Creating a Digital Twin of an Insider Threat Detection Enterprise Using Model-Based Systems Engineering

James Lee
Dept. of System Engineering and Operations Research
George Mason University
Fairfax, USA
jlee194@gmu.edu

Ahmad Alghamdi
Dept. of System Engineering and Operations Research
George Mason University
Fairfax, USA
aalgham8@gmu.edu

Abbas K. Zaidi
Dept. of System Engineering and Operations Research
George Mason University
Fairfax, USA
szaidi@gmu.edu

***Abstract*—Inference Enterprise Modeling (IEM) is a methodology developed to address test and evaluation limitations that insider threat detection enterprises face due to a lack of ground truth and/or missing data. IEM uses a collection of statistical, data processing, analysis, and machine learning techniques to estimate and forecast the performance of these enterprises. As part of developing the IEM method, models satisfying various detection system evaluation requirements were created. In this work, we extend IEM as a digital twin generation technique by representing modeled processes as UML Activity Diagrams and tracing solution processes to problem requirements using ontologies. Using the proposed framework, we can rapidly prototype a digital twin of a detection system that can also be imported and executed in systems engineering simulation software tools such as No Magic Cameo Enterprise Architecture Simulation Toolkit. Cyber security and threat detection is a continuous process that requires regular maintenance and testing throughout its lifecycle, but there often exists access issues for sensitive and private data and proprietary detection model details to perform adequate test and evaluation activities in the live production environment. To solve this issue, organizations can use a digital twin technique to create a real-time virtual counterpart of the physical system. We describe a method for creating digital twins of live and/or hypothetical insider threat detection enterprises for the purpose of performing test and evaluation activities on continuous monitoring systems that are sensitive to disruptions. In this work, we use UML Activity Diagrams to leverage the integrated simulation capabilities of Model-Based Systems Engineering (MBSE).**

Keywords—Digital Twin, MBSE, Insider Threat, System Engineering

I. INTRODUCTION

With the increasing concern about threats from adversaries trying to exploit both human and machine related vulnerabilities, organizations are actively seeking to implement methods to help mitigate this risk. Cyber security and threat detection is a continuous process that requires regular maintenance and testing throughout its lifecycle. However, data availability issues related to privacy, lack of ground truth, and limited visibility into the parameters used by the commercial off-the-shelf (COTS) products pose a challenge in evaluating these detection systems sensitive to disruption.

An insider threat is a malicious threat that comes from actors within an organization. It includes incidents such as a former employee of a vendor stealing customer data and a former employee stealing intellectual property. While these incidents are infrequent and hard to detect, a realized threat has significant security and financial consequences. To mitigate this risk, the government has mandated that all contractors must have an insider threat program in place, and there has been a growing interest in developing insider threat detection systems. However, due to lack of ground truth and limited understanding of detection accuracy and performance, there have been an overwhelming number of false alarms and challenges with reducing the population to manageable subpopulations for more detailed analysis. To address these challenges, a group of researchers developed a best-in-class Inference Enterprise Modeling (IEM) methodology, a modeling and simulation methodology to better evaluate the performance of insider threat detection enterprises [1], [2].

The technology of Digital Twins opens a new level of efficiency in system development. But the creation of virtual systems becomes more challenging as an exact digital representation is needed [3]. The work in [4] and [5] has described Digital Twin as a dynamic digital representation of a physical system and recommends integrating digital twin technology within the Model-Based System Engineering (MBSE) methodology. Moreover, MBSE can be used in conjunction with digital twin to improve operational models, predict and analyze faults and perform system-wide analysis.

In this work, we re-introduce IEM as a technique to create digital twins for insider threat detection enterprises. There is a breadth of IEM knowledge in academic papers, technical reports, and repositories for future IEM researchers to refer to. However, like any software development project, searching, selecting candidates, and confirming whether the potential solution is a feasible option is often a time-consuming activity. We develop a framework to document, formalize, and scale IEM modeling knowledge by employing systems engineering techniques.

Systems engineering is an interdisciplinary field of engineering and engineering management that focuses on designing, integrating, and managing complex systems over their life cycles [6]. As the engineering community transitions

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

from document-based to model-based design, there is an increasing need for applications capable of supporting the digitization of the design and modeling process. This transition can be made possible by adopting Model-Based Systems Engineering (MBSE), which is a formalized application of modeling to analyze and document key aspects of the systems engineering life cycle [7], [8]. MBSE provides a way to coordinate system design activities and satisfy stakeholder requirements through improved communications, rigorous requirements traceability, and continuous requirements validation and design verification [9]. There are many available MBSE tools in the market, providing system engineers with resources, repositories, and tools to support the collaboration between teams and the required analysis for their system [10]. These modeling tools enable the development of digital engineering models to integrate the modeling and analysis into a single environment.

The paper is organized as follows. In section 2, we review the Inference Enterprise Modeling methodology and discuss the research opportunities. In section 3, we review the relevant technologies used in the solution method. In section 4, we describe the solution method and demonstrate it using a small example. In section 5, we conclude and outline the future direction for this work.

II. Inference Enterprise Modeling

An Inference Enterprise (IE) is an organizational entity that uses data, tools, and people to make mission-focused inferences. In this work, we focus on enterprises that seek to detect and identify a small proportion of the population that are potential insider threats. An IE makes predictions about events which lack ground truth, and without the knowledge of the ground truth, it is challenging to evaluate the accuracies of these predictions and adjust the system.

IEM uses available data and information about the organization to design and develop models that will be used to forecast the performance of a system. Through IEM, modelers can gain an understanding of how various factors impact the performance and analyze these results to suggest changes to, or tune, the model before deploying it in a live operational environment to make predictions about future events. The typical activities involved in the IEM process are shown in Fig. 1.

The IEM process starts with a problem statement that outlines the purpose of the modeling task. For example, we may be asked to predict the future performance of an insider threat detection enterprise given historical data and information about the classifier it is using. Along with the problem objective, historical data and information about the classifier or anomaly detector is provided to the modelers. The historical data can be provided in the form of complete user-level records or aggregated data. Aggregated (or redacted) data is usually delivered in the form of a set of data summaries such as histograms and correlations to protect personal identifiable information. Modelers use the provided data to create models that generate simulated populations, including simulated target labels. Statistical variations are applied to simulated populations to account for uncertainty, especially for future or unknown scenarios. Different simulation methods are used depending on the characteristics of the data. This describes the Population Modeling phase which results in many versions of the historical and potential future CULRs the summary statistics were based on.

In the second phase, the modeler designs and develops algorithms that make predictions of a target behavior. We call these Down-Select (DS) algorithms because they seek to identify a subset of the target population. Examples include classification algorithms such as Support Vector Machine and

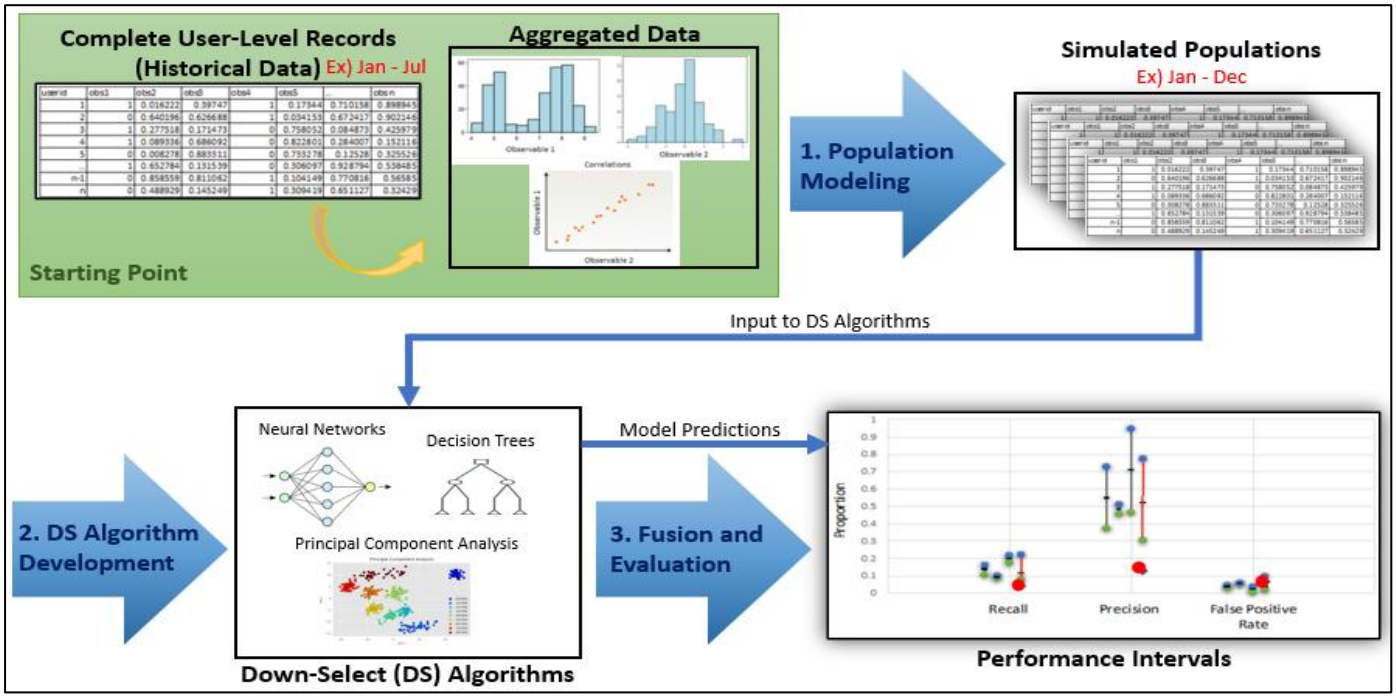

Figure 1. Inference Enterprise Modeling

Neural Networks. The specific algorithms are often given as part of the problem specification, but sometimes the IEM team is asked to identify and evaluate suitable algorithms to be considered for a future IE implementation. The problem description specifies the data to be used to train the algorithms make predictions on. Since the premise of this problem is that the user behavior and ground truth of the future are unknown, the predictions are compared to the simulated labels created in the population modeling phase to evaluate performance metrics such as precision, recall, and false-positive rates. Since there are multiple populations to represent the possibilities of the unknown, there will also be multiple predictions and performance metrics to match the number of populations generated.

In the final Fusion and Evaluation step, modelers fuse the predictions from different population models to calculate central tendencies for the performance metrics which represent the uncertainty in the estimates. These estimations are based on the stochastic simulation described in the population modeling step. In the model evaluation phase, the estimates are reviewed by the modelers and may reassign the weights used to fuse the results together for each model. Modelers may also go back and adjust the assumptions and parameters defined in the software assets for Population Modeling and Down-Select algorithms. After model validation, the modelers can identify ways to improve future performance and apply those changes to the live operational environment.

A. *IEM Knowledge Management Opportunities*

As part of developing the best-in-class IEM method, a collection of quality IE models were built, and the modelers gained expert knowledge in modeling insider threat detection systems. Currently, this expert knowledge can be accessed through academic papers, technical reports, and repositories, which makes creating a new IEM solution for new researchers a heavily manual process where either a new solution is created from scratch or potential previous solutions are searched for and modified for the new problem. The process of searching, selecting candidates, and evaluating whether it is a feasible solution is often a time-consuming activity. There is an opportunity to manage IEM knowledge using systems engineering methods for better knowledge scaling. Formalization of this knowledge in a computable representation enables partial automation and rapid prototyping, which can shorten the time to prototype new potential solutions allowing for more time to refine the models.

III. RELEVANT METHODS

As mentioned in the previous chapter, there are opportunities for utilizing knowledge management techniques to better preserve and scale IEM knowledge. With the opportunities in mind, the high-level objectives are to facilitate preservation and sharing of IEM knowledge, enable the ability to combine modules from previous solutions to solve new problems, and define the relationship between problem requirements (or characteristics) to specific solutions so that they can be queried. More specifically, my research objectives are to develop a comprehensive knowledge base for the IEM method to provide a reuse formalization for discovery of new potential solutions which are also executable through systems modeling simulation software. To meet these objectives, I would need the following capabilities:

- Knowledge Representation: to document processes, skills, artifacts used in previous projects
- Instance storage: store new knowledge
- Reasoning capabilities: make recommendations based on new requirements

This makes the use of ontologies and process modeling languages as an ideal solution candidate for this work.

A. *Ontologies*

Ontologies are a formal description of knowledge as a set of concepts within a domain and the relationships between those concepts. The benefits of ontologies are that they provide a coherent and easy navigation between one concept to another. As formal and explicit specifications of representing knowledge about a domain, ontologies are helpful in making information more shareable across people and computer models to support knowledge-based reasoning. The Web Ontology Language (OWL) provides a formal representation of an ontology that describes classes, properties, and individuals in semantic web documents or datasets [11], [12]. The OWL semantic web technology has been used to develop ontologies for a variety of disciplines, including biology, medicine, enterprise architecture, and web services, to better manage knowledge and information within these domains [13]–[15]. In this work, we use Protégé as an ontology development tool.

Ontologies are enablers of good modeling in that it focuses on establishing well defined domain concepts in terms of the terminology, definitions, and relationships to model real world applications [16]. For these reasons, ontologies have been used to model system specifications, enterprise architectures [17], and groups such as the Object Management Group (OMG) have been leading initiatives to leverage ontology practices to improve MBSE. Ontologies representing requirements and software assets [18], [19].

B. *Process Modeling*

Process modeling is the graphical representation of business processes, which are a collection of tasks an organization performs to create products, reach specific goals, and provide value to stakeholders. These tasks may include manual and automated activities, such as developing or automating execution of software assets. Traditional process modeling methods focused on increasing production efficiency and quality of goods in the industrial sector, but there is an increasing need for process modeling and improving business processes in the rapidly advancing information technology and software-intensive system domain [20].

There are several process modeling languages such as Business Process Modeling Notation (BPMN), Event-Driven Process Chain (EPC), and Petri nets. In this work, I use UML (Unified Modeling Language) activity diagrams, which are particularly good at modeling processes [21], that are not only equally suitable for human and machine processing but also widely used in the software development community [22]. Activity diagrams help designers express the behaviors and

This work has been funded by

event occurrence over time and gain the advantage of readability over other behavioral diagrams [23]. For that reason, modelers use activity diagrams as an analysis tool to capture the system behaviors and their actors. In this work, we use No Magic Cameo Enterprise Architecture (Cameo EA) to model and execute the processes using Cameo Simulation Toolkit. Simulation Toolkit allows designers to execute designed systems to endorse the validation and verification. It also supports built-in scripting engines to customize different actions and allows designers to better understand the system without manipulating the actual system [9].

IV. SOLUTION METHODOLOGY

The solution is an extension of the work described in [24], [25], in which a solution is instantiated using a pre-defined template based on the output from querying the ontology knowledge base. In this approach, the IEM Process Ontology represents the expert knowledge of the IEM methodology described in section 2. We used UML Activity Diagrams as the representation for the Process Template. Since there was no official Activity Diagram ontology available for reuse, we reverse-engineered a version of an Activity Diagram ontology by inspecting an activity diagram XML and identifying the necessary XML. We also designed a Model Generator that produces an Activity Diagram of a suggest solution workflow that can be executed in simulation tools such as Cameo.

A. IEM Process Ontology

The proposed knowledge management framework, which utilizes systems engineering tools, process modeling, and ontologies, follows the three-step process of:

1. using formal process modeling to document the generalized solution workflows,
2. generalizing solution methods by creating a process template and categorizing each part of the solution as a section of the template, and
3. defining the connection between problem requirements to each solution method.

This enables instantiation of new solutions of the generalized workflow that covers the new problem instances and use of simulation plug-ins to execute the solution instance.

The IEM Process Ontology is comprised of IEM Problem Requirements, IEM Solutions, and IEM Process Template. The problem requirements component formalizes the characteristics that define each problem, and the solution component documents the software assets and manual activities that fulfill the requirements. The process template is used to organize the solution modules into one of the steps of IEM: A. Population Modeling, B. Down-Select, and C. Fusion and Evaluate. The relationship between the ontology components are described in Fig. 2.

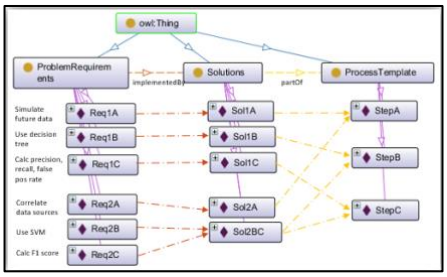

Figure 2. IEM Process Ontology Architecture

The structure creates a connection from the IEM problem requirements to a sequence of solution modules which represents the process that satisfies the requirements. As a whole, the IEM Process Ontology serves as an IEM knowledge base with links to the IEM assets repository. We can see how certain problem requirements are *implementedBy* specific solutions, and how each solution is *partOf* a step of the process template.

Considering a new problem with requirements to correlate data from different data sources, use decision trees to make predictions, and ask for precision, recall, and false-positive rate. We can see connect each requirement to a corresponding solution, and as a result, end up with a suggested solution process template that can use Sol2A, Sol1B, Sol1C in order.

1) Process Modeling

For this step, we started documenting previous solutions as Activity Diagrams. An example of a solution workflow is shown in Fig. 3.

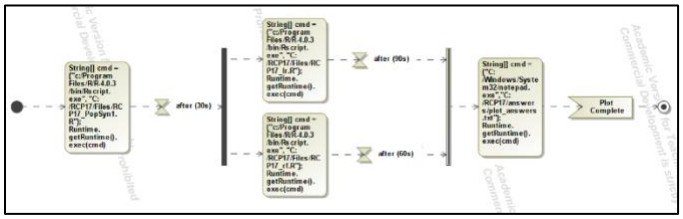

Figure 3. Solution Workflow Example

After creating activity diagrams for the previous solution workflows, we identified the minimum set of common UML Activity Diagram elements, including the Initial Node, Activity Final Node, Fork and Join Nodes, Call Behavior Action, Opaque Action, Accept Event Action, and Control Flows. This list of activity diagram elements is later translated to classes in the process template ontology, which is described in the next subsection.

The Call Behavior Actions represent manual tasks that modelers performed where Opaque Actions correspond to software tasks. Each of these action types will contain the path from which the documentation related to the task can be found. Each software script takes a certain amount of time to complete execution. The scripts are executed through Opaque Actions, and each Opaque Action is followed by an Accept Event Action that is triggered by a time event that indicates the time it takes to execute a script. Since the execution time for manual tasks can vary from modeler to modeler, the Accept Event Action is triggered by a signal representing task completion. We

specified the location of the script and software executable in the Body and Language field of the specifications. Using BeanShell as the language, we input the following:

```
String[] cmd = {"path to software executable", "path to file"};
Runtime.getRuntime().exec(cmd)
```

This BeanShell script executes the specified file, using the specified executable in command line. The different software options we use in this work include Python, R, Excel, and Notepad.

2) Process Tamplate

From the documented solutions in the form of Activity Diagrams, we identified the distinct steps of the IEM process and created the process template as in Fig. 4, where each of the fork/join pairs represents each step. This structure ensures that the model is syntactically correct even when there is no instance for a specific step of the process.

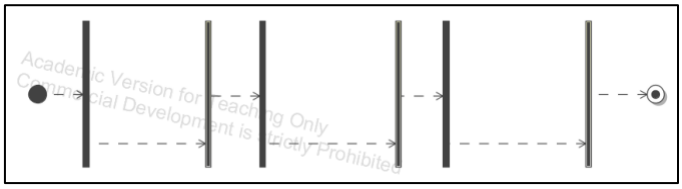
Academic Version for Teaching Only
Commercial Development is Strictly Prohibited

Figure 4. Process Template

The resulting class structure of process template ontology is shown in Fig. 5(a). Each of the subclasses for process template corresponds to the common activity diagram elements that were identified in the previous section. For the Workflow class, we have 3 subclasses for each of the IEM Process Steps outlined in the previous section. The object properties shown in Fig. 5(b) are used to describe the binary relationship between two instances. *hasElement* is used to indicate which UML Activity Diagram elements a specific part of the workflow has, *hasSource* and *hasTarget* define the source and target nodes (or elements) for each Control Flow.

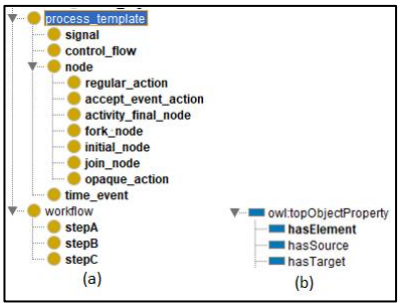

Figure 5. Process Template Ontology Properties

3) Problem and Solution Modeling

a) IEM Problem Requirements Ontology

Modeling the connection between problem requirements (or characteristics) to each solution module started with identifying all the possible problem requirements of the previous challenge problems that were solved by the modeling teams. Characteristics of an IEM problem can be broken down to categories such as the type of data provided, type of down-select algorithm to be used, and types of performance evaluation questions to be answered. Each of these categories would have a unique choice that would be specific to a problem. By creating an object property for each of the categories and instances for each choice within the categories, we defined problem requirements for each problem. An example of an IEM problem (instance "IRCP3") described using IEM problem requirements instances is shown in Fig.6.

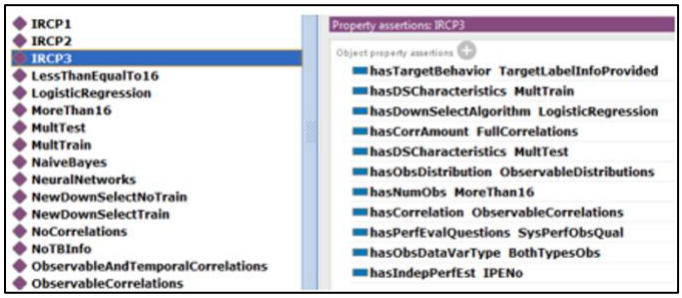

Figure 6. IEM Problem Requirements Example

b) IEM Solution Ontology

Once the problem requirements were defined, we cataloged the solution modules, both software, and manual, including information such as the time it takes to execute the software as well as the file path of the artifacts. By importing the IEM Process Template ontology, we can re-use the two classes *opaque_action* and *accept_event_action* to create instances and the imported object properties for assertions. By importing the IEM Problem Requirements ontology, we can connect the requirements and solutions using an object property *implementedBy*. In addition, the following properties presented in Fig. 7 were defined to specify the time it takes to execute a software module (*hasDuration*), which Opaque Action the Accept Event Action represents (*representsDurationFor*), how the Accept Event Action is triggered (*triggers*), and the command that executes the software script (*hasBody*) and the scripting language of which the command was written in Cameo (*hasLanguage*).

Figure 7. IEM Solution Ontology Properties

Finally, we define the rules that connect a set of problem requirements to modules that are part of the solution process. Considering a module "WF_rcp17_downselect" that satisfies a requirement to use linear regression as a down-select algorithm, the rule can be modeled in Protégé as seen in Fig. 8.

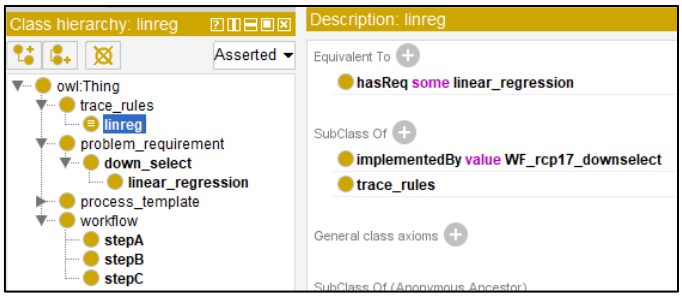

Figure 8. Requirement Implementation Rules

Each rule is created as a class where the *Equivalent To* field is used as the IF statement and *SubClass Of* is used as the THEN statement. The rule "linreg" can be interpreted as "the requirement for linear regression being implemented by the WF_rcp17_downselect module."

B. Ontology Instantiation and Demonstration

Given there is a problem that requires running a linear regression classifier for its prediction model, this is a small example solution workflow that contains a single task where a linear regression script is executed. Since this example only contains a down-select step (step B), the ontology instances that need to be added will look like Fig. 9. The fork and join nodes, in this case are the same fork and join nodes for step B in the process template.

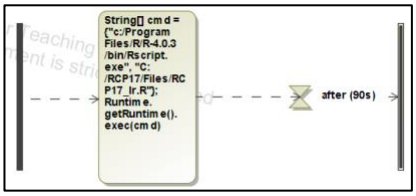

Figure 9. Small Example for Demonstration

First, all the necessary activity diagram elements are created as instances in the process template ontology depicted in Fig. 10(a). Next, the instances that correspond to the activity diagram elements are linked to part of the workflow using object property assertion *hasElement* as shown in Fig. 10(b). Since the process template already contain the forks and joins, they are not added in the step above. Each control flow is given a source and target node as in Fig. 10(c), and the scripting language that's calling the script from within Cameo EA and the location/path of the script to execute are defined as data properties of the instance that represents the Opaque Action (Fig. 10(e)). Finally, we define the time event that triggers the Accept Event Action that indicates the duration of the script that Opaque Action represents (Fig. 10(d)).

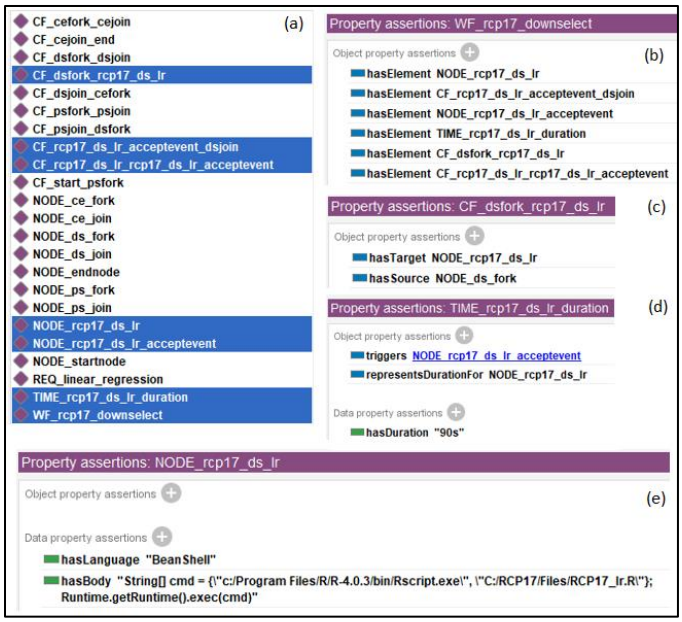

Figure 10. Small Example Ontology Implementation

Once the ontology is set up, we can run a query for the activity diagram elements that satisfy the linear regression requirement as seen in Fig. 11.

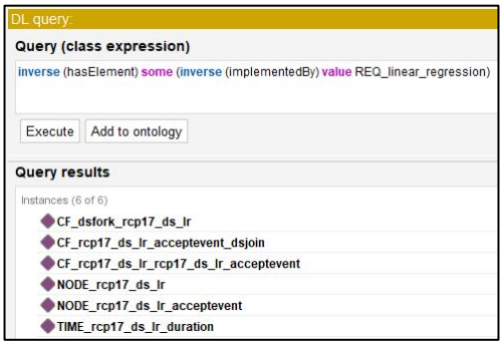

Figure 11. Small Example Ontology Query

Taking the activity diagram elements from the query, we use the activity diagram model generator that extracts the information in the object and data properties from the owl file and generates an XML (Fig. 12).

Figure 12. Small Example XML Output

When importing this XML into Cameo, we get the following Activity Diagram (Fig. 13), which can be executed using the Cameo Simulation Toolkit.

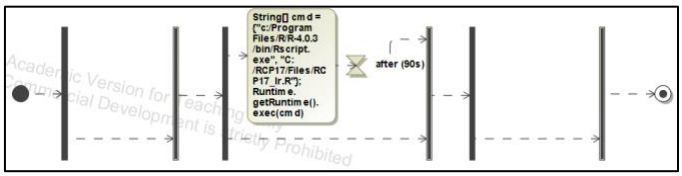

Figure 13. Small Example Suggested Workflow

V. Conclusion and Future Works

In this paper, we re-introduce Inference Enterprise Modeling (IEM) as a technique to create digital twins for insider threat detection enterprises. We identify an opportunity to scale IEM knowledge by employing systems engineering techniques. We develop a framework to document and formalize IEM modeling knowledge that can be used to rapidly prototype digital twin solutions for insider threat detection enterprise modeling problems. Finally, we demonstrate that this can be done by modeling activity diagrams and executing these processes in Cameo EA using Cameo Simulation Toolkit. This method can also be extended as a knowledge management system that can capture the knowledge for various engineering projects involving a process-oriented workflow that involves manual and software-related tasks.

There are some limitations to this work. The quality of the suggested solution depends on the assumption that the future user of this solution understands the new modeling problem correctly. If the future modeler does not use the correct problem requirements query, there is no guarantee that the output will be of good quality. This challenge could be resolved by employing a natural language processer and data pre-processing script to automatically extract the problem requirements of a new potential problem. Also, the template used in this approach is based on IEM activities, so it follows a linear process where one activity follows the other. This may not work in other domains where the problem-solving process looks different. Supporting additional templates works as a short-term solution, but a more flexible approach to generalizing various workflows or processes is an interesting research direction. Other future work includes extending this work to integrate with parametric diagrams to take advantage of the full simulation capabilities of the systems modeling software.

Acknolwedgement

The Research reported here was supported under IARPA contract 2016 16031400006. The content is solely the responsibility of the authors and does not necessarily represent the official views of the U.S. Government.

References

[1] D. M. Buede *et al.*, “Inference enterprise models: An approach to organizational performance improvement,” *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 6, p. e1277, 2018.

[2] K. B. Laskey *et al.*, “Modeling inference enterprises using multiple interoperating models,” in *INCOSE International Symposium*, 2018, vol. 28, no. 1, pp. 1764–1777.

[3] Y. Wang, T. Steinbach, J. Klein, and R. Anderl, “Integration of model based system engineering into the digital twin concept,” *Procedia CIRP*, vol. 100, pp. 19–24, 2021.

[4] A. M. Madni, C. C. Madni, and S. D. Lucero, “Leveraging digital twin technology in model-based systems engineering,” *Systems*, vol. 7, no. 1, p. 7, 2019.

[5] M. Hause, “The Digital Twin Throughout the SE Lifecycle,” in *INCOSE International Symposium*, 2019, vol. 29, no. 1, pp. 203–217.

[6] Wikipedia contributors, “Systems engineering,” *Wikipedia*. Wikipedia, The Free Encyclopedia, Oct. 09, 2021. Accessed: Nov. 12, 2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Systems_engineering&oldid=1049109928

[7] “Transitioning Systems Engineering to a Model-based Discipline - SEBoK,” *SEBoK GUIDE TO THE SYSTEMS ENGINEERING BODY OF KNOWLEDGE*, Oct. 14, 2020. https://www.sebokwiki.org/w/index.php?title=Transitioning_Systems_Engineering_to_a_Model-based_Discipline&oldid=60100 (accessed Mar. 25, 2021).

[8] A. L. Ramos, J. V. Ferreira, and J. Barceló, “Model-based systems engineering: An emerging approach for modern systems,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 1, pp. 101–111, 2011.

[9] A. Morkevicius, “Making the Most of an Enterprise Architecture Modeling Tool.” No Magic, Inc., 2011. Accessed: Sep. 06, 2021. [Online]. Available: https://www.nomagic.com/mbse/images/whitepapers/Making_the_Most_of_EA_Modeling_Tool.pdf

[10] N. M. Inc, “Cameo Enterprise Architecture.” https://www.nomagic.com/products/cameo-enterprise-architecture (accessed Mar. 24, 2021).

[11] J. Carroll, I. Herman, and P. F. Patel-Schneider, “OWL 2 Web Ontology Language Document Overview (Second Edition).” W3C OWL Working Group, Dec. 11, 2012. Accessed: May 31, 2018. [Online]. Available: https://www.w3.org/TR/2012/REC-owl2-overview-20121211/

[12] A. Doan, J. Madhavan, P. Domingos, and A. Halevy, “Learning to map between ontologies on the semantic web,” in *Proceedings of the 11th international conference on World Wide Web*, 2002, pp. 662–673.

[13] G. O. Consortium, “The Gene Ontology (GO) database and informatics resource,” *Nucleic acids research*, vol. 32, no. suppl_1, pp. D258–D261, 2004.

[14] Z. Rajabi, B. Minaei, and M. A. Seyyedi, “Enterprise architecture development based on enterprise ontology,” *Journal of theoretical and applied electronic commerce research*, vol. 8, no. 2, pp. 85–95, 2013.

[15] J. de Bruijn, D. Fensel, U. Keller, and R. Lara, “Using the web service modeling ontology to enable semantic e-business,” *Communications of the ACM*, vol. 48, no. 12, pp. 43–47, 2005.

[16] H. Graves, “mbse:ontology [MBSE Wiki],” Jun. 11, 2013. https://www.omgwiki.org/MBSE/doku.php?id=mbse:ontology (accessed Nov. 12, 2021).

[17] Henson Graves and Matthew West, “Current State of ontology in engineering systems,” 2012. https://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:oat:graves-current-state-ontology_.pdf (accessed Nov. 12, 2021).

[18] D. Gašević, N. Kaviani, and M. Milanović, “Ontologies and software engineering,” in *Handbook on Ontologies*, Springer, 2009, pp. 593–615.

[19] H.-J. Happel and S. Seedorf, “Applications of ontologies in software engineering,” in *Proc. of Workshop on Sematic Web Enabled Software Engineering"(SWESE) on the ISWC*, 2006, pp. 5–9.

[20] R. Seethamraju and O. Marjanovic, “Role of process knowledge in business process improvement methodology: a case study,” *Business Process Management Journal*, 2009.

[21] R. Miles and K. Hamilton, *Learning UML 2.0*, 1st ed. Beijing ; Sebastopol, CA: O’Reilly, 2006.

[22] Z. D. Kelemen, R. Kusters, J. Trienekens, and K. Balla, “Selecting a process modeling language for process based unification of multiple standards and models,” Budapest, Technical Report TR201304, 2013.

[23] L. Delligatti, *SysML distilled: a brief guide to the systems modeling language*. Upper Saddle River, NJ: Addison-Wesley, 2014.

[24] J. D. Lee, A. K. Zaidi, and K. B. Laskey, “Rapid Prototyping Insider Threat Inference Enterprise Model Workflows Using Ontology-Template Approach,” in *Systems Engineering in Context*, Springer, 2019, pp. 643–652.

[25] J. D. Lee, S. Matsumoto, A. K. Zaidi, and K. B. Laskey, “Towards Automating Design and Development of Inference Enterprise Models,” in *2019 IEEE International Systems Conference (SysCon)*, Orlando, FL, USA, Apr. 2019, pp. 1–6. doi: 10.1109/SYSCON.2019.8836882.