PLAN SYNCHRONIZATION IN THE RSAC ENVIRONMENT

David Shlapak, Patrick Allen and William Schwabe

February 1986

P-7195

The Rand Paper Series

Papers are issued by The Rand Corporation as a service to its professional staff. Their purpose is to facilitate the exchange of ideas among those who share the author's research interests; Papers are not reports prepared in fulfillment of Rand's contracts or grants. Views expressed in a Paper are the author's own and are not necessarily shared by Rand or its research sponsors.

The Rand Corporation, 1700 Main Street, P.O. Box 2138, Santa Monica, CA 90406-2138

PLAN SYNCHRONIZATION IN THE RSAC ENVIRONMENT

INTRODUCTION[1]

This paper addresses some of the issues involved in the synchronization of Analytic War Plans (AWPs) within the Rand Strategy Assessment Center's (RSAC) automated war-gaming system.[2] Following a brief introduction to the RSAC, we will explore some of the issues involved in the planning process and how those issues affect the creation and maintenance of AWPs. Next, we will examine the script-like nature of many AWPs and expose some of the problems which may arise in coordinating multiple scripts; we will then discuss the techniques used in addressing these problems within the RSAC system.

THE RAND STRATEGY ASSESSMENT CENTER

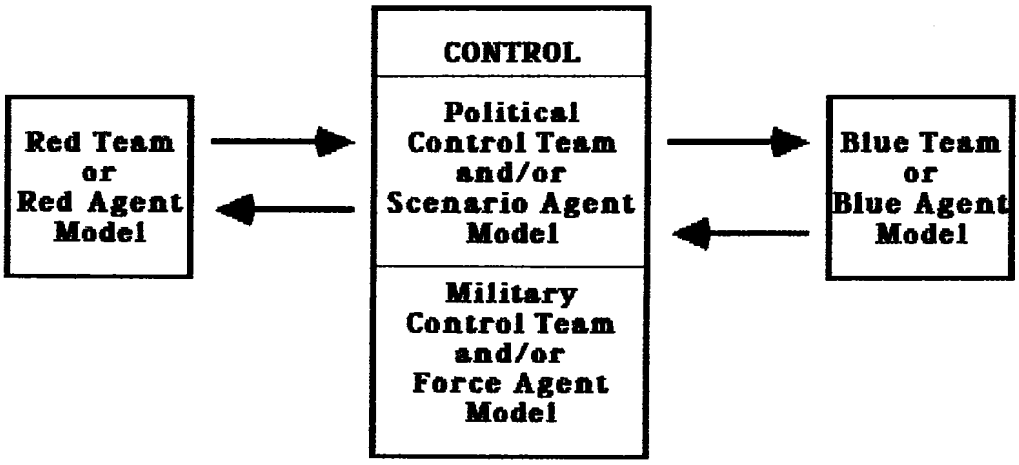
The Rand Strategy Assessment Center (RSAC) is developing a computer-supported, interactive war-gaming system. In this system, the functions of the traditional "Red," "Blue," and "Control" teams can be simulated or supported by automated decision models. The system may be configured for a game pitting computerized decision models against one another, a man-machine exercise in which human players or teams replace one or more of the automated decision-makers, or to support a more traditional political-military game.

The RSAC system consists of four principal modules, or *agents*, which are shown in Figure 1. The Red and Blue Agents represent the Soviet Union and the United States, respectively. Each of these agents is headed by a **National Command Level** (NCL) model which functions as the highest political-military authority, assessing the world situation, establishing objectives, and selecting plans and strategies. Below the NCL in the hierarchical structure of each agent are several other levels

---

[1]This paper was prepared for presentation at the TIMS/ORSA Joint National Meeting, April 14-16, 1986, in Los Angeles.

[2]Related Rand Corporation papers describing the RSAC include: Davis (1984); Shukiar (1985); Davis (1985); Schwabe (1985); and Shlapak (1985).



Red Agent: **Models Soviet and Warsaw Pact decisions**
Blue Agent: **Models United States and NATO decisions**
Scenario Agent: **Models nonsuperpower countries' decisions**
Force Agent: **Simulates results of combat and other military operations; also controls simulation time**

Fig. 1 -- RSAC system organization

of decision models; our concern in this paper is largely with these lower command levels.

Scenario Agent is a third rule-based model in the RSAC system; it models the behavior of nonsuperpower countries within the general RSAC context of superpower conflict.[3] Finally, the **Force Agent** is an integrated complex of interactive models designed to support analysis of multi-theater conflict across the full spectrum of potential combat intensities.

These models operate within a sophisticated support environment under the UNIX operating system;[4] this environment consists of four principal elements:

---

[3]Scenario Agent is documented in Shlapak et al (1985).
[4]UNIX is a Trademark of Bell Labs.

- The **System Monitor** coordinates the activity of all other parts of the RSAC system, making sure that the proper agent is running at all times.
- The **Data Dictionary** defines the data items which can be accessed by the various decision-making entities. It can also enforce restrictions on that access so that, for example, Blue cannot "peek" at Red's operational plans.
- The **World Situation Data Set**, or WSDS, contains all the data items themselves.
- The **Data Editor** serves as the man-machine interface between the user of the RSAC system and the WSDS.[5]

The RSAC system exists today as an operational prototype; current plans are to begin making it available to initial government users by the end of FY 1986.

The concept of an Analytic War Plan is central to the design of the Red and Blue Agents, and we will next examine the role AWPs play in the RSAC system in somewhat greater depth.

Scripts and AWPs

A *script* may be thought of as a linked set of instructions meant to be executed sequentially over a period of time. For example, a set of directions such as "Turn north at the corner of Pico and Main and pull into the fourth driveway on your left" constitutes a simple script. Scripts may be linear, meaning that one follows a series of steps never repeating any step twice; the simple example above represents a linear script. Alternatively, a script may be cyclical, executing the same steps repeatedly if the situation so requires. Instructions for telephoning a friend could be represented as a cyclical script like this:

[5]The computer environment of the RSAC system is described in Shukiar (1985).

Step A: Dial your friend's number. Go to step B.

Step B: If you hear a busy signal, go to step C; else go to step D.

Step C: Hang up. Wait one minute. Go to step A.

Step D: Have your conversation. Hang up. End.

In this script, steps A, B, and C may be repeated any number of times.

In expert systems and artificial intelligence (AI) work, scripts are used in much the same way scenarios are used for planning and analytic purposes: they provide a means for creating a focused context potentially containing a great deal of information and guidance while remaining themselves manageably compact. This makes them a powerful tool for both implementing and managing complex simulations and models.

The principal script-like entities within the RSAC are the Analytic War Plans utilized by the various Area Command Levels (ACLs) within the Red and Blue Agents. An AWP is a representation for analytic purposes of a war plan suitable for execution by the Red or Blue Agent in an RSAC war game. ACL-level AWPs are similar to cyclical branched scripts in that they are made up of a group of sequential actions each of which is related to and to some extent dependent upon those which preceded it, with conditional tests choosing which particular course the plan will follow from a pre-defined set of options.[6] A plan, for example, may incorporate a branch which conducts an offensive operation using conventional weapons only along with another in which nuclear weapons are utilized. The script will track down one path or another depending upon the situation it is confronted with.

RSAC efforts to mimic the output of real-world planning processes have led to a concomitant recognition that AWP writers will also face some of the same hazards encountered in those processes. In the next section we will briefly examine that process and those hazards.

---

[6]The programs utilized at the Global Command Level (GCL) and the Supertheater Command Levels (SCLs) are also AWPs. However, they do not share the script-like qualities of ACL plans; both GCLs and SCLs run from the top every time they execute rather than picking up where they left off in a sequence of actions. Problems of synchronization do not arise with these AWPs, since, unlike ACL plans, they start with a clean slate every time they move.

THE PLANNING PROCESS

Military plans are written based upon sets of assumptions, including estimates of future conditions. However, there are two time- and experience-honored rules of forecasting, and hence of planning itself:

1. The forecast is wrong,
2. The forecast will change.

Essential to the creation of successful plans is the development of some mechanism which can cope with the inherent unpredictability of the future.

There are many factors which a planner must consider when plying his trade. Timing, available resources, and likely rules of engagement must all be taken into account. However, the most important factor associated with a plan is its *purpose* or *goal*.

Modifying and Switching Plans

If the purpose of the plan has been accomplished or is no longer possible, the plan becomes obsolete. One example is the capitulation of the major opponent during the course of a war; a contrary one is the outbreak of hostilities when a plan's goal was to maintain the peace through a strategy of deterrence. As long as the purpose of a plan is still obtainable and the plan is still feasible, the plan is more likely to be modified than discarded when circumstances demand some alteration in approach.

Changes in national objectives generally necessitate switching plans. Operations plans may also be switched if they aren't working or if an operation has "run out of plan."

Even when a plan has clearly failed, however, there are risks involved in replacing it. For example, the longer the time it takes to complete the transition from one plan to another, the more likely that the enemy will gain the initiative. The old adage "Don't switch horses in midstream" is an appropriate caution in such circumstances.

A second clear problem results from the potential mismatch between steps excuted by the previous plans and the context with which the newly-implemented plan was designed to deal. For example, an American plan intended to enhance deterrence against a Soviet invasion of Iran might involve the deployment of U.S. forces to the Arabian Peninsula. Should a war in Europe break out shortly thereafter, the availability of those forces might be a requirement, or at the very least an assumption, of the plan chosen to guide the defense of NATO. Alternatively, a plan based upon massive use of nuclear weapons might be substituted for one which countenanced only conventional conflict. The new plan, or those charged with executing it, must be smart enough to verify that friendly forces are dispersed for protection, and if they are not, to issue the proper orders. It is problems such as these, which arise both *between* and *within* theaters that we have identified as issues of plan synchronization. These problems arise not only in the world of operational planners, but in the tasks of RSAC AWP writers as well.

AWP DESIGN AND STRUCTURE

To explain the impact of synchronization problems in the RSAC system, we must first examine in greater detail the design and structure of AWPs. We will also introduce the two conceptual modes in which AWPs are executed, and across which they must therefore be coordinated; we have labeled these modes *synchronous* and *asynchronous*, and their significance will be made clearer in the following section.

Synchronous and Asynchronous Processes

Unlike most AI-type applications of scripts, within an RSAC war game there are many scripts running in parallel. Currently, the Red and Blue Agents are each able to support about a dozen AWPs running simultaneously. Each of these plans exists in an individual location which is created by the RSAC system software when a run is initialized. Each of these locations can be thought of as a *socket* into which an AWP can be plugged. For example, the Blue Agent's Central European theater (the CEUR ACL socket) can be filled with any one of a number of CEUR AWPs. Proper control of each script must be exercised both by the RSAC

control software (to avoid system crashes should one script begin running amok) and the top-level player, human or automated (to ensure, for example, that attacks intended to occur simultaneously do so). Thus, these numerous sockets require some measure of coordination. Because all sockets are created and terminated simultaneously (at the beginning and end of an RSAC system run) and mark time by a common clock (the system clock maintained by the RSAC Force Agent) we may refer to them as *synchronous* processes.

A typical ACL-level AWP is composed of three levels of control. The main control mechanism is a loop in which the basic decisions affecting plan execution are made. As the script is executed, control passes from this central loop to one of any number *phases*, each of which in turn is made up of several *moves*. For example, a Red (Warsaw Pact analogue) plan for the invasion of the Federal Republic of Germany (FRG) might have the following structure:

PHASE I: Preparation

Move i: Alert and deploy forces

Move ii: Poise forces for assault

PHASE II: Attack and Combat

Move iii: Attack

Move iv: Adjust operation as objectives are achieved or not (includes branch for use of nuclear weapons if necessary).

PHASE III: Termination

Move v: End hostilities.

Execution passes from move to move according to logic embedded at the phase level. The appropriate phase to enter is determined via a closed loop at the beginning of the script. A phase may return control to the top level logic at any point should it be unable to appropriately deal with a situation; the top level may then choose to execute another phase of the script or return control to its superior command with an explanation for its condition and a recommended course of action to rectify its difficulties. When, however, the plan returns to the central loop it needs to know what it has done in order to properly generate the next cycle. Ensuring that such information is available to the script is a second aspect of the synchronization task.

Further, most uses of scripts do not require that they be arbitrarily interchangeable; once activated, a script will run itself out from beginning to end. It is a design requirement of the RSAC system, however, that a player, whether the automated agents or a person sitting at a terminal, be able to change plans at any moment of his choosing. Thus, a given socket may have several different plans plugged into it at different times in the course of one game. These successive plans within a single socket can be thought of as *asynchronous processes* which can be activated at any moment and turned off in an equally arbitrary manner. This means that a script with, say four phases, might be installed at a point where it should skip the first two. Such a plan switch can only be successfully accomplished by providing a mechanism which orients the new script properly and provides it with a fairly rich context.

A fundamental element of the RSAC approach to AWP coordination is the distribution of responsibility for ensuring this coordination throughout the RSAC system. Some facets of the synchronization problem are addressed in the system software itself, while others are dealt with in the design of the AWPs at each command level. The nature and purpose of this division of labor are discussed at greater length in later sections.

COORDINATING RSAC AWPS

Synchronous Processes

Proper coordination of the many synchronous processes is accomplished in a two-fold manner:

- The sockets themselves are hierarchically arranged, and
- Each plan determines the timing and nature of its own activities, consistent with control exercised from higher levels of the agent hierarchy.

As was noted above, the synchronization of a single plan within a given socket is a special case of the more general issue of coordinating synchronous processes. Thus, the proper functioning of a particular AWP script move after move is ensured by applying the principles sketched in this section.

AWP sockets within the RSAC are arrayed according to their functional position within the Red or Blue Agent's command structure. Each socket is immediately subordinate to one other socket, and may itself be in command of one or more lower-level sockets. In common AI usage, a socket has one *parent* and may have any number of *children*.

For example, Figure 2 shows part of the command structure of the RSAC's Blue Agent. The Blue GCL is immediately below the Blue National Command Level (NCL), which is the highest level of the Blue Agent. In turn, the GCL is the *parent* (superior) of the Blue European SCL (BEUR), and the Blue ACLs for intercontinental strategic warfare (BIC), and Southwest Asia (SWA). BEUR, a *child* (subordinate) of the Blue GCL, is itself the parent of the Central European (CEUR) and Northern European (NEUR) ACLs. These relationships exist *regardless* of the specific AWPs which are running in the various sockets.

The parent-child relationship is important in a number of ways. First, each plan is responsible for starting and stopping the AWPs in its child sockets, if any: an AWP is always initiated and terminated by

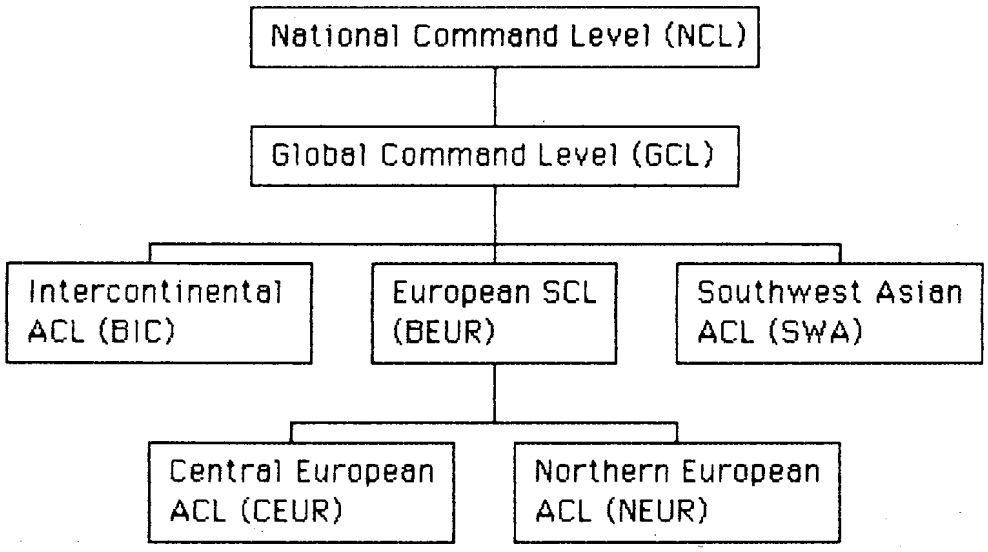


Fig. 2 -- An illustrative Blue Agent command structure

its immediate superior. Similarly, when a plan is removed from a socket, all the AWPs in its child sockets are likewise unplugged.[7]

Further, certain AWPs are designed to be run in conjunction with certain others; that is, the context built into the AWP script includes assumptions about what's happening in the game. These assumptions mean that not all arbitrary combinations of AWPs can be expected to run. For example, a plan running in a GCL socket may be designed to operate in a crisis environment in which war has not yet broken out. This AWP will work properly only with subordinate plans having consistent objectives. A war-fighting SCL AWP (one intended to control actual combat operations) will not work in conjunction with such a GCL plan; the crisis GCL plan is not designed to coordinate a wartime subordinate plan. By establishing these relationships between plans and incorporating assumptions regarding them in the built-in context of AWP scripts, we believe we have significantly reduced the dangers of costly disconnects between socket and socket or plan and plan.

The hierarchy outlined above is analogous to real-world military command structures and is therefore intuitively attractive and comprehensible. In the RSAC, however, it serves primarily to coordinate AWPs in their guise as bits of software floating around in a highly complex processing environment. The second synchronization measure mentioned earlier, the constrained independence of the AWP in each socket, is largely responsible for coordinating each plan as an operational, analytically meaningful entity.

It is first important to understand that an AWP, when it awakes, has no knowledge of its history or the history of the socket it occupies. It utilizes the same clues to line itself up whether it is moving for the 22nd time or the first. Thus, much of what is discussed below is applicable also to the coordination of asynchronous processes (i.e., synchronizing a new plan in a previously occupied socket).

---

[7]Note that removing a plan is *not* the same thing as removing a socket; as was discussed earlier, sockets remain from start to finish regardless of plan changes within them.

AWPs orient themselves from move to move via *situational data* ("What time is it? Is there a war going on? Are nuclear weapons in use?") and *authorizations and controls* ("Am I authorized to use nuclear weapons? Have I been ordered to terminate?") set by higher levels in the RSAC hierarchy.

We have established three primary situation indicators for each AWP:

- *Ground situation*, which is set according to the current level of army activity in a given socket. Examples of ground situation might include deploying, poising, in-combat, or phaseline-A.[8]

- *Air situation*, which reflects a qualitative assessment of the air war in the theater associated with an AWP. Exemplary values might be inferiority and superiority.

- *Nuclear situation*, reflecting the extent to which nuclear weapons have been used, or preparations for their use have been made, by the AWP or its predecessors. Sample values could include dispersing, demonstrative-use, and strategic-use.

A plan might resume executing to find, for example, a ground situation of "in-combat." This would indicate that, if faced with a choice, it should enter its war-fighting phase rather than, say, its preparation phase (which it has either executed already or, in the event of a surprise attack, might in large measure have been rendered moot by the enemy's attack). The precise actions to execute would be determined using this contextual information in conjunction with any rules of engagement imposed by higher authorities.

Authorizations and controls may be either *prescriptive* or *advisory*. A prescriptive authorization or control mandates the immediate execution of a particular action or set of actions. For example, an AWP receiving termination authorization from its controlling NCL will ignore all other

[8] "Phaselines" in military terminology usually refer to intermediate goals of a plan. They can be expressed in a variety of ways but frequently include elements of territory lost or gained, attrition inflicted and suffered, and elapsed time. Phaseline-A, in these terms, would represent the first intermediate goal of a given AWP; a ground situation of phaseline-A would indicate that it had been achieved.

considerations and promptly enter its termination phase with the intent of ending hostilities.[9]

An advisory control or authorization, on the other hand, either restrains a plan from carrying out a specific action or allows it to; it does not require the plan to do anything in particular. An AWP receiving advisory authority to use nuclear weapons will not necessarily begin launching such strikes immediately. However, if it reaches a branch where nuclear use is an option, it will ascertain whether or not this permission has been granted before choosing the script calling for atomic use.[10]

Permissions and controls were originally intended as a means of allowing the NCL flexible control over actions taken at lower levels of the Red and Blue Agents. They have evolved into a mechanism which, when combined with the situational data which the plans leave like a trail of breadcrumbs, provides the orienting context for all AWP scripts.

Asynchronous Processes

A final tool used for plan coordination in the RSAC is used when a new plan is installed in a socket.[11] As explained earlier, parent sockets are responsible for starting up all AWPs in their child locations. Orienting these newly-begun plans is an important part of their initialization.

Synchronization is relatively trivial when an NCL starts a new GCL, or a GCL an SCL, since both GCLs and SCLs are not script-like but bootstrap with every execution. Similarly, it would be possible for the

---

[9]The exact orders given within the termination phase itself might be sensitive to external factors, however. For example, a plan may want to straighten out the front lines of its forces before implementing a ceasefire in order to avoid maintaining exposed salients and flanks. Such mopping up activities could be ordered from within the termination phase of a plan.

[10]Many authorizations and controls are two-sided in that they can be either prescriptive or advisory, depending upon the value of the attribute in question. Nuclear-use, for instance, might have values of "no," "allowed," and "required."

[11]It is largely irrelevant whether or not another plan had previously occupied the socket in question.

GCL or SCL to simply initiate an ACL script and let it find its own way via the techniques outlined in the previous section. It was decided, however, that such a blind start-up method was both inelegant and prone to failure. In true parental fashion, then, GCLs and SCLs give their ACL children a good start in life by pointing them towards an appropriate starting point in their script.

In determining this starting point, the GCL or SCL relies upon the situational information posted by the socket in question and the identity of the ACL-level plan being initiated. Thus, the starting point passed to the ACL represents a crude indicator, but one which saves the script a significant amount of processing: on its first move it need only check its starting point and permissions to get oriented.

Table 1 is extracted from the RAND-ABEL instructions used by a Blue European SCL to determine the starting points for each of its subordinates.[12] Note that only one of the three situation descriptors

Table 1

CHOOSING PLAN STARTING POINTS

gnd-sit	air-sit	nuc-sit	prev-plan	curr-plan	point-in-curr-plan
Any	Any	Any	Any	CEUR0	Top-of-plan
Any	Any	Any	Any	CEUR1	Top-of-plan
Any	Any	Any	Any	CEUR2	Top-of-plan
Alerted	Any	Any	Any	CEUR2	Poise-pt
Deploying	Any	Any	Any	CEUR2	Poise-pt
Poised	Any	Any	Any	CEUR2	Combat-pt
Combat	Any	Any	Any	CEUR2	Combat-pt
Termination	Any	Any	Any	CEUR2	Terminate-pt.

[And so on for all plans in CEUR, NEUR, and SEUR.]

NOTE: gnd-sit = ground situation in the theater
air-sit = air situation in the theater
nuc-sit = nuclear situation in the theater
prev-plan = previous plan in the theater (socket)
curr-plan = current, or new, plan in the theater (socket)
point-in-curr-plan = starting point selected for new plan

[12]RAND-ABEL" is an English-like, C-based computer language developed by the RSAC for automated gaming applications. The definitive reference is Shapiro et al (1985).

currently implemented is actually used by this SCL; nor do the CEUR plans shown require that the full range of starting points be utilized. There is therefore a considerable amount of built-in growth room in this mechanism for allowing much more precise coordination of ACL plans should such control be called for as AWP development continues.

CONCLUSIONS

This paper has been by no means an exhaustive or detailed account of either the potential problems in synchronizing plans within the RSAC war gaming system or the solutions implemented; it has rather tried to describe the basic elements of both realms. We have devised simple and apparently robust mechanisms for controlling multiple scripts in a complex environment, and we believe that our techniques are extensible to other applications requiring multiple scripts in a dynamic simulation.

SELECTED BIBLIOGRAPHY OF RSAC PUBLICATIONS

Davis, Paul K., *Applying Artificial Intelligence Techniques to Strategic-Level Gaming and Simulation*, P-7120, November 1985.

---, *Game-Structured Analysis as a Framework for Defense Planning*, P-7051, January 1985.

Davis, Paul K., and William Schwabe, *Search for a Red Agent to be Used in War Games and Simulations*, P-7107, July 1985.

Schwabe, William, *Knowledge Engineering for Rand Strategy Assessment Center Military Command Level Models*, P-7115, July 1985.

Shapiro, Norman Z., et al., *The RAND-ABEL Programming Language: Reference Manual*, N-2367-NA, October 1985.

Shlapak, David A., *The Rand Strategy Assessment Center and the Future of Simulation and Gaming*, P-7162, November 1985.

Shlapak, David A., et al., *The Mark III Scenario Agent: A Rule-Based Model of Third-Country Behavior in Superpower Crises and Conflict*, N-2363-NA, October 1985.

Shukiar, Herbert J., *Automated War Gaming: An Overview of the Rand Strategy Assessment Center*, P-7085, May 1985.