A Unifying Framework for UML Profile-based Cognitive Modeling: Development and Experience*

Zhi Zhu[1,2], Yonglin Lei[1,2], Yifan Zhu[1]
[1]Institute of Simulation Engineering
College of Information System and Management, National University of Defense Technology, Changsha 410073, P.R. China
zhuzhi|yllei|yfzhu@nudt.edu.cn

Abdurrahman Alshareef[2], Hessam S. Sarjoughian[2]
[2]Arizona Center for Integrated Modeling and Simulation
School of Computing, Informatics & Decision Systems Engineering, Arizona State University, Tempe, AZ 85281, USA
alshareef|sarjoughian@asu.edu

ABSTRACT

To achieve model reuse, cognitive decision behaviors are usually implemented using a scripting language which is separate from the programming language used to implement simulation models. Therefore, it is desirable to establish a much better grounding for cognitive modeling. In the context of Domain-Specific Modeling (DSM), metamodeling from scratch for designing such a scripting language poses some limitations, among which is the issue of integrating various models that are represented by different customized languages, together with a large expenditure of designing, implementing, and maintaining these languages and their supporting resources. Instead, UML Profile-based metamodeling is adopted in this research, as a light weighted extension to capture the cognitive domain concepts, relationships, and constraints. Moreover, we propose a unifying framework for designing the domain specific profile where the expressiveness of cognitive domain specification is increased and the development cost and time decreased. Also, we present the development process by constructing a profile of Anti-Submarine Tactics (AST) and illustrate it by demonstrating an example with a scenario of armed escort.

CCS CONCEPTS

• **Software and its engineering** → Model-driven software engineering; Domain specific languages; Unified Modeling Language (UML) • **Computing methodologies** → Modeling and Simulation

KEYWORDS

DSL, UML profile, cognitive modeling, anti-submarine

1 INTRODUCTION

Cognitive modeling has long been a significant field in military system domain. Inspired from model driven engineering (MDE), decomposition and increasing the level of abstraction are two effective means to reduce system specification complexity. However, it poses some challenges especially regarding human-in-loop systems where the combat rules depend on human wills and the dynamics of the environmental situation. With fixed pre-implemented interfaces, also namely a script-based tactics manager [1], cognitive behaviors are usually developed dynamically outside the simulation models. In other words, simulation models can be reused in various scenarios without any modification when tactics change, by modifying only the tactics written in scripting languages. In order to enable such a language to describe the flexible tactics, metamodeling mechanism has recently been adopted to improve the abstract hierarchy in the context of Domain-Specific Modeling (DSM), together with the model transformation technology [2]. Currently, there are two major kinds of metamodeling mechanisms. The first is to provide a lightweight extension for the existing language or profile, e.g., UML Profile [3]. The second is to establish a heavyweight metamodeling from scratch based on Meta-Object Facility (MOF), e.g., Eclipse Modeling Framework (EMF) [4]. Determining which one is better has been discussed in the literature [5], and yet there are no definitive answers for constructing a Domain-Specific Language (DSL).

Tactics for cognitive behaviors of combat effectiveness simulation system are featured with high flexibility and richness in domain knowledge. First, it requires a separation of cognitive models from simulation models. Without maintaining this separation, the simulation models would have to be rewritten and recompiled every time when the tactics change. Second, a domain specific language tailored in a more understandable and modifiable way to represent the rich domain knowledge. Third, a set of domain specific constraints enforced at a higher level of abstraction to minimize the faults and errors found in later stages of the development process. Such considerations should be accomplished in a well-established framework due to the large number of tactics embedded in the battlefield and the complexity of open environment. There are also many constraints that need to be considered. These constraints may crosscut various entities, which make the development task very challenging.

In this research, we adopt the lightweight method and present a unifying framework. The structure of the paper is as follows: Section 2 describes the research background and related work. Section 3 proposes the unifying framework. In section 4, an Anti-Submarine Tactics (AST) profile is constructed. Also, an armed escort case is designed, which is followed by conclusions in Section 5.

SIMUTOOLS '17, September 11–13, 2017, Hong Kong, China

ACM ISBN 978-1-4503-6388-4/17/09…$15.00
https://doi.org/10.1145/3173519.3173524

2 BACKGROUND AND RELATED WORK

To better know what the cognitive modeling is, it is necessary to learn its source and the relationships with relative parts. In this section, the effectiveness simulation domain is explored through decomposing it into static structure, physical behavior, and cognitive behavior parts. After that, relative standard profiles are introduced as a good guideline to cognitive modeling.

2.1 Domain of Effectiveness Simulation

Combat effectiveness simulation domain (ES domain) is usually decomposed into static structure, physical behavior, and cognitive behavior with the objective to decrease the level of system complexity [6]. An entity of the static structure applies its physical behavior to constitute a simulation model as distinct from its cognitive model. Fig.1 shows the decomposition of the ES domain.

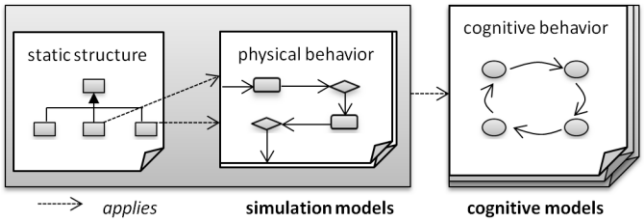

Figure 1: Decomposition of the ES domain

Static structure contains the inherent entities and their relationships of a specific system. It is the most stable part which is determined by the nature of a given system. Physical behavior refers to the dynamic behavior of a specific entity in physical and information domains. It is also relatively stable but somewhat varies across different situations. The static structure and physical behavior are basic models for cognitive models.

In this paper, we concentrate on the specification of cognitive models. Cognitive behavior belongs to the cognitive and social domain in which human's will or the outside environment plays a key role. Due to the difference between decision-maker's abilities, cultural traditions, and educational background, it is the most flexible part which should be developed independently from the simulation models. Hence, a similar simulation model can be reused to schedule multiple cognitive models through predefined interfaces.

2.2 Related Profiles

Based on Model Driven Engineering (MDE), Lei et al. [7] proposed a model architecture-oriented method to handle model composability [8], domain-specific modeling [9], and model evolvability [10] in Combat Effectiveness Simulation Systems (CESS). Based on this work, a common framework-based domain-specific composable modeling method is proposed by Li et al. [11] to handle composable modeling and multi-domain modeling requirements. Both of them established the ground for effectiveness simulation modeling. They handle the encountered complexity to a large extent. However, their work on cognitive modeling is based on EMF which belongs to the kind of heavyweight metamodeling method. Thus, there is the issue of how to establish an interface between the various sub-models which are represented by different customized languages.

To address the integration problems and reducing maintenance cost of various DSLs, many modelers tend to select the lightweight metamodeling method since all languages share a common UML syntactic and semantic foundation. Also, they can benefit from the existing available UML resources. Currently, there are several main standard profiles released by Object Management Group (OMG) in which some concepts can also be used in effectiveness simulation systems.

OMG defines that "SysML is a general-purpose graphical modeling language for specifying, analyzing, designing, and verifying complex systems that may include hardware, software, information, personnel, procedures, and facilities"[12]. We can also view SysML as a domain specific language since it is defined as an extension of a subset of UML using the UML profiling mechanism. Recently, with a semantic foundation of graphical representations, the language has been successfully applied in complex systems for modeling system requirements, behavior, structure, and parameters. There is also another profile extended from a subset of SysML for specific modeling requirements, namely the SafeML [13] and its supporting tool for modeling safety information for designing safety-critical systems.

To enhance the UML structuring and behavioral expression power, the UML SPT profile focuses on capturing the concept of Schedulability, Performance and Time (SPT) [14]. In particular, for the real-time systems domain which has strong timing constraints, this profile is progressively gaining popularity as it provides a set of stereotypes and tagged values to model quality of service, resource, time, and concurrency concepts and to support predictive quantitative analysis.

As an extension of SPT, the MARTE (Modeling and Analysis of Real Time and Embedded systems) profile [15] is intended to replace the existing UML-SPT Profile. In order to add more capabilities to specify real time and embedded systems, MARTE defines several packages such as Non-Functional Properties (NFPs), Generic Resource Modeling (GRM), Allocation Modeling (Alloc), Generic Component Modeling (GCM), and Generic Quantitative Analysis Modeling (GQAM) among others.

Table 1 summarizes the comparison of previously mentioned profiles with the AST profile for the capability of specifying cognitive behaviors. From the comparison, AST profile designed directly using the AST domain concepts therefore showing higher capability in serving for this specific domain.

Table 1: Comparison of previous profiles with the AST Profile

Name	Specialty	Capability
SysML	Requirements/Behavior/..	medium
SafeML	Safety-critical Systems	medium
SPT	Schedulability/Performance/..	low
MARTE	NFPs/ GRM/ Alloc/..	low
AST	Cognitive behavior	high

In fact, the existing profiles do not include many of the concepts in the cognitive domain. On the other hand, they come along with definitions that are not needed and may impose

needless complexity. Therefore, our contribution lies significantly in providing a well-suited profile for the domain of interest. To deliver a better solution, the domain concepts, such as tactics, need to be handled at both the abstract as well as the concrete syntactic levels. For instance, we define a set of domain specific syntactic constructs in order to facilitate the decision-making process for the military professionals. These constructs are tailored at the different levels of abstraction to be suitable for the selected domain. Specific notations are created to provide better and consistent communication with the domain experts.

3 THE UNIFYING FRAMEWORK FOR DESIGNING COGNITIVE DOMAIN SPECIFIC PROFILE

Currently, there have been little guidelines applied to define a domain specific language for AST. Consequently, many profiles can be either invalid because they are conflict with the standard UML principles, or incomprehensive because they do not adequately capture necessary domain knowledge. Therefore overcoming these issues is a key question that an effective framework needs to answer for defining well-formed and flexible profiles. It requires not only the domain expertise but also proficiency in UML modeling.

In this section, we progressively present our unifying framework for designing cognitive domain specific profile. We believe that the framework proposed to develop the profile in this research can be adapted to produce technically correct quality profiles in other domains. As shown in Fig. 2, we follow the Meta-Object Facility (MOF) four-layer metamodeling architecture and identify its language engineering and model implementation parts. The framework is illustrated by constructing an AST profile and demonstrated by an example (see Section 4).

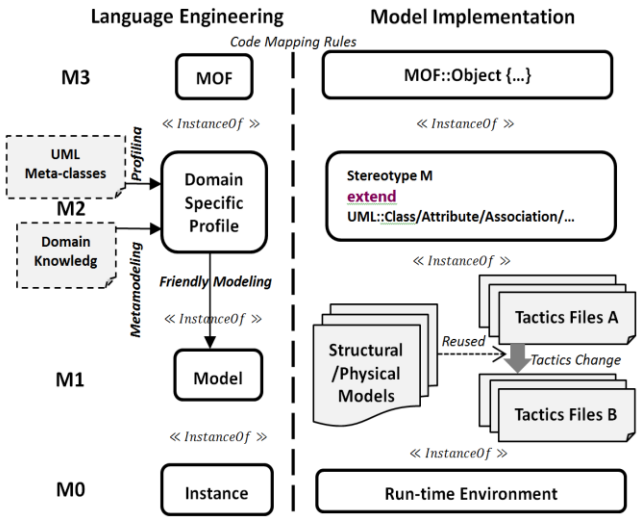

Figure 2: The unifying framework for designing cognitive domain specific profile

3.1 Meta-Object Facility

MOF is a strict four-layer metamodeling architecture in which every model element on every layer is strictly in correspondence with a model element of the layer above [16]. M3-layer defines the language for describing metamodels. The models in this layer, called meta-metamodels, conform to themselves. M2-layer defines the language for specifying a model. Its models, called metamodels, are instances of a meta-metamodel. The logic can be extended to M1- and M0-layer, too. M1-layer defines a language to describe the real world and its model is an instance of metamodel. M0-layer defines the real world and its model is an instance of a model. The four-layer conceptual architecture creates an infrastructure for customizing a new modeling language or making future language extensions. In general, it is widely used as a principle by the language engineering community.

3.2 Language Engineering based on UML Profile

To understand any enterprise, it is necessary to analyze the process: which activities are preformed, what entities are operated on, and what the casual relationships are [17]. In this framework, the language engineering defines three distinct but closely related activities. First, the metamodeling activity specifies domain knowledge then outputs the domain specific metamodel which consists of the fundamental language constructs, relationships, and constraints. Second, the profiling activity identifies the most suitable UML metaclass which is semantically closest to the semantics of domain concepts. Third, the friendly-modeling activity applies the domain specific profile to represent the concrete cases.

3.2.1 *Metamodeling*. The process of language engineering commences with the initial definition of the domain model, which is the purpose of the metamodeling activity. Metamodeling is the explicit modeling to capture the abstract syntax of a class of models, i.e., of a modeling language [18]. A domain model is the specification of what needs to be represented and how. In general, a domain model is a metamodel of the DSL that should include a set of fundamental language constructs that represent the essential domain concepts, a set of valid relationships that exist between the domain concepts, a set of constraints that govern how the languages constructs can be combined to produce valid models, the concrete syntax or notation of the language, and the semantics or meaning of the language [19].

It is not an easy task to define the above key elements because they become woven together when defining such a domain model. Moreover, the domain model is related with the subsequent activity profiling. To get a valid profile of good quality, it is of importance to identify some experience with defining profiles that will avoid some of the common pitfalls.

Firstly, specify the domain model without any consideration of UML metamodels.

This experience aims to acquire an unpolluted specification of domain model being designed. To achieve such a domain model, it is necessary to isolate domain metamodeling from the UML profiling. With such a separation of concerns, domain experts and modelers can concentrate on their domains of interest based on their specialties.

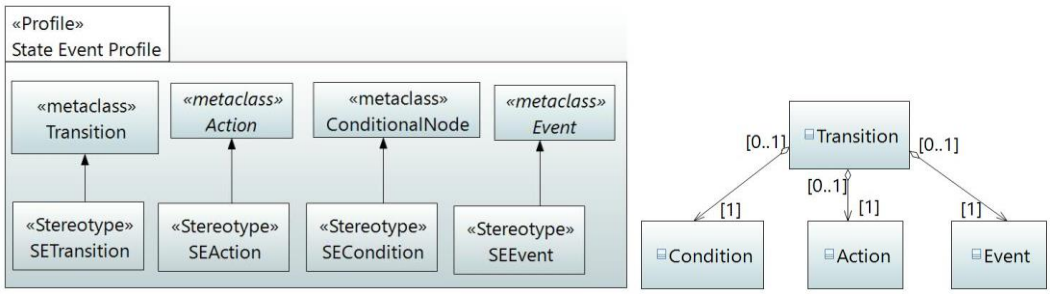

Figure 3: Part of the state event metamodel by fitting domain elements one by one

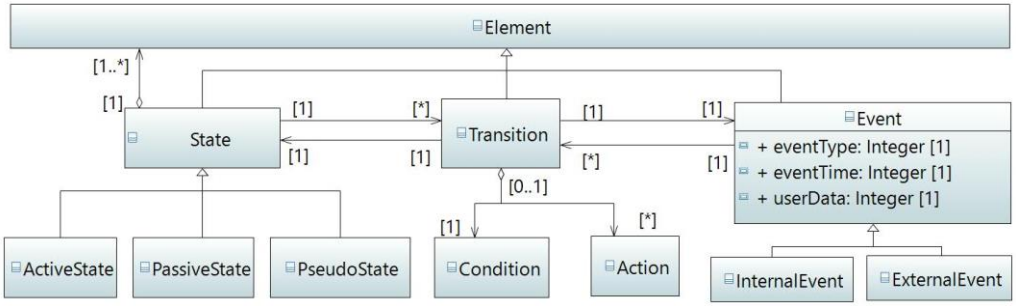

Figure 4: The state event metamodel without any consideration of the standard UML

Unfortunately, the development of far too many profiles tends to map a domain concept or relationship into the UML metaclasses once they are identified. At first glance, it seems to be reasonable because the light weighted profile-based method conforms to the standard UML metamodels. But a further examination reveals that the domain model is of poor quality to represent the domain knowledge even though the standard UML metamodel is well satisfied.

To illustrate such a situation, consider a simple extension of the Statecharts formalism for adding an explicit specification with regard to event. In this example, we want the event to be an independent element which has itself graphical concrete syntax, attributes and operations.

If we seek a well aligned UML metaclass once a domain concept is identified, shown in the left part of Fig. 3, the resulting profile seems an "ideal" domain specific language because its semantics keeps a maximum similarity with the standard UML. But unfortunately the implied domain model, shown in the right part of Fig. 3, is not the best conceptualization of the domain definitions or at least it can be significantly enhanced.

Thus, without any consideration of the standard UML metamodels at this stage, we concentrate on domain modeling and focus on the new framework in terms of its capability to provide a valid representation from a purely domain point of view. Except the basis of Statecharts formalism that includes "State", "Transition", "Condition", "Action", and "PseudoState", an additional event specification that consists of "Event", "Internal Event", and "External Event" as well as some affiliated attributes are added. This metamodel, shown in Fig. 4, can be viewed as a class diagram.

Unlike the shared relationship between the element "Transition" and the element "Event" shown in Fig. 3, transition connects states and events, indicated by the directed relations. Each event may have 0 or more transitions exiting it, as marked by the [*] cardinality. Similarly, each state may have [*] transitions entering it. On the other hand, a transition can exit from one and only one state and enter one and only one other, indicated by the [1] cardinality. In addition, Event has itself attributes such as "eventType", "eventTime", and "userDate". As such, the state event metamodel of Fig. 4 should be able to capture the domain characteristics of what we require originally.

Secondly, adjust the domain model with some loss of expressiveness if conflict occurs.

Once the domain model is completed, it does not mean that we can proceed to the subsequent work without any change of the domain model. In practice, the domain model may have its domain specific constraints, attributes, and relationships which are in conflict with the standard UML metamodels. Consequently, we cannot always map the domain concepts precisely to the UML metaclasses. As a result, we need to adjust the domain model with some loss of expressiveness if conflict occurs.

Take the similar example discussed above, the state event metamodel is an ideal model since it is purely defined by fitting elements one by one as mentioned above. In other words, the profile based on the metamodel is perfectly able to represent the domain knowledge. However, the relation between the element "Event" and the element "Transition" does not exist in the standard state machine metamodel as the domain requires. In order to comply with the standard UML rules, we may delete the relation and generate event from the element "PseudoState".

Although the conformance issues are addressed by adjusting the state event metamodel at the cost of expressive power, the profile strays too far from the ideal because event and state are distinct. Therefore, it may be the case that the light weight profile-based method is inappropriate for developing such a domain specific language, and the heavy weight metamodeling should be adopted. When that happens, the advantage is that the domain model established from the light weighted method has been created considering other extension mechanisms. Therefore, it can be reused in the heavy weighted method.

Thirdly, check the domain model to reduce the complexity of constraints.

The complexity in the domain model lies in two related perspectives. The model needs to address the domain experts' requirements while complying with the UML metamodel. A compromise must be worked out in a case of a conflict that may arise between the domain experts' needs and the technical modelers. The design of the metamodel can significantly impact the subsequent activities such as instantiation and constraint formulation. The readability of the created metamodel is also an important aspect. The computational speed of enforcing a constraint relies on the used algorithm for formulating that constraint which depends heavily on the constituted structure by the metamodel. In some cases, the domain specific constraints can become unreadable or even cannot be represented in OCL.

Consider the state event metamodel example shown in Fig. 4. The entity "ActiveState" and the entity "StaticState" are both generated from the common entity "State" to relate with each other through the entity "Transition". Referring to the principle one of activity cycle diagram (ACD) [20], the active state and the static state should alternate. In other words, the similar type of entities cannot be connected continuously with each other. OCL describes such a domain specific constraint as follows:

context *Transition*
Inv: *Alternation*
self.*stateSource->oclIsTypeOf(ActiveState)* **implies**
self.*transitionTarget->oclIsTypeOf(StaticState)*

The above constraint can be formulated simpler if we make a change on the structure of the metamodel. As shown in Fig. 5, we add an enumeration "stateStatus" which has two enumeration literal "ACTIVE" and "STATIC" to represent the different status of the entity "State" which has an attribute "status" that is typed by the enumeration. Thus, the two deriving entities "ActiveState" and "StaticState" as well as their generated relations are all deleted (see Fig. 4). Therefore, the structure of domain model has changed. When describing the alternating relationship between static and active states as mentioned above, OCL describes it as shown in the bottom note of Fig. 5 with the same context and the same constraint name. The function "oclIsTypeOf()" is replaced by the direct attribute assignment to further improve the time for examining the constraint hence an additional data is being stored which is the state status.

3.2.2 Profiling. Once the metamodeling activity is completed, the second activity named profiling can start. This is an activity of mapping the domain concepts to the most suitable UML base classes. There are also several experiences which can be viewed as guidelines to find a most suitable UML base class for each domain concept.

Firstly, the UML base classes should be selected semantically similar to the domain concepts. This is important for new stereotypes to reuse the existing UML tools if their semantics are most closely related. In this case, less domain specific constraints are needed or at least they can become simpler to be described.

Furthermore, not all the stereotypes are from the UML base classes. Instead, some of them can be constructed by generation from super stereotypes. For example, the stereotype "ActiveState" and "StaticState" can be directly generated from the super stereotype "State". It is not necessary for them to be extended from the base UML class "State" again.

Lastly, the selected UML base classes are not always well aligned with the domain concepts. They can be contradictory. When this happens, special care must be taken to add proper constraints to resolve conflicts. For instance, we can add a constraint of forcing the cardinality to be 0 for eliminating a relationship connecting some entities in the UML metaclass. Note that if the constraint is described poorly or may be unreadable across many entities, we need to consider another metaclass to be extended.

3.2.3 User Friendly-Modeling. With the customized profile, one can model by using the language constructs that they are familiar with. User friendly-modeling is not only a benefit from using domain specific modeling, but also a mean in which we check whether the profile is qualified to represent the real world or not. In practice, we often find some practical modeling issues when using the domain specific language. This is inevitable because the domain specific rules are always discovered in practice even though domain experts have very professional domain knowledge. Thanks to the unifying framework for developing UML based domain specific profile, we can more easily add the new discovering domain specific rules and clearly renew the language based on the previous one.

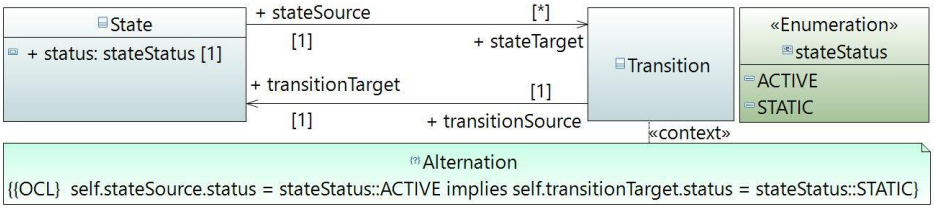

Figure 5: Improve the computing speed of constraints by changing the structure of domain model

3.3 Model Implementation

The remaining model implementation of the unifying framework is divided into simulation models and tactics files to emphasize their decoupling [21]. Through the code mapping rules, elements of language engineering can be mapped into the code snippets of model implementation. In M2- and M1 layers, model implementation is divided into simulation models (e.g., structural and physical models) and cognitive models (e.g., tactics files) with the fixed pre-defined interfaces. With such a separation, simulation models call the pre-defined interfaces to connect cognitive models, thus they do not need to be recompiled when cognitive models change. In M3- and M0 layers, model implementation has the same elements to instantiate the modeling language from MOF, and to create the run-time instances in the run-time environment, respectively.

The demonstrating example is about an armed escort which applies various anti-submarine tactics when an adversary submarine is possibly cruising around the escort route. Thanks to the proposed framework, we develop the case gradually from metamodeling to profiling to friendly-modeling, which prevent us from getting into those pitfalls stated earlier. Furthermore, we use OCL to describe the constraints on the metamodel and profile elements for specifying well-formedness rules.

4 AST PROFILE UPON THE FRAMEWORK

In this section, we demonstrate our approach to achieve the unifying framework for modeling AST. We analyze the AST domain concepts in conjunction with the construction of their realizations according the framework. We argue that the framework provides a capability to capture the cognitive behavior embedded in the tactics in a disciplined manner. Accompanied with provided extensibility, the framework is likely to accommodate changes even when encountering complexity.

4.1 Domain Conceptual Analysis

Tactics are *"the employment and ordered arrangement of forces in relation to each other, also the procedures, techniques"* [22]. Generally, Anti-Submarine Warfare (ASW) process includes three phases search, attack, and evasion and in each phase there are various corresponding AST tactics.

The best defense is certainly to stay undetected. Once detected, the best way of survival is to break contact as soon as possible [23]. To achieve the goal, there are usually two ways to disappear from the enemy's vision. One is active evasive maneuver. The other is to employ passive deceptive countermeasures such as decoy and jammer (the "soft kill" method). A successful anti-submarine tactic depends on close-cooperation of these two maneuvers.

When it comes to the effective tactic to stay undetected, keeping the platform going in the baffles of the enemy should be safe. For a submarine's aft (shown as shaded part in Fig. 6), its baffles are a cone about 15 degrees wide extending aft from the stern, underwater disrupted by the water environment and the submarine itself. In such a blind area, the sonar is ineffective, thus this tactic enable a platform to track and trail a target as well as remain undetected from the enemy. However, it is not always safe to stay in the baffles of the enemy, since there are tactics against it. The tracking platform may suddenly make 90 to 360 degree turns to listen its former baffles, e.g. the famous "Crazy Ivan" [24].

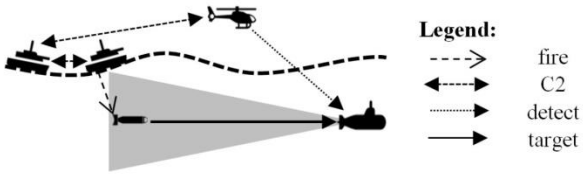

Figure 6: Tactic to stay undetected in the baffles of the adversary submarine

When detected, the platform begins an evasive process which consists of a set of actions. It launches the decoy with the direction perpendicular to the bearing of the target, and turns to the opposite direction at maximum speed to get the maximum evasive time while the enemy runs to the decoy. Considering the maximum turning radius and decrease in acceleration, it is of importance for the platform not to turn more than necessary. So in order to always turn fewer degrees, the decoy is usually launched on a course into the two aft quadrants of the platform. As shown in Fig. 7, the incoming submarine is to the starboard side of the surface ship from the third quadrant. The surface ship fires the decoy into the second quadrant, perpendicular to the bearing of the submarine. Then, the surface ship takes a turn-angle θ into the fourth quadrant to evade by accelerating to maximum speed. More details about the classic evasive tactic can be found in [25, 26]. Meanwhile, it deploys two pairs of jammers with an interval of 14 seconds, forming a cluster of noise around the platform so that the target might lose contact if already detected.

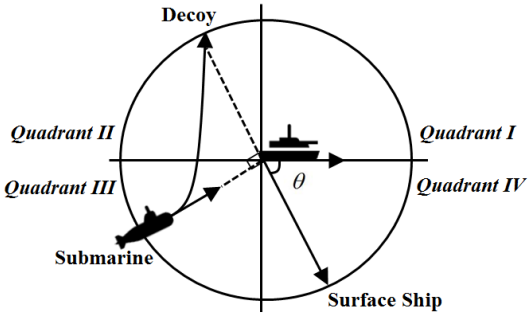

Figure 7: Tactic to launch the decoy when detected

4.2 Cognitive Behavior Metamodeling

All of these factors, such as the received information, categories of threats, commanders' abilities, cultural traditions, etc., should be considered in the process of cognitive modeling. To capture the overall process that human beings or organizations use to think, grow, and thrive in a rapidly changing environment, OODA (Observe-Orient-Decide-Act) [27], as a methodology for explicitly specifying ambiguity and uncertainty, is well-suited to represent human cognitive behaviors.

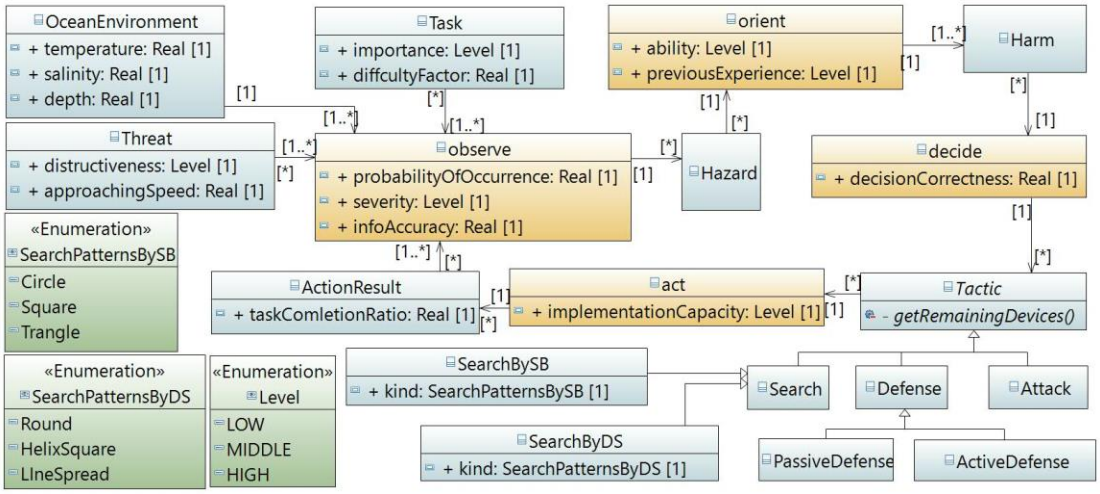

Figure 8: The AST cognitive metamodel

Fig. 8 depicts the AST cognitive metamodel, which is organized into three kinds of elements. The first deals with observe, orient, decide, and act for the OODA loop. The second deals with task, ocean environment, threat, hazard, harm, tactic, and action result for the loop inputs or outputs. The last include three kinds of enumerations: search patterns using sonobuoys, search patterns using dipping sonar, and level. Note tactic is abstract and it is inherited by search, defense, and attack according to the tactical phased process as depicted earlier.

As for the well-formedness rules, OCL, as a declarative language that can be used to define model constraints in terms of predicate logic, is adopted to describe the constraints on elements of the AST cognitive metamodel. A constraint is defined as [28]: "A constraint is a restriction on one or more values of (part of) an object-oriented model or system."

In general, language has syntax and semantics. Syntax has an abstract syntax and a concrete syntax. The abstract syntax defines the set of syntactically correct models which describe the concepts of modeling language, relationships that may exist among the concepts, and well formedness rules specifying how the concepts can be combined. The concrete syntax gives notations for constructing and presenting models. Notations can be textual and/or visual. The textual syntax uses constructed text to represent models, such as XML. The visual syntax represents the language diagrammatically, which is used in this research. While, semantics, describing the meaning of the language, has also two types: static and dynamic. Static semantics defines the well formedness of the constructs in the language in the form of invariant conditions that must hold for any model created using the language. Dynamic semantics then is the interpretation of a given set of constructs in the specific context of model instances themselves, and this is the reason why dynamic semantics cannot be specified in metamodels [16].

Accordingly, we define the domain specific constraints on the cognitive metamodel mainly from two aspects. One is the syntactical constraints that ensure the correctness of attributes, entities, and the structural well formedness rules.

Consider such two possible syntactic constraints on the attribute "probabilityOfOccurence" of the entity "observe", and its cardinality related to entity "Task". As stated using English, might be:

First, the value range of threat occurrence probability has a specific level of severity. Second, one observer can be assigned a maximum of 10 tasks. OCL specifies them as follows:

```
context observe
Inv: AttributeCompatible
probabilityOfOccurrence >= 0.8 implies severity = Level::HIGH and
probabilityOfOccurrence >= 0.5 and probabilityOfOccurrence < 0.8
implies severity = Level::MIDDLE and
probabilityOfOccurrence >= 0.0 and probabilityOfOccurrence < 0.5
implies severity = Level::LOW

context observe
Inv: TaskCapability
self.Task.allInstances()->size() <= 10
```

The other aspect is semantic constraints which specify meaningful constraints and ensures the language satisfies them [29]. We know that a good tactic is limited to the available devices. Suppose during a combat the number of the available sonobouys is not enough to decide one of the search patterns using sonobouys. Such a constraint can be specified by OCL as:

```
context SearchBySB
Inv: RemainingDevicesCons
getRemainingDevices() < 1 implies kind <>
SearchPatternsBySB::Circle
```

Many constraints are very complex and not straightforward to express. Note that one key principal for reducing the complexity of constraints is that the models to be restricted should be carefully designed, if not, it would be difficult to define useful constraints.

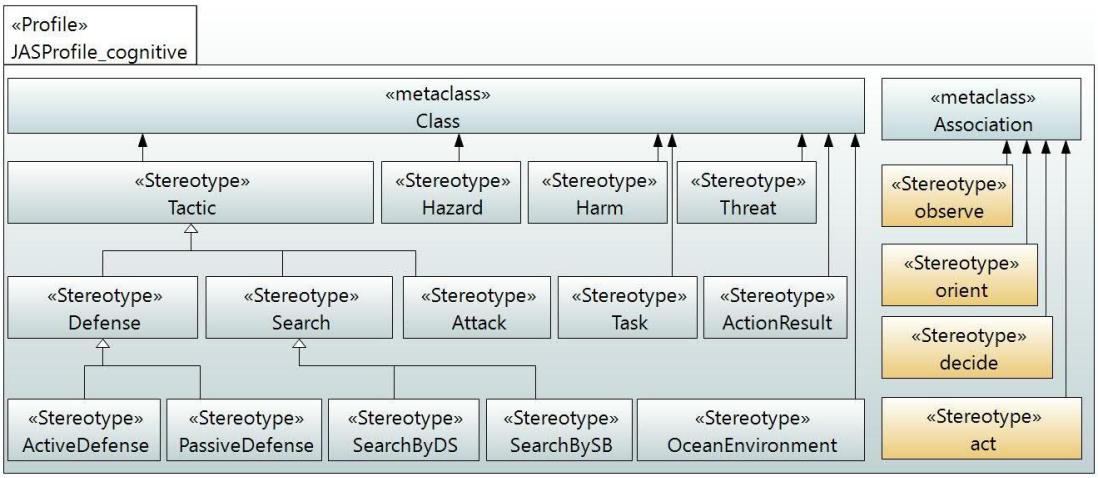

Figure 9: The cognitive profile

4.3 Profiling based on Domain Model

Profiling is a process of mapping domain model elements to UML metaclasses that requires them to be semantically aligned. The more semantic similarity, the more features could be reused, and the less resulting complexity for the language. From this perspective, the mapping is not a one-time process, it is a selection which requires further investigation if the current one might not be the most appropriate one. There are mainly two categories of mapping extensions for AST profiling, as shown in Fig.9.

In the first category, all of the inputs and outputs of OODA (Observe-Orient-Decide-Act) loop are directly or indirectly mapped to the UML "class". In UML, "a class describes a set of objects that share the same specifications of features, constraints, and semantics" [30]. This is closer to the meaning of the loop inputs or outputs. Take "Tactic" as an example, which is basically defined as such a logical entity that generally has similar employment and ordered arrangement of forces, also the procedures and techniques. Consequently we select the UML base metaclass "Class" to be extended by the stereotype "Tactic" which is generated by stereotypes "Defense", "Search", and "Attack". They are further generated by stereotypes "ActiveDefense", "SearchByDS", etc.. Similar to the rest of elements, such as "Hazard", "Harm", and "Task", etc., tactics are all mapped to the same metaclass "Class". We note that not all the stereotypes are designed by only extending the UML metaclasses. In fact, we can also construct stereotypes by the inheritance from another stereotype based on their semantics. In addition, based on the AST cognitive metamodel as shown in Fig. 8, a stereotype may have tags which can strengthen its expressiveness, which are not shown in Fig. 9 and detailed here for brevity.

In the second category, the four nodes in the OODA loop are all mapped to the UML "Association" and detailed in Table 2. Similar to above descriptions for simplicity, Fig. 9 just shows their mapping relationships not the tagged values.

Table 2: The stereotypes for the relationships defined in AST Profile.

Stereotype	Generation	Description
observe	*metaclass Association*	The relationship between $\ll OceanEnvironment \gg$, $\ll Threat \gg$, $\ll Task \gg$, $\ll ActionResult \gg$ and $\ll Hazard \gg$.
orient	*metaclass Association*	The relationship between $\ll Hazard \gg$ and $\ll Harm \gg$.
decide	*metaclass Association*	The relationship between $\ll Harm \gg$ and $\ll Tactic \gg$.
act	*metaclass Association*	The relationship between $\ll Tactic \gg$ and $\ll ActionResult \gg$.

Designing such a profile not only requires abstractions of aspects such as human activities (*observe, orient, decide, and act*), tactics (*active defense, passive defense, search using dipping sonar, and search using sonobuoys*), the surrounding environment *(the coming threat, the ocean environment, task, and the result of action*), and the security (*harm and hazard*), but also a flexible framework which can account for external changes. In other words, if the modeling requirements need to be updated under new situations, we just need to modify the concrete implementations not the framework. So the cognitive behavior framework should be able to represent the common domain knowledge, and for the concrete implementations we just need to do direct instantiation or simple extensions.

In order to ensure the consistency of the profile with the semantics of the domain model, a set of constraints need to be defined.

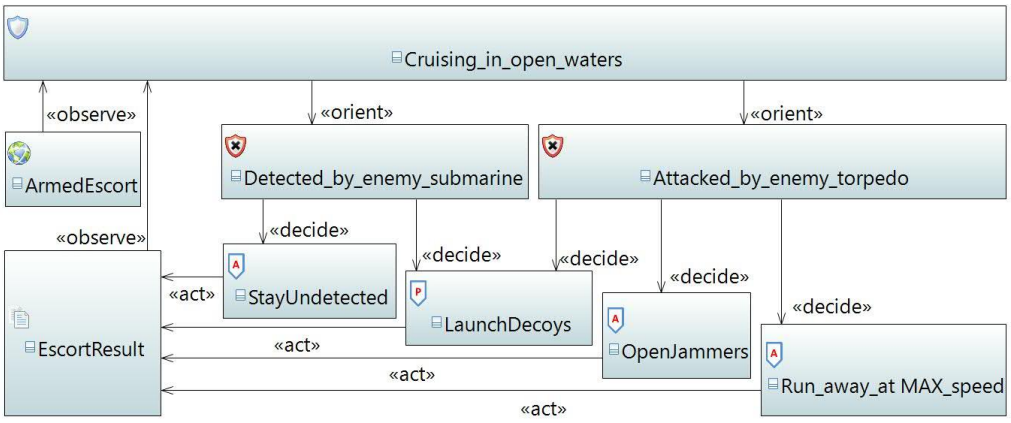

Figure 10: The armed escort model

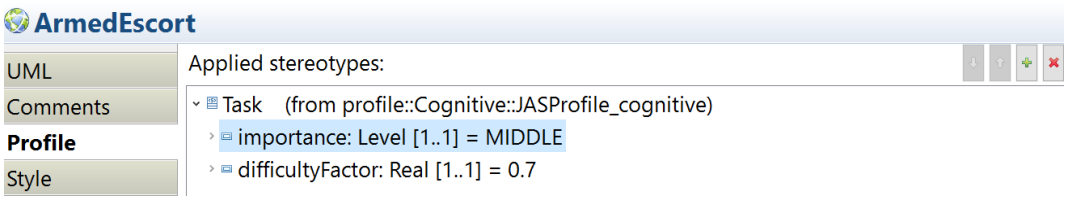

Figure 11: A snapshot of tagged values for the "ArmedEscort" task

In the context of "Hazard", consider such an invariable named "TaskMustBeAssigned" which represents that one task is assigned for any hazard at least. This is because the hazard occurs on the premise of conducting some tasks. OCL defines this constraint as:

context *Hazard*
inv: *TaskMustBeAssigned*
not self.*base_Class.ownedAttribute.association.memberEnd.class.*
getAppliedStereotypes() -> *select*(s | *s.name* = 'Task') -> *isEmpty*()

4.4 User Friendly-Modeling Using AST Profile

Fig. 10 shows the armed escort model. We can identify the following hazards when assigned the escort task, harms while taking these hazardous actions, and the corresponding tactics if those harms occur.

The task of armed escort, if assigned, is assumed to be conducted through cruising in open waters, which may cause harms such as being detected by enemy submarine, or even attacked by enemy torpedo. To survive in such situations, various AST tactics stated in domain conceptual analysis can be applied to disturb the threat's vision or even destroy the enemy. For instance, staying in the blind area of the adversary submarine, or countermeasures such as decoys and jammers should be launched to deduce or deny the enemy's detection. If no decoys and jammers are available, run away at the maximum speed to evade the enemy's detection range. Meanwhile, available weapons could be launched if possible.

We do not show all the tagged values to improve readability, and just present the snapshot of tagged values for the "ArmedEscort" task as shown in Fig. 11. The task has two tagged values "importance" that is assumed to be "MIDDLE", which represents that the task is not an really urgent one, and the "difficultyFactor" that is set to "0.7", which indicates that the task is considerably not easy to be accomplished. Thus, many of cognitive domain characteristics that cannot be specified by general purpose modeling languages are well-defined by using AST profile.

5 CONCLUSIONS

We presented a new domain-specific modeling language for specifying Anti-Submarine Tactics aiming to significantly reduce the communication gap between domain experts and simulation modelers. Using this profile, one can not only represent various underwater tactics by directly using domain specific concepts, relationships, and constraints with which they are very familiar, but also able to benefit from the standard UML design notations and existing tools. To achieve this goal, a unifying framework is proposed to guide how to develop the cognitive domain specific profile based on UML domain-specific language design and development. Following this unifying framework, a case study using the AST profile is discussed to highlight the overall development process from metamodeling to profiling and friendly-modeling. Meanwhile, Object Constraint Language (OCL) is applied to capture the domain specific constraints at earlier phases, i.e., metamodeling and profiling help reduce errors that often reveal themselves in latter stages of development including testing and evaluation.

The profile has been shown to represent various aspects of anti-submarine tactics such as tasks, hazards, harms, tactics, results, and each node in typical OODA lifecycle. The development of the AST profile has been also shown from initial domain specific concepts to concrete models implementation. However, the current effort still needs further evaluations from domain experts to promote the model syntax and operational semantics consistency. Thankfully, we account for the future changes and enhancement via the framework presented in this paper. As a result, incorporating such changes is made simpler and disciplined. The future work concentrates on the accessibility of the AST profile in effectiveness simulation domain by making it more complete and understandable.

ACKNOWLEDGMENTS

We are grateful to the anonymous referees for their helpful reviews, and all the volunteers who wrote and provided helpful comments on previous versions of this document. We gratefully acknowledge the National Natural Science Foundation of China (NSFC) (NO. 61273198) for supporting this research.

REFERENCES

[1] M.J. Son, D.Y. Cho et al. 2010. Modeling and simulation of target motion analysis for a submarine using a script-based tactics manager. *Advances in Engineering Software* 41, 506-516.

[2] J. Miller and J. Mukerji. 2003. *MDA Guide Version 1.0.1.* Retrieved from http://www.omg.org/omg/2003-06-01

[3] B. Selic and S. Gerard. 2014. An introduction to UML profiles. In *Proceedings of Modeling and Analysis of Real-Time and Embedded Systems with UML and MARTE,* 27-43.

[4] Eclipse Foundation. 2014. *Eclipse Modeling Framework* (EMF). Retrieved from http://www.eclipse.org/modeling/emf/

[5] S. Gerard. 2011. *Papyrus User Guide Series about UML profiling, version 1.0.0.* Retrieved from http://www.eclipse.org/papyrus/resources/PapyrusUserGuideSeries_AboutUMLProfile_v1.0.0_d20120606.pdf

[6] Y.L. Lei, Q. Li, F. Yang et al. 2013. A composable modeling framework for weapon systems effectiveness simulation. *System Engineering-Theory & Practice* 33, 11, 2954-2966.

[7] Y.L. Lei, N. Zhu, J. Yao, Z. Zhu, and H. S. Sarjoughian. 2016. Model-architecture oriented combat system effectiveness simulation. In *Proceedings of Winter Simulation Conference,* 3190-3191. DOI: 10.1109/WSC.2015.7408464

[8] H.S. Sarjoughian. 2006. Model Composability. In *Proceedings of Winter Simulation Conference*. 149-158.

[9] S. Kelly and J.P. Tolvanen. 2008. *Domain-Specific Modeling: Enabling Full Code Generation.* Wiley-IEEE Society Press, New York, NY.

[10] W.G. Wang, W.P. Wang, F. Yang, and Y.F. Zhu. 2010. Service-Oriented Simulation Framework: An Overview and Unifying Methodology. *Simulation* 87, 3, 221-252.

[11] X.B. Li, Y.L. Lei et al. 2013. Domain-specific decision modelling and statistical analysis for combat system effectiveness simulation. *Journal of Statistical Computation and Simulation*, 1-19.

[12] Object Management Group. 2010. *SysML specification, version 1.2 formal/10-06-02.* Retrieved from http://www.sysml.org/specs.htm2010.

[13] G. Biggs, T. Sakamoto and T. Kotoku. 2016. A profile and tool for modeling safety information with design information in SysML. *Software System Model* 15, 147-178

[14] Object Management Group. 2005. *UML profile for schedulability, performance, and time specification, version 1.1, formal/2005-01-02.* Retrieved from http://www.omg.org/spec/SPTP/1.1/2005

[15] Object Management Group. 2011. *A UML profile for MARTE specification, version 1.1, formal/2011-06-02.* Retrieved from http://www.omg.org/spec/MARTE/1.1/PDF.

[16] G. Nordstrom, J. Sztipanovits, G. Karsai, and A. Ledeczi. 1999 April. Metamodeling - Rapid Design and Evolution of Domain-Specific Modeling Environments. In *Proceedings of IEEE ECB'99 Conference.* Nashville, Tennessee, 68-74.

[17] H. Vangheluwe. 2000. *Multi-Formalism Modeling and Simulation.* Ph.D. Dissertation. Gent University, Gent, Belgium.

[18] P.J. Mosterman and H. Vangheluwe. 2004 September. Computer Automated Multi-Paradigm Modeling: An Introduction. *Simulation* 80, 9, 433-450.

[19] B. Selic. 2007. A Systematic Approach to Domain-Specific Language Design Using UML. In *Proceedings of 10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing* (ISORC).

[20] W.P. Wang et al. 2007. *Discrete Event System Modeling* (2nd. Ed.). Science Press, Beijing, China.

[21] W.B. Frakes and K. Kang. 2005. Software Reuse Research: Status and Future. *IEEE Transactions on Software Engineering* 31, 7, 529-536.

[22] Joint Doctrine Division. 2011 April. *DOD Dictionary of Military and Associated Terms*. Joint Publication 1-02.

[23] N.R. Bourassa. 1993. *Modeling and Simulation of Fleet Air Defense Systems Using EADSIM*. Master's thesis. Naval Postgraduate School, Monterey, CA.

[24] Watts and J. Anthony. 1999. *Jane's Underwater Warfare System* (11th. Ed.). Jane's Information Group Limited, UK.

[25] K. R. Armo. 2000. *The relationship between a submarine's maximum speed and its evasive capability.* Master's thesis. Naval Postgraduate School, Monterey, CA.

[26] K.H. Liang and K.M. Wang. 2006. Using simulation and evolutionary algorithms to evaluate the design of mix strategies of decoy and jammers in anti-torpedo tactics. In *Proceedings of Winter Simulation Conference*. 1299-1306.

[27] J.R. Boyd. 1987. *A Discourse on Winning and Losing* (Unpublished briefing slides). Air University Library, Maxwell AFB, AL.

[28] J. Warmer and A. Kleppe. 1999. *The Object Constraint Language-Precise Modeling with UML.* Addison-Wesley, Boston, MA.

[29] M.D. Petty and E.W. Weisel. 2003. *A Formal Basis for a Theory of Semantic Composability.* Spring Simulation Interoperability Workshop, 03S-SIW-054.

[30] Object Management Group. 2012. *Unified Modeling Language (UML) superstructure, version 2.4.1, formal/2011-08-06.* Retrieved from http://www.omg.org/spec/UML/2.4.1/Superstructure