Proceedings of the 2017 Winter Simulation Conference
W. K. V. Chan, A. D'Ambrogio, G. Zacharewicz, N. Mustafee, G. Wainer, and E. Page, eds.

CONCEPTUAL FRAMEWORK FOR AN AUTOMATED BATTLE PLANNING SYSTEM IN COMBAT SIMULATIONS

Byron Harder
Curtis Blais
Imre Balogh

Modeling, Virtual Environments, and Simulation (MOVES) Institute
Naval Postgraduate School
1 University Circle
Monterey, CA 93943, USA

ABSTRACT

Automated planning is a key to unlocking the next generation of human behavior modeling for military simulations. Automated planning is distinguished from other dynamic behavior by its ability to reason with predictions or assumptions about future outcomes, which has traditionally been left to the meticulous effort of human modelers. In this paper, we present a conceptual planning framework as an architectural roadmap for the development of this kind of capability in support of modeling and simulation. This paper also presents an initial implementation of the framework in a representative combat simulation for exploring potential future opportunities for community advancement of the planning techniques.

1 INTRODUCTION

Automated planning is a key to unlocking the next generation of human behavior modeling for military simulations. Effective automated planning can speed up the scenario design process, but its potential goes beyond simple efficiency gains. A trustworthy automated planner enables the realization of an increased number of scenarios that could be evaluated for a study. For training simulations, it could reduce manpower requirements and help improve the realism of modeled units, allowing system operators to focus on higher levels of abstraction and better attend to the needs of the training audience. In general, combat models and simulations used in training and analysis do not have an automated planning capability for multiple units in a command structure, and they lack tools for forward reasoning about effects such as suppression by firepower. To address this, we present an architectural framework for automated battle planners for the design and management of complex planning systems, also providing an overview of a representative implementation. Full details are presented by Harder (2017).

2 AUTOMATED BATTLE PLANNING SYSTEM

Battle planning is a type of planning where the agents are units in a combat model, the initial contextual situation is a *scenario*, and the desired outcomes are *military objectives*. The latter may include *goals*, given as a logical combination of predicates evaluated against the world state, as well as numerical *objective functions* to be optimized. An *automated battle planning system* (ABPS) is a software system (or subsystem) that accepts a scenario and military objectives as input and produces a battle plan that is executable by designated units. This work is focused on battle planning for ground-based units operating in a simulated battlespace. The *effectiveness* of the ABPS is the degree to which execution of its plans are successful at achieving all military objectives. Effectiveness is just one part of an ABPS's *validity*.

Our ultimate objective is to produce realistic plans executable by units in a combat model implemented on a computer. Therefore, we take an artificial intelligence (AI) point of view to define planning as the formulation of a partially ordered set of instructions for a set of agents in an attempt to modify the world state from an initial configuration towards a desired outcome. This is a more general definition than that of classical state-space AI planning because we will need extensions such as plan-space, hierarchy, and continuous time. Harder (2017) provides an explanation about how the combat modeling domain violates several classical planning assumptions.

It is convenient to distinguish an ABPS, assuming we have one, from the rest of the executable model. To that end, we define a *combat simulation environment* (CSE) as an executable combat model, along with its supporting tools (i.e., for scenario design, behavior development, etc.), with the exclusion of any ABPSs. We assume that an ABPS is only useful if it produces battle plans that resemble those of human planners; i.e., an automated battle planner is a model of the activities of a human commander and, if appropriate, a planning staff. Development of an ABPS is a modeling and simulation (M&S) activity subject to the same control and evaluation processes as a combat model. Since this work is restricted to ABPS development, we assume that the CSE is a fixed version. This lets us narrow our focus for issues such as verification and validation to the ABPS. We introduce other terms related to ABPS development:

- *Planning Simuland:* What ABPs represent; i.e., real-world military and paramilitary commanders and planning staffs of tactical units, in the context of their development of battle plans.
- *Planning Referent:* What we use as the authoritative source to guide ABP development; i.e., doctrinal, training, and technical publications of U.S. and other nations' military organizations.
- *Conceptual Planner Model:* A collection of data structures, logical components, and algorithms used to model the plan-generating capability of commanders and planning staffs, written in some formal way (e.g., Unified Modeling Language).
- *Executable Planner Model,* or *ABPS:* An executable implementation of the conceptual planner model for a particular CSE.
- *Unit* and *command:* For combat models, a unit is a representation of one or more people and assigned equipment that work together on a shared task in close geographic proximity. A *command* is a collection of units in a hierarchical relationship.
- *Invocation:* A single execution of an ABPS planning capability, resulting in an executable battle plan (or failure to produce one).

Initial planning is the generation of the first plan for a set of units in a replication. *Replanning* is planning activity for units that already have a plan, including both modification and complete replacement. In theory, an ABPS could replan several times during a single replication. Depending on the design of the combat model, there could be more than one ABPS for a scenario (one for each of the opposing forces, for example). These variations demonstrate that the number of invocations of an ABPS is usually not equal to the number of replications of a particular scenario in simulation execution.

3 CONCEPTUAL PLANNING FRAMEWORK

An ABPS produces plans for use in a combat model. Harder (2017) reviews several examples of effective systems that fit this basic description; however, no single approach stands out as dominant, in part because the combat models they support have different purposes and structures. To proceed, we define some additional terms. An *architecture* is the organizational scheme of a specific executable model or part thereof. Architectures delineate the major components and subcomponents of implementations and the interfaces between them. A *framework* is an abstract architecture applicable to a variety of implementations. It distills the design principles, common themes, and vocabulary of a class of architectures. This paper describes a framework for ABPSs. We remain intentionally abstract about

implementation details. The goal is to establish concepts that apply regardless of certain modeling decisions. Harder (2017) describes a particular architecture (for a specific executable combat model) conformant to this framework. There, the author is specific about modeling choices, but for now we remain agnostic about the following: level of aggregation (whether the smallest modeled unit is a person, vehicle, team, squad, or some larger echelon); types of combat units and their capabilities; dynamic behavior models governing small-unit or individual entity decisions; specific instructions that can be issued to units and composed together in a plan; the way terrain is represented and its effect on combat outcomes; the model and resolution of time advancement; and details of the combat adjudication methodology (which may be deterministic or stochastic).

3.1 Top-Level Framework

Our conceptual planning framework is organized at the highest layer of abstraction into three principal components: Planning Data, Planning Input, and the Plan Generator, as shown in Figure 1, which also includes blocks for the executable plan (the output) and the corresponding CSE. The dashed line in the figure delineates the system boundary of the ABPS (or the subsystem boundary if it is packaged together with a CSE).

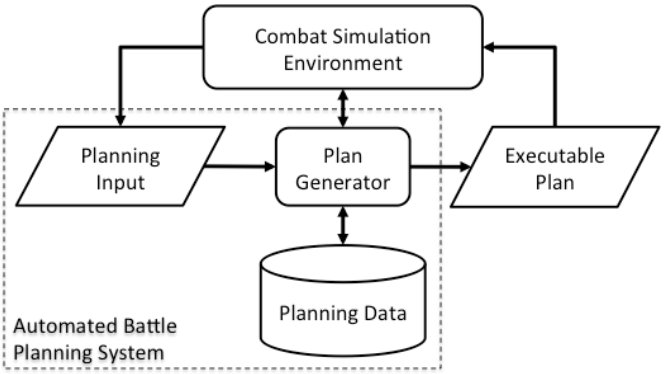

Figure 1: Top-level conceptual planning framework.

Planning Data includes tasks, methods (rules for achieving tasks), and derived models that characterize the structure and dynamics of the battlespace. Planning Input takes data from a scenario in the CSE, and from a user interface, to formulate a planning problem for the ABPS. The Plan Generator provides the algorithms, heuristics, data structures, and interfaces that produce the battle plans. The Plan Generator uses Planning Input, its internal data structures, algorithms, and tools, along with the rules and heuristics of the Planning Data, to produce a plan in its own internal representation. Finally, it translates this plan into a format that can be processed by the particular CSE during one or more replications.

When we speak of taking a *separation-of-duties* approach, it is to explicitly recognize that the Planning Data decouples military planning logic from more general AI-based algorithms found in the Plan Generator, in the same way that the developers of a planning software product (like a generic hierarchical task network (HTN) planner) are separate from the teams that use the product to produce plans for a specific domain. This separation is meant to allow distinct configuration control on these two components, which can be managed by different groups with different skill sets. This is similar to the video game development community where there is a distinction between "level" (scenario) designers and AI developers. A combat model with an ABPS may actually need a three-way separation: (1) *scenario designers*, who configure the CSE and use the ABPS; (2) *behavior developers*, who encode tactical planning logic within the architecture of an ABPS; and (3) *automated planning developers*, who are responsible for the architecture and algorithms of the ABPS. We need this additional silo because, unlike most video games, the scenario design and AI development for a combat model are often not together on

a single team. Scenario designers typically do not have the programming background needed to encode tactical planning logic in a programming language, so behavior developers with specialized skills are needed when the default Planning Data is insufficient. Finally, we distinguish automated planning developers from behavior developers because automated planning is an even more specialized skill than behavior programming.

The three components of the conceptual planning framework are described in the following subsections.

3.2 Planning Data

The Planning Data component of the conceptual planning framework is shown in Figure 2. This component represents a persistent data store intended to support a data-driven approach to the development of functionality. This component is meant to be an enterprise-managed database, not just a local user's data file, to support the sharing of products across organizations and to reduce overlapping and duplicated effort. Each conceptual component of the Planning Data (the Task Dictionary, for example) may be realized with one or more separate data stores. Ideally, everything in the Planning Data adheres to a single, seamless data model to more easily support sharing across components of the Plan Generator. Subcomponents of the Planning Data component are described in the following subsections.

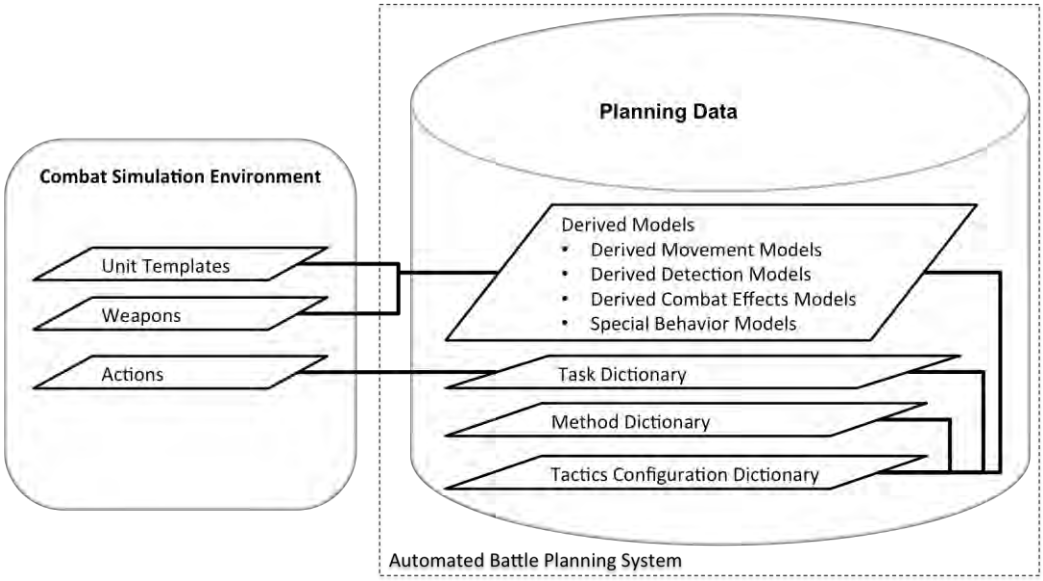

Figure 2: Planning data.

3.2.1 Units and Templates

Unit here refers to a software object that can be manipulated by the ABPS and corresponds to a particular combat unit instance in a scenario. Each unit has an assigned *echelon:* one of a discrete set of values such as squad, platoon, company, or battalion. A unit's echelon does not determine its command relationships; it is simply a clue to the level of abstraction that the ABPS needs to use during planning. A *unit template* is a configuration of the attributes of a unit with the exception of explicit planning rules or planning guidance. Every unit is an instance of a unit template. If units contain entities, then each unit template also includes a multiset of *entity templates*. Similarly, an *entity template* is a configuration of the parameter values for an entity. The unit template includes reactive behavior models (which we abbreviate to *behaviors* henceforth), which are not part of the planning framework but are considered by it.

Behaviors may be defined at the unit or entity level and may be built-in or configurable, depending on the CSE.

3.2.2 Commands and Hierarchies

S*ide* is defined as a set of units that are aligned together under a common commander or within the same alliance, coalition, or unification. A *force* is a subset of a side. Every unit is *commanded by* at most one other unit, called its *commanding unit*. The *commanded by* relationship is between individual units, not unit templates. There are different ways that combat models may interpret hierarchical unit organization, such as contingent aggregation, explicit command and control (C2) nodes, abstract C2 nodes, or a hybrid approach. The planner must account for the approach in order for its output to be meaningful for the CSE.

3.2.3 Actions

The *actions* of a CSE are instructions that may be included in the Executable Plan. Actions are issued to units and entities, and they result in a change to the state of the CSE. Typical actions include movement to a waypoint, switching engagement modes (i.e., from "only when fired upon" to "engage upon target detection"), firing on a designated target area, and preparing defensive positions.

3.2.4 Derived Models

In general, an ABPS requires *derived models* to estimate the effects of many highly detailed actions. These allow planning to occur at a higher and less computationally demanding level of abstraction. The derived models are the specific functions, algorithms, equations, and other tools that make up the conceptual world of the ABPS. The CSE is the external world; the real world is not even part of the planning framework. This can be a source of confusion, because the derived models are "models of the model" of the real world. The development of derived models is a characteristic and rather unique aspect of automated battle planning, compared to other combat modeling skills. For example, assume that some CSE has fine-grained equations to represent acceleration, effects of rough terrain on rate of movement, and realistic turning radii for each entity. It offers actions to instruct units to travel to a given waypoint, but the actual travel time depends on the frame-by-frame movement model, which is affected by the terrain type, number of turns, size and formation of the unit, and so on. Rather than use this detailed movement model to calculate paths or travel times for planning, a derived model could be a simpler, piecewise constant-velocity movement model with a small adjustment for unit size. If the simpler model can estimate the travel time within less than a second with high probability, say, then we have a derived movement model that is accurate enough for planning purposes and much less costly to calculate.

3.2.5 Task Dictionary

The Task Dictionary stores the set of *tasks* that can be assigned to units. We borrow the term *task* from HTN planning, but the framework does not necessarily require a task network representation or an HTN planning algorithm. Some of the tasks map to the CSE's actions, but the planner may augment the task language with other tasks. The tasks do not, by themselves, provide any guidance about when they *should* be used (see Method Dictionary in section 3.2.6), only when they *may* be used (preconditions). Parameters (within the allowed value ranges) allow behavior developers to write tasks that are applicable for a variety of different situations. They tend to reduce the number of different tasks needed at the price of greater complexity per task. Tasks may be *instantaneous* (a discrete value change at a single point in simulation time) or *enduring* (affecting the unit state in a certain way between initiation and termination). Although these two types are analogous to the events and persistence conditions of continuous-time planning, the framework does not stipulate a particular time representation.

3.2.6 Method Dictionary

The Method Dictionary conceptually groups all methods together to help explain other concepts such as method sets. A method is defined as an optional subroutine, function, or mapping that generates planning branches. Different types of methods may be present in the Method Dictionary if the Plan Generator uses multiple planning representations and algorithms. Each entry in the Method Dictionary may include a set of partial unit templates called its *applicable template set*. The *applicable method set* of a unit template is the set of all methods for which it matches a member of the applicable template set.

3.2.7 Tactics Configuration Dictionary

A *planning style* restricts individual units to the planning assumptions and tactics appropriate for their training, experience, motivation, goals, or national military culture, according to the scenario designer. A *tactics configuration* is a mapping from units to planning styles, or the combination of a method set, a task set, and a derived model set.. Every scenario that employs the ABPS must have a tactics configuration. A *standard tactics configuration* is a mapping from unit *templates* to method sets—in other words, it specifies a set of unit profiles. Standard tactics configurations are shortcuts for quickly generating or resetting the tactics configuration of a scenario. A scenario designer can avoid dealing with tactics configurations by using a standard tactics configuration that has been verified and, ideally, validated for the class of scenario in use. The Tactics Configuration Dictionary is a collection of standard tactics configurations.

3.3 Planning Input

The Planning Input component of the framework is shown in Figure 3. Most of the input to the planner is drawn from the CSE. Concrete settings for all of the required elements of the Planning Input constitute a single instance of an automated battle planning problem.

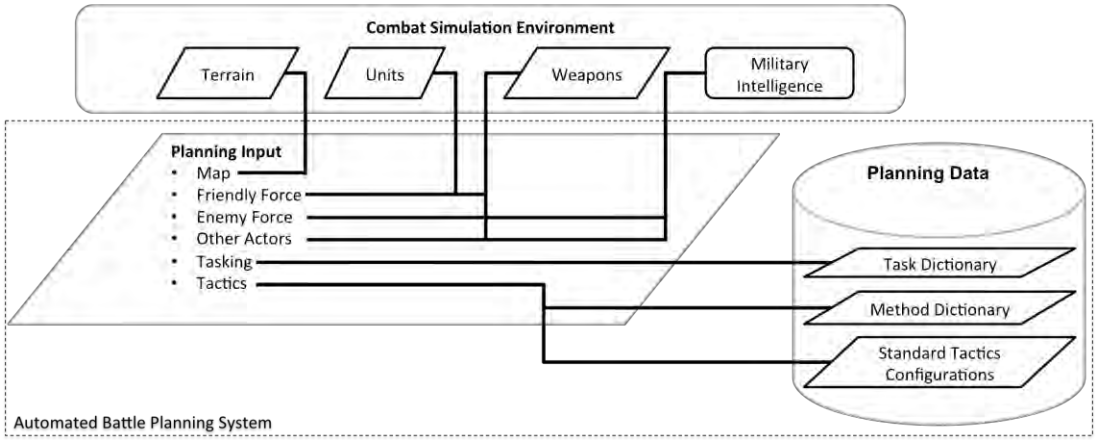

Figure 3: Planning input.

3.3.1 Map

The map is a digital representation of the geographical terrain where the combat occurs. The details included in the map depend on the terrain representation of the combat model, but the important point here is that the planner can access those details to support the prediction of combat outcomes.

3.3.2 Friendly Force

The Friendly Force is the set of combat units for which the ABPS will generate a plan. All units in the Friendly Force are assumed to be on the same side, but it is possible to run the planner with a Friendly Force that does not constitute an entire side. If that is the case, all other units on the same side are considered Other Actors (see 3.3.4 below). Friendly Force information is expected to include unit organization and unit starting locations.

3.3.3 Enemy Force

The Enemy Force input has all of the same parameters as the Friendly Force input, with the addition of an assumed plan. However, in general the planning system cannot be assumed to have accurate information about the enemy. For this reason, we show Enemy Force data connected to a Military Intelligence module in the CSE, as if the ABPS is the recipient of the final stage of the intelligence collection, processing, and dissemination process. The ABPS can assume the accuracy of Military Intelligence output, weight risk versus reward based on a confidence level, or even plan against multiple Enemy Force courses of action.

3.3.4 Other Actors

We include an input space for Other Actors, referring to any decision-making agents not contained in the Friendly Force or Enemy Force. As with the Enemy Force, the Friendly Force cannot be expected to have intrinsic knowledge of Other Actors' plans or other parameters, so we say that their information comes from the Military Intelligence component of the CSE. Other Actors could include neutral combat units, local citizens, international aid workers, or even herds of livestock.

3.3.5 Tasking

The Planning Input must include a *Tasking*, which provides requirements and guidance to the planner. The Tasking could come directly from a user or from a software system (such as another instance of the ABPS). In doctrinal planning, the mission statement is developed by the commander and staff of the planning unit, not by the higher echelon command tasking the planning unit. If we think of the planning system as a simulation of the commander's and staff's planning activity, then the responsibility of developing a mission statement lies within the system boundary, not as an input. The Planning Input provider plays the role of the higher echelon tasking command, which has the responsibility of providing taskings to its subordinates in its own plan (operations order, or OPORD). The Tasking content consists of required tasks (i.e., task instances that describe what must be done by the Friendly Force), prior plan, area of operations, user cost function (whereby the user provides nuanced guidance to the ABPS by weighting measurable features of candidate plans), and tactics.

3.3.6 Tactics

While Tasking specifies what to perform, tactics specify how to perform a particular action, as given in doctrinal publications or defined for unique behaviors for a particular purpose.

3.4 Plan Generator

The Plan Generator component of the framework is shown in Figure 4. The Plan Generator consists of processing components that use the Planning Input and Planning Data to produce an executable plan.

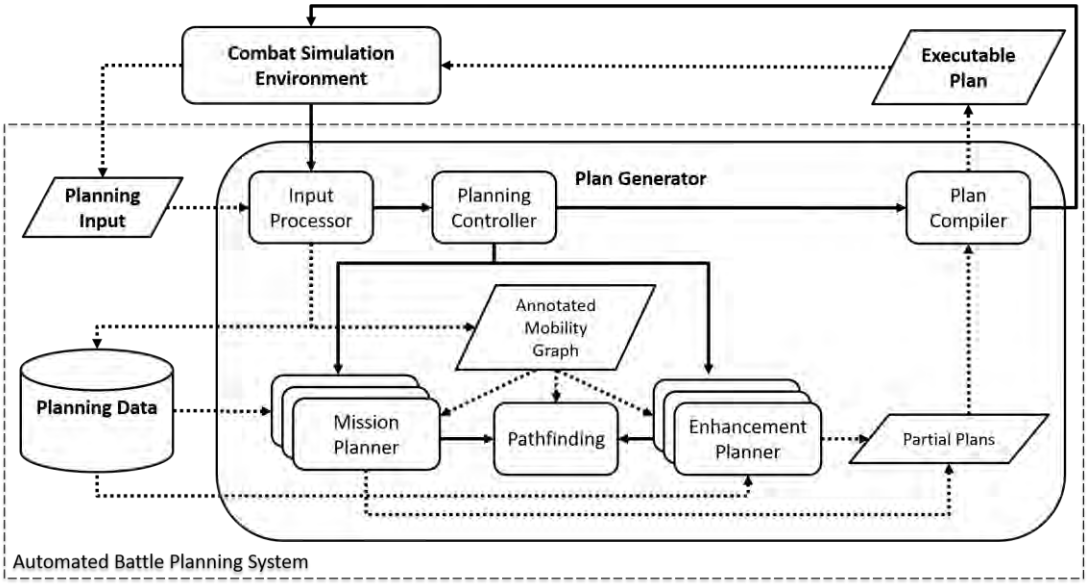

Figure 4: Plan generator.

3.4.1 Input Processor

The Input Processor is invoked by the CSE or by the user to initiate execution of the ABPS. It retrieves necessary information from the Planning Data into local memory, generates the Annotated Mobility Graph (if it does not already exist), and passes control to the Planning Controller. If the Planning Input includes a Prior Plan, the Input Processor is responsible for translating it into the Partial Plan format.

3.4.2 Planning Controller

The Planning Controller contains the logic for control flow between the Mission Planners and Enhancement Planners. This could be a fixed sequence of function calls or a conditional, iterative logic based on Partial Plan features, Cost Function values, or other Planning Input data. Each Mission Planner and Enhancement Planner is a self-contained automated planning algorithm. Since each subordinate planning algorithm may be computationally expensive, the Planning Controller should be as simple as possible. Mission Planners and Enhancement Planners are each allowed to use different objective functions and heuristics. The Planning Controller is responsible for dealing with these different measures to select a single Partial Plan (the *final plan*) that achieves all Required Tasks, if possible.

3.4.3 Annotated Mobility Graph

The *Annotated Mobility Graph* is a representation of the Map used for planning computations (e.g., Pathfinding. Annotations add information to each node to use for tactical reasoning about units, such as visibility and risk. Its structure and contents may be dependent on the specific locations of units, which may change even during the execution of a single scenario. A node in the Annotated Mobility Graph is a location in discretized space; i.e., a point, area, or volume that may be occupied by one or more units.

3.4.4 Mission Planner

A Mission Planner is a processing component that constructs or expands partial plans using a tactics configuration in order to achieve tasks. It reasons about units in relation to the Annotated Mobility Graph

by using the Planning Data's derived models and the Enemy Force's Plan (if provided in the Planning Input). It may invoke the Pathfinding component to consider possible movement of units. The Plan Generator may contain any number of Mission Planners, and there is no requirement for any particular component to achieve any particular tasks as long as the overall system generates acceptable plans for the CSE. The Mission Planner's effort is likely to involve searching the vast space of "goal" plans for one with an acceptable score—or at least best discoverable score—according to some objective function.

3.4.5 Pathfinding

Most conceivable tactical plans involve some kind of movement. Tactical movement planning is a relatively well-studied problem with some individually-packaged solutions, so we put this part of the problem into its own component. The Pathfinding component is expected to use the Annotated Mobility Graph as its search space, and it may use the derived movement models of the Planning Data for its calculations.

3.4.6 Enhancement Planner

The only fundamental difference between a Mission Planner and an Enhancement Planner is that the latter does not qualitatively achieve tasks; i.e., it does not have a meaningful goal test. An Enhancement Planner expands or modifies a partial plan using a tactics configuration to improve a plan according to an objective function. Enhancement Planners may be thought of as plan "optimizers," with the caveat that "optimal" plans are generally not achievable (at least not tractably).

3.4.7 Partial Plans

The Task Dictionary is the lexicon of the language needed by Plan Generator components to communicate with each other. Its grammar is the *partial plan representation* of the Plan Generator. We use this term in a general sense here to represent a set of task instances assigned to the units of the Friendly Force with a timing specification. It may contain gaps in time where no tasks are assigned to a particular unit, and it does not necessarily achieve all—or any—of the Required Tasks (yet). The parameters of each task instance may be partially bound. A Plan Generator subcomponent that receives a partial plan as input must be able to operate on some of its tasks. Each subcomponent should be able to harmlessly ignore partial plan tasks that it cannot process.

3.4.8 Plan Compiler

The Plan Compiler translates the Planning Controller's final plan into a format that can be executed by the CSE. If the planning system is built for a specific CSE, it may be possible to use the Executable Plan format of the CSE as the partial plan representation. In this case, a Plan Compiler is not required.

4 PROOF-OF-CONCEPT IMPLEMENTATION

An automated fire support planner has been developed which employs a greedy, best-first plan-space search with a continuous model of time (Harder, Balogh, and Darken 2016; Harder and Darken 2016). The planner accepts a maneuver plan (notionally generated by a Mission Planner) as input; it adds fire support tasks, including movement, for available units to suppress enemy threats and improve the chances of plan success. The CSE used for testing and demonstrating the planner is a custom-made simulation called WOMBAT XXI (WXXI), loosely patterned after the operational Combined Arms Analysis Tool for the 21st Century (COMBATXXI) simulation developed and employed by the US Army and US Marine Corps. WXXI is implemented in the Unity Game Development Environment (Unity Technologies

2016); see Figure 5. The implementation includes representations of entities, targeting and protection, stochastic range-based probability of hit and probability of kill, a three-state suppression model, hierarchical units, unit formations, waypoint-based movement, movement modes, operational tasks and actions, and terrain imported from real-world elevation data, as well as a simulation engine, run manager, and data collection mechanisms.

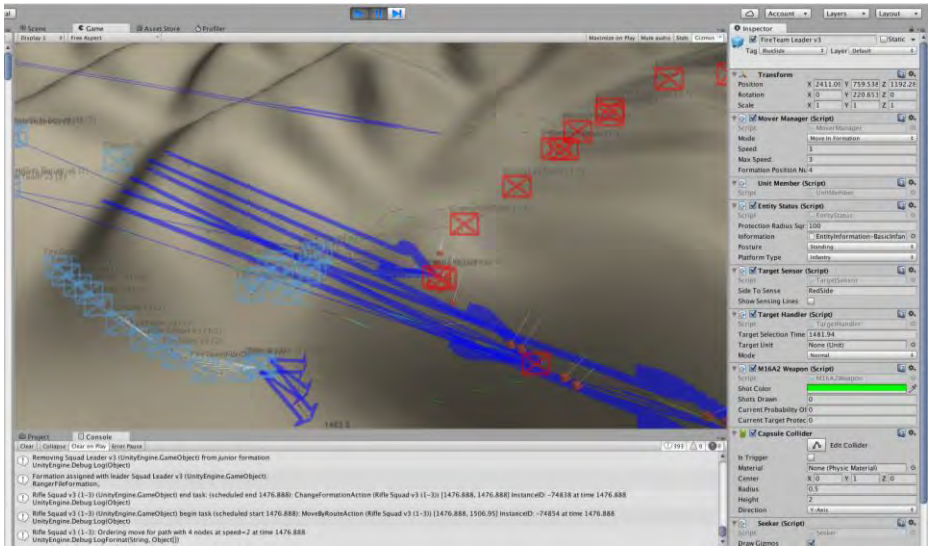

Figure 5: Screen capture of WXXI Combat Simulation Environment developed in Unity3D.

Implementation of the plan generator following the framework presented earlier is shown in Figure 6. A maneuver planner provides a partial plan for the fire support planner to modify. The fire support planner creates and modifies risk intervals (regions where a maneuvering unit is considered susceptible to enemy fire due to range and line-of-sight considerations) and fire support tasks to update the partial plan. Table 1 tabulates some outcomes from evaluation of generated fire support plans, where performance measures for mission accomplishment and fractional exchange ratio are computed as:

1. Mission Accomplishment = $[C(0) - C(t_{END})] / C(0)$, where C(t) is number of cleared objectives at time t.
2. Fractional Exchange Ratio = $[[B(0) - B(t_{END})] / B(0)] / [[A(0) - A(t_{END})] / A(0)]$, where A(t) is number of attacking entities alive at time t and B(t) is number of defending entities alive at time t.

In testing, the planner exhibited some interesting emergent properties such as well-timed suppressions, shifting and lifting fires, supporting follow-on attacks, consolidating fire support positions, ignoring low threats, and bounding movement. However, some negative outcomes were also possible, such as unsafe directions of fire, firing position collocation, and incorrect cease-fire times. Additional constraints can be added to the plan generation scheme to address such issues. Even so, the rapidity of execution to generate numerous acceptable plans will enable analysts to generate and explore a larger range of scenarios more quickly.

5 SUMMARY AND CONCLUSIONS

The planning framework described here supports a modular design for ABPSs, which in turn allows a division of labor in its development. The data-driven approach prescribed by the Planning Data may allow some components, such as the Planning Controller, Mission Planners, and Enhancement Planners, to be

re-used across different CSEs (or at least simplify migration). The Partial Plan data structure and Annotated Mobility Graph also may be reusable if references to units and terrain objects are carefully encapsulated. Agreement on a modularization approach can help with the re-use and improvement upon ideas, algorithms, and data structures. The framework makes few assumptions about the domain, but is general and modular enough to be applied to a wide range of combat M&S uses.

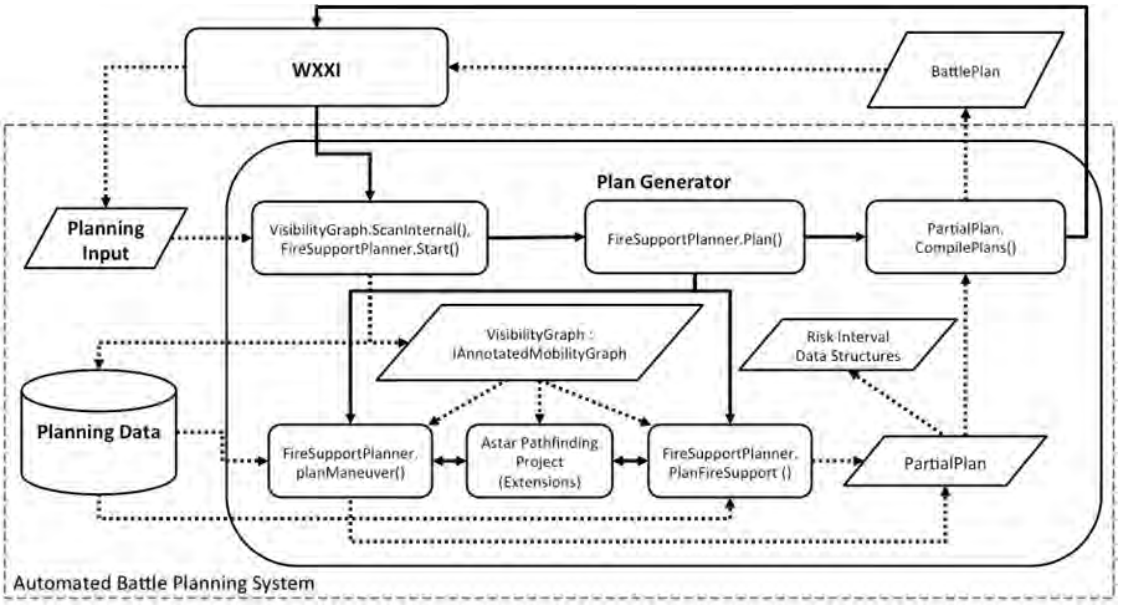

Figure 6: Plan generator implementation for the fire support planner.

Table1: Sample of quantitative results of the effectiveness of generated fire support plans.

Scenario				Mission Accomplishment		Fractional Exchange Ratio	
Group	**Forces**	**Map**	**Fire Support Planning Type**	***μ***	**std. err**	***μ***	**std. err**
1	Plt vs Squad	A	Manual	0.367	0.092	0.88	0.225
			Best = Max	0.667	0.000	5.13	0.601
			Close-quarters	0.800	0.054	6.49	1.362
2	Plt vs Squad	B	Manual	0.567	0.071	1.73	0.177
			Best = Max	0.833	0.056	19.50	3.744
			Close-quarters	0.600	0.083	1.91	0.419
3	Co vs Plt		Manual	0.267	0.030	0.40	0.047
			Best	0.733	0.038	1.87	0.148
			Max	0.656	0.042	1.65	0.138
			Close-quarters (Best)	0.733	0.050	2.59	0.364
4	Plt vs Squad	C	Manual	0.450	0.050	0.53	0.087
			Best = Max	0.875	0.042	8.88	1.307
			Close-quarters	0.775	0.058	2.69	0.470
5	Co vs Plt		Manual	0.460	0.031	0.89	0.102
			Best = Max	0.750	0.031	3.17	0.326
			Close-quarters	0.690	0.023	2.21	0.129
6	Bn vs Co		Manual	0.156	0.020	0.38	0.024
			Fastest	0.359	0.012	0.61	0.021
			Fast	0.741	0.023	2.06	0.132
			Best	0.841	0.037	2.56	0.173
			Max	0.874	0.021	2.80	0.181
			Close-quarters (Best)	0.933	0.011	4.42	0.222

As envisioned, a variety of different functions could be provided by a single ABPS. We offer the framework and implementations as starting points for the organization of research in this domain, but improvements on its structure, vocabulary, and assumptions are expected. Future open community competitions may serve to advance the state-of-the-art in much the same way as has occurred in the real-time strategy gaming community.

ACKNOWLEDGMENTS

Information in this paper reflects the opinions of the authors and is not necessarily the official position of the Naval Postgraduate School, the United States Navy, or the United States Marine Corps.

REFERENCES

Harder, B., I. Balogh, and C. Darken. 2016. "Implementation of an Automated Fire Support Planner". In *Proceedings of the Twelfth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE-16)* , edited by N. Sturtevant and B. Magerko, 51-57. Palo Alto, California: AAAI Press. http://www.aaai.org/ocs/index.php/AIIDE/AIIDE16/paper/viewFile/14021/13594.

Harder, B., and Darken, C. 2016. "Automated Fire Support Planning for Combat Simulations". Poster Session, *2016 International Conference on Social Computing, Behavioral-Cultural Modeling, & Predication and Behavior Representation in Modeling and Simulation*. Online Proceedings: http://sbp-brims.org/2016/proceedings/IN_7.pdf.

Harder, B. 2017. "Automated Battle Planning for Combat Models with Maneuver and Fire Support". Ph.D. thesis, Modeling, Virtual Environments, and Simulation, Naval Postgraduate School, Monterey, California.

Unity Technologies 2016. *Unity*. Video game development software. Version 5.4.2f2. https://unity3d.com/.

AUTHOR BIOGRAPHIES

BYRON HARDER is the Chief Integration Officer for the Live, Virtual, and Constructive Training Environment at the Marine Corps Training and Education Command. He holds a Ph.D. in Modeling, Virtual Environments, and Simulation (MOVES) from the Naval Postgraduate School (NPS), Monterey, California. His research interests include automated planning, behavior modeling, computer networks, software engineering, IT management processes, algorithms, and theory of computation. His e-mail address is byron.harder@usmc.mil.

CURTIS BLAIS is a research associate in the NPS MOVES Institute. He holds a Master of Mathematics and Bachelor of Mathematics degrees from the University of Notre Dame, South Bend, Indiana. His research interests are combat modeling, human behavior modeling, unmanned system behavior modeling, and web technologies for knowledge representation and reasoning. His e-mail address is clblais@nps.edu.

IMRE BALOGH is an Associate Research Professor and Director of the NPS MOVES Institute. He holds a Ph.D. in Computer Science from New Mexico State University, Las Cruces, New Mexico. His research interests include foundations of Modeling and Simulation, agent based models and AI. His email address is ilbalogh@nps.edu.