# *Managed Readiness Simulator (MARS) V2*

*Design of the Managed Readiness Model*

Stephen Okazawa
Mike Ormrod
Chad Young
DRDC CORA

DRDC CORA TM 2009-057
November 2009

**Defence R&D Canada**
**Centre for Operational Research and Analysis**

Land Forces Operational Research Team

National Defence Défense nationale

Canada

# Managed Readiness Simulator (MARS) V2

*Design of the Managed Readiness Model*

Stephen Okazawa
Mike Ormrod
Chad Young
DRDC CORA



## Defence R&D Canada – CORA

Technical Memorandum
DRDC CORA TM 2009-057
November 2009

Principal Author

*Original signed by Stephen Okazawa*

Stephen Okazawa

Defence Scientist

Approved by

*Original signed by Dean Haslip*

Dean Haslip

Section Head Land and Operational Commands

Approved for release by

*Original signed by Dale Reding*

Dale Reding

Chief Scientist DRDC CORA

Defence R&D Canada – Centre for Operational Research and Analysis (CORA)

# Abstract

The first Managed Readiness Simulator prototype (MARS V1) successfully demonstrated the modelling concepts on which MARS is based, but it lacked technical flexibility and the potential to address subsequent development goals. The most important development goal which was considered infeasible in MARS V1 was support for a dynamic establishment in which CF resources are recruited into, advance through and retire from the establishment at the same time that operational deployments are occurring. Other features were also desired in order to improve the generality and flexibility of the MARS application. MARS V2 was developed to achieve these goals. The feature enhancements and additions involved in the development of MARS V2 affected nearly all aspects of the original functional design. Therefore, this paper serves as a comprehensive reference of all the functionality now available in MARS V2.



# Résumé

Le premier prototype de simulation de gestion de la disponibilité opérationnelle (MARS V1) a fait la preuve des principes de modélisation sur lesquels MARS est fondé, mais ce logiciel n'offrait pas la souplesse technique ainsi qu'un potentiel suffisants pour répondre aux objectifs de développement subséquents. L'objectif de développement le plus important qu'il semblait impossible de réaliser au moyen de MARS V1 était la prise en charge de l'établissement dynamique, dans le cadre duquel les ressources des FC sont engagées, mises en œuvre et retirées de l'établissement au même moment que les déploiements opérationnels sont effectués. D'autres caractéristiques étaient aussi souhaitées afin d'améliorer la portée générale et la souplesse de l'application MARS. MARS V2 a donc été développé pour répondre à ces objectifs. Les améliorations et ajouts apportés aux fonctions et intégrés au développement de MARS V2 ont touché presque tous les aspects de la conception fonctionnelle initiale. Le présent document constitue une référence complète décrivant les fonctionnalités offertes par MARS V2.

This page intentionally left blank.

# Executive summary

## Managed Readiness Simulator (MARS) V2: Design of the Managed Readiness Model

**Stephen Okazawa; Mike Ormrod; Chad Young; DRDC CORA TM 2009-057; Defence R&D Canada – CORA; November 2009.**

**Background:** The Managed Readiness Simulator (MARS) is a software application being developed at Defence Research & Development Canada - Center for Operational Research & Analysis as an Applied Research Project managed by the Land Force Operational Research Team. MARS is designed to quickly simulate a wide range of readiness scenarios to determine if the resources of an establishment are able to satisfy the requirements of planned operations.

The first version of MARS (termed V1) successfully conducted a preliminary analysis of the Army's plans to generate forces for Task Force Afghanistan. In the process of conducting this analysis, several aspects of the MARS V1 architecture and design were identified as limiting the potential of MARS to address future problems. In particular, modelling the dynamics of the Establishment (the creation, advancement and release of Establishment Resources), was not considered feasible within the MARS V1 software architecture and feature set. Certain design aspects of MARS V1 were also restrictive in terms of the types and complexity of scenarios that could be represented.

**Results:** Prior research produced a new technical platform for MARS, termed V2, which greatly expanded the potential of MARS to address more complex scenarios. This technology was exploited to implement more advanced modelling capabilities including Establishment dynamics. In order to achieve these new modeling capabilities, substantial modifications to MARS V1 functionality and the implementation of entirely new functionality was carried out. In total, the feature changes and additions implemented in the MARS V2 platform affected nearly every aspect of the original functional design. Therefore, the purpose of this paper is to fully document the functionality now available in MARS V2.

**Significance:** This paper serves as the principal document describing the capabilities of MARS V2. It will allow both the military client and the civilian analyst to assess what can and cannot be achieved using the current MARS application.

# Sommaire

## Managed Readiness Simulator (MARS) V2: Design of the Managed Readiness Model

**Stephen Okazawa; Mike Ormrod; Chad Young; DRDC CORA TM 2009-057; R & D pour la défense Canada – CORA; Novembre 2009.**

**Contexte:** Le Programme de simulation de gestion de la disponibilité opérationnelle (MARS) est une application logicielle que l'on développe à Recherche et développement pour la défense Canada – Centre de recherche opérationnelle et d'analyse comme projet de recherche appliquée gérée par l'équipe de recherche opérationnelle de la Force terrestre. Le programme MARS a été conçu pour simuler rapidement un vaste éventail de scénarios de disponibilité opérationnelle afin de déterminer si les ressources d'un établissement peuvent répondre aux exigences des opérations prévues.

La première version du programme MARS (appelée V1) a réalisé avec succès une analyse préliminaire des plans de l'Armée visant à mettre sur pied des forces pour la Force opérationnelle interarmées Afghanistan. Cependant, on a décelé dans la conception et l'architecture de MARS V1 plusieurs aspects qui limiteraient le potentiel de MARS en ce qui a trait à la résolution de problèmes futurs. Tout particulièrement, la capacité de modéliser la dynamique des établissements (création, évolution et libération des ressources de l'établissement) serait impossible avec l'architecture et l'ensemble de caractéristiques de la première application MARS. Par ailleurs, certains aspects conceptuels de MARS V1 étaient aussi restrictifs sur le plan des types et de la complexité des scénarios qui pouvaient être représentés.

**Résultats:** Les recherches précédentes ont produit une nouvelle plateforme technique pour MARS, appelée V2, qui a beaucoup élargi le potentiel de MARS à analyser des scénarios plus complexes. Cette technologie a été exploitée dans le but de la mise en œuvre de capacités de modélisation plus avancées, en particulier en ce qui concerne la dynamique de l'établissement. Afin d'obtenir ces nouvelles capacités de modélisation, on a apporté des modifications substantielles aux fonctionnalités existantes et on a mis en œuvre de toutes nouvelles fonctionnalités. Finalement, les modifications et ajouts sur le plan des fonctionnalités offertes par MARS V2 ont eu une incidence sur presque tous les aspects de la conception fonctionnelle initiale. Par conséquent, le présent document a pour objet de documenter l'intégralité des fonctionnalités de MARS V2.

**Importance:** Le présent rapport est le principal document décrivant les capacités de MARS V2. Il permettra au client militaire et à l'analyste civil d'évaluer ce qui peut ou ne peut pas être réalisé avec l'application MARS actuelle.

# Table of contents

# List of figures

# List of tables

This page intentionally left blank.

# 1 Introduction

The Managed Readiness Simulator (MARS) is a versatile program that allows the user to quickly simulate a wide range of Canadian Forces (CF) readiness Scenarios to determine if the Resources of an Establishment are able to satisfy the requirements of a set of operational Tasks. The flexibility of MARS allows diverse operational tasks to be defined as processes composed of activities that place specific resource demands on the Establishment. The software also provides a graphical user interface that facilitates the creation and execution of simulation scenarios and the analysis of simulation output. The output analyzer allows the user to view aggregated simulation results and drill down to view the status of specific tasks and units over time. This provides the user with a powerful tool to anticipate problems that may arise and to identify their cause. Ultimately, MARS is intended to be used as a decision support tool for senior commanders of the CF. It provides them with forecasts of the impact of proposed changes to lines of operation, the Establishment, the readiness plan, CF policy, and other factors that may affect the CF's ability to satisfy operational demands and to maintain the health of the Establishment. A more detailed description of the motivation behind the development of MARS and its potential applications can be found in [1].

MARS is being developed by Defence Research & Development Canada - Center for Operational Research & Analysis (DRDC CORA) as one of its Applied Research Projects. DRDC CORA's Land Force Operational Research Team (LFORT) is managing the project and is using an evolutionary development approach where new capability is added incrementally to the existing model. This approach allows the model to be used in its current state with its existing capability while new capability is being added to the next version. It also allows the developers to learn from and refine the existing model while adding new capabilities to the next version.

The first version of MARS (termed MARS V1) was built as an initial prototype to demonstrate the managed readiness modeling concept. Previous publications [2, 3] document the design and implementation of MARS V1. Completed in 2007, its primary application was to assess the capability of the existing Army Establishment (the current supply of qualified personnel and equipment Resources) to meet the demands of planned operations. It was implemented as a Discrete Event Simulation (DES) using Rockwell's Arena software [4].

MARS V1 was successfully applied to a preliminary analysis of the Army's plans to generate the forces required for Task Force Afghanistan [5]. However, in the process of conducting this analysis, several aspects of the MARS V1 architecture and design were identified as factors limiting the potential of MARS to address future problems. In particular, modelling the dynamics of the Establishment (the creation, advancement and release of Establishment Resources), was not considered feasible within the MARS V1 software architecture and feature set.

The incorporation of a model of the dynamics of the Establishment is crucial to the full assessment of the readiness of the Establishment over time. The majority of deployable CF personnel and equipment resources spend most of their time in garrison preparing for deployed operations and serving roles that sustain the Establishment. For personnel, the ratio of time on deployment to time in garrison is typically 1:4 or 1:5. Therefore, a given resource can only be considered "ready" if, in addition to being qualified for a deployment, it has had sufficient time to perform these Establishment sustainment roles and to prepare for the deployment. In the case of

personnel resources, sustaining the Establishment means that resources must devote time to training in order to advance in qualification and progress toward promotion, and they must also provide training as instructors and serve in administrative and planning roles to maintain a functioning organization. Further, all personnel spend a portion of their time on various forms of leave for vacation, illness or injury. Ultimately, all personnel retire and this provides openings for junior personnel to rise through the ranks and for recruitment of new personnel. Thus, without the incorporation of models of recruitment, training, promotion, leave and attrition, which are the dynamic aspects of the Establishment, the assessment of the readiness of CF resources is incomplete.

The first step to addressing these limitations was to make significant improvements to the technology underlying the implementation of MARS V1. The analysis of potential technical approaches and an assessment of the chosen solution was documented in [6]. The re-implementation of MARS using this new software architecture produced the second major version of MARS (termed V2) which provided a more powerful platform on which the development of more advanced modelling capabilities was possible, including the incorporation of Establishment dynamics.

In order to achieve these new modeling capabilities, substantial modifications to existing functionality and the implementation of new functionality were carried out. In total, the feature changes and additions present in MARS V2 affected nearly every aspect of the original functional design. Therefore, the purpose of this paper is to fully document the functionality now available in MARS V2. For comparison, the modelling features of MARS V1 are documented in [2,3].

The following two sections, 2 and 3, provide a review of MARS concepts, terminology and components. Section 4 describes the specific aspects of the MARS V1 design that motivated the feature changes and additions implemented in MARS V2. Section 5 begins the detailed description of the MARS V2 feature-set with a discussion of Attributes which are used to define the properties of Organizations, Resources, and Slots (positions or roles that resources can occupy within an organization). Section 6 describes the structure of the Establishment and Theatre Organizations. Section 7 describes Tasks and Task Rotations. Section 8 describes the extensive functionality of Activities. Section 9 describes the life-cycle of an Activity as it moves through the simulation. Finally Section 10 provides concluding remarks.

# 2 MARS Concepts and Terminology

The MARS program is designed to simulate a given readiness *Scenario* by forecasting the ability of an *Establishment* to generate the *Resources* required to satisfy a set of *Tasks* occurring over time under a given set of conditions. The program also records the state of every Resource throughout the simulation; therefore the results can be used to determine the utilization level of a unit within the Establishment or of a specific group of Resources.

The program currently models three types of Resources: Personnel, Equipment and Facilities. Every Resource occupies a *Slot* in an *Organization* as shown in Figure 1. In general, the *Attributes* of each Slot define a particular *Resource Requirement* of the Organization and the Slot can only be occupied by a single Resource that satisfies the requirement. For example a Slot may assert that it can only be filled by a Resource that satisfies the criteria: *rank == Captain AND occupation == Infantry*. However, special *Overflow* Slots can also be built into the Establishment that are allowed to contain many Resources. These Slots are sometimes needed to store extra Resources that do not satisfy the Resource Requirement of any Slot or that cannot be assigned to a Slot because eligible Slots are currently occupied.

An Organization consists of a group of Slots that define the Resource Requirements of a unit. They are typically arranged in a hierarchical tree structure where only the terminal nodes of the tree contain Slots. There are two types of Organizations. Establishment Organizations define the units that contain the Resources available in the Scenario. Theatre Organizations define templates of the units required by the Tasks being modelled and do not contain any Resources. During the simulation, the program uses these templates to create a group of Slots that will be filled by Establishment Resources selected to perform a Task.

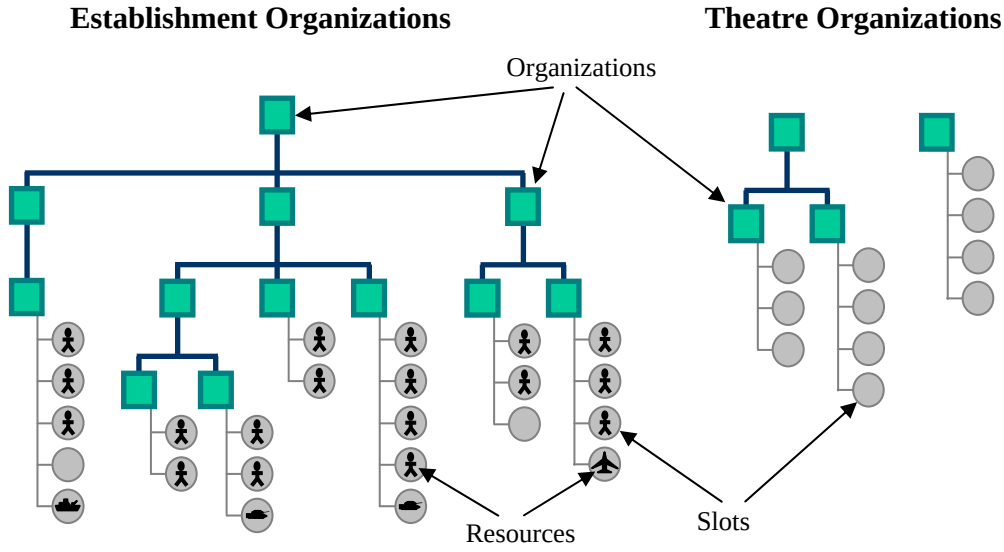


*Figure 1: Establishment Organizations and Theatre Organizations showing Slots and Resources*

Each Resource has Attributes that define its current state. Attributes store the information that determines whether a Resource can be chosen for a particular Task. Examples of Resource Attributes include a person's rank and qualifications. Other Attributes indicate whether the Resource is currently busy and whether there are any restrictions on what the Resource is allowed to do. In general, Attributes define the capability and availability of a Resource. These Attributes along with the Organization to which the Resource is attached are used to determine the eligibility of the Resource to be used by a given Task.

The operations and events being simulated in a MARS Scenario are represented by *Tasks* within the model. Each Task is broken down into *Activities* which are scheduled within the Task so that they will occur in a specified order. Activities are responsible for assembling the Resources they require. Activities can also be linked together to pass Resources from one Activity to another or to enforce dependencies. There are two types of Activities in the model:

1. A Process Activity temporarily employs Resources for a certain period of time and may alter their state upon commencement and completion.

1. An Event Activity changes the state of the selected Resources at a single point in time.

Each Activity is triggered in the simulation according to timing and Resource constraints. Activities simulate everything from training and operational Tasks to recruitment and retirement events or, if referring to equipment, acquisition and disposal events. The Activity construct is illustrated below in Figure 2.

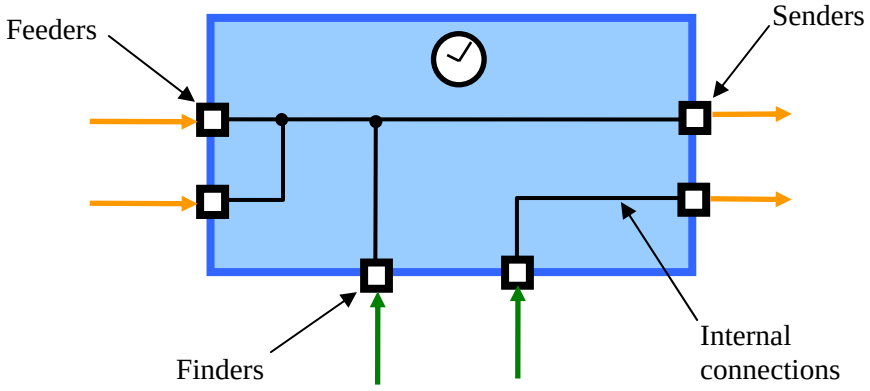


*Figure 2: The Activity Construct showing Feeders, Finders and Senders and internal connections.*

When an Activity is triggered and starts processing, it must acquire the Resources it needs to carry out its function. Resources enter an Activity as part of a resource group (*ResGrp*) through either a *Feeder* or *Finder* node and exit through a *Sender* node. An Activity may have multiple Feeders, Finders and Senders, and must have at least one Feeder or one Finder in order to act on at least a single ResGrp. A ResGrp is a set of Resources and Slots where each Resource occupies a single Slot and each Slot is either empty or occupied by at most one Resource. ResGrps are created by Finders which select Resources from the Establishment to participate in the Activity. Figure 3 illustrates the steps carried out by the Finders.

First, the Finder identifies the Theatre Organizations that contain the Slots that define the *Resource Requirement* for the Activity. For example, a Disaster Assistance Response Team Triage Unit might contain Slots for a medical officer, a medical technician, and a nurse.

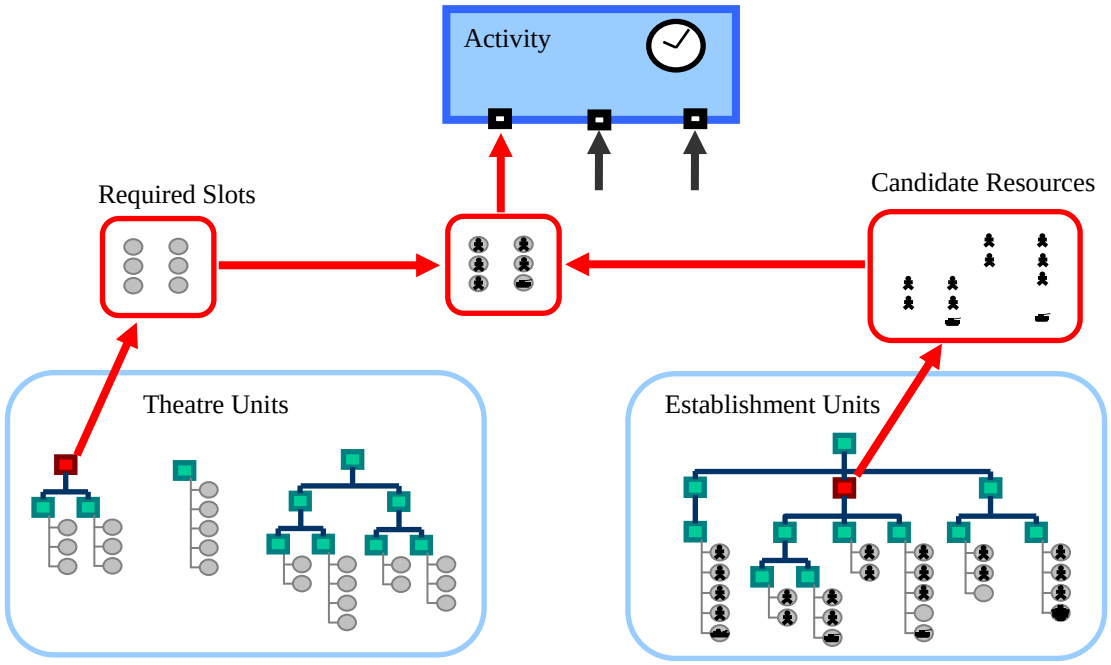


*Figure 3: Resource selection process used by an Activity Finder.*

Next, the Finder specifies a prioritized list of Establishment Organizations that may be searched to find Resources to participate in the Activity. This prioritized list is subject to constraints that can be used to limit the number of Resources taken from a given Organization and to filter for Resources with certain Attributes. The Finder also verifies that these Resources have not already been assigned to a conflicting Activity. This produces a list of *Candidate Resources*.

The Finder then attempts to fill each Required Slot with one of the Candidate Resources by comparing the Attributes of the Resources to the requirements of each Slot. If a suitable match is found, the Resource is assigned to the Slot and becomes part of the ResGrp being created by the Finder. To maximize the number of successful matches, the Finder attempts to assign the least qualified Candidate that meets the requirements of each Slot. For example, a Slot that can be filled by either a Major or Lieutenant Colonel will be preferentially assigned a Major because Lieutenant Colonels, being of higher rank, are in shorter supply and may be required by other Slots with more stringent requirements. This process is repeated for all Finders, with each Finder creating a ResGrp.

Activities also acquire Resources through Feeders which receive ResGrps that were created by a preceding Activity and passed on through one of that Activity's Senders. After acquiring its Resources through its Finders and Feeders, the Activity verifies that a specified minimum number of the required Slots have been filled. If this minimum requirement is not met, the Activity fails and the Resources are released. If sufficient Resources are found, the Activity takes control of

the selected Resources, altering their Attributes to reflect the nature of the Activity and employing them for the duration of the Activity.

Each Feeder and Finder is connected internally to a Sender. Upon Activity completion, each ResGrp is passed to a Sender which alters the Attributes of the Resources within the ResGrp to reflect the completion of the Activity. Each Sender is then responsible for either passing the ResGrps to the Feeder node of a follow-on Activity or for releasing the Resources within the ResGrp back to their Establishment unit. Senders and Feeders are the connection nodes that allow Activities to be linked together within a larger Task. When the Activity's processing time has finished and each ResGrp has exited through a Sender, the Activity is complete. Similarly, when all of the Activities of a Task are finished, the Task is complete.

To allow Tasks to be reused within a Scenario and to control when they begin, *Task Generators* are used to assign a start time to a Task. Multiple instances of a Task can be generated on a *Rotation* schedule to model the repetition of a Task such as a cycle of deployments that make up a continuous operation. When all of the Task Generators have been processed, and all of their associated Tasks are finished, the MARS Scenario is complete and the simulation stops.

From the outputs generated by the MARS simulation, the extent to which the Establishment was able to supply the Resources required for all the Tasks being modelled can be measured. More specifically, the output is analyzed to determine how successful each Finder was in creating its ResGrp from the Establishment. By aggregating the results from the Finders, the user can determine how successful the Establishment was in generating the required Resources for each Activity, Task, Task Generator, and the entire Scenario. Similarly, the states of the Resources within the Establishment can be tracked over time. These results can be combined to plot the state over time of a selected group of Resources, a unit or group of units, or the entire Establishment.

MARS is a versatile tool with many potential applications. Its strengths are its generic constructs that allow users to quickly simulate virtually any force generation Scenario and its outputs that provide users with the ability to aggregate and drill down into the simulation results to identify the causes of a particular outcome. The ability to forecast how successfully an Establishment can generate the forces required to satisfy both operational and sustainment demands will provide decision makers with invaluable information that currently is unavailable.

# 3 MARS Application Components

The MARS application consists of three major components, shown in Figure 4: a Scenario Database, a Managed Readiness Model, and a Graphical User Interface (GUI). The Scenario Database stores all the data that define a specific Scenario. The Managed Readiness Model is the discrete event representation of the process of selecting and employing Resources. The GUI allows the user to interact with the database and the model by performing three management functions. As the Input Manager, it is responsible for facilitating the transfer of Scenario data into the Scenario database. The input data consist of the Tasks, the Establishment, plans, and policies to be modelled in the proposed Scenario. The data may be input directly through the GUI input screens or imported from an external source such as a Corporate database or spreadsheet. As the Simulation Manager, the GUI controls the execution of the Simulation Scenario. Finally, as the Output Manager, it allows a user to analyze the simulation results and to generate output reports.

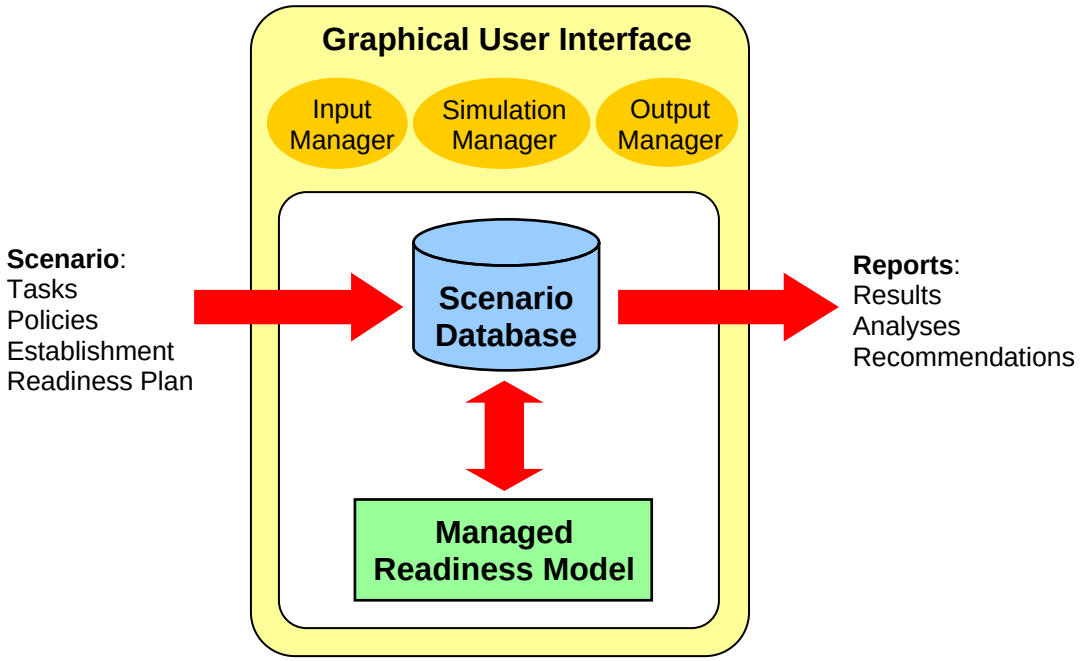


*Figure 4: Components of the MARS Application with data transfer shown as red arrows.*

# 4 Motivation for Feature Changes and Additions

Significant changes were made in MARS V2 to address limitations in the original functional design and to add the features required to achieve the next development goal of modelling the dynamics of the Establishment. The specific aspects of the MARS V1 design that motivated undertaking these feature enhancements are described below. A detailed description of the MARS V1 functionality discussed in this Section can be found in [2].

1. The Establishment Organization, its Slots and the assignment of Resources to those Slots were static over the course of the simulation. Therefore, it was not possible to move Resources from one Slot to another, nor to create new Resources entering the Establishment, nor to retire Resources from the Establishment. In order to support Establishment dynamics, all these features were required.

2. Only limited operations on Attributes were supported and these operations were only available for Resource Attributes. Attributes that applied to Slots and Organizations were not updatable. The only comparative operation that the Resource Attributes supported was testing for equality. For example, a Theatre Slot in an Activity could specify the requirement, *Rank = Captain*, but it could not specify the requirement, *Years of Service > 5*. Similarly, the Attribute updating instructions used by Feeders, Finders and Senders only supported setting Resource Attributes to a specified value. It was not possible to increment or decrement an Attribute value. For example, it was not possible to implement a *Years of Service* Resource Attribute because this Attribute must be incremented once per year. Therefore, it was necessary to expand these Attribute operations to add greater flexibility and to generalize Attributes so they can also be used for Slots and Organizations in a dynamic Establishment.

3. The mechanism that allowed different Candidate Resources to be drawn from different Establishment units depending on the current Task Rotation was overly complicated. This Rotation mechanism consisted of three cycles at the area, unit and sub-unit levels. Each Finder would search the Establishment for Resource Candidates based on the positions of the area, unit and subunit cycles. Various rules determined when the cycles would rotate to search different parts of the Establishment for Resource Candidates. However, this method of managing Rotations proved difficult to work with. First, it forced the Establishment to be organised into area, unit, and subunit levels. This is not generic as other applications may have different Establishment structures. Second, because of the independence of the three cycles and the complexity of the rules that determined when each cycle would rotate, it was difficult to determine exactly which units would be tasked for specific Activities. In practice, the Tasking of units to future Activities is set out in the CF planning process, but it was difficult to rig the Rotation mechanism such that these units would in fact rotate into position at the appropriate time in the simulation. The desired functionality was the ability to explicitly identify Establishment Organizations for each Finder in each Task Rotation.

4. The specification of Theatre units and Establishment units used by Finders to assemble ResGrps was restrictive. In MARS V1, each Activity Finder specified a single Theatre unit defining the Resource Requirement and a single Establishment unit, depending on the state of the Rotation mechanism, to supply Resources for the Activity. However, it is often necessary to be able to identify multiple Theatre and Establishment units and to refine the list

of Slots and Resources that are supplied. This would give Finders more precise and flexible control over the creation of ResGrps that participate in simulation Activities.

5. The method of specifying Establishment units to supply Resources to Activities prevented generalization of Finders and Activities. By requiring the Finder to direct which Establishment unit should supply the Resource Candidates through the Rotation mechanism, the Finder becomes fixed to a particular Task. Other Tasks must draw their Resources from different Establishment Organizations on a different Rotation schedule. Thus, Finders, and by extension, the Activities they belong to were not portable across different Tasks. For example, a training Activity that grants a qualification could not be inserted arbitrarily in any Task where that training was needed because the Finder specified not only the number and type of Slots available on the course but also who in the Establishment would attend, which varies from Task to Task. Thus, to train a different set of Resources in a different Task on the same course, a different Activity and Finder was typically required. In order to generalize Finders and Activities so they can be placed in any Task, the specification of which Establishment Organizations will supply Resources to Finders and Activities should be controlled by the Task.

In order to address these issues, nearly all aspects of the MARS functional design were affected. The following sections (5 through 9), will provide a complete description of the functionality of MARS V2 starting at the most fundamental level, Attributes, and proceeding to the high-level process that manages the Activity Lifecycle.

# 5 Attributes

Attributes are generic variables that belong to Organizations, Slots and Resources and define their states. Attributes for these objects can be created by the user and used in the simulation. Each Attribute may restrict itself to be used only if another Attribute holds a certain value. This allows an object to adopt the correct Attributes depending on the value of some other Attribute. For example, personnel, equipment and facilities are all considered Resources but each has different Attributes. Each of these types of Resources has an Attribute called 'Resource class' whose value indicates whether the Resource is a person, a piece of equipment, or a facility. Other Attributes specifically associated with personnel (rank, occupation, qualifications etc.) are then attached to the Resource if and only if the Resource class value is set to personnel. The same scheme is applied to the Attributes of equipment and facilities.

The types of Attributes supported in MARS V2 are shown in Table 1, and a more detailed description of each Attribute type follows below.

*Table 1: Description of Attribute Types available in MARS V2.*

| Attribute Type | Description |
|---|---|
| 1 | A single value from a set (for Resources) |
| 2 | A list of Attribute Requirements (for Slots) |
| 3 | An integer value |
| 4 | A real value |
| 5 | A list of values from a set |
| 6 | A Slot ID |
| 7 | A Resource count |
| 8 | A single value from a set (for Organizations) |

A Type 1 Attribute is used for Resources and can hold one value from a set of possible values. For example, the value of a personnel rank Attribute can be one of Private, Corporal, Master Corporal, Sergeant, etc.

A Type 2 Attribute is used for Slots and contains the ID of a set of Attribute Requirements that restrict which Resources can occupy the Slot. A typical personnel Slot will have five Type 2 Attributes that define the required class, rank, occupation, component and qualifications. Attribute Requirements are described in more detail in Section 5.1.

Attribute Types 3 and 4 are used for regular integer and real values respectively.

A Type 5 Attribute is a list that can contain multiple values from a set of possible values. Individual values can be added to or removed from the list. For example, the personnel qualifications Attribute stores all qualifications that an individual has attained over time in a Type 5 Attribute.

A Type 6 Attribute is a special Resource Attribute that stores the Establishment Slot ID that a Resource currently occupies. This Attribute is partially managed by the simulation to perform special functions that will be described in greater detail below.

A Type 7 Attribute is a special Slot Attribute that stores the number of Resources currently occupying a Slot. In most cases this value is 0 for an empty Slot or 1 for a filled Slot, but special Overflow Slots are allowed to hold more than one Resource at a time. While this Attribute holds an integer value, it is given its own Type because, unlike a Type 3 Attribute, it is updated exclusively by the simulation.

A Type 8 Attribute is used for Organizations and can hold one value from a set of possible values. For example, the Establishment or Theatre designation of an Organization is stored in a Type 8 Attribute. These Attributes cannot be changed during a simulation.

A wide range of functionality is built around these Attributes. Activities can update the Attributes of Resources that pass through their Feeders, Finders and Senders; Organizations, Slots and Resources can be filtered based on their Attributes; and Resource Attributes can be matched to Slot Attribute Requirements in order to fill Slots with qualified Resources. These functions will be discussed in more detail in the following sections.

## 5.1 Attribute Requirements

Attribute Requirements define criteria for acceptable Attribute values. They are used in several parts of the MARS managed readiness model:

1. Slots define what Resources are allowed to occupy them using Type 2 Attributes that point to Attribute Requirements.

2. Lists of Candidate Resources that Finders assemble to participate in Activities are filtered using Attribute Requirements.

3. Lists of Slots that define the Resource Requirements of an Activity are filtered using Attribute Requirements.

Attribute Requirements that act on specified Attributes can be activated and deactivated. For example, the simulation developer may want to compare the effect of running a simulation with

qualification requirements turned off and then on. This can be achieved by activating or deactivating the qualification Attribute.

An Attribute Requirement consists of a set of criteria describing acceptable Attribute values. For example, a rank requirement might be defined as *Rank == Captain OR Rank == Major*. Each individual criterion consists of an Attribute, a Boolean operator, and a value. During a comparison, the specified Attribute of a simulation object is tested against the given value using the Boolean operator. All the criteria in the set are then specified as being related by a logical AND or a logical OR.

In most cases, the criteria provide a list of acceptable values, in which case they are related by a logical OR as in the rank example above. If the Attribute Requirement specifies criteria for a list Attribute (type 5), such as personnel qualifications, then the criterion *Qual == Blackbelt* means Blackbelt must be a member of the Resource's Qual list. Similarly, the criterion *Qual <> Chessmaster* means Chessmaster must not a member of the Qual list. Criteria that apply to list Attributes are typically related with a logical AND to indicate that a specific combination of values must be present in the list, such as a list of pre-requisites for a course Activity.

Typically, several Attribute Requirements, each consisting of multiple criteria, are applied at once, and it is assumed that they must all be true. Thus multiple Attribute Requirements are related with a logical AND.

The comparative operators equal (==) and not equal (<>) are supported for Attribute Requirements on all Attribute Types. For numerical Attributes (Types 3 and 4) the full range of comparative operators are available (==, <>, <, <=, >, >=). Table 2 describes the effect of applying each of the operators in comparing an Attribute, *A*, with a value, *v*.

The behaviour of an Attribute criterion can be modified by adjusting a parameter called its Variable ID. The standard behaviour is to compare an object's Attribute to the value specified in the criterion using the specified operator. But, the Variable ID can be used to specify that any Attribute value is acceptable (equivalent to ignoring the criterion), or it can be used to indicate that a probability distribution should be used to determine the value of the criterion at runtime. This is used in some cases where it is not known ahead of time what type of Resource will occupy a Slot. For example, a certain proportion of Slots on deployed operations are typically filled by reserve force members. It is not typically known which Slots will be filled by reserve personnel, only that some Slots can be filled by reserve personnel and an overall percentage is usually targeted. To simulate this behaviour, the Variable ID can be used to determine the value of the criterion probabilistically at runtime. Thus, for the simple criterion, 'Attribute == value,' the value can be selected randomly at runtime from a list of possible values each having a specified probability. For example, to model a 20% chance that a particular Theatre Slot will require a reserve force member, the component criterion for a Slot would be defined as in Equation 1, below.

$$component == \begin{cases} regular\ force,\ 80\%\ of\ the\ time \\ reserve\ force,\ 20\%\ of\ the\ time \end{cases} \quad (1)$$

*Table 2: Description of the effect of each comparative operator on each of the Attribute types*

| **Attribute Type** | **Comparative Operator** | | | | | |
|---|---|---|---|---|---|---|
| | == | <> | < | <= | > | >= |
| 1 | *A == v* | *A <> v* | *n/a* | *n/a* | *n/a* | *n/a* |
| 2 | *A == v* | *A <> v* | *n/a* | *n/a* | *n/a* | *n/a* |
| 3 | *A == v* | *A <> v* | *A < v* | *A <= v* | *A > v* | *A >= v* |
| 4 | *A == v* | *A <> v* | *A < v* | *A <= v* | *A > v* | *A >= v* |
| 5* | *A.mem(v)* | *A.notmem(v)* | *n/a* | *n/a* | *n/a* | *n/a* |
| 6 | *A == v* | *A <> v* | *n/a* | *n/a* | *n/a* | *n/a* |
| 7 | A == v | A <> v | n/a | n/a | n/a | n/a |
| 8 | A == v | A <> v | n/a | n/a | n/a | n/a |

*For list Attributes, *A.mem(v)* means *v* is a member of list *A*, and *A.notmem(v)* means v is not a member of list *A*.

## 5.2 Attribute Updating

Feeders, Finders and Senders have the ability to update the Attributes of Resources that pass through them as they enter or leave an Activity. Feeders and Finders perform their Attribute updates before the Activity runs; Senders perform their updates after the Activity finishes whether it succeeds or not.

These Attribute updates are accomplished with a series of instructions associated with the Feeder, Finder or Sender. Each instruction consists of an Attribute to be updated, a value and an operator. The supported operators are: =, +, and -. These operations and their effects on the various Attribute types used for Resources are summarized in Table 3.

Each Resource has a special Type 6 Attribute that contains the Establishment Slot ID that the Resource currently occupies. This Attribute is partially controlled by the simulation and partially controlled by the simulation developer. In order to model a dynamic Establishment, Resources must be recruited into, advance through, and eventually retire from the Establishment Organization. This means the Resource Slot Attribute must be updated during the simulation to reflect the Resource's movement within the Establishment.

*Table 3: The effect of the Attribute update operators on Resource Attribute types, where the Attribute, A, is being updated with the value, v.*

| **Attribute Type** | **Update Operator** | | |
|---|---|---|---|
| | = | + | - |
| 1 | *A = v* | *n/a* | *n/a* |
| 3 | *A = v* | *A = A + v* | *A = A - v* |
| 4 | *A = v* | *A = A + v* | *A = A - v* |
| 5* | *A.append(v)* | *A.append(v)* | *A.remove(v)* |
| 6 | *A = v* | *n/a* | *n/a* |

*For type 5 Attributes, *A* is a list of values, thus *A.append(v)* means to append the value, *v*, to the list, *A*. Likewise, *A.remove(v)* means to remove *v* from *A*.

Newly created Resources are explicitly assigned to a new Resource Slot using standard Attribute updating instructions. Similarly, retiring Resources are explicitly assigned to a retired Resource Slot. But the movement of Resources within the Establishment is a more complicated process that requires special Attribute updating functionality. In order to move Resources from Slot to Slot within the Establishment, an Activity Finder builds a list of empty Establishment Slots and builds a list of Resource Candidates to fill those Slots. The Finder then attempts to fill each empty Slot with one of the Candidates using the normal matching process. But in order to complete the move, the Slot Attribute of each matched Resource must be updated to the new Slot ID that it occupies. To achieve this, the simulation developer adds a special value to the Slot update instruction, *Slot = current Slot ID*, that tells the simulation to update the Resources' Slot Attributes according to the new Slots they were matched to.

## 5.3 The Resource Utilization Level (RUL) Attribute

MARS allows individual Resources to simultaneously belong to several ResGrps. Each of these ResGrps may take part in a different stream of Activities whose timing may overlap. Without the inclusion of constraints, this functionality can lead to the utilization of a Resource beyond its intended capacity. To protect against over-utilization, each Resource has a Type 1 Attribute called its Resource Utilization Level (RUL), which can take on a value of 0 (unutilized) or 1 (fully utilized).

Each Activity specifies RUL Attribute update instructions for its Feeders, Finders and Senders reflecting whether or not the Activity takes up the time of participating Resources. For example, a training Activity fully occupies the time of participating Resources for the duration of the Activity so it would increment the RUL Attributes of those Resources to 1. Whereas a waiver Activity alters waivers-related Attributes but does not actually occupy the time of the

participating Resources so it would increment the RUL Attributes of participating Resources by 0.

Once a Finder has assembled a preliminary list of Resource Candidates for an Activity, it checks each Candidate for RUL conflicts by calculating its total RUL over a future time horizon assuming the Resource participated in the Activity. This calculation is based on existing Activity sequences of which the Resource is already a part and the new Activity sequence that will result from participation in the current Activity. The Activity sequences that the Resource will participate in are known based on the Activity-to-Activity connections defined by the Task. Resources are removed from the Candidate list if assignment to the Activity will cause their RUL to exceed 1 at any time in the future. Figure 5 shows an example of a Resource being considered for participation in Activity C, which is followed by Activity D in the future. However, the Resource's prior assignment to Activities A and B results in an RUL conflict in the future between Activities B and D. This Resource would therefore be rejected as a Candidate for Activity C.

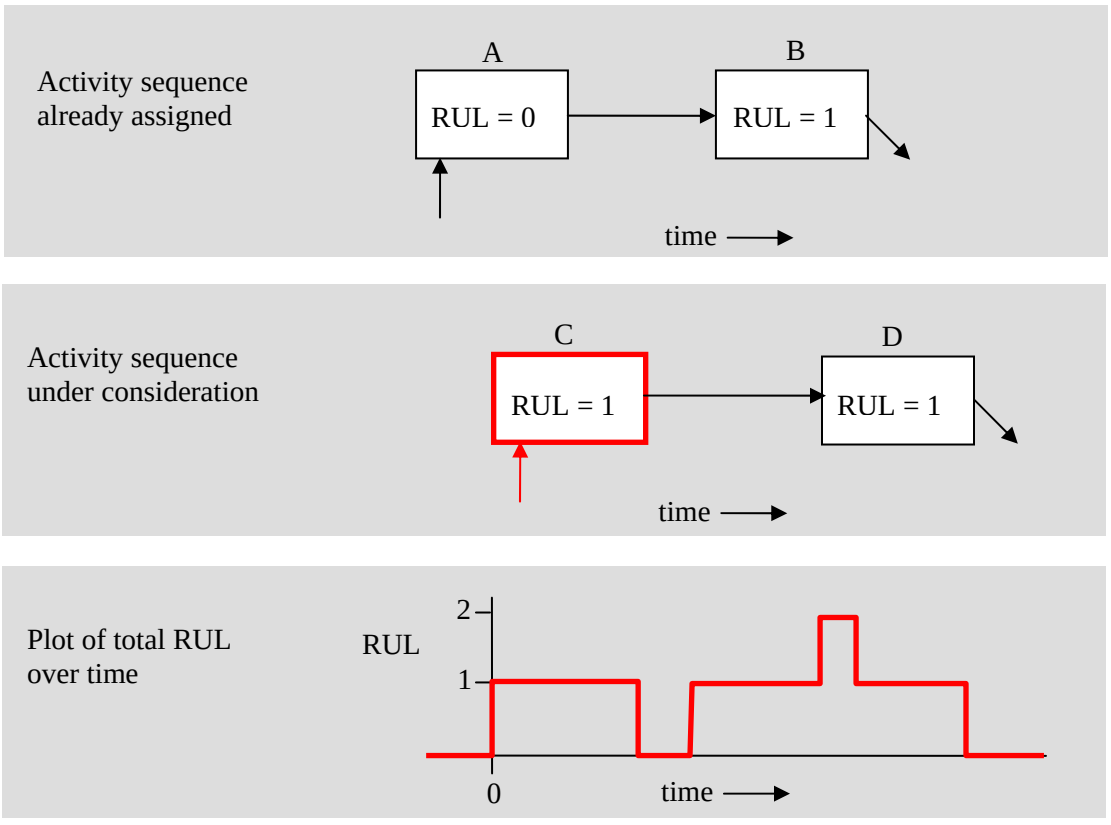


*Figure 5: RUL verification for a Resource who is being considered for Activity C, which is followed by Activity D, and who has already been assigned to Activities A and B.*

# 6 Establishment and Theatre Organizations

The main function of the MARS managed readiness model is to select available and qualified Resources in the Establishment and assign them to Activities. Therefore, MARS operates on two main data structures: the Establishment Organizations which define the available Resource supply and the Theatre Organizations that define the Resource Requirements of Activities. In MARS, Organizations form a hierarchical structure. Each Organization unit has Attributes that indicate its type (Establishment or Theatre), branch, hierarchy level, and component. Each Organization unit also points to a parent unit such that the Organizational hierarchy is represented as a tree structure. Multiple root units (with no parent) are allowed within in the Establishment and Theatre Organizations.

Terminal units in the Organization trees contain Slots. A Slot is a generic term used to describe a defined Resource Requirement which may represent a required piece of equipment such as a LAV III, a personnel requirement which is normally called a position, or a facility requirement such as a firing range. Each Slot has an Attribute that indicates which Organization unit it belongs to, timing information indicating when the Slot is active, a count of the number of Resources occupying the Slot and Attribute Requirements that define the criteria that must be met by any Resource that occupies this Slot.

The timing information is used to simulate planned changes to the Establishment structure. For example, in order to model the recent growth plans of the CF, new Slots must be added to the Establishment over the course of the simulation. The timing information allows the simulation developer to specify that certain Slots will come online at specified times during the run.

Every Resource in MARS occupies a Slot in an Establishment Organization. At any given time, most Establishment Slots are occupied by a single Resource, some Slots are empty, and special Overflow Slots may be occupied by multiple Resources. The Overflow Slots are used to accommodate an excess of Resources assigned to an Establishment unit or Resources that do not meet the requirements of any Slot within the unit. The availability of Overflow Slots facilitates the setting up of initial conditions because, in the real world, it is not guaranteed that the Resources currently present in an Organization exactly match the Slots that the Organization contains. Figure 6 illustrates the structure of an Establishment Organization.

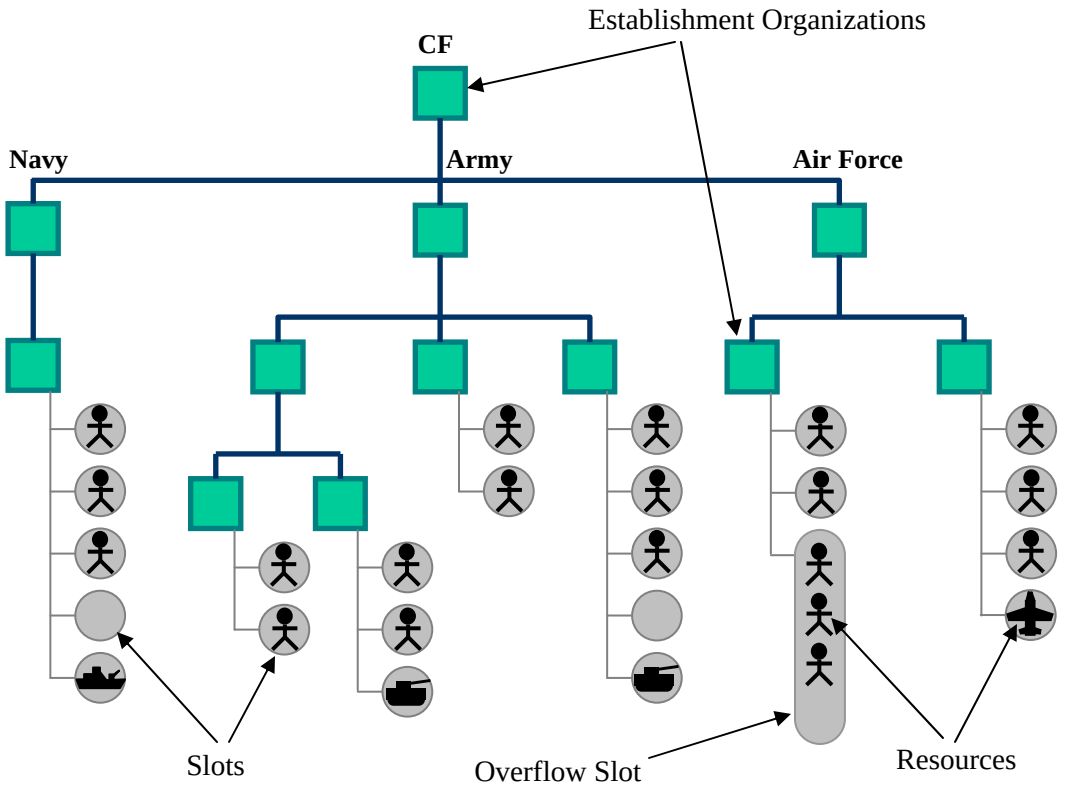


*Figure 6: Example Establishment Organization structure with Slots and Resources*

In contrast, the Theatre Organization Slots are never directly filled by Resources. Instead, each Theatre Organization acts as a template defining the Resource Requirement for an Activity that is part of an operation. This means that an Activity will identify a Theatre unit or combination of Theatre units and create its own copy of the list of Slots attached to those units. It will then attempt to assign Resources to the copied Slot list rather than the original Slots in the Theatre Organization. This allows multiple Activities to independently and concurrently assemble Resources using the same Theatre unit as a template. Each copied instance of a Theatre Slot can only be filled by a single Resource. Figure 7 illustrates the structure of a Theatre Organization.

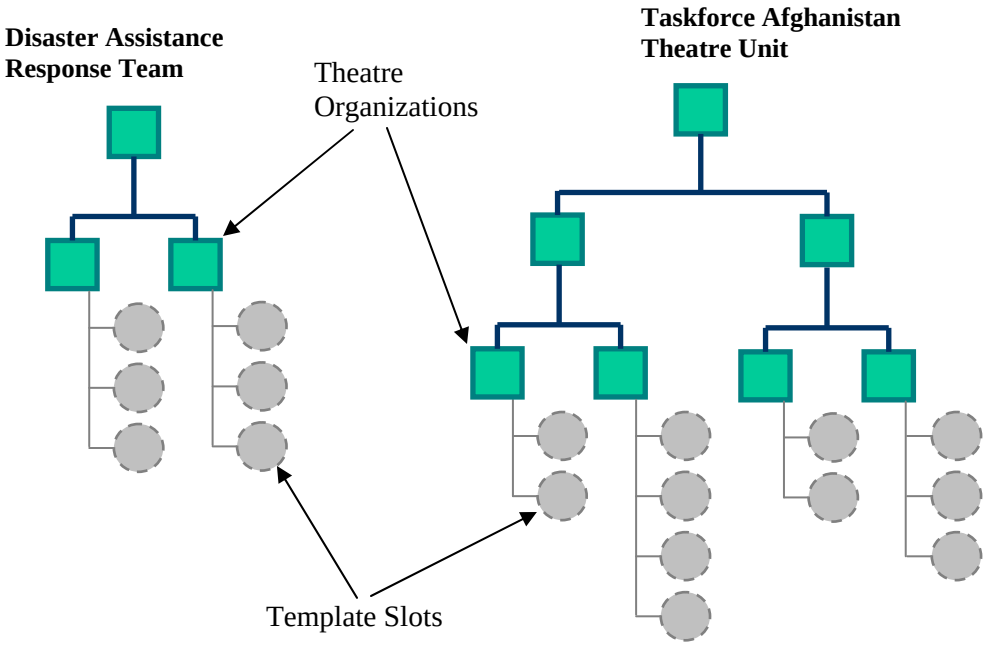


*Figure 7: Example Theatre Organization structure showing Slot Templates*

# 7 Tasks and Rotations

A Task is a collection of related and possibly interconnected Activities that make up a part of planned operations. Tasks may be repeated according to a schedule in which case each instance is referred to as a Rotation. The number of Rotations to be generated is specified by the user and the scheduling of each Rotation can be explicitly defined or can be generated according to a probability function. The available probability functions for non-deterministic Rotation scheduling are exponential, normal and constant. The instantiation of each Rotation of a Task on the appropriate schedule is managed by a Task Generator. Individual Task Generators can be activated or deactivated by the user to control what parts of the model will be used in a given Scenario

Each Task defines a set of Activities and how they will behave in the simulation. The Task gives each Activity timing constraints indicating start-no-earlier-than (SNET) and start-no-later-than (SNLT) times relative to the Task start time. The Task defines the connections between Activities which direct the movement of Resources from one Activity to the next. Resource passage between connected Activities imposes the constraint that the source Activity must complete before the receiving Activity starts. The Task can also specify Activities that are dependent on other Activities without the transfer of Resources. Dependencies impose the constraint that an Activity's dependants must complete before it starts. For example, some of the Activities involved in the simulation of establishment dynamics must be executed in a particular order but do not involve the transfer of resources. On an annual cycle, first an attrition Activity runs, creating vacancies in the Establishment Organization. Then a promotion activity pulls qualified resources into vacant slots one rank level up starting from the highest rank and proceeding to the lowest. Then a recruitment activity fills vacancies at the lowest rank level. The correct execution order of these activities is controlled by making recruitment depend on promotion and making promotion depend on attrition. Figure 8 illustrates an example of a Task composed of multiple Activities with timing information, connections and dependencies.

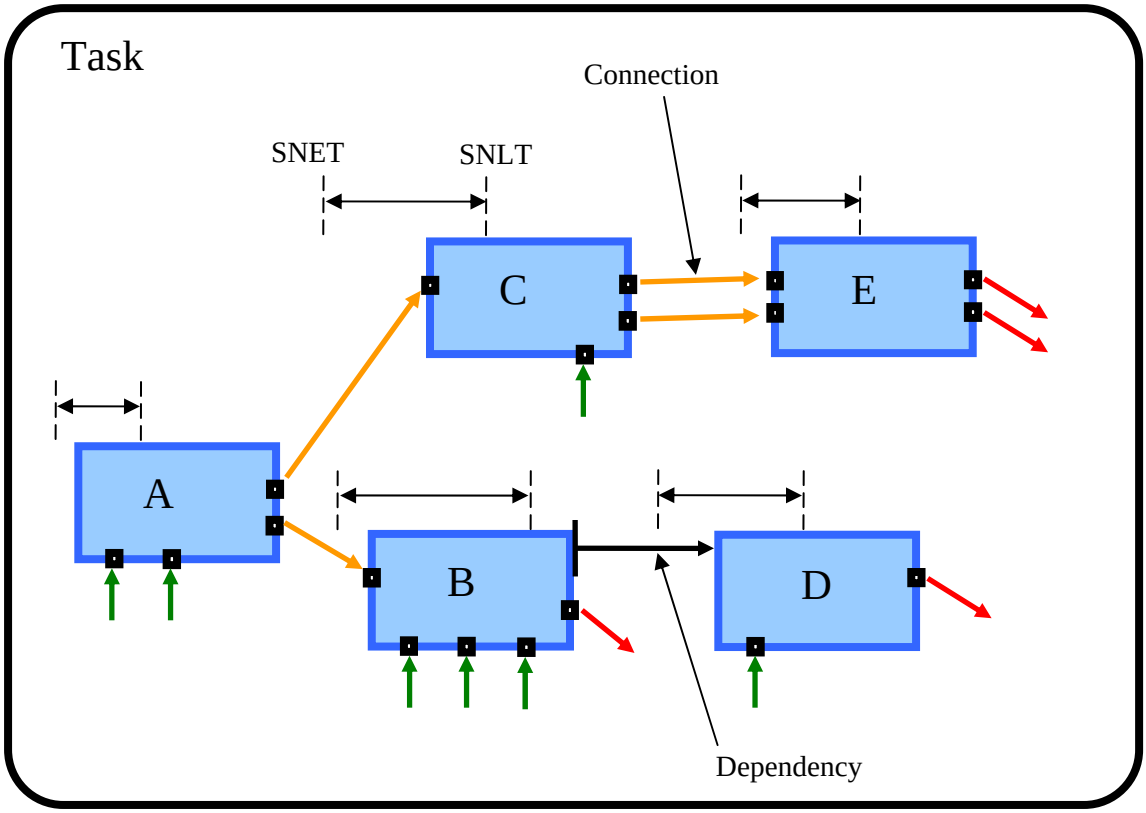


*Figure 8: A Task composed of multiple connected Activities (A through E) with SNET and SNLT timing constraints and a Dependency constraint between Activities B and D.*

Typically, the purpose of supporting multiple Rotations is to run the same Task multiple times, each time drawing Resources from different parts of the Establishment Organization. Therefore, the list of Establishment Organizations that supply Resource Candidates to a particular Activity Finder is actually one of several lists in a sequence. As each Task Rotation is instantiated, it is assigned a new sequence number so that it draws its Resource Candidates from the next list of Establishment Organizations in the sequence.

# 8 Activities

Activities are the primary building blocks of a MARS simulation and each Activity carries out a complex process which will be described in this section. The role of an Activity is to collect the Resources it requires, to employ those Resources for some time and then to pass them on to subsequent Activities or release them. An Activity assembles its required Resources through entry nodes called Feeders and Finders. Feeders receive ResGrps from preceding Activities. Finders assemble new ResGrps by searching for and selecting Resources from the Establishment Organization. When the Activity completes, exit nodes called Senders either pass the ResGrps on to subsequent Activities or return them to the Establishment. All entry and exit nodes are able to adjust the Attributes of Resources that pass through them.

## 8.1 General Attributes of Activities

Each Activity has Attributes indicating its type, Resource utilization level (RUL) and duration. The type is a Boolean variable indicating whether the ResGrps created by Finder nodes should be saved for potential passage to subsequent Activities. This concept was introduced to reduce the data output storage requirements of the model. The normal operation (called a Type 1 Activity) is to create ResGrps at Finders, but in some instances it is convenient to assemble a collection of Resources only to update their Attributes at the Finder without taking any further action (called a Type 2 Activity). For example, an Activity that simulates attrition would search randomly for a certain percentage of Resources across the Establishment and update their status to show that they have released, but this ResGrp will not participate in any future Activity so it is not saved.

The RUL is a Boolean value indicating whether the Activity precludes any participating Resources from also participating in a concurrent Activity. Activities with an RUL of 0 do not occupy the time of participating Resources. Activities with an RUL of 1 fully occupy the time of participating Resources for the duration of the Activity. Each Activity will have corresponding Attribute update instructions at its Feeders, Finders and Senders to ensure that the Resource RUL Attributes are updated accordingly. In general Resources can participate in any number of concurrent RUL-0 Activities and one RUL-1 Activity at a time.

The duration indicates the amount of time for which the Activity will employ its assembled Resources. This is the time during which participating Resources are considered occupied for RUL-1 Activities. The duration is measured from the time that Resources are assigned to the Activity. Once this time passes, the participating Resources are routed to the Senders and from there to subsequent Activities or to the Establishment.

## 8.2 Initial conditions

In order to establish the initial conditions of the simulation, it must be possible to create Activities that are already in progress and employing a group of Resources. It must also be possible to pre-assign Resources to future Activities where Resources have already been specified by CF planners. To accommodate these capabilities, the ResGrps that are created by Activity Finders can be pre-filled with Resources before the simulation starts. These Resources are guaranteed to

participate in the Activity when it occurs within the simulation. Therefore, Activities that are already in progress will acquire fully pre-filled ResGrps at the start of the simulation and employ them for the remainder of their duration. For future Activities, ResGrps may be fully pre-filled, partially pre-filled, or empty depending on data available from the readiness planning process. Fully pre-filled ResGrps are employed as is by the Activity. Partially pre-filled ResGrps can have additional Slots and Resources added to them by the Finder's Resource selection process. Empty ResGrps must proceed through the Finder Resource selection process to supply Resources to the Activity.

## 8.3 Slot and Resource Limits

At various points during the Resource selection process, the number of selected Slots or Resources can be limited to a certain number to model situations where the number of Slots or Resources needs to be restricted. For example, an Activity representing a course may be able to accept exactly 10 Candidates from a certain Establishment unit, regardless of the total number of qualified personnel available in that unit. Or an Activity that simulates injuries, illness and leave may need to periodically select a certain percentage of the Resources in an Establishment unit and set their status to Left out of Battle (LOB). Or an Activity representing attrition may need to randomly select personnel Resources according to a known annual attrition probability. Limits provide this functionality.

Three types of limits are available to be applied to collections of Slots or Resources: number, percentage and chance. A number limit randomly selects objects to remove from the collection until a specified number remain. A percentage limit randomly selects objects to remove from the collection until a specified percentage of the original count remain. A chance limit performs a Bernoulli trial with a specified chance of success on each object in the collection and removes the successes from the collection.

## 8.4 Feeders

Feeders receive pre-assembled ResGrps from preceding Activities. Each connection to a Feeder originates at a Sender from a preceding Activity. Each preceding Activity routes all its ResGrps through its Senders at the same time, and multiple ResGrps may be passed through a single connection. Thus, when the Activity has received at least one ResGrp per connection per Feeder, it is guaranteed that all ResGrps have been received. It is the responsibility of Sender nodes to route Resources to Feeders, so the Activity must wait until all connections to all Feeders have each received at least one ResGrp.

When the Feeders have received all the required ResGrps, each Finder performs its assigned Attribute updating instructions on the members of the ResGrps entering the Activity.

## 8.5 Finders

Finders assemble Resources from the Establishment to participate in Activities. The typical functionality (termed a Type 1 Finder) is to match Candidate Resources from the Establishment to a Resource Requirement defined by a selection of Theatre Organizations. However, in some

cases the Resources required by an Activity are already properly assembled in the Establishment Organization structure. Thus a second type of Finder (termed Type 2) assembles the Resources as they appear in the selected Establishment Organization. For example, an Activity that selects certain personnel Resources from the Establishment to be in a LOB state does not need to match these Resources to Theatre Slots, it simply processes all the Resources in the selected Establishment Organizations. In special cases, the function of a Finder is to create Resources rather than to select them from the Establishment. This is termed a Type 3 Finder and is used to create new Resources, such as new recruits, during the simulation in order to model the dynamics of the Establishment. In this case, the created Resources are processed without matching to a set of required Slots.

### 8.5.1 Required Slot List

The first step for a Type 1 Finder is to build the list of required Slots. (Types 2 and 3 Finders proceed directly to building the list of Resources from the selected Establishment Organizations.) This process is shown in Figure 9. Typically the Slot list is derived from a selection of Theatre Organizations, but it can also be created from a pre-existing ResGrp that already has Slots in it, or it can be created from an Establishment Organization. Thus each Type 1 Finder specifies a list of Theatre Organizations, Establishment Organizations and/or ResGrps. This list is divided into groups having their own refinement parameters. Refinement is optional and consists of applying a filter and/or a limit. For each group, the Organization units are searched to find all Slots belonging to these units and their descendants in the Theatre and Establishment Organizations. If a ResGrp is specified, then the Finder selects the Slots belonging to the ResGrp. This forms the preliminary Slot list for the group.

The Finder then performs various actions to reduce this preliminary list. First, it removes Slots whose timing information indicates that the Slots are not active. Next, it removes any Slots that are already present as a pre-filled Slot in the Finder's ResGrp. If a filter is specified, the Finder removes any Slots from the preliminary list that do not meet the filter criteria. If a limit is specified, the Finder limits the number of Slots according to the limit parameters. These steps allow the simulation developer to adjust the composition and number of Theatre Slots derived from the originating Organization units and ResGrps.

This process of building and refining Slot lists is repeated for each group of Organization units and ResGrps specified by the Finder. The remaining Slots from each pass are appended to a final Slot list. Once all the groups of Organization units and ResGrps have been processed, the Finder applies a second limit to the final Slot list to control the total number of Slots that the Finder will then attempt to fill.

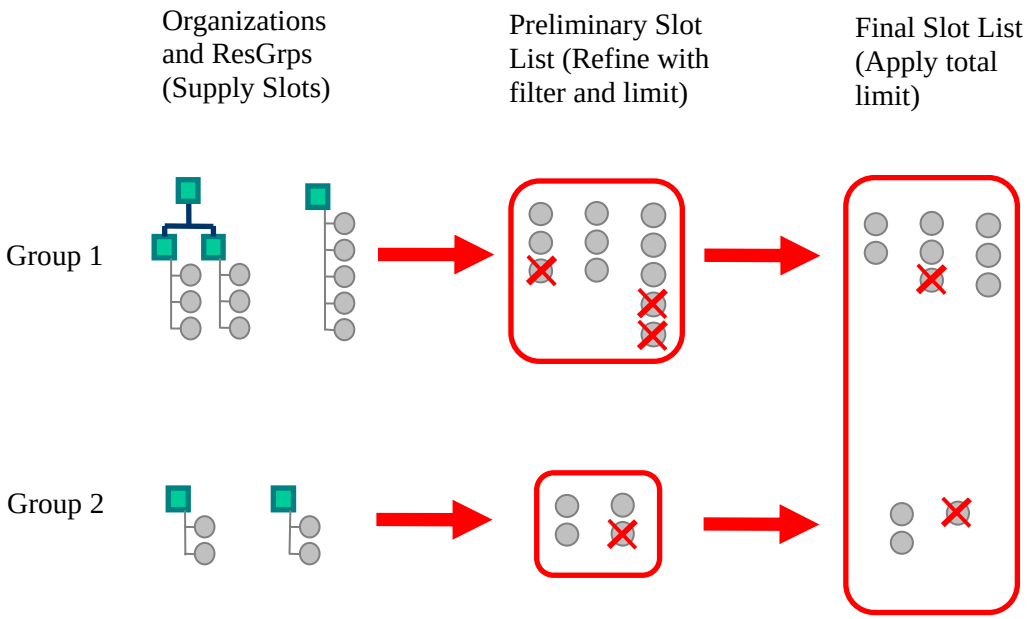


*Figure 9: The Slot selection process showing two groups of Organizations and ResGrps that supply Slots that are refined to form the final Slot list.*

Once the final Slot list is created, each Slot is checked to see if any of its Attribute Requirements depend on a probability distribution. If a probabilistic Attribute Requirement is found, then the value of the Attribute criteria is selected randomly from the list of options according to the specified probabilities. Having resolved any probabilistic Attribute criteria, the final version of the Attribute Requirement rules for each Slot are saved for the matching process.

Slot Attribute Requirements can also indicate that the Slot should be automatically filled by a Generated Resource. This feature is used when certain parts of the Establishment are not explicitly modeled in a given Scenario. For example, if it is not important to the results of the study that the Reserve Force be explicitly modeled, their availability can be assumed without modelling individual Reserve Resources. Therefore, before proceeding, the Slot list is scanned to see if any of the Slots have Attribute Requirements that indicate that the Slot should be immediately filled by a Generated Resource. Any such Slots are recorded as filled in the ResGrp (though no Resources actually occupy them) and are removed from the list of Slots that proceed to the matching step.

Finally, because some Slots are much harder to fill than others, the Slot list is sorted in decreasing order of the expected difficulty of finding Resources that meet the Attribute Requirements of the Slots. These expected difficulty ratings for each Attribute value are defined by the user by assigning a weight to each Attribute value. Thus the Slots that are hardest to fill (because of a high rank, unusual occupation, or rare qualification requirement) will be matched first.

### 8.5.2 Resource Candidate List

In the next phase, the Finder builds a list of Resources that may participate in the Activity. For Type 1 Finders, these Resources are Candidates that can be used to fill the list of required Slots. Only those Resources that are matched to a Slot will participate in the Activity. For types 2 and 3 Finders, all the Resources found in this step will participate in the Activity.

In most cases a list of Establishment units is specified to supply the Resources. However, pre-existing ResGrps may also be specified to supply Resources. The particular list of units and ResGrps depends on the sequence number associated with the current Task Rotation and the specific Activity and Finder that are currently executing.

For Type 1 Finders, the list of Establishment units and ResGrps is broken down into two categories, called Primary and Augmentee. Primary indicates that the Resources selected by the Finder should be preferentially supplied by these units and ResGrps. Augmentee units and ResGrps may be used to fill any Slots that remain unfilled after a matching attempt using the primary Resources. Both Primary and Augmentee categories are further broken down into levels that indicate the search order to be used. Thus, primary level one Establishment units and ResGrps are searched first to fill as many Slots as possible, followed by Primary level two, and so on until the Primary list has been exhausted. Then the simulation will begin match attempts using Resources from Augmentee level one units and ResGrps, followed by Augmentee level two Resources, and so on until the Augmentee list has been exhausted. This process stops either when all Slots are filled or when the search reaches the end of the Establishment unit and ResGrp list. This process of locating Candidates and matching them to open Slots in the ResGrp for Type 1 Finders is illustrated in Figure 10. Note that the ResGrp being filled with Resources in Figure 10 is the same ResGrp whose required Slots were identified in Figure 9.

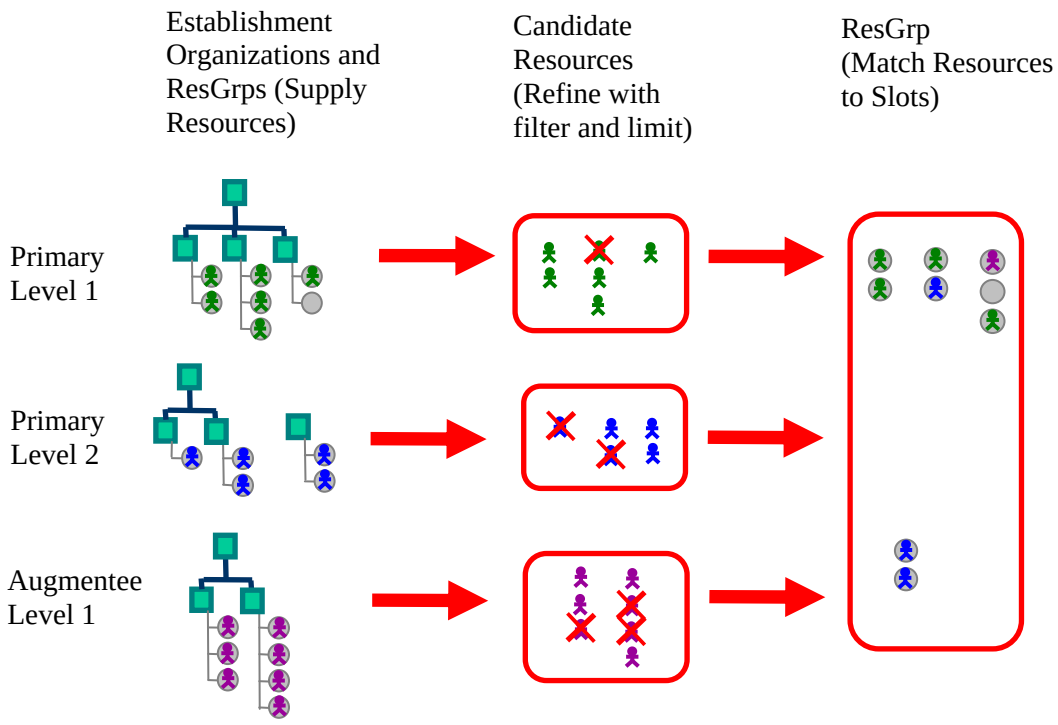


*Figure 10: The Resource selection and matching process for type 1 Finders in which a first match attempt is made using Primary Level 1 Establishment Organizations and ResGrps (green Resources), followed by Primary Level 2 (blue), and finally Augmentee Level 1 (purple).*

For Types 2 and 3 Finders, the Activity uses the Establishment and ResGrp Resources directly without matching them to a list of required Slots. Therefore, a search order defined by multiple categories and levels is unnecessary, so the simulation only processes the Primary level one list of Establishment units and ResGrps to create the list of Resources.

The Establishment units are first searched to find all descendants within the Establishment tree. Then the Resources occupying Slots in those units are assembled in a list. ResGrps are searched to retrieve the Resources they contain and are added to the Resource list. This forms the preliminary Resource Candidate list for a given Primary or Augmentee level.

Any Resources in the Candidate list that are already pre-filled in the Finder's ResGrp or that will encounter an RUL conflict as a result of being assigned to this Activity are removed from the preliminary list. The list is then filtered based on Resource Attribute criteria and limits to control the size of the Resource pool. The remaining Resources make up the final Resource Candidate list.

### 8.5.3 Resource Group Creation

For Type 1 Finders, the Resources from the Candidate list that are eventually matched to the required Slots will form the Finder's ResGrp. For Type 2 Finders, this Resource list is immediately assembled into the Finder's ResGrp without a matching process. For Type 3 Finders, the Resource list is used as a template population from which a selection of new Resources will be created. In this case, the simulation randomly selects a specified number of Resources from the template population and creates new Resources that are duplicates of the selected Resources. The final list of duplicates then forms the Finder's ResGrp. Note that the number of duplicates created can be larger than the size of the template population because the random selection does not remove the selected Resource from the template population. This allows the simulation developer to create a template population in the Establishment Organization that accurately represents the demographic make-up of new recruits. Then an unlimited number of new Resources can be created by randomly selecting and creating duplicates from this population.

At this stage, the Resources groups for Types 2 and 3 Finders are complete and proceed to the Attribute update step. Type 1 Finders continue on to the matching process.

The final list of Candidates from the Primary or Augmentee level being searched is sorted in increasing order of the weight assigned to their Attribute values. For example, typically a Sergeant is given a higher weight than a Private and will be placed later in the Candidate list.

Finally, the simulation attempts to fill each required Slot with a Resource from the Candidate list. The matching proceeds by starting at the first Slot in the sorted Slot list and making a pass through the Candidate list searching for the first Resource that meets the Slot's Attribute Requirements. If a match is found, the Resource is assigned to the Slot. If no match is found, the Slot is left unfilled. In either case, the matching process then continues to the next Slot, and so on until each Slot has been attempted or the Candidate list is empty.

The relative sorting of the Slot and Candidate lists ensures that the most difficult-to-fill Slots are matched first to the least qualified Candidate that meets the Slots requirements. This ensures that the maximum number of Slots will be filled from a given Candidate pool by avoiding the case of placing a high-valued Resource in an easy-to-fill Slot.

If the Slot list is not completely filled by the Candidate Resources, the matching process repeats by assembling another Candidate list based on the next group of Establishment units. For example, if the Primary level-one Resource Candidates are not successful in filling all the required Slots, the simulation will repeat the matching attempt using Primary level two Candidate Resources, and so on until match attempts have been made using all Primary levels. If the primary Establishment units and ResGrps fail to fill all the Slots, the simulation continues with the Augmentee levels. Once this process completes, Activity processing will proceed whether or not there remain unfilled Slots after all Primary and Augmentee levels have been attempted.

For Type 1 Finders, the full list of required Slots and any Resources matched to those Slots form the Finder's ResGrp. Subsequent logic, described in the next Section, will examine all the ResGrps present at the start of an Activity to determine if there are sufficient Resources of the right type for the Activity to run.

Once the ResGrp has been created by either of the three Finder types, the Attribute update instructions attached to the Finder are applied to all the Resources in the ResGrp. Table 4 summarizes the steps carried out by the three Finder types.

*Table 4: Summary of actions performed by the three Finder types.*

| Action | Finder Type 1 | Finder Type 2 | Finder Type 3 |
|---|---|---|---|
| 1. Build Required Slot List | ✓ | | |
| 2. Build Resource Candidate List | ✓ | ✓ | ✓ |
| 3.a Build ResGrp from Candidate Resources successfully matched to Required Slots | ✓ | | |
| 3.b Build ResGrp from Resource Candidate List | | ✓ | |
| 3.c Build ResGrp from newly created Resources that are randomly sampled from the Resource Candidate list | | | ✓ |

## 8.6 Activity Firing Rules

Each Activity defines rules that establish the minimum number of Resources that must be found in order to run the Activity. Each rule is expressed as a minimum percentage of Slots that must be filled for a given Resource type. Resource types are arbitrarily created by associating each Feeder and Finder with a Resource type number. As ResGrps are received at Feeders and assembled at Finders, the number of Slots filled compared to the total Slots is tallied for each type. Once all ResGrps have been counted, if any of the types is not filled to the required percentage, the Activity fails and will immediately route its ResGrps through its Senders onto subsequent Activities or back to the Establishment. For example, a training Activity might have three Resource types: instructors, a group of students, and some equipment and facilities. Each of these Resource types would be acquired at a different Feeder or Finder and each Feeder or Finder would be assigned a corresponding type number. The Activity Firing Rules would then enforce minimum percentages for each type: 100% for the instructors, 75% for the students, and 90% for equipment and facilities. A violation of any of these rules would result in the training Activity being cancelled and the instructors, students, equipment, and facilities being released.

## 8.7 Internal Routing

Each Type 1 (process) Activity defines internal connections for routing ResGrps from its Feeders and Finders to its Senders. Type 2 (event) Activities do not save the ResGrps created by their Finders and do not have Feeders thus no internal routing of ResGrps occurs. The routing links each Feeder and Finder to a single Sender. Two internal routing maps are defined for each Type 1 Activity. The first is used if the Activity meets its Firing Rules and executes normally; the second is used if the Activity fails one or more of its Firing Rules and is cancelled. This allows the simulation developer to apply a different set of Resource Attribute update instructions and to route the Resources to different follow-on Activities depending on whether the current Activity succeeded or failed. In either case, the ResGrps are routed to the appropriate Senders for exit processing just prior to ending the Activity. For a successful Activity, the Activity ends after employing its assembled Resources for the specified duration. For a cancelled Activity, the Activity ends immediately. Figure 11 shows an example of the two internal routing maps where, if the Activity succeeds, it passes its Resources groups on to a subsequent Activity (black connectors), and if the Activity fails, it returns the Resources groups to the Establishment (grey connectors).

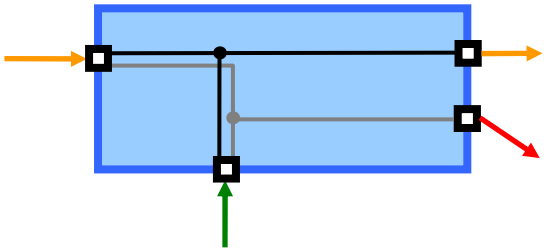

*Figure 11: Example of Activity's internal routing showing the two mappings that apply if Resource acquisition satisfies the Activity Firing Rules (black) and if it fails(grey).*

## 8.8 Senders

When a Type 1 Activity ends, whether it ran successfully or was cancelled, its ResGrps exit the Activity through Senders. Each Sender first applies Attribute updating instructions to the Resources contained in the ResGrps passing through it. Depending on the Activity-to-Activity connections defined by the Task, the Senders then route the ResGrps either to the Feeder of a subsequent Activity or back to the Establishment. The exact Attribute update instructions and routing will reflect whether the Activity met its Firing Rules and ran successfully or failed its Firing Rules and was cancelled. For example, a training Activity would have two Senders for its students: one that applies an Attribute update granting the new qualification and that routes to subsequent training if the Activity ran successfully, and one that simply returns the Resources to the Establishment if the Activity failed. In some instances, a failed Activity Sender may still route its ResGrps to subsequent Activities because each Activity assembles Resources in different ways and has different Firing Rules. Thus the receiving Activity may still run successfully while the sending Activity failed. This would only be possible if the subsequent Activity was not dependent on the ResGrp having successfully completed the failed Activity.

# 9 The Activity Lifecycle

Execution of a simulation Scenario in the MARS managed readiness model consists of instantiating all the Activities that will be active in the Scenario and then managing the lifecycle of each Activity within a Discrete Event process. The entire Activity Lifecycle from simulation start to finish is illustrated in Figure 12. The simulation begins with the creation of Task generators. Each Task generator is associated with a single Task and manages the creation of each Rotation of that Task. The Task generator also attaches timing information to each Rotation according to the specified Rotation schedule. Each Task Rotation then instantiates the full collection of Activities that make up the Task. The timing information associated with each Activity (which is defined relative to the start time of the Rotation) is then converted to an absolute time relative to the start of the simulation.

The entire collection of Activities from all Task Rotations is then placed in a hold queue. At this point, the simulation time starts advancing and the Activities wait in the queue until three conditions are satisfied:

1. The simulation time is within the absolute SNET time and SNLT time for the Activity.

2. All preceding Activities that are connected to the Activity have completed and have routed their ResGrps through their Senders.

3. All preceding Activities that the Activity depends on have completed.

When an Activity has satisfied all of these conditions, it is released from the queue and becomes active. For Type 1 (process) Activities, the Activity begins by processing its Feeders and Finders to assemble the Resources groups that will participate in the Activity. In so doing, it updates the Attributes of the acquired Resources, and keeps track of the number of Resources that it has acquired of each type. The Activity then checks whether the number of Resources found satisfies the Activity's Firing Rules. If the Firing Rules are not satisfied, the Activity is cancelled and its ResGrps are immediately routed internally to its Senders according to the failed Activity routing map. If the Firing Rules are satisfied, the Activity starts running, and employs its chosen Resources for the duration of the Activity. When this duration has passed, the Activity routes its ResGrps to its Senders according to the successful Activity routing map.

Whether the Type 1 Activity succeeded or failed, its ResGrps are then processed at its Senders which perform Attribute updates and then either route the ResGrps onto subsequent Activities or return the Resources to the Establishment. This completes the lifecycle of a Type 1 Activity, and the Task Rotation to which the Activity belongs registers its completion and whether it succeeded or failed.

When a Type 2 (event) Activity is released from the queue, it begins by processing its Finders. Event Activities do not save or manage ResGrps so there are no Feeders or Senders. They also have no duration. Only type 2 and 3 Finders are allowed on event Activities. Since these Finders select and use Resources directly without matching them to a list of required Slots, it is impossible to have unfilled Slots in an event Activity. Therefore, Activity Firing Rules are not enforced for event Activities. Once the Finders have been processed and the Attributes of the

selected Resources have been updated, the event Activity has completed its life cycle. Its parent Task Rotation then registers its completion.

When all the Activities in a Rotation have completed, the Rotation is complete. Once all the Rotations of a Task have completed, the Task generator is complete. Once all Task generators have completed, the simulation is complete.

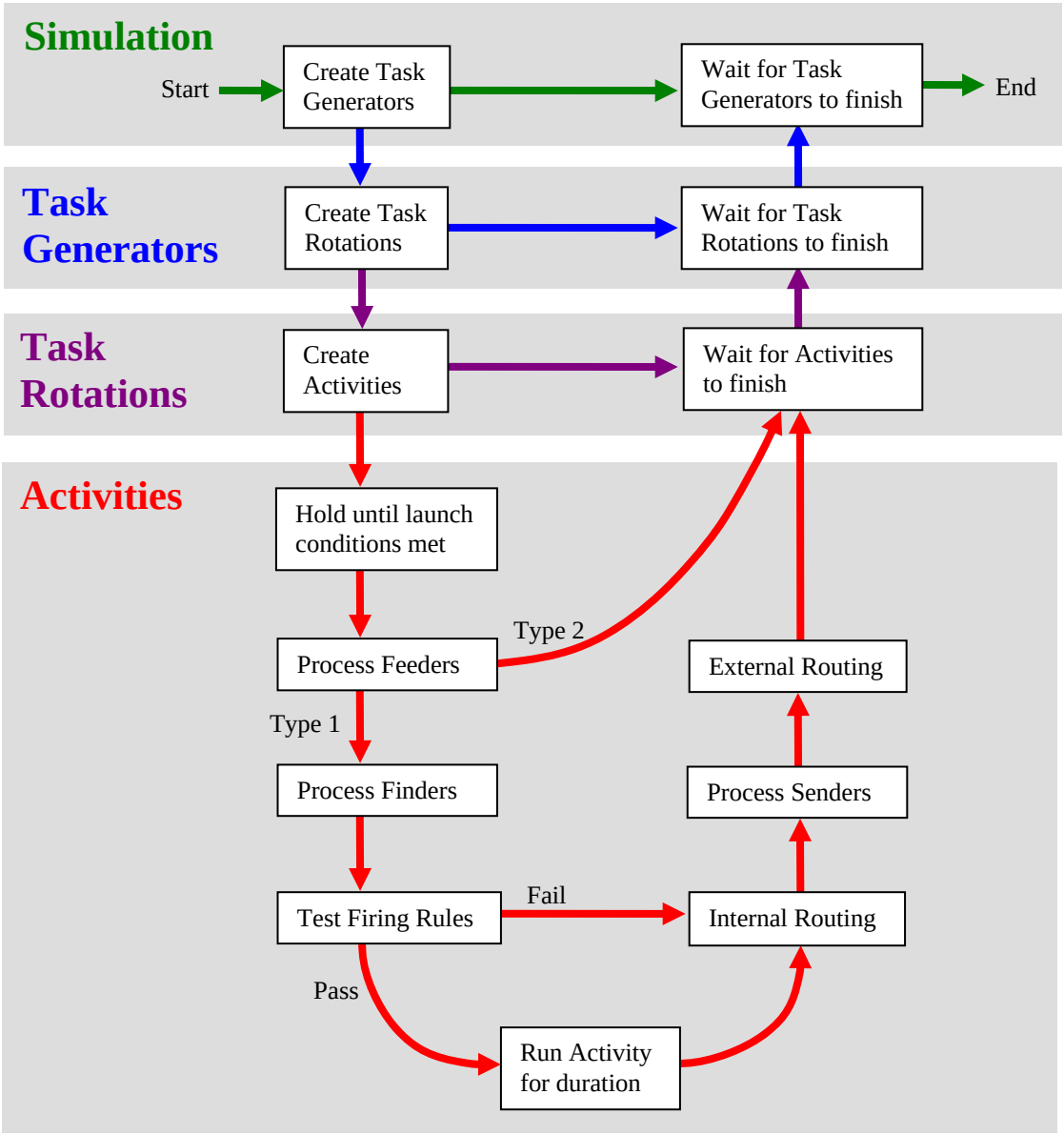


*Figure 12: The Activity Lifecycle showing the creation of Task Generators, Task Rotations, Activities and the Activity Process.*

# 10 Conclusions

This paper provides a complete reference describing the functionality available in MARS V2. This new functionality addressed the limitations identified in the functional design of MARS V1 and provided modelling capabilities required to implement Establishment dynamics. This will allow MARS V2 to conduct simulations of readiness scenarios that incorporate a wide range of modelling constructs. These constructs include operational deployments and Resource status changes used to simulate basic readiness scenarios. However, the new functionality available in MARS V2 also allows for the incorporation of constructs that model Establishment dynamics including recruitment, training, promotion and attrition. This integration of Establishment dynamics into simulations of force generation scenarios will provide military decision makers with previously unavailable assessments of current and future readiness.

# References


[1] Ormrod, M., Young, C., and Pall R. (2007). Modelling Force Generation with the Managed Readiness Simulator (MARS): Modelling Concept and Requirements for MARS v1.0. DRDC CORA Technical Memorandum TM 2007-65.

[2] Pall, R., Young, C., and Ormrod, M. (2007). Modelling Force Generation with the Managed Readiness Simulator (MARS): Implementation of MARS v1.0 in a Discrete Event Simulation Environment. DRDC CORA Technical Memorandum TM 2007-52.

[3] Young, C., Pall, R., and Ormrod, M. (2007). A Framework & Prototype for Modelling Army Force Generation. DRDC CORA Technical Memorandum TM 2007-54.

[4] Kelton, D.W., Sadowski, R. P., Sturrock, D. R. (2004) Simulation with Arena. 3rd ed. Boston: McGraw-Hill Higher Education, 2004.

[5] Ormrod, M., Young, C. (2007). Preliminary Analysis of Task Force Afghanistan Sustainability Using MARS. DRDC CORA Technical Memorandum TM 2007-40

[6] Okazawa, S., Ormrod, M., Young, C. (2009). Managed Readiness Simulator (MARS) V2: Assessment of a Simulation Runtime Database Approach. DRDC CORA Technical Memorandum TM 2009-043.

This page intentionally left blank.

# Distribution list

Document No.: DRDC CORA TM 2009-057

(Report distributed by CD unless otherwise noted)

**LIST PART 1: Internal Distribution by Centre**

| | |
|---|---|
| 3 | Authors (Hard Copies and emails) |
| 1 | DG DRDC CORA (Email: Ross.Graham@forces.gc.ca) |
| 1 | DDG DRDC CORA (Email: Eric.Fournier@forces.gc.ca) |
| 1 | Chief Scientist DRDC CORA (Email: Dale.Reding@forces.gc.ca) |
| 1 | Section Head, Land OR (Email: Dean.Haslip@forces.gc.ca) |
| 1 | LCDORT (Email: Fred.Cameron@forces.gc.ca) |
| 1 | DRDC Valcartier OR Team |
| 2 | DRDC CORA Library (1 Hard Copy, 1 CD) |
| 1 | DGMPRA Personnel Generation Research Section |
| 14 | TOTAL LIST PART 1 |

**LIST PART 2: External Distribution by DRDKIM**

| | |
|---|---|
| 1 | ADM(S&T) (for distribution) |
| 1 | Director S&T Land |
| 1 | DRDKIM 3 |
| 1 | DG DRDC Valcartier |
| 1 | CF College Library |
| 1 | Fort Frontenac Library |
| 1 | COS(Land Ops) [DGLS] |
| 1 | COS(Land Strat) [DGLCD] |
| 1 | LFDTS |
| 1 | DLS [DLSP] |
| 1 | G1 [DLPM] |
| 1 | G3 [DLFR] |
| 1 | G4 [DLSS] |
| 1 | DLCD |
| 1 | DLFD |
| 1 | DLR |
| 1 | DLCI |
| 1 | DAT |
| 1 | DAD |
| 1 | DLSE |

1 Document Exchange Manager
DSTO Research Library
Defence Science & Technology Organisation
PO Box 44
Pyrmont NSW 2009
AUSTRALIA

1 Dr. Neville J Curtis
Research Leader Land Operations Research
75 Labs
Land Operations Division
PO Box 1500
Edinburgh SA 5111
AUSTRALIA

1 Chief Analyst
Land Battlespace Systems
Dstl Integrated Systems
Room 31, Bldg A3, Fort Halstead
Sevenoaks, Kent, UK, TN14 7BP

1 Dr. Jason Field
Land Battlespace Systems
Dstl Integrated Systems
Fort Halstead
Sevenoaks, Kent, UK, TN147BP

1 Director, US AMSAA
ATTN: AMSRD-AMS-S)
392 Hopkins Road
APG, MD 21005-5071

1 Mr. Patrick O'Neill
Chief, Combat Support Analysis Division USAMSAA (ATTN: AMSRD-AMS-S)
392 Hopkins Road
APG, MD 21005-5071

1 Dr. James T. Treharne
OCA Division
Center for Army Analysis
6001 Goethals Road
Fort Belvoir, VA 22060-5230

1 Mr. Robert Barrett
Chief, International Activities
Center for Army Analysis
6001 Goethals Road
Fort Belvoir, VA 22060-5230

1 Mr. John Hughes
HQ, TRADOC Analysis Center (TRAC)
Programs & Resources Directorate (PRD)
255 Sedgwick Avenue
Fort Leavenworth, Kansas 66027-2345

1 Ms. Belinda Smeenk
TNO Defence, Security and Safety
Information and Operations
P.O. Box 96864, 2509 JG
The Hague, The Netherlands

1 Mr. Bob Barbier
TNO Defence, Security and Safety
Information and Operations
P.O. Box 96864, 2509 JG
The Hague, The Netherlands

33 TOTAL LIST PART 2

**47 TOTAL COPIES REQUIRED**

This page intentionally left blank.

## DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)

1. ORIGINATOR (The name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Centre sponsoring a contractor's report, or tasking agency, are entered in section 8.)

Defence R&D Canada – CORA
101 Colonel By Drive
Ottawa, Ontario K1A 0K2

2. SECURITY CLASSIFICATION (Overall security classification of the document including special warning terms if applicable.)

UNCLASSIFIED

3. TITLE (The complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S, C or U) in parentheses after the title.)

Managed Readiness Simulator (MARS) V2: Design of the Managed Readiness Model

4. AUTHORS (last name, followed by initials – ranks, titles, etc. not to be used)

Okazawa, S.; Ormrod, M.; Young, C.

| 5. DATE OF PUBLICATION (Month and year of publication of document.) | 6a. NO. OF PAGES (Total containing information, including Annexes, Appendices, etc.) | 6b. NO. OF REFS (Total cited in document.) |
|---|---|---|
| November 2009 | 50 | 6 |

7. DESCRIPTIVE NOTES (The category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

Technical Memorandum

8. SPONSORING ACTIVITY (The name of the department project office or laboratory sponsoring the research and development – include address.)

Defence R&D Canada – CORA
101 Colonel By Drive
Ottawa, Ontario K1A 0K2

| 9a. PROJECT OR GRANT NO. (If appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant.) | 9b. CONTRACT NO. (If appropriate, the applicable number under which the document was written.) |
|---|---|
| | |

| 10a. ORIGINATOR'S DOCUMENT NUMBER (The official document number by which the document is identified by the originating activity. This number must be unique to this document.) | 10b. OTHER DOCUMENT NO(s). (Any other numbers which may be assigned this document either by the originator or by the sponsor.) |
|---|---|
| DRDC CORA TM 2009-057 | |

11. DOCUMENT AVAILABILITY (Any limitations on further dissemination of the document, other than those imposed by security classification.)

Unlimited

12. DOCUMENT ANNOUNCEMENT (Any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in (11) is possible, a wider announcement audience may be selected.))

Unlimited

13. ABSTRACT (A brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual.)

The first Managed Readiness Simulator prototype (MARS V1) successfully demonstrated the modelling concepts on which MARS is based, but it lacked technical flexibility and the potential to address subsequent development goals. The most important development goal which was considered infeasible in MARS V1 was support for a dynamic establishment in which CF resources are recruited into, advance through and retire from the establishment at the same time that operational deployments are occurring. Other features were also desired in order to improve the generality and flexibility of the MARS application. MARS V2 was developed to achieve these goals. The feature enhancements and additions involved in the development of MARS V2 affected nearly all aspects of the original functional design. Therefore, this paper serves as a comprehensive reference of all the functionality now available in MARS V2.

Le premier prototype de simulation de gestion de la disponibilité opérationnelle (MARS V1) a fait la preuve des principes de modélisation sur lesquels MARS est fondé, mais ce logiciel n'offrait pas la souplesse technique ainsi qu'un potentiel suffisants pour répondre aux objectifs de développement subséquents. L'objectif de développement le plus important qu'il semblait impossible de réaliser au moyen de MARS V1 était la prise en charge de l'établissement dynamique, dans le cadre duquel les ressources des FC sont engagées, mises en œuvre et retirées de l'établissement au même moment que les déploiements opérationnels sont effectués. D'autres caractéristiques étaient aussi souhaitées afin d'améliorer la portée générale et la souplesse de l'application MARS. MARS V2 a donc été développé pour répondre à ces objectifs. Les améliorations et ajouts apportés aux fonctions et intégrés au développement de MARS V2 ont touché presque tous les aspects de la conception fonctionnelle initiale. Le présent document constitue une référence complète décrivant les fonctionnalités offertes par MARS V2.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (Technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus, e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

MARS; MARS V2; managed readiness simulator; readiness; modelling; simulation;


DRDC CORA

www.drdc-rddc.gc.ca