Conference on Cyber Conflict
Proceedings 2010
C. Czosseck and K. Podins (Eds.)
CCD COE Publications, 2010, Tallinn, Estonia.

DOMAIN ENGINEERING FOR CYBER DEFENSE: A CASE STUDY AND IMPLICATIONS

Jaak TEPANDI[a], Gunnar PIHO[a,b], Innar LIIV[a]

[a]*Tallinn University of Technology, Estonia,* [b]*University of Leeds, UK*

Abstract: Efficient processing of large amounts of data gathered about real-world objects, activities, attacks, and other relevant entities is vital for successful cyber defense. The value of data processing depends critically on the quality of utilized data models. To be more practical, data models should be integrated and reused. Domain engineering addresses the challenges of model quality, integration and reuse. This paper analyses the possibilities of using domain engineering for cyber defense, exemplified and motivated by a case study of predicting the results of terrorist behavior. The presented case study demonstrates the need for adequate domain engineering for simulation and cyber defense tasks. An approach to modeling and integrating cyber defense and simulation data with archetype-based domain engineering is presented.

Keywords: cyber defense, simulation, domain engineering, modeling of terrorist behavior, archetype-based engineering

INTRODUCTION

The early motivation for this research emerged after the 1995 Oklahoma City bombing that killed 168 people. The components of the truck bomb used in this event were easily accessible to an ordinary person: ammonium nitrate, an agricultural fertilizer, and nitromethane, a motor-racing fuel. In the same way, information about usage of these and other components for preparing explosives was also easily available. The more technology develops and information spreads, the more one can expect such incidents to occur – a fact that certainly worries many people. In the study (Tepandi 2002, Tepandi&Vassiljev 2008) we modeled, simulated, and investigated a terrorism spreading problem closely related to similar issues in cyber defense.

While working on this study it soon became evident that the value of simulation results critically depends on the correctness of the world model used in simulation. Adequate methods for data representation – and more generally, for domain engineering – are also important for effective cyber defense (Presidency of the Council of the European Union (PCEU) 2009, Department of Homeland Security(DHS) 2009, Thomas&Cook 2005).

Powerful methods for domain modeling have been developed by the software engineering community (Sommerville 2006, Bjorner 2006). In the next section we outline the relationships between cyber defense and domain engineering. The second section presents a summary of the case study aimed at modeling and simulation of a terrorism spreading problem. The third section is devoted to principles of archetype-based engineering of domains, requirements, and software for cyber defense (Arlow&Neustadt 2003, Piho, et al., 2009). The final section presents some open challenges and directions for further work.

1. CYBER DEFENSE AND DOMAIN ENGINEERING

An important aspect of cyber defense is processing of large amounts of data gathered about real-world objects, activities, attacks, and other relevant entities (DHS 2009). Such data processing may give answers to specific information requests, enable mining of significant clues that help to prevent or counter cyber attacks, or provide cost-effective simulation solutions to critical information infrastructure protection (CIIP) exercises (PCEU 2009).

Efficient data processing depends critically on data representations (Thomas&Cook, et al., 2005). If the data entities are fragmented and difficult to relate to each other, then solving each new problem begins from scratch and significant data relation-

ships may be lost. In contrast, in case of integrated data representations, various objects may be associated with each other and hidden relationships may be discovered more easily. In addition, new problems may be defined and solved based on already existing world models. In (Thomas&Cook 2005, p. 133), one of the actions recommended for advancing the community's capabilities for data representation and transformation is to "create methods to synthesize information of different types and from different sources into a unified data representation so that analysts, first responders, and border personnel may focus on the meaning of the data".

Traditionally, mathematical and statistical representations are widely used to present analytical data. These representations enable efficient transformation of data to be utilized in analysis and simulation tasks. As the area to be represented widens, mathematical and statistical models tend to be less comprehensible. Therefore it is useful to utilize domain modeling experience gained in software engineering.

A proper domain model is essential for a successful realization of an information system; therefore, various representation tools and methods have been developed by the software engineering community. Requirements specified for information systems must be well agreed with the customer. Hence, significant emphasis has been put by software engineers on understandability of system models by both the developer and the customer (e.g. (Sommerville 2006, Bjorner 2006)).

As in cyber defense, different systems may reflect diverse aspects of the same domain. Therefore, it is important to have integrated domain models. In addition, the reality changes and the systems must reflect this change. Domain engineering addresses the challenges of both model integration and reuse (Bjorner 2006). It attempts to build reusable models for application domains – knowledge areas that cover important fields of reality and share common concepts. An application domain model may represent terrorism simulation, cyber defense, CIIP, medical laboratory, or other areas.

Critical IT infrastructure protection is an example of an application domain which may benefit from domain engineering. Critical IT infrastructure is important in itself, providing critical services (e.g. communications and access to vital registers). It is also vital as a supporting framework without which other critical services (e.g. banks, health services) would be impossible. Any significant future cyber conflict will most probably comprise attacks on the critical IT infrastructure. Critical information infrastructure protection (CIIP) needs to be supported by regular exercises (Enisa 2009). CIIP exercises may be expensive and sometimes impossible to perform full-scale. Simulation is a viable alternative. It cannot provide full participation experience, but enables evaluating influence, resources, consequences, and so on. Usefulness of simulation depends on the quality of data representations used and consequently – on the quality of domain modeling.

2. A CASE STUDY: AGENT-ORIENTED MODELING OF TERRORIST BEHAVIOR DYNAMICS

The application domain of the case study (Tepandi 2002, Tepandi&Vassiljev 2008) is spreading of terrorist acts. The objective of this study is to comprehend the dynamics of terrorism spreading as a function of certain world properties, such as access to information, availability of material resources, and others. The analysis is based on the indication that the probability, power, and influence of terrorist attacks – both in physical and cyber reality – are increasing with growing access to information and material resources.

2.1 MAIN HIGHLIGHTS OF THE STUDY

We begin with analyzing two extreme cases. On one extreme, when the resources, both the information and materials, are on a low level, one cannot do much harm. For example, prior the twentieth century it was practically impossible to affect the living conditions on Earth significantly by even a large number of people. In this case the probability that the population will be terminated as a result of cumulative effect of terrorist attacks may be evaluated to zero. The situation has been changed, but powerful resources have still usually been out of reach of an ordinary person. Little by little, this encouraging situation is also changing. Like in Oklahoma and subsequent events, ordinary people have more and more information and resources available for terrorism.

The other extreme is a hypothetical situation of extremely large resources being available to everyone. It seems clear that in such a case the world would not last long. There will inevitably be people who are stressed enough, or who believe that passing away is the best option for everyone. The probability that the population will cease to exist due to the cumulative effect of terrorist attacks may be evaluated to one. This extreme situation is unlikely, for the governments recognize the danger and are building barriers to available resources. Still it seems inevitable that the power in the hands of individuals is growing, and the governments are taking more measures to prevent that power from growing too high.

The study addresses the question whether the transition between the above two extremes is evolutionary (for example linear) or stepwise (for example exponential). An evolutionary transition would allow taking measures when the situation indicates that the level of resources is too high and it is time to take a more restrictive approach. In the case of a stepwise transition there might be no way back after a

certain level of resources has been exceeded.

The study involved designing the domain model for terrorist behavior, development of the simulation environment, performing simulation experiments, and drawing conclusions.

2.2 THE DOMAIN MODEL FOR TERRORIST BEHAVIOR

The domain model for terrorist behavior is based on a world of agents. The world has certain properties and so do the agents. The world evolves in a discrete time, where each unit represents a world cycle. The initial properties of the agent are determined by the world properties. The properties of each agent at the next moment are determined by the agent's individual properties, the world properties and the values of its neighbor agents at the current moment.

The world is determined by its shape and size, overall access to information and access to resources values, the level of interaction between the agents ("sociality"), initial distributions of information, resources, violence, and charity, as well as the rules for activating violence or charity acts and for changing the agent values. Example: a condition that in a specific world W, the overall value of the AccessToInformation property is in the range of 0 to 1, may be expressed by $0 \leq$ WorldValue(W, AccessToInformation) ≤ 1.

Some properties of the world determine characteristics of the simulation, for example the rules which determine whether the world is considered to be evolving, stable, or extinct for the purpose of analysis.

The agent properties include the amount of information and resources, as well as the levels of violence and charity. Example: a condition that for a specific agent A in the world W, the value of the AggressivenessLevel property is in the range of 0 to 1, may be expressed by $0 \leq$ AgentValue(W, a, AggressivenessLevel) ≤ 1.

In each world cycle, the agents go through various interactions. The agents are born and die; they acquire new and lose existing information and resources according to certain laws. The agents also perform violence or charity acts according to their property values. The nature, probability, and influence of the act depend on the agent mood, its access to resources and information, and other factors. Each violent act enlarges the aggressiveness of the neighbors and may kill other agents; each charity act enlarges the charity of the neighbors and may bring new agents into being.

An example of an agent's interaction: an object A performs a terrorist act with probability proportional to the overall violence level and its own knowledge, resources, and aggressiveness levels. As the result of the act, the neighbors of A will be removed with probability adversely proportional to the distance from A (nearer neighbors suffer to a greater extent). The Aggressiveness property of the neighbors will increase in adverse proportion with the distance from A (nearer neighbors are more influenced)

Given a world with its agents, properties, and interactions, this world may be started, letting the agents act and interact. This process may exist in a long-term or infinite continuous interaction. It may also end in termination of the population (most or all agents are destroyed as a result of terrorist attacks) or in stable non-interacting situation. The first situation occurs most probably when the opposite properties, such as violence and charity are balanced, the other two – when they are out of balance.

2.3 THE SIMULATION ENVIRONMENT

The simulation environment developed for the study provides a simulation model description language (SMDL), tools for executing the model defined in SMDL, facilities for visualizing the results of the simulation, as well as tools for saving and analyzing the results of the simulation.

The SDML defines the simulation general properties, the world general properties, the initial distribution intervals for the agent property values (Aggressiveness, Violence, Knowledge, Charity, and Resources), the rules determining change of the agent property values in each cycle, and agent properties. Example: an assertion "aggressiveness_for_act=80" specifies that if an agent's aggressiveness value is higher than 80 (on a scale 0...100) it will consider a terrorist act.

The properties defined in SDML allow a wide variety of specific populations to be simulated using the tools for executing the SDML model.

Facilities for visualizing the results of the simulation include the main window and auxiliary windows. The agents are represented as squares of different colors. The black color of a square depicts a dead agent. The other colors indicate the state of aggressiveness of an agent, varying from light green (zero aggressiveness level) to yellow (average aggressiveness) to red (very aggressive). The auxiliary windows provide graphs for average aggressiveness of the population and the number of agents alive with respect to the number of turns passed.

2.4 EXPERIMENTS AND CONCLUSIONS

At the start of a simulation run the world and the agents are specified in the SMDL. The rules for changing the world and agent properties during each world cycle are also given in SMDL. At the end of each cycle, a check is performed for the end of the simulation. The simulation run is finished and the final results are output in the following cases: the population has survived; the population has stabilized; the population has terminated.

Experiments have been performed using agent models of different complexity. In a typical experiment, the world properties, such as access to information, varied from minimum to maximum. For each intermediate value, a series of simulation runs were performed to evaluate the probability of population termination due to the cumulative influence of terrorist acts. The resulting graphs of the relationship between the world properties and the termination probability were analyzed.

The results of a typical experiment portraying the relationship between the probability of population survival and access to information for different levels of access to material resources demonstrate that the relationship tends not to be linear. Rather, the graphs represent a stepwise or constant relationship. Therefore the model does not necessarily lead to destabilization of the population with the growth of access to information. But in the case it does, the resulting dependency is rather step-wise than smooth. These experiments allow concluding that this property may be not an incidence, but regular behavior.

Thus the findings indicate that the results of terrorism activities can start spreading very quickly with the growing amount of information and material resources in individuals' hands, allowing no point of return. These results should be taken into account when designing political, social and technical systems to prevent terrorism.

The case study used both simulation and visualization for delivery of results. Simulation helps to have deeper insight into the cyber defense problems and explore risks of critical infrastructure protection situations that typically have previously not been experienced in reality. For example, as the cyber defense systems must predict behaviors and situations unspecified beforehand, simulation helps to cost-effectively predict the need for resources for these systems. Visual analytics tools and techniques help to better synthesize information, derive insight, discover the unexpected, and communicate assessment effectively for action (DHS 2009).

This case study has also demonstrated that trustworthy world models comprising terrorist activities are vital for these kinds of experiments, are complex, and require much development effort. The simulation environment must utilize multiple models for diverse tasks and experiments.

3. TOWARDS ARCHETYPE-BASED DOMAIN MODEL OF CYBER DEFENSE

The SMDL introduced in this case study is based on mathematical notations (sets, relationships, formulas) and a language for presenting the world life cycle. The practicality, integrity, and other properties of this kind of models are not easy to comprehend and analyze. To make building and analysis of domain models more feasible we propose principles of archetype-based development (ABD) (Piho, et al., 2009) for cyber defense. We use ABD at Clinical and Biomedical Proteomics Group (University of Leeds, UK) for developing software factory (Greenfield 2004) for laboratory information management system (ASTM 2006) software. In ABD we combine triptych software development (from domain via requirements to dependable software) (Bjorner 2006) with archetype and archetype patterns initiative (Arlow&Neustadt 2003).

An archetype is defined as a primordial thing that occurs consistently and universally in various domains (business, manufacturing, transportation, defense, etc.) and in systems supporting such domains. Examples of archetypes are product, feature, money, address, person, organization and so on. An archetype pattern is a collaboration of archetypes. Arlow and Neustadt have the following archetype patterns: party and party relationship, product, order, inventory, quantity and money, and rule. In the following we exemplify how to build archetype and archetype patterns based domain models for defense. These models are utilized by simulation and visualization environments to further explore critical situations and problems, as well as to obtain a deep insight that directly supports assessment, planning, and decision-making in this domain.

3.1 ZACHMAN FRAMEWORK AND ABD

Components of the ABD are represented within the Zachman Framework (ZF) (Zachman 1987, Zachman 2003a, Zachman 2003b). The ZF (Fig. 1) is a framework for enterprise architecture, which provides a formal and structured way for describing an enterprise. It is presented as a two dimensional matrix consisting of 6 rows and 6 columns. Each column of the ZF describes single, independent phenomena within the analytical target (Zachman 2003b). The rows present conceptual model, business model, system model, technology model, detailed representations, as well as functioning enterprise aspects of the domain. In what follows we characterize the contents of the columns in ZF with examples.

Column 1 (What) describes what the things are, what the features of those things are and how these things are related to each other. In ABD we use both product and

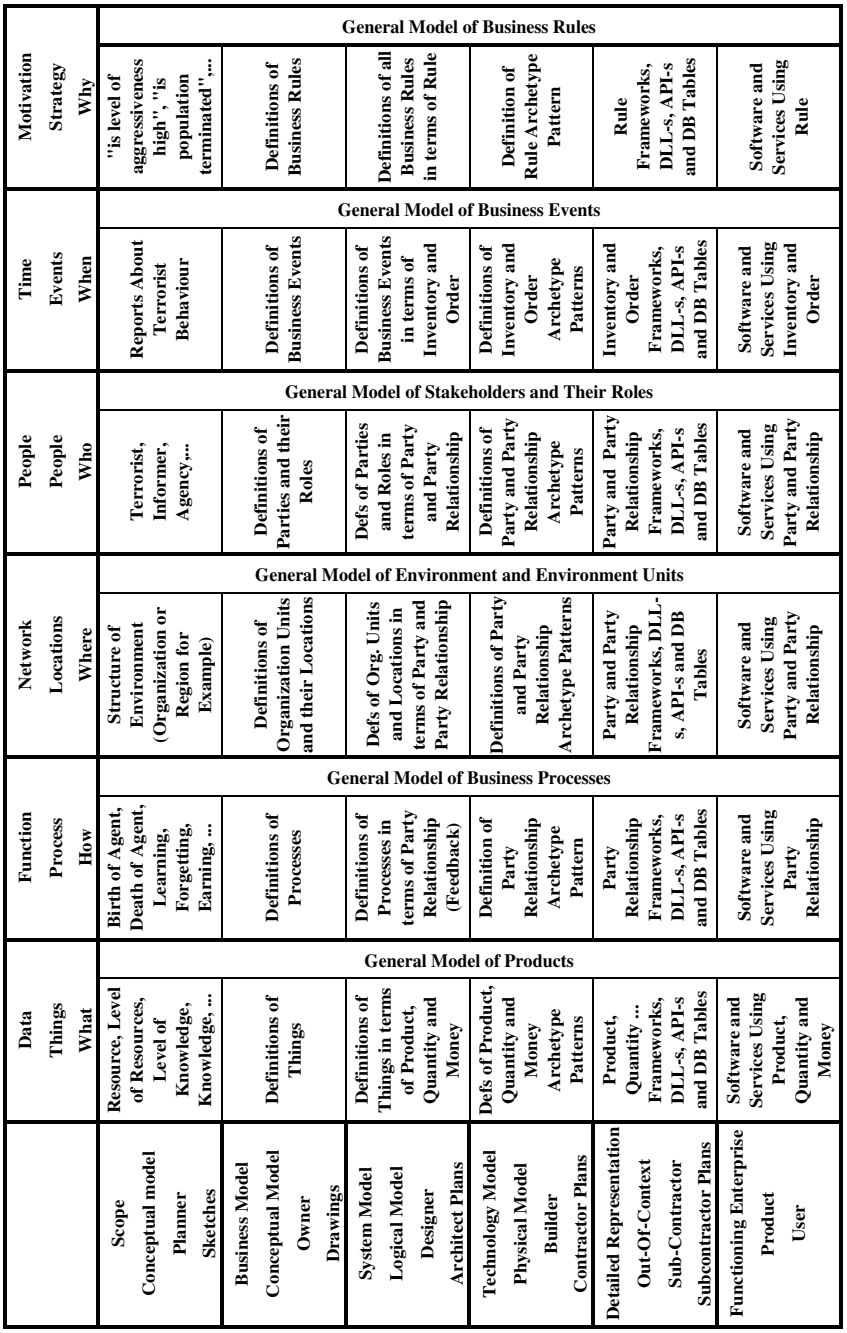

Figure 1. Zachman framework with archetype patterns

quantity (Arlow&Neustadt 2003) archetype patterns for modeling things. Examples of things (derived from agent-oriented model for terrorists behavior (Tepandi 2002)) in the domain of defense can be: resource, level of resources, level of knowledge, knowledge, level of aggressiveness, aggressiveness, probability of reproduction, probability of expiration, level of access to resources, level of access to information, information, etc.

Column 2 (How) describes processes. In ABD we model processes using their feedbacks (Fig. 2) given by one party to other. We use a party relationship archetype pattern *Arlow&Neustadt 2003) for modeling such feedbacks. The examples of processes in the domain of terrorism simulation (Tepandi 2002) are birth of agent, death of agent, learning, forgetting, earning, spending, social interaction, terrorist act, charity act, etc.

These processes can be modeled as reports from one party (informer for example) to other (central agency for example) or from one party (informer) about another party (terrorist for example). More reports from trusted and different parties means better and more implicit picture about the whole process.

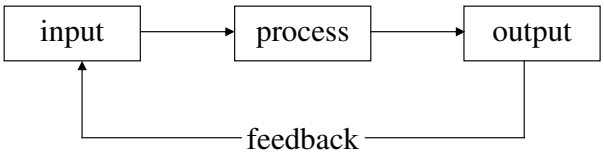

Figure 2. Process and feedback

Column 3 (Where) describes environment. School, hospital, organization, district, region, state, world, infrastructure, and computer network are examples of environments. In ABD we use party and party relationship (Arlow&Neustadt 2003) archetype patterns for modeling environments. We describe the structure of environment in terms of environment units (for instance, organizations are described in terms of organization units – division, department, team, group, etc). The role types each environment unit has to play in the environment are presented. The responsibilities (assigned, mandatory and optional), requirements for responsibilities, as well as conditions for their satisfaction for each role type and for each environment unit are depicted.

Column 4 (Who) describes the agents (persons, organizations, artificial agents) and their roles somehow related to the environment described by Column 3. In ABD we use the party and party relationship (Arlow&Neustadt 2003) archetype patterns for modeling agents and agent roles. Examples of roles of agents in domain of defense are terrorist, informer, agency, etc.

Column 5 (When) describes the events related to the processes described by Column 2. The events must be logged for audit trail or for the late analysis. In ABD we use order and inventory (Arlow&Neustadt 2003) archetype patterns for modeling events. This means that every event will generate (or will change or amend) some order to change something in the inventory. The inventory is a repository for important information about the environment. An example of an event is a report about terrorist behavior.

Column 6 (Why) describes the strategies and strategic questions such as "Is the level of aggressiveness high?", "Is the population terminated?", "Is this person a terrorist?", etc. In ABD we use the simple propositional calculus-based rule archetype pattern (Arlow&Neustadt 2003) as the basic model for strategies.

3.2 EXAMPLES OF ARCHETYPE-BASED MODELS

The following examples are archetype-based models of defense domain. For modeling of things (Column 1 in ZF) we use either product or quantity archetype patterns. As an example, for modeling agent properties like knowledge and aggressiveness, we use quantity (Fig. 3). A quantity is an amount of something characterized according to some measure and corresponding units.

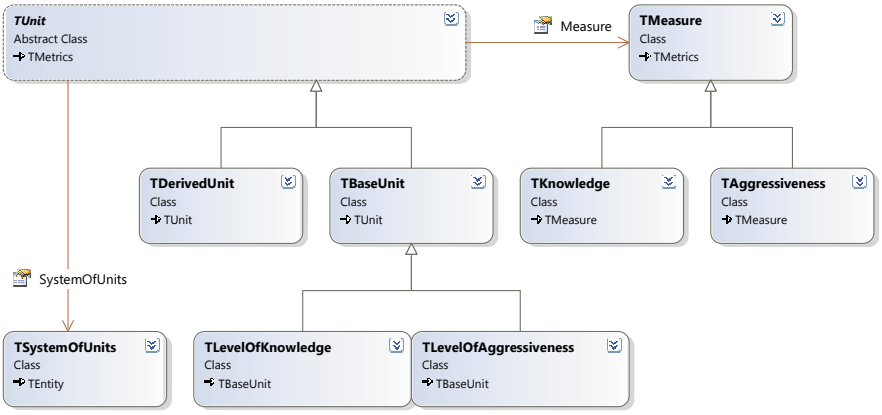

Figure 3. The agent properties

The class diagram in Fig. 3 (prefix "T" in class names comes from "type" or "archetype") comprises two measures (*TKnowledge* and *TAggressivness*) and two units (*TLevelOfKnowledge* and *TLevelOfAggressivness*). Both units are inherited from the *TBaseUnit*. As a result, the units inherit automatically the functions associated with the base unit such as arithmetic operations (addition, subtraction, and so on), round-

ing, or translation of quantity from one unit to other.

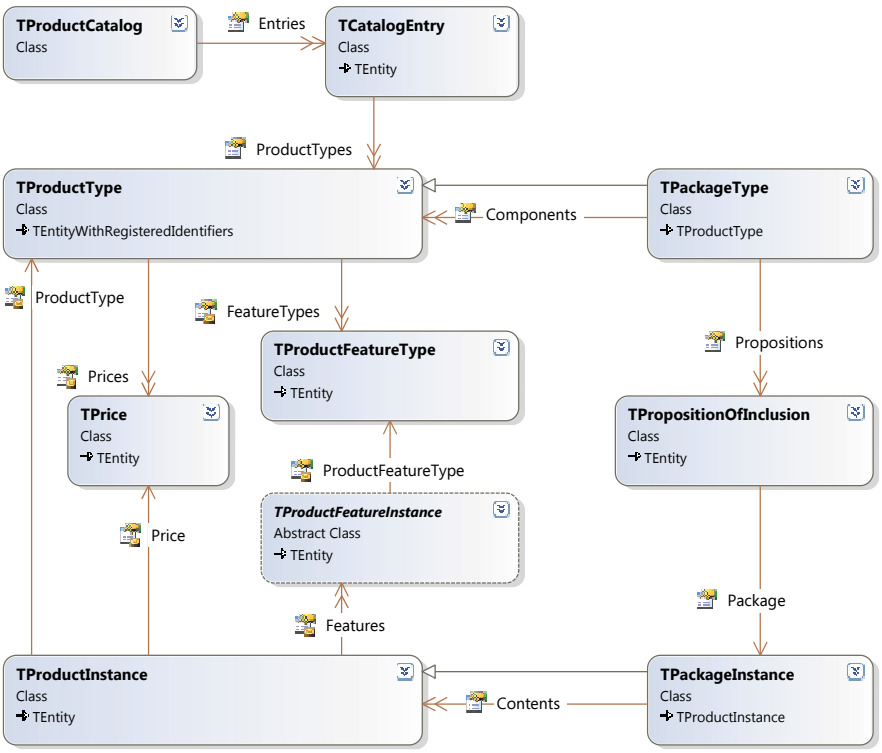

Figure 4. Product archetype pattern abstraction

Some objects in the domain of defense, for example agent resources, are different kinds of products and/or services. All those things that an agent can in principle buy or sell can be modeled by using the product archetype pattern (parts of this pattern are presented in Fig. 4). *Product type* describes the common properties of a set of goods or services and *product instance* represents a specific instance of a product type. *Product feature type* and *product feature* are used either to represent possible product type features (like set of possible colors) or to represent concrete features of specified product instance. Packages (*package type* and *instance* respectively) are selections of products grouped together as a product unit. *Components* in *package type* are used when a package consists of a fixed set of products; a *product set* is used to represent a set of *product types* from which selection by some rule may be made. A *product relationship* is a relationship (upgrade, substitute, replace, complement, compatible, and incompatible) between product types. A *price* is the amount of money that must be paid in order to purchase a product. A product type has pos-

sible *prices* whereas the product instance has an *agreed price*. The *pricing strategy* determines how a price is calculated for a package type. Product *catalog* is a store of product information where *catalog entry* holds the information about a particular type of product in a product catalog. Similarly, a *batch* describes a set of product instances of a specific product type that must be tracked together, for example, for quality control purposes.

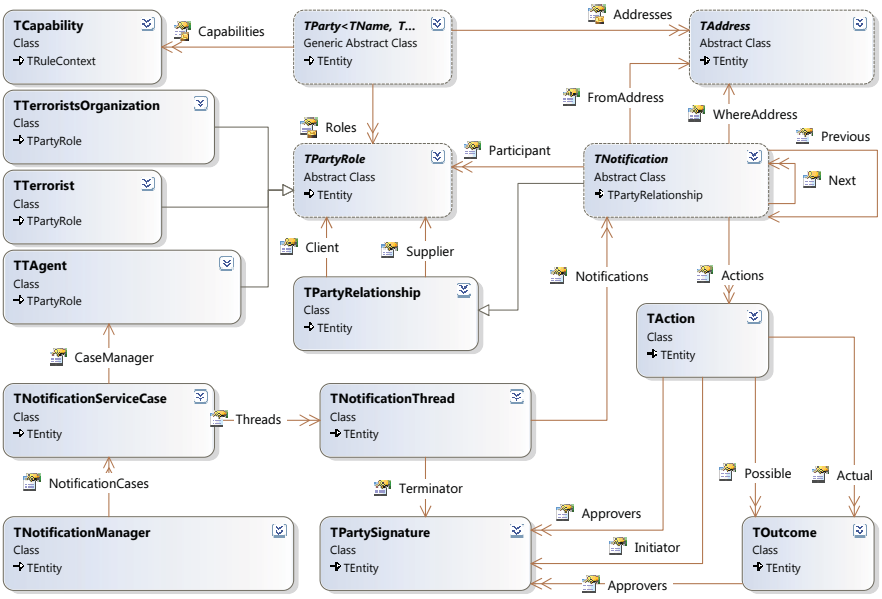

Figure 5. Notification archetype pattern abstraction

For modeling of processes (Column 2 in ZF) we use the party relationship archetype pattern as a base. For example the notification archetype pattern (Fig. 5) concretizes the party relationship archetype (Arlow&Neustadt 2003) and is similar to the customer relationship management archetype pattern (Arlow&Neustadt 2003). In notification, the agent (*TAgent*) "from" address notifies about some event which has happened in the "where" address. In case of terrorism simulation, this may be a party relationship where one agent informs the agency about the behavior of the terrorist's organization or about the behavior of someone who acts on behalf of a terrorist's organization. More than one terrorist or terrorist's organization – participants – can be involved in event the about which the agent has reported. A notification routing is a special case of notification, which represents notification handovers from agent to agent. Notification case tracks all notification threads (sequence of notification) about a specific topic related to a specific terrorist's organization or terrorist. Action represents something that can or must happen (logging of information

or some other action for example) after the notification.

Notifications and actions (logging of information) these notifications generate may be, for example, information about birth (new member of a terrorist's organization or new terrorist's organization), death (death of a terrorist or terrorist's organization), earning (terrorist or terrorist's organization has got more resources), learning (terrorists have got new information) and so on.

For modeling of the environment (Column 3 in ZF) we use the party and party relationship (Arlow&Neustadt 2003) archetype patterns. The *party archetype* (Fig. 6) represents a (identifiable, addressable) unit that may have a legal status and has some autonomous control over its actions. *Persons* and *organizations* are parties.

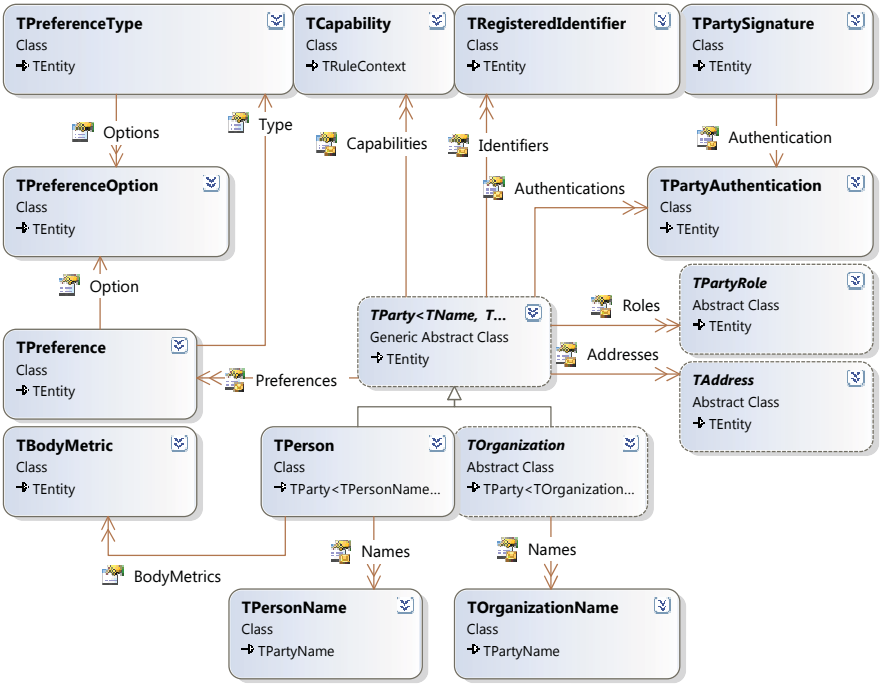

Figure 6. Party archetype pattern abstraction

Party has zero or more *addresses* (phone number, e-mail, web address, postal address) where one and the same address can belong to more than one parties. Party has zero or more *registered identifiers* (passport, VAT number, domain name, stock exchange symbol, etc). Party *authentication* is a way to confirm that the party is who they say they are. Each party can play different *roles* (one and the same person can be for example a student and a member of terrorist's organization). *Preference*

stands for a party's (or a role's) choice of or linking for something (like dietary preference) and is typically selected from a set of options. The *capability* is a collection of facts about what a person or organization is capable of doing as well as *body metric* stores information about the human body. For example the world global properties (Tepandi 2002) like *InitialPopulation, GlobalEndOfPopulation, GlobalAccessLevelToInformation,* etc., are capabilities of a party called the world (set of agents).

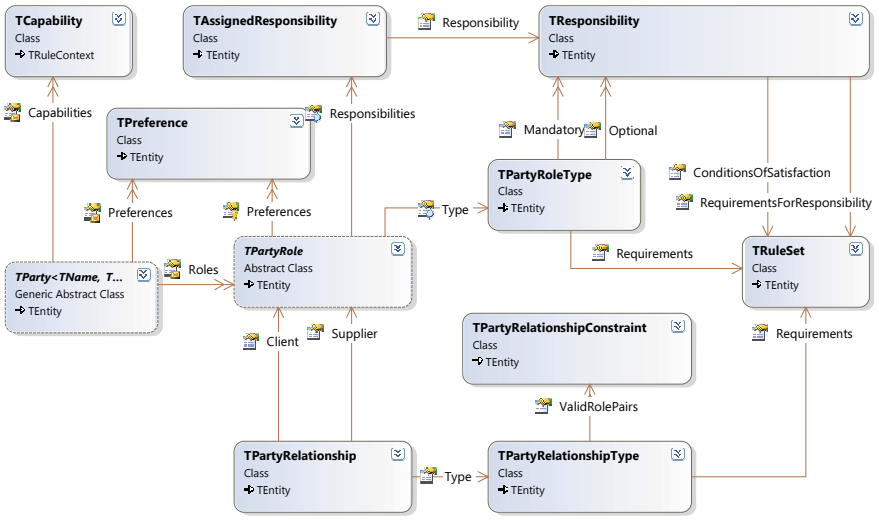

Figure 7. Party relationship archetype pattern abstraction

Fig. 7 abstracts the *party relationship archetype pattern,* which captures a fact about the semantic relationship between two parties in which each party plays a specific role. Binary (more flexible and cleaner than n-ary) relationship is used, which means that one *relationship* binds two roles called "client" and "supplier". It has to be clarified that the role is always solely used to store information that belongs to the role itself and not either to a party or to a relationship. *Role type* is used to store common information for a set of similar roles; *relationship type* is used to store common information for a set of a similar relationship instances. *Responsibility* describes a particular activity that a party, playing a role, may be expected to perform, where the *assigned responsibility* captures the fact that responsibility is assigned to concrete party playing that role. *Conditions of satisfaction,* as well as the *requirements* for *party role type*, for *party relationship type* and for *responsibility* are *rule sets* (see rule archetype below). Here the *capability* (rule context) contains information needed for the execution of rules; in case of a party, this information states whether a party can complete necessary responsibilities for its role in relationship. In ABD we use party relationship archetype pattern for modeling of internal structure (for

example all immobile under the defense) and chains of command in the environment.

The same party and party relationship archetypes are used for modeling of persons (Column 4 in ZF). While in the case of Column 3 (location) the target is to model the environment where the business takes place, in case of Column 4 we model all the stakeholders somehow active or related to the business in question.

Although the location and stakeholder-related domain aspects are both modeled by the party relationship archetype pattern, these models themselves are different. In the case of locations we use the party relationship archetype pattern to model the internal structure of the environment, for example the organizational structure of the enterprise. In the case of stakeholders, we model specific employers, customers, sellers, patients, terrorists and other independent agents with their relationships with the environment in question. We also model the possible relationships between independent agents (e.g. employer A is wife of employer B, and so on).

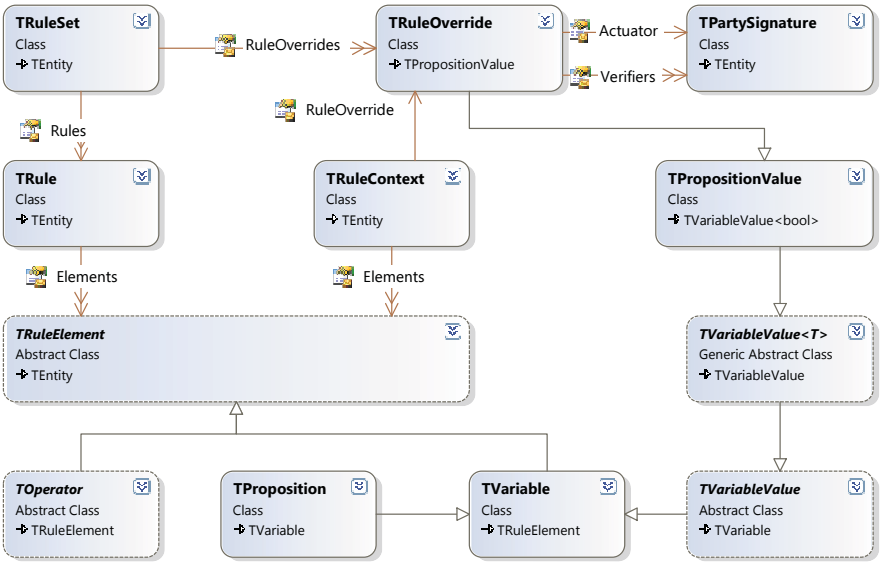

Figure 8. Rule archetype pattern abstraction

The processes (Column 2 in ZF) comprise actions. These actions are triggers for events modeled by Column 5 of ZF (when). For modeling of events, we use the order and inventory archetype patterns. An action generates a document. This document is some type of order, and according to this order, the inventory of environment (Column 3) will be updated.

The last column of ZF (why) describes strategies. For modeling strategies, we use rule archetype patterns. A *rule archetype pattern* (Fig. 8) is a constraint on the operation of the software systems of the business. The rule semantics is defined by sequence of *rule elements*. Rule elements can be *operators*, *propositions* (a statement that has a truth value) and *variables*. Operator is either a Boolean operator (and, or, xor, not) or quantifier operator (=, !=, <, >, <=, >=). While a *rule* represents some kind of mask or pattern, the *rule context* contains the informational context for the execution of a rule. Rule context represents this information as a collection of rule elements that may be propositions or variables, but not operators. The following sets are examples of simple rule (R) and respective rule context (C) (Arlow&Neustadt 2003).

R = {IsGoldCardHolder, IsSilverCardHolder, OR,
CarryOnBaggageKg, AllowedBaggageKg, LESS, AND}
C = {true, false, 4.5, 5.0}

IsGoldCardHolder and *IsSiverCardHolder* are propositions which take the actual value from the context C (true and false, respectively). *CarryOnBaggageKg* and *AllowedBaggageKg* are variables which take the actual values (4.5 and 5.0) from context. OR, LESS and AND are operators.

3.3 FROM DOMAIN VIA REQUIREMENTS TO SOFTWARE

The triptych software process (Bjorner 2006) – from domain model via requirements to software – has a very simple informal description: before starting to write software, we need to know the requirements; before knowing requirements, we have to understand the domain; to understand the domain we have to study one. The interpretation on ZF rows in terms of triptych (requirements, domain, and software) development can be as follows.

Row 1 (conceptual model) is just the glossary (list of things, objects, assets, etc.) that defines the scope or boundary of requirements. For example the cell defining the scope for Column 4 (people) for the domain of defense can include terms like *agent*, *terrorist*, *informer*, *agency*, and so on.

Row 2 (business / semantic model) is a definition and a model of the actual requirements. It defines the concepts (terms and facts) actually needed. This can be represented as simple narratives (for instance "terrorist has an alias", "informer has a codename", etc.) or in some more formalized (for example class diagrams) notation.

Row 3 (system / logical model) describes the requirements in terms of domain

model. For Column 4 this means for example, that "terrorist is the role for person", "terrorists alias is a name for person whose role is a terrorist", etc.

Row 4 (technology / physical model) is the actual model of the domain. For Column 4 this is the model of the party archetype pattern.

Row 5 (detailed definition) is the party archetype pattern realized for example as API, or as a database scheme supporting this pattern under some specific database engine.

Row 6 (product) is the software or service which fulfils the requirements from row 1 and row 2.

Due to the technological infrastructure and nature of attacks, it is possible to have two strategies for implementation of cyber defense domain engineering. One approach is to model the normal behavior in systems with a goal of detecting abnormal events, behavioral outliers, etc. Another, complementary approach is to model specific attack types (e.g. DDoS attacks (Mirkovic&Reiher 2004, Douligeris&Mitrokotsa 2004)). Archetype-based domain engineering allows simultaneously both top-down (building models from existing taxonomies of attacks) and bottom-up (generalizing data-driven models from detailed event logs) approaches. Archetypes representing normal behavior of the system and a specific attack must be described in a way to allow semi-automatic synthesis of simulation procedures.

4. CONCLUSIONS

We have described a case study of predicting the results of terrorist behavior, stressed the need for adequate domain engineering for simulation and cyber defense tasks, and proposed an approach to integrating cyber defense and simulation data representations using domain engineering with archetype-based domain engineering.

As open challenges and directions for further work, the cyber defense domain can be viewed as an integration of object and process views. Domain engineering for the processes and integration of object and process representations are some directions for further work.

This work was partially financed from ESF grant No. 6839 and target financing grant SF0140013s10.

REFERENCES

- Arlow, J., Neustadt, I., 2003. *Enterprise Patterns and MDA: Building Better Software With Archetype Patterns and UML* : Addisson-Wesly.
- ASTM. 2006. *E1578-06 Standard Guide for Laboratory Information Management Systems (LIMS)* : ASTM International.
- Björner, D., 2006. *Software Engineering, Vol. 3: Domains, Requirements, and Software Design.* Texts in Theoretical Computer Science, the EATCS Series : Springer.
- C. Douligeris, C. and A. Mitrokotsa, A, 2004. *DDoS Attacks and Defense Mechanisms: Classification and State-of-the-art*, Comp. Networks, vol. 44, pp. 643–66.
- Department of Homeland Security. 2009. *National Infrastructure protection Plan.*
- ENISA. 2009. *Good Practice Guide on National Exercises. Enhancing the Resilience of Public Communications Networks.*
- Greenfield, J., et al., 2004. *Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools* : Wiley.
- Mirkovic, J. and Reiher, P., 2004. *A taxonomy of DDoS attacks and defence mechanisms. ACM SIGCOMM Computer Communications Review, 34(2):39–54, Apr. 2004.*
- Piho, G., Roost, M., Perkins, C., and Tepandi, J., 2009. Towards Archetypes Based Software Development. *CISSE.* (accepted for publication).
- Presidency of the Council of the European Union., 2009. "Conference conclusions." Tallinn : s.n., 27-28 April, 2009. European Union Ministerial Conference on Critical Information Infrastructure Protection.
- Sommerville, I., 2006. *Software Engineering.* : Addison-Wesley.
- Tepandi, J. and Vassiljev, S., 2008. Conflict Expansion in an Information Rich Society: Feasibility of Corrective Actions. [ed.] E. Khaled. *Innovations and Advanced Techniques in Systems, Computing Sciences and Software Engineering* : Springer, pp. 231-236. ISBN 978-1-4020-8734-9.
- Tepandi, J., 2002. Simulation of Conflict in an Agent World: Access to Resources and Possibility of Termination of the Population. *Informatica.*, Vol. 13, 4, pp. 501-512.
- Thomas, J. J. and Cook, K. A., 2005. *Illuminating the Path. The Research and Development Agenda for Visual Analytics.* : National Visualization and Analytics Center.
- Zachman, J. A., 1987. A Framework for Information Systems Architecture. *IBM Systems Journal.* Vol. 26, 3.
- Zachman, J. A., 2003b. *The Zachman Framework: A Primer for Enterprise Engineering and Manufacturing.* 2003b.
- Zachman, J. A., 2003a. *The Framework for Enterprise Architecture – Cell Definition.* ZIFA .