and memorized in the form of a policy.

We further extend the generative policy model by associating a violation estimated cost (VEC) with each new policy. Policy violations often occur in those situations where two or more constraints must be met and the device may not able to meet all these constraints. The device may thus have to violate one or more such constraints. For example, one device attempting to maintain a constant speed and an altitude of 50 ft may be affected by external events that the device cannot control. Other neighboring devices may be taking actions that would prevent this device from maintaining its required altitude of 50 ft. The VEC represents the equivalent monetary loss that the device operator would incur if the constraint is violated. As such, it provides a quantitative measurement to the PEP in those situations when a constraint violation is likely to occur. The possible values for a VEC are defined in the policy management system based on levels of severity. The VEC for each policy is initially assigned by the PRFD process when the policy is first generated and subsequently revised by the system administrator.

3. MOTIVATING SCENARIO

In order to motivate the need for behavior analytics to generate policies, we look at a common scenario that is likely to arise in future coalition operations. We envision an environment where coalition forces are operating in an area where a set of bunkers in the forward operating base is supported by a supply depot. Robotic mules are used to ferry supplies, e.g., Meals Ready to Eat (MREs) from the depot to the soldiers in the bunkers. We can further imagine that the supplies in all the sites, including the bunkers and the depot are managed by robotic devices. A smart refrigerator keeps track of the MREs at the bunker that is needed for managing the inventory, and notifies the robot at the depot that it needs additional supplies. At the depot, a pool of autonomous vehicles (mules) is maintained. The robot asks the computer system managing the mules to assign one to the task of delivering the supplies. The robot would then load the mules with the appropriate supplies as needed. The mule then delivers the carton to the bunker. The scenario and the sequence of steps needed to do automated delivery and logistics management is shown in Figure 2.

Under normal operating conditions, the mules follow the same route from the depot to the bunker, and usually do not encounter any problems. However, in some instances, the mules may encounter some problems, e.g. they may find their path blocked due to a flooding of the street, or they may find that the depot may be running short of supplies of an item that is needed urgently at the forward operating base. In those situations, it would be highly desirable if the mules are able to deal with the anomalous situation in an appropriate manner.

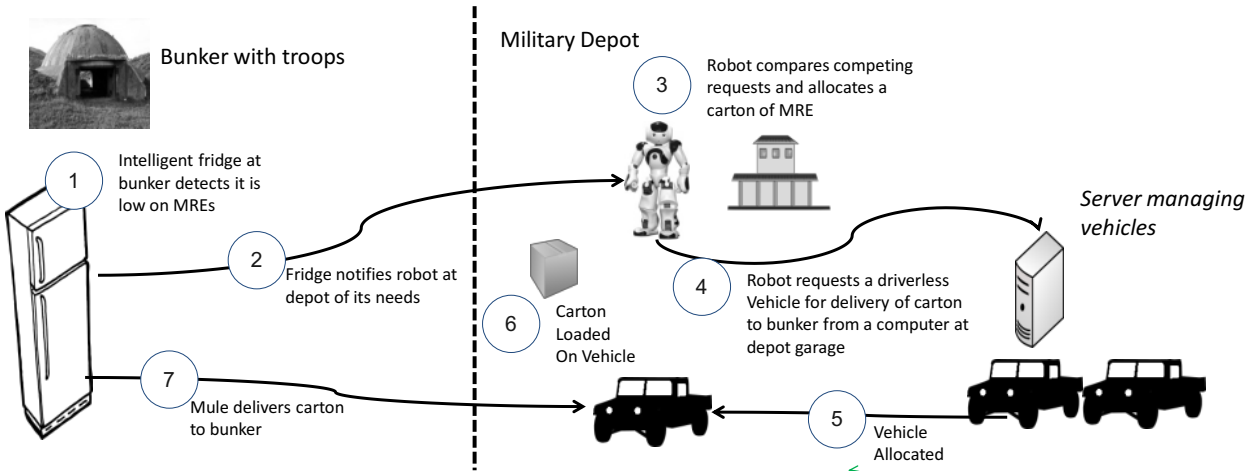

Figure 2. Motivating Scenario for Behavioral Analytics

In this scenario, the smart fridge with an assigned Role (F) is a fully autonomous system that maintains its own inventory and is aware of other assets with which it can interact. Such assets include a robot with an assigned role (R) responsible for replenishment, and a mule with an assigned role (M) that is responsible for delivering the goods to the bunker. Within the depot, the robot interacts with a fleet management system with a role (S) that is responsible for assigning the mule for the delivery. The robot also interacts with the mule for loading the mule with the required good. In turn, the mule interacts with the fridge for the final delivery.

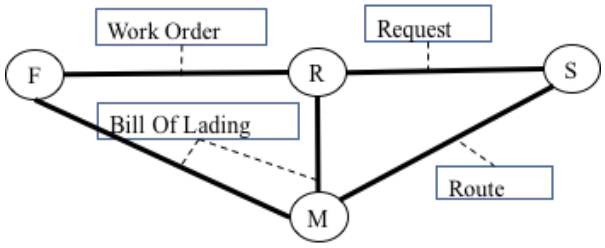

Figure 3. Interaction Graph

Figure 3 represents a typical interaction graph for this scenario where each of the axis will have a set of policies governing the interactions between the various roles: a) The interaction between roles F and R is governed by the work order that identifies the required items, quantity and time constraints generated by role F and associated with each item. b) The interaction between R and S is defined by the formal request for transportation vehicle and constraints generated by role R for the destination, the estimated payload weight and size, and the committed delivery date. c) The interaction between S and M is defined by the route that the assigned asset will follow to its destination and associated constraints generated by role S on specific path segments and observed speed. d) The interaction between R and M and the interaction between F and M is defined by the bill of lading and constraints generated by role R on packaging and special handling. Those constraints could be further extended by role M depending on specific circumstances such as re-enforced packaging that is appropriate for the route defined by S.

In considering the role of the mule assigned for delivery, an anomaly could occur as the mule follows the normal route for the delivery or if the bill of lading includes a missing item. For example, the mule may found an obstructed portion of the route and thus finds, with the help of maps it has on-board, a different route. While on the different route, the mule passes by another bunker and be asked by the fridge local to this bunker to unload two cartons of a specific supply. As fridges are in the interaction graph of the mule, the mule can then interact with this fridge (provided that some authentication steps are executed). The mule has then to decide whether it can unload the two requested cartoons. The mule has no specific policies about such “unforeseen requests”; it is however aware, from statistical data stored onboard, that an error rate of 5% in delivery is acceptable (in the sense that there could be 5% error rates in delivery with respect to what is requested).

In addition the mule is aware that the fridge requesting the cartoons is from a bunker of the same platoon. Therefore the mule will trust that the request is legitimate. Then it will check whether, after delivering the two cartoons, it will remain in the acceptable error rate. If not, the mule will check whether it may perhaps deliver only one cartoon. If not, the mule will not deliver the cartoons. More complicated reasoning will also be possible. Notice however that the mule will record all information about this circumstance, including the decision it has taken, and report the information to the military depot. The robot at the military depot will determine that this is a circumstance that has never arisen before and can thus create new generative policies for handling this type of situation. Such new policies will be distributed to all the mules, so that they will all be able to learn from the experience of the mule that experienced the anomaly.

Other types of "anomaly" that a mule could experience are requests of interactions by devices or even humans that are not in its interaction graph. Various actions are possible. The most conservative action is to limit the interaction to requesting information from the other device and then report this anomaly to the military depot that can determine whether to expand the interaction graph and define appropriate generative policies.

4. BEHAVIORAL ANALYTICS FOR GENERATIVE POLICIES

The generative model of policies enables a degree of autonomy that will not be possible otherwise. However, when just the concept of an interaction graph is used, the autonomy available in the system is limited. Figure 2 shows the typical roles that can be assigned to different devices. The four roles shown are that of the bunker manager, which has to keep track of local supplies, e.g. the refrigerator; the role of the depot manager who has to take care of the requests coming from various bunkers, the role of the vehicle fleet manager, and the role of the delivery vehicle. Each of the devices can discover other devices in different roles, and use the attributes known about those devices to generate policies related to security, resource control, privacy as other related attributes.

Using the generative policy model, the different devices that are in the various roles can discover each other, determine the attributes that are associated with each of the devices, and then generate their own policies for a variety of functions, including which information or service at the local device should become accessible to others, as well as figuring out when to contact other parties. Anticipated relationships between the policies can be specified within the limits and assumptions of the system. Policies generated in this manner are intended to deal with the normal affairs that are ongoing when interacting among different entities. However, they are not designed to cope with situations that are out of ordinary.

In order to apply behavioral analytics, one needs to understand the normal and usual expected behavior in any link on the interaction graph. Behavioral analytics can provide an approach to this. This allows a way to generate a tighter bound on the policies that can help to deal with the situation at hand.

The relationship between policies generated from the interaction graph and behavioral analytics can be illustrated in Figure 4. Assume that a system has two parameters that have some relationship. Without a policy based management system, the system has just some universally accepted physical limits on these two parameters. With policy based management, a human can put in bounds on what the two parameters ought to be. The interaction graph model of generative policies allows the bounds to be determined dynamically by the devices that are involved. In the figure, we are showing the policies as bounds that are independent of the relationship between the two parameters. Generative policies may be able to capture some relationships that may be easily specified. However, in many cases the relationship is complex, and can only be learnt online.

This can be further illustrated using the four scenarios in Figure 4: In the first scenario for Standard System, the two parameters are only bound by some upper limit that the laws of physics impose. In the second scenario, we introduce Policy-based Systems as a way for a human to define tighter limits. In the third scenario title Generative Policy System, we illustrate how each device may generate its own policy and impose its own limit based on its own situation and following a well define and prescriptive rule for policy generation. In the fourth scenario, we illustrate how Behavioral Analytics can provide a further refinement to scenario 3 and define a bounding rectangle with a range of acceptable values that were derived from a large number observations from a multitude of devices performing the same task under similar circumstances.

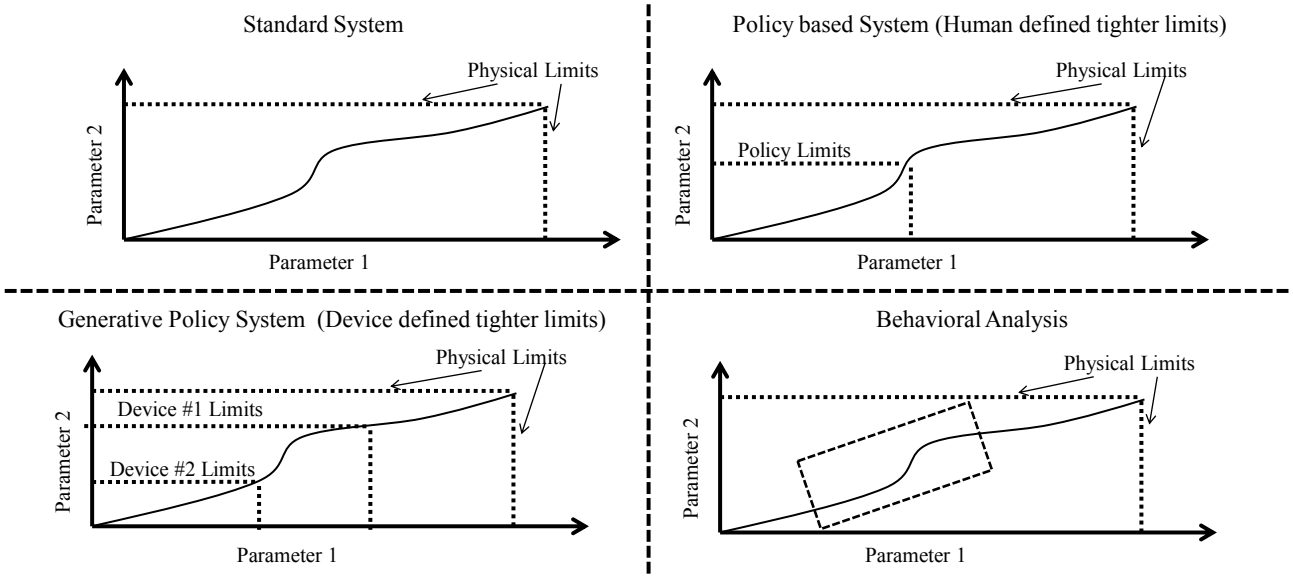

Figure 4. Relationship between different policy models explained using two parameters

The behavioral analytics model allows the system to get the relationships between the two parameters, identify when the relationship goes out of kilter, and put even tighter constraints among the different parameters. This can allow the determination of the relationship, and also adjust it over time. In this way, behavioral analytics allow a better adjustment of policies and augment the generative policy framework.

5. BEHAVORIAL ANALYTICS BASED POLICY FRAMEWORK

In order to create a behavioral analytics based system for generating policies, we propose a specific framework for policy refinement within the device which takes into account the information collected from different previous excursions and activities performed within the network. Starting with the policy grammar, each device follows the refinement process to generate its initial set of policies. However, overtime and as more observations become available, behavioral analytics is used to re-trigger the refinement process resulting in additional constraints added to the policy set. The framework is shown in Figure 5.

The generative policy framework using interaction graphs provides a mechanism that can create the information model that can be used to generate policies, i.e., the definition of the attributes and control functions that can go into the definition of conditions and actions within a policy relevant to the system. Using that information model, one can determine the attributes that need to be monitored for performing behavioral analytics. The source of information for the behavioral analytics comes from field observations and measurements as a result of the systems' interactions with each other. In our framework, we consider two types of system interactions: Peer-to-peer and Hierarchical.

Peer-to-peer (P2P) interactions occur when two independent systems need to cooperate, such as exchange information or goods to complete a joint task, with each system responsible for the full completion of its own tasks. The interaction between the mule and the robot in the scenario above is one example of a P2P interaction and can easily be modeled using an interaction graph. However, P2P interactions also occurs when two systems need to be aware of each other but must avoid each other in order to complete their respective tasks. A simple example for the latter case is when mule A needs to pass another moving object B and must adjust its speed according to the speed of object B. This type of interactions is often missing from the interaction graph but need to be accounted for when analyzing the behavior of system A in the absence of a specific linkage to system B. In a P2P interaction, systems are usually interchangeable: In an interaction between systems A and B, if system B is not available or fails, we assume that it can be replaced by system B' with similar capabilities and features. P2P interaction is also characterized by the time dependency between the two systems and the

presence of a well defined sequence of events that is a primary source of information for defining normal behavior and therefore detecting anomalies when they occur.

Hierarchical interaction occurs when a system A is fully dependent on another system or subsystem X to complete its own task. A mule for example must rely on a navigation system in order to perform its mission. This type of interactions is often missing in the interaction diagram as it is assumed to exist for system A to perform its function. In this hierarchical model, it is impractical to substitute a system for another one without severely impeding the course of the mission as those hierarchical interactions tend to present rigid links and complex inter-dependencies between the two systems. Detecting anomalous behavior in a hierarchical interaction presents some specific challenges related to the proper identification of the source of the anomaly and therefore the specific parameter for imposing the constraints. For example, when a mule deviates from its normal route, it could be due to a loss in its GPS satellite link, other system malfunctions or simply due to changes in the terrain.

These two types of interactions define the basis for the behavior history that is collected every time the mission tasks specific to the domain of policy are being performed by the device. The behavior history consists of all the attributes and their values that were observed during different times when the required mission task was being performed. The analysis of the behaviors recorded in this manner can then be analyzed to determine a normal and prevailing envelope for the values different attributes can take in those environments.

Once the envelope is developed, the limits of that envelope can be combined with the other constraints that may be available from the interaction graph available from management system input. This can identify additional limits or conditions under which alerts may need to be made. Thus, policy is determined as a combination of both learnt behavior as well as behavior that needs to be within a given limit.

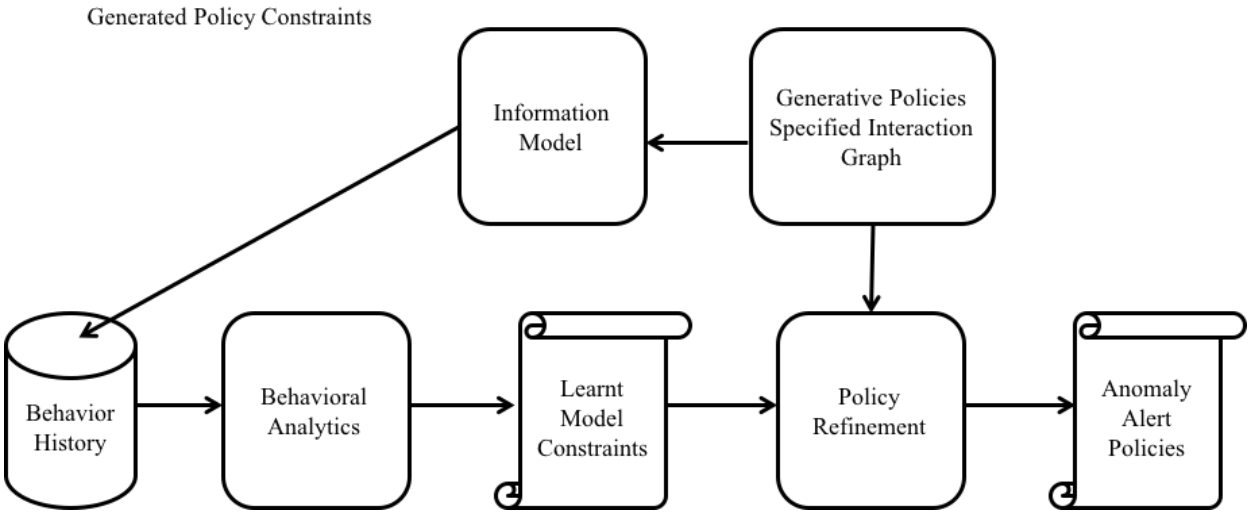

Figure 5. Behavioral Analytics based Policy Refinement Framework

The creation of the limits on policy using the behavioral analytics model can be understood further by the illustration in Figure 6. For simplicity, we show the example using only two attributes in the behavioral model. However, the same framework can be easily extended to more than two attributes.

For each of these two attributes, a user defined limit can be specified by a human operator as part of the refinement limit. That part is shown in Figure 6(a). Since the relationship is complex, the operator cannot be expected to understand the relationships between the two, and so such constraints look like horizontal rectangles on the graph. The relationship between parameters is as shown in figure 6(b). The behavioral analytics, performed by the device, can be used to estimate a boundary envelope between the two parameters. That aspect is shown in Figure 6(c). Finally, the result of the composition of both of these aspects is the result shown in Figure 6(d). The resulting policy space that is allowed is shown as shaded regions in 6(d). The resulting value has policies that combine human insights along with the result of machine learning, thus resulting in overall combination of both those aspects.

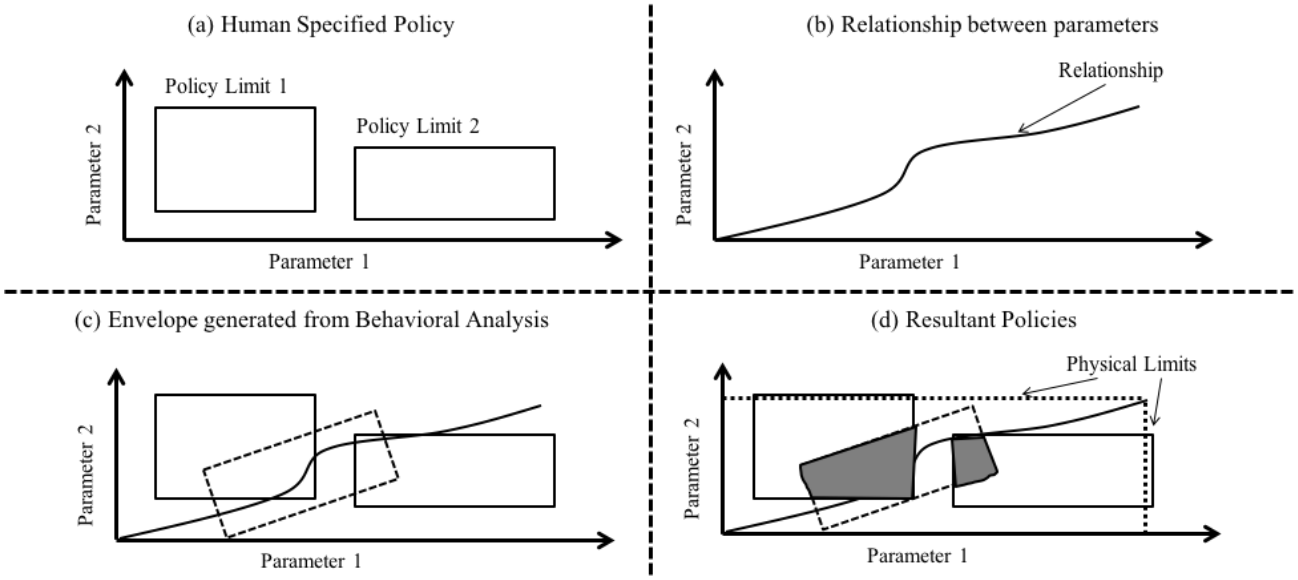

Figure 6. Behavioral Analytics and Policy Generation

The framework can be applied within various application use-cases where repetitive functions are performed, While the work is still in early stages, our ongoing activities will explore it further and apply it to various different use-cases.

6. INFORMATION MODEL AND POLICY EXPRESSION FOR AUTONOMOUS SYSTEMS

The information model supporting the analytics defines a system in terms of its features vector. Features vectors identify all the attributes that are known about the system. Some of the attributes may have fixed values (e.g., Serial Number) but the majority of the features that define a system will have an assigned value that changes over time (e.g, speed). Typically, each attribute will have an upper limit and a lower limit that define the physical boundary of the system and cannot be violated.

A system is also defined in terms of its Parametrized Constraints (PC) that should not be violated. Parametrized Constraints define the desirable optimal value, or range of values, of each of the attributes as needed. They are intended to preserve the long running/operation of the system at the lowest possible cost. In this initial work, we express constraints as they relate to four dimensions: time, location, actor and associated action. The actor often expresses a role or specific skill required to perform the action. PC also has an associated cost if and when the constraint is violated. Cost is always defined in term of equipment material loss or equivalent monetary value.

For example, the following parametrized constraint allows speed up to 140 MPH but only when driven by a professional driver on a closed highway. It also defines the cost of violation as being a partial or total loss of the vehicle.

PC (system=system id, speed<=140MPH, time=any, location=closed highway,

actor=professional driver, action= drive, Cost=[partial loss ..total loss])

Parametrized constraints fall into one of two categories: Predefined and Generative. Predefined Parametrized Constraints (PPC) are defined by entities that are external to the system such as the system owner/operator or a government body with jurisdiction over the operations of the system. For example, a predefined constraint that requires a drone to be flown at an altitude no higher than 500 feet is expressed as:

PPC (system=Drone X11, altitude<=500ft, time=any, location=any,

actor=licensed pilot, action= operate, cost= [total loss + Penalty])

Generative Constraints (GC) are used throughout the mission to adjust specific limits on the system operation even when the actual reason for imposing the constraint is not available beforehand, and therefore the input for the generative process is not available. For example: During the preparation phase, the system connects to the operator's database and identifies the specific team members who are also licensed pilots and generate the corresponding refined policies with the specific pilots' names. During the operation phase, the system does not have access to the database but already has the specific policy to match the licensed pilot when assigned. As such, the specific instantiation of a GC depends on the specific context and available data sources during the generation phase. For illustration, the GC expressed below will generate a different policy depending on the data source used in a given use case:

GC (system=s, feature=any, time=any, location=any,

actor=licensed pilot, action= operate, cost= [total loss + Penalty])

Use Case 1: GC (system=s, feature=any, location=any, time=any,

actor="Duke Cunningham", action= operate, cost= [total loss + Penalty])

Use Case 2: GC (system=s, feature=any, location=any, time=any,

actor="Steve Ritchie", action= operate, cost= [total loss + Penalty])

The information model also includes the definition of Observable Restrictions (OR) that are learnt. They are generated from observations, using deductive reasoning or other refinement techniques. OR control how systems interact with each other. They define the upper or lower bounds of each of the features as it influences or is being influenced by an external system or peer. ORs also identify the external agent (i.e. peer) involved in the interaction. Like other constraints, they generally have an associated cost for violation. The example below describes an OR that requires a drone to maintain a distance of 1 mile from any other drone:

OR (system id=Drone X11 , distance > +1 mile, time=any, location=any,

actor=any, action=any, cost=poor performance,

agent=<class: drone>}

7. CONCLUSION

In this paper, we outline a framework for future research exploring the use of behavioral analytics for policy generation in autonomous systems. We use systems-to-system interactions to define the boundaries for the operational context of the analysis. By comparing repetitive tasks that occur when two autonomous systems interact, we can detect anomalous behavior and prevent it from re-occurring by introducing new constraints in the form of policies. The next step of our work will define a formal model for the analysis that is aligned with the framework for generative policy and use field experimentation for training the model on a subset of coalition scenarios.

8. ACKNOWLEDGEMENTS

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copy-right notation hereon.

REFERENCES

[1] D. Agrawal et al., “Policy technologies for self-managing systems,” Pearson Education, (2008).

[2] S. Illner, A. Pohl, H. Krumm, I. Luck, D. Manka, and F. Stewing, "Policy-based self-management of industrial service systems," Proc. of 4th IEEE International Conference on Industrial Informatics, 492-497 (2006).

[3] D. Verma, "Simplifying network administration using policy-based management." IEEE network vol 16 no 2, 20-26 (2002).

[4] R. Bhatti, A. Ghafoor, E. Bertino and J. Joshi, "X-GTRBAC: an XML-based policy specification framework and architecture for enterprise-wide access control," ACM Transactions on Information and System Security (TISSEC) vol 8 no. 2, 187-227 (2005).

[5] M. Masullo and S. Calo, "Policy management: An architecture and approach," Proc. of the IEEE First International Workshop on Systems Management, 13-26 (1993).

[6] A. Ganek, A. Nadalin, N. Nagaratnam, and D. Verma, "An autonomic approach for managing security and identity management policies in enterprises," Journal of High Speed Networks vol 15 no 3, 291-300 (2006).

[7] D. Agrawal, J. Giles, K. Lee, K. Voruganti, and K. Filali-Adib, "Policy-based validation of SAN configuration," Proc. of Fifth IEEE International Workshop on Policies, 77-86 (2004).

[8] D. Verma, G. Cirincione, and T. Pham, "Policy enabled interconnection of sensor networks using a message queue infrastructure." Proc. of SPIE Defense and Security Symposium, (2008).

[9] A. Sallam, E. Bertino, S. R. Hussain, D. Landers, R. M. Lefler, D. Steiner, "DBSAFE—An Anomaly Detection System to Protect Databases From Exfiltration Attempts," in *IEEE Systems Journal* vol PP no 99, 1-11 (2015).

[10] V. Chandola, A. Banerjee,V. Kumar, “Anomaly detection: A survey” ACM Computing Surveys (CSUR) vol 41 no3, 1-58 (2009).

[11] E. Aharoni, R. Peleg, S. Regev and T. Salman, "Identifying malicious activities from system execution traces," in IBM Journal of Research and Development vol 60 no 4, 5:1-5:7 (2016).

[12] M. Better, F. Glover and M. Laguna, "Advances in analytics: Integrating dynamic data mining with simulation optimization," in IBM Journal of Research and Development vol 51 no 3.4, 477-487 (2007).