DARWARS Developers' Guide

A BBN Technologies Document

Version 3.0

31 January 2005

Please e-mail comments to: darwars@bbn.com

A copy of this document will be available online at: https://www.darwars.net

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **31 JAN 2005**	2. REPORT TYPE **N/A**	3. DATES COVERED **-**

4. TITLE AND SUBTITLE **DARWARS Developers' Guide: A BBN Technologies Document, Version 3.0**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Office of Naval Research 800 North Quincy Street Arlington, VA 22217-5660**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES
The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**UU**	**37**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

Document History

29 October 2003	Initial Publication (v1.0)
31 July 2004	Publication of v2.0
31 January 2005	Publication of v3.0

Change Log

Date	***Change***	***Section***
31 January 2005	2.3 Training Record	New Section.
	3.8 Web Services	Revised.
	5.3 Voice over IP	Revised.
	5.2 Instant Messaging	Revised.
	Appendix A – Client Host Configuration	Revised.
	Appendix B – SOAP API	Revised.
	Appendix C – DARWARS Installation	New Section.
	Appendix D – DARWARS Administration	New Section.
	All Sections.	Minor Edits.

Contents

Figures

1 Introduction

1.1 The DARWARS Program and DARWorld

U.S. military personnel who emerge from the Combat Training Centers (CTC) are the best trained in the world. DARWARS aims to bring the level of excellence achieved at the CTCs to all our forces, all the time, everywhere and to do so at a lower cost. BBN will achieve this vision with the creation of DARWorld – an innovative training environment that encompasses communities of members including trainees, instructors, subject matter experts, and authors of training content.

In this document, DARWARS is the DARPA program integrating training components, tools, and infrastructure into DARWorld, which is a distributed training system consisting of many interconnected components and applications ranging from single-user training systems to multi-user and massively multi-user training systems to authoring and administrative tools.

1.2 DARWorld Development

DARWorld has three development areas:

- Develop the infrastructure services on which other software can be developed.
- Develop training systems to train students.
- Develop tools and other support software to weave together an effective overall training system from the individual training system and the infrastructure.

1.3 Scope of this Document

This document defines the services available to tool and training system developers as well as the background required to understand how these services fit into DARWorld. This is not intended as an infrastructure developer's guide, though it will provide an appropriate backdrop for the infrastructure development work. DARWorld services are not divided into Training System services and tools services. Rather, there is a range of services – some having greater applicability to training system developers and others having greater applicability to tool developers. This document will be specifically of interest to developers of training systems who wish to take advantage of DARWARS services and make their software available to DARWARS users.

This document assumes that the reader is familiar with the basic concepts of the DARWorld Architecture as described in the DARWorld Architecture Document.

This document describes the current version of the DARWorld services; it will be updated as these services are modified and expanded.

This document is complemented by other documentation as specified in section 6.1.

1.4 Document Organization

This document primarily describes the interfaces (APIs) and protocols used to connect to and use DARWorld distributed services. The services have been grouped into functional areas and a section is devoted to each area with subsections describing the separate services. For each service, we name the service and describe its functionality and usage. We give guidelines for its proper use and, finally, a detailed description of the interface and its associated protocol. Other sections round out the document with an introduction, references, glossary, etc. The sections are as follows:

> Section 2 introduces several concepts that are essential for the creation of training.
>
> Section 3 describes how training systems can be integrated with DARWARS.
>
> Section 4 discusses the DARWARS server-side services that are available.
>
> Section 5 introduces the client-side services that are provided by the DARWARS infrastructure.
>
> Section 6 includes references to other DARWARS documentation as well as external links.

1.5 Acknowledgements

This document has been produced by the BBN architecture team. Many valuable inputs to this document have come from discussions with the training system and component developers (CHI Systems, ISI, Acuitus). We would like to thank them for their contributions of ideas on architecture, infrastructure, and cross-cutting components. Special thanks go to DARPA PM, Dr. Ralph Chatham, for his overarching vision and guidance, and to the rest of the government team: Dr. Harold Hawkins, Ray Perez, Dexter Fletcher, Jason Robar, and Avron Barr. We also benefited from informative and insightful conversations with many other individuals: Philip Dodds, Michael Zyda, Harry O'Neil, and many others.

2 Concepts

The DARWARS training infrastructure uses several terms that you should be familiar with. The most important concepts for the training system developer are the *training system profile*, the *training package*, and the *training event*. A more complete list of terms can be found in the glossary included in the appendix. The following diagram illustrates the relationships among the training package, profile, and event and the editors used to produce these.

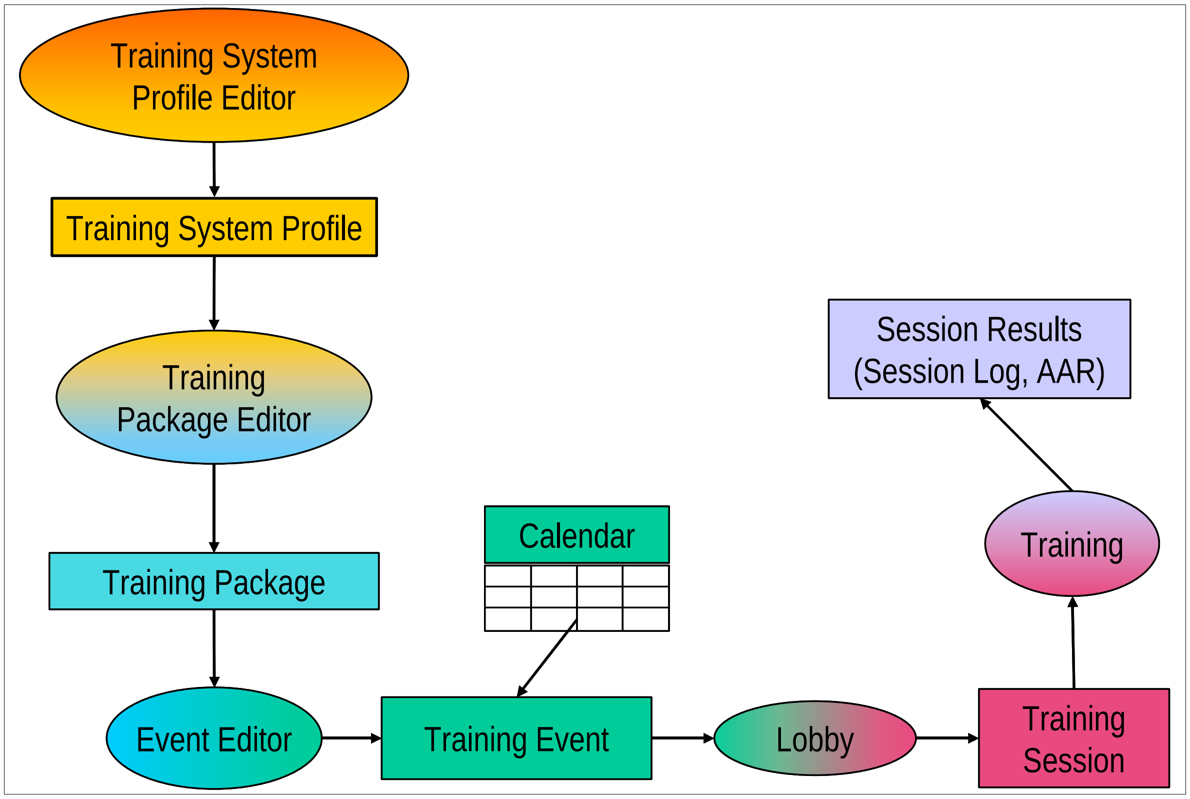

Figure 2-1: The Complete DARWARS Training Development Sequence

2.1 Training System Profile

The training system profile specifies a name, description, and permissible values for any operating parameters that may be included on the command line for invoking the training system. For example, a training system that can run with different scenario files might have a parameter named "scenario file" that could accept any filename that has the ".snr" extension. Similarly, a training system that can run at different difficulty settings might have a parameter named "difficulty level" that could accept any integer value from 0 to 10.

It is assumed that these parameters may be set to different values in order to provide varying training experiences. The profile may include parameter descriptions to provide guidance on the setting of parameter values that can be used to achieve different training objectives. For example, the difficulty level description might note that levels 3-5 are appropriate for users that are somewhat familiar with navigation, while expert navigators should use levels 8-10. These descriptions are useful for developing training packages (see below).

Training system profiles are produced with the Training System Profile Editor that will be documented in the DARWARS Instructor's Guide. Training system developers, or instructors working with guidance from training system developers, create training system profiles for integrating a new training system into DARWARS.

2.2 Training Packages

Training packages are sets of parameter specifications that define different learning experiences using training systems. A single training system might be used in many different training packages, if it has many possible parameter values. For example, training packages may be defined to use different scenarios, or to accommodate varying numbers of participants, or to concentrate on particular training objectives. A single training package may also use multiple training systems. For example, a battlefield tactics-training package for three participants might require each participant to be running a separate but intercommunicating training system.

A training package lists the different training roles available and specifies a set of parameter values for running the training system for that role. A training package also defines the training objectives addressed for each participant who takes on a training role.

Training packages are produced from training system profiles using the Training Package Editor that will be documented in the DARWARS Instructor's Guide. Training packages are generally produced by instructors in order to craft specific training opportunities for their trainees.

2.3 Training Record

The following assumes familiarity with the report of the training impact group ("Designing a DARWARS Architecture that Supports Learning"; cf. section 6.1 "Related DARWARS Documents"). That report defines proficiency and exposure but does not suggest how they should be recorded. The current implementation records proficiency and exposure with an entity called a "training record", or TrainingRecord (in java).

A training record is essentially a "score" for an Objective of a user during a particular training session. In database terms, a training record is an entity with a primary key consisting of a username, ObjectiveId, and TrainingSessionId and an integer attribute that is a "score". For objectives that are competencies, the score indicates the proficiency demonstrated for that competency during the training session. For objectives that are experiences, the score is a measure of exposure to the experience.

For a training record to be understood by a training system, the training system must be able to interpret the ObjectiveId (an integer). The objective identified by the id exists in the space of all DARWARS training systems and so is not associated with a particular training system. In order for a training system to understand the objectives in a training record, there must be a mapping from those ids into equivalent concepts in the training system.

There are two places where this mapping could reside: as part of the training system profile, or as part of the training role. The latter is more general since the same training system objective could be mapped to different DARWARS objectives (the superset of all objectives of all training systems) depending on the scenario or other parameterization of the training system. But, it is also more cumbersome since it would have to be defined as part of every training package. The former is more restrictive and requires that the training system have a set of specific objectives for all possible training scenarios.

The current implementation does not attempt to perform this mapping and training systems should assume that the DARWARS objectives have the same names as the training system objectives. This means that for a training system to "understand" a TrainingRecord, it must translate the ObjectiveId of the record to the name of the objective (using the getObjective(ObjectiveId) API method). To compose a TrainingRecord, the training system must map from objective name to ObjectiveId. The DARWARS API does not provide a direct method for doing this. Instead, the training system must use getAllObjectives() and build a map from the names of those objectives to their ObjectiveId.[1]

The training system may also have to "understand" the TrainingSessionId of the TrainingRecord. For example, if the training system has a model of skill decay, it may need the time of the individual TrainingRecord instances. This information is available in the TrainingSession returned by the getTrainingSession(TrainingSessionId) API method. For adding new TrainingRecords, the training system must know the current TrainingSessionId. This is a standard parameter and could be trivially passed to the training system on the command line when it is launched provided that the training system accepts such a command line option.

2.4 Training Events

A training event is an instantiation of one training package. Using a training package as a template for creating learning experiences, users are able to schedule and partipate in training events. An event may be scheduled for a future time or for immediate participation, and may be created for one or more users. The event creator/owner may specify individuals to participate in training roles, or trainees may "sign up" for previously created events. Shortly before an event begins, participants can access the training lobby, a virtual gathering place for all participants. Once all the required participants have checked into the event lobby, the training event can be initiated to start a training session.

Training events are created from training packages using the DARWARS web-based user interface that will be documented in the DARWARS User's Guide. Training events may be created by instructors or by the trainees themselves.

[1] Currently, the objectives are simply text strings. Objectives are created using the DARWARS web interface. The ids are assigned automatically by DARWARS when the objectives are created.

3 Training System Integration

This section describes the integration of training systems with DARWARS. DARWARS has adopted a model of variable integration for training systems. This means that training systems need only integrate to the level that they gain value from DARWARS. DARWARS does not impose a minimum level of integration. DARWARS recognizes that there is a cost/benefit analysis for any given training system, and allows training systems to integrate as tightly or loosely to DARWARS as they desire. Training system authors are expected to pay the cost of integration only to the extent that they derive benefits from DARWARS.

The loosest integration with DARWARS is launch. As described below, DARWARS can launch a training system without any modifications to the training system on behalf of DARWARS. Tighter integration, however, requires that the training system author make modifications to their existing training systems. This section outlines the various ways a training system can integrate with DARWARS.

3.1 Simple Launch

The simplest form of training system integration is launching the training system with no command line arguments and no mechanism for retrieving information from the training system.

In order to support this level of launching the following are required:

- A training system Profile (which is specified using the Training System Profile Editor; see the DARWARS Instructor's Guide).

- A Training Package (which is specified using the Training Package Editor; see the DARWARS Instructor's Guide).

- A Training Event (created using the DARWARS web-based user interface; see the DARWARS User's Guide).

- An entry in the client computer's application map. This is a text file that maps the training system name (as specified in the training system profile) with the command line execution string on the trainee's computer. See the Client Host Configuration Appendix for an example, and for the location of this file.

With the above information, DARWARS starts the training system, and records that the student has run the specified training package in the student's profile. The next sections describe how DARWARS launches more tightly integrated training systems including delivering scenario files, enabling users to specify settings for the training system, and collecting data from the training system.

If a training system requires launching multiple components, there are two options for integration with DARWARS. The simplest option is to install a single script on the client computer to launch multiple systems, and invoke this script from DARWARS as described above. The other option is to define each sub-component within DARWARS, and then create a training system profile that refers to these sub-components as required resources. DARWARS will then launch all the required resources of the training system immediately prior to launching the training system as described in section 3.3 below.

3.2 Command Line Arguments

Tighter integration with DARWARS can be achieved by adding command line arguments to the training system so that DARWARS can pass information to the training system at launch time. A training system can then be configured to run with certain settings in one training package, and other settings in a different training package. For instance, a command line argument might indicate whether a scenario takes place at night or during the day.

The training system profile specifies all the command line arguments. A training package that incorporates this training system specifies values for these command line arguments. This process is referred to as binding. These bindings (i.e. command line argument values) are passed to the training system as part of the command line at launch time.

DARWARS handles a variety of command line arguments, including switches, integers, and strings. In addition, DARWARS can install files on the client system prior to launching so that those files can be passed on the command line to the training system. This content delivery enables a training system to run the correct scenario for a training event even though that scenario was not previously installed on the client system.

Through the use of command line arguments, DARWARS seamlessly configures a training system to provide specific training experiences for students. The training system is automatically configured with appropriate settings and files to achieve specific training objectives.

3.3 Launch Order and Multiple Application Launch

More complex training systems may have several supporting applications that need to be launched along with the main training system application. For example, a training system that supports verbal interaction may require the use of a separate speech recognizer. The supporting applications are called "required resources" in DARWARS. (Required resources are configured in the same way as training systems, using the Training System Profile and Training Package editors.) The DARWARS server computes the global launch order dependencies and chooses resource servers to use. The launcher and resource server applications handle the actual launching of the applications.

DARWARS has flexible support for specifying and instantiating supporting resources for training systems. The resources may be co-located with each training system instance, located at one such instance and shared by all, shared by training systems in a particular family of training systems, or run on separate server machines and shared. Required resources may require other resources. Each resource required by another has an associated start ordinal ranging from negative to positive values. Resources are started in order with negative ordinals being started before the requiring resource. Conflicting starting order constraints are detected and forbidden. The launcher coordinates the startup of all resources by interacting with the launchers on other machines and waiting as necessary for prerequisite resources to be available.

Resource watchers watch launched applications. These watchers answer two questions: has the application started and has it terminated. There are currently four such watchers:

- The `BasicResourceWatcher` simply launches the application and immediately says it has started. The application is deemed to have finished when it exits.

- The `StdoutResourceWatcher` watches the standard output stream of the application looking for a particular pattern of characters to appear. The pattern is a regular expression with a default value of ^STARTED$. It declares the application has started when that pattern appears. It also deems the application to have finished when it exits.

- The `ResourceServerWatcher` watches applications running on a resource server. It establishes a connection with the remote resource server and waits for the resource server to indicate that the application has started or finished.

- The `LaunchCoordinatorWatcher` watches applications running on other client machines. It establishes a connection with the other launchers on the client machines and waits for the other launcher to indicate that the application has started or finished.

DARWARS has no direct support for optional resources. However, there is no requirement that a required resource continue running throughout the session. Optional resources can be implemented with a command line parameter that causes the resource to say it has started and then exit immediately. The setting of this parameter would then be configured in the training package editor; some training packages would specify the command line parameter enabling the optional component to run and others would disable it from running. For example, the resource that is optional would have a command line option like `--run=<boolean>`. The training system profile would be created with an `Option1` parameter having a boolean validation. Then in training packages where the `Option1` resource is to be run, we would set its value to true and in those where it is to be omitted we would set it to false. The resource would be started in all cases, but would immediately exit if its command line has the option `--run=false.`

3.4 Standard Parameters

DARWARS supplies bindings for a number of standard parameters that the launcher passes as arguments to the training system on the command line. These bindings are generally dynamic and have values that are only known once DARWARS begins to launch a training event. These parameters are defined when the training system profile is created using the Training System Profile Editor. The parameters and their names (as specified in the Training System Profile Editor) are described below. In addition, some of the parameter values may be specified by special escape patterns in command line or file name representations, for example, to specify a file name that incorporates the training session id.

3.4.1 Session ID

The session ID represents a unique identifier for the training session. This session id is used, for instance, to enter information in the session log. The unique session id allows DARWARS to associate a variety of information together as part of a common training session. The standard name for a session id parameter is `SessionId`.

The session id can be incorporated into the representation of other parameters using %s in the Training System Profile Editor.

3.4.2 JSession ID

The JSession ID is a security tag that allows the currently running training system to connect securely to DARWARS via the SOAP API. The standard name for a JSessionId parameter is `JSessionId`.

3.4.3 EndPoint

The endpoint is a SOAP URL that indicates how to access DARWARS via SOAP to communicate with the DARWARS server. The standard name for the EndPoint parameter is `EndPoint`.

3.4.4 Session Master Host

This parameter has the value of the name of the host on which the session master is running. The standard name for the session master host parameter is `sessionMasterHost`.

3.4.5 Session Master User

This parameter has the value of the name of the user who is the session master. The standard name for the session master user parameter is `sessionMasterUser`.

3.4.6 Host

This parameter has the value of the host where this training system is launched. This is generally not used on its own, since the training system can determine what host it is running on. It is expected to be used as part of a dynamic parameter, described below. The standard name for the host is `host`.

3.4.7 User

This parameter has the value of the DARWARS user that is executing this training system. This parameter will allow a training system to annotate data with a DARWARS username, which may not be known to the training system. This parameter is expected to be useful as a dynamic parameter, described below. The standard name for the user is `user`.

3.5 Dynamic Parameters

A training session comprising either multiple training systems or a training system and remote required resources needs a standard way to convey the locations of those other training systems and resources. Dynamic parameters relating to required resources are an extended form of standard parameters. The specification for this capability is currently incomplete, but it is clear that for each such other resource, the parameter must convey the identity of the resource, the identity or name of the parameter and the value of the parameter. Dynamic parameters referring to attributes of required resources are named by prefixing the name of the required resource to the standard name of the parameter.
For example, if the parameter name is `TicTacToeServer.host` (where `host` is a standard parameter name, described above), a binding will be created by the launcher with a value that is the IP address of the host running the `TicTacToeServer` required resource.

A training system may also need to know information about the other training system instances that are running on other client machines. Trying to convey this kind of information on the command line quickly breaks down and DARWARS does not attempt to support it. Instead, the training system should use the DARWARS services directly to translate the `SessionId` standard parameter into a training event and examine the training roles of that event to enumerate the participants and the details about them. The procedure follows:

`DarWorld.GetTrainingSession(sessionId)` returns a `TrainingSession` object. `TrainingSession.getTrainingEvent()` extracts the `TrainingEvent`. `TrainingEvent.getTrainingRoles()` returns a collection of `TrainingRole` objects. Each `TrainingRole` object has a great deal of information. `TrainingRole.getTrainingResource()` returns an object describing the training system including its name and profile. In addition, the `TrainingRole` object has all the `Bindings` that were used to launch the training system. Of particular interest is the `user` and `host` standard parameters that have the user name and host address of the training system instance.

Other DARWARS services can be used to obtain more information about the user, training package, the author of the training package, the owner of the training event, session master, etc.

3.6 File Download

DARWARS provides automatic content download when launching a training system. DARWARS maintains a cache of previously downloaded material on the client and uses MD5 digests to detect material that is already present to eliminate superfluous downloads. This also means that material that is shared among multiple training systems and packages does not need to be downloaded again.

DARWARS currently provides four types of file download. Three are generic and one is training system specific (MAGTF/ BC2010). These are discussed individually below:

3.6.1 File

This is the simplest download validation. The binding of a parameter of this type supplies a data file id that is used to download the file. The file is placed in a temporary location and the full filename of the downloaded file is available as the value of the binding. New training systems should be designed to use this kind if validation if at all possible.

3.6.2 Manifest

A manifest collects together a number of files and presents them to the training system in a single parameter. Each file in a manifest is stored using a path relative to a directory specified by an environment variable. The manifest specifies the name of the environment variable. The command line representation is built using another value from the manifest. Manifests are created by the training package editor and will be discussed further in the DARWARS Instructor’s Guide (cf. section 6.1 “Related DARWARS Documentation”).

3.6.3 Session Log File

This type of download is the inverse of the Log File upload validation (see `UploadableFileValidation`). It downloads a log file that was uploaded in a previous training session. The actual file id is determined at launch time based on the previous training session using the information recorded in the session log.

3.6.4 MAGTF/BC2010

This type of download is functionally similar to the manifest download type, but downloads a zip file that is then unzipped into a directory. Since it is not general purpose, further explanation is omitted. It is mentioned to illustrate that the set of download (and upload) validations is open-ended.

3.7 Data Collection

When a training system (or any resource application running on behalf of a training system) terminates, files that it has written during the session may be automatically uploaded to and stored on the DARWARS server. Each such uploaded file corresponds to a parameter specification of the training system profile. The parameter validation must be an `UploadableFileValidation`. This validation allows for two representations of the parameter binding: a command line representation and a file name representation. The former specifies how the name of the file should be presented to the program (including no representation at all) and the latter specifies the name that the training system will give to the file and thus the name by which the file can be found for uploading. Both representations may use environment variables as part of their value. Such environment variables usually correspond to the directory in which the training system is installed.

An example will help make this notion clear. Consider the MAGTF training system that optionally writes a log file. The name of the log file can be specified on the command line using the "`–F <logfile>`" option. The training system writes the log file into the "`data/loggertapes`" directory under its installation directory. The installation directory is specified by the "`MAGTF_HOME`" environment variable. We choose to name the parameter "`LogFile`". In the Training System Profile Editor, we create a new parameter and name it `LogFile`. We enter '`–F %v`'' in the representation field. The `%v` indicates a binding value. Bindings are created in the training package editor. Certain bindings are created automatically.

Next, we show the validation details and choose the "Log File" parameter type (named for its only current use, but subject to change). Next we fill in the "File Name Representation" with, "`%eMAGTF_HOME%/data/loggertapes/%v.lgr`". The %e---% construct specifies that the launcher on the client machine should substitute the value of the MAGTF_HOME environment variable. As before, the %v specifies substitution of the binding value.

Variations are possible. For example, the training system might always write the same file. In this case we would omit the command line representation and specify the filename with no %v. We might also omit the environment variable, if the training system always writes to /temp, for example.

Files are stored in DARWARS using a unique file id. In the case of an uploaded file, the launcher records the id in the session log with an entry type of "`logfile`" and an entry value of `<ts id>:<parameter name>:<file id>`.

Future extensions include possible file encoding and ways of dealing with directories of files. Another possible extension is to support coupling an upload parameter with another parameter so for example, the upload file name could be specified in terms of the binding of a scenario name parameter.

3.8 Invoking Web Services

Training systems can access server-side DARWARS services via a Simple Object Access Protocol [SOAP] API. SOAP is a World Wide Web Consortium standard for invoking server-side services in an object-oriented, RPC style via HTTP. SOAP provides a programming language independent API, affording DARWARS training system and tool developers maximum flexibility in their choice of programming language.

There are presently two client-side implementations that are known to work with DARWARS. For Java language clients, the Apache project's Axis [AXIS] SOAP client can provide the appropriate security tokens to make calls through the DARWARS security layer. BBN can provide examples of how to do this.

C language clients can use gSoap [GSOAP] to access the DARWARS SOAP API. We expect that most applications written in languages other than C or Java can also make use of the gSoap client by importing it as a native library. Languages like C++, Perl, Python, etc. can all make calls to a C library. DARWARS can provide both C source code to be compiled into a library and a pre-compiled Microsoft Windows DLL. The code provided by DARWARS includes additions to enable gSoap to communicate through the DARWARS security layer as well as a set of C functions that make the SOAP calling sequence easier to use.

3.8.1 Web Service Basics

3.8.1.1 Security

In order to access DARWARS SOAP methods, the proper security context must be established. The DARWARS HTTP server uses a JSessionID cookie to mark an authenticated session. Each SOAP call must include the JSessionID in the HTTP headers in order to pass through the DARWARS security layer. Both Axis and gSoap can provide the capability of adding the JSessionID cookie to the HTTP headers.

There are many other SOAP client implementations, but none have been tested against the DARWARS security layer. Training system developers are welcome to try to use other SOAP client implementations, but we expect some, if not most, will not provide sufficient hooks to set up the necessary HTTP cookies to allow them through the DARWARS security layer.

3.8.1.2 Client Stubs

In order to invoke SOAP services, it is necessary to generate stubs for use within the calling program. DARWARS provides a Web Services Description Language (WSDL) file describing the SOAP interface. This WSDL file can be used to generate the client stubs necessary to invoke DARWARS web services. The WSDL can be accessed at `http://<darwars_server>/member/services/darWorld?wsdl`.

Each SOAP client implementation will provide a translator to convert the DARWARS WSDL into its desired format. Both Axis and gSoap, for instance, have WSDL parsers.

3.8.1.3 The DARWorld SOAP Endpoint

All DARWARS SOAP methods are contained within a single interface, called `DarWorld`, or `com.bbn.darwars.service.DarWorld`. The endpoint passed to a training system at launch time can be used to acquire an instance of `DarWorld` so that methods can be run. Methods returning data will return it in the form of objects defined in the `com.bbn.darwars.common` package. These classes, and the DARWorld endpoint are documented in the Appendix DARWARS SOAP API.

3.8.2 Accessing DARWARS from Java

DARWARS clients written in Java should use Apache Axis to communicate with DARWARS via SOAP. Axis can parse the WSDL to generate Java stubs. Programs can then be compiled against the stubs, and can use the library classes in the DARWARS common package (com.bbn.darworld.common) to send data to and receive data from the DARWARS server via the SOAP API calls.

The DARWARS security token is handled in Axis by establishing a custom socket factory. Axis allows the default socket factory to be overridden by setting the system property `axis.socketFactory` to a fully qualified class name. In DARWARS, this is done in the `init` method of class `com.bbn.darworld.util.axis.DarWorldAxis`. The custom socket factory used in DARWARS is `com.bbn.darworld.util.axis.DarWorldSocketFactory`. The custom socket factory creates standard `java.net.Socket` sockets, but adds the DARWARS security token to the HTTP stream by adding the JSessionId to the `otherHeaders` field.

Use of the SOAP API should appear familiar to Java programmers. Standard method calls are made, and return values received. Axis handles all of the network communication automatically.

All of the DARWARS source code and class files for use with Apache Axis are available upon request from BBN Technologies. For more information, see the documentation for Apache Axis, and `java.net.Socket`.

3.8.3 Accessing DARWARS from C

Because C is not an object-oriented language, the calling sequence for SOAP can be difficult and confusing. To remedy this, the DARWARS C API library (libcapi) includes an additional layer of C functions that should make it easier for training system developers to use.

Below is a code fragment that initializes the DARWARS SOAP structure with an endpoint and a JSessionId. Both the endpoint and the JSessionId would be handed to the training system as command line arguments when launched from DARWARS. After initialization, a SOAP call is made to determine the username of the current DARWARS user. If the call is successful, the name is printed out. If the SOAP call fails for any reason, the fault is printed to standard error.

```
dw_init_darwars(&darwars, soap_endpoint, jsession_id);

result = dw_getCurrentUsername(&darwars, &username);
if (result == SOAP_OK) {
   printf("current user name is \"%s\".\n", username);
} else {
   dw_print_fault(&darwars, stderr);
}
```

The full DARWARS C SOAP API is documented in Appendix B. The API is currently smaller than the full Java API, and is focused only on the functionality that is currently expected to be useful to training system developers. Over time, we expect this API to grow as more server side capabilities are offered, and as more training systems are more deeply integrated.

4 Server-Side Services

4.1 Introduction

This section describes the DARWARS server-side services including support for a session log, the ability to upload and download data between client machines and the DARWARS file repository, and support for user profiles. Each section gives an overview of the capability, and then describes the SOAP interface to the service.

4.2 Session Log

The session log provides a repository for information gathered during a single training session. This is a server-side DARWARS service available to all DARWARS applications. There are two SOAP API methods for using the session log.

```
void addSessionLogEntry(TrainingSessionId sessionID, String host,
String source, long timeInMillis, int sequence, String type, String
entry)
```

`addSessionLogEntry` is used to insert new entries into the session log. All entries are associated with a training session denoted by the `sessionID`. Applications can specify the host computer where the entry is coming from, the software application that is the source of the entry, the time when the event occurred, the sequence number of the event (useful for distinguishing events that happen at the same time), the type of the event, and an arbitrary entry string.

```
SessionLogEntry[] getSessionLog(TrainingSessionId sessionId)
```

`getSessionLog` is used to retrieve session log information about a training session. All entries for a given session are returned. Applications can then filter the entries based on host, source, time, type, etc.

4.3 Data Upload & Download

This topic is covered in the context of launching training systems (cf. section 3.6) and that coverage is the primary documentation from a training system developer's point-of-view. Here we discuss the DARWARS file repository in isolation without reference to launching training systems for those who need to understand more about the capabilities and limitations of this feature.

DARWARS data files are cataloged in the DARWARS database and either stored in the database itself (for short files) or in the server file system. Each file has a unique integer id, a description, and an MD5 digest. An MD5 digest is a cryptographically generated signature of the content of the file. Such a digest almost uniquely identifies the content of the file – the probability that two different files have the same MD5 digest is vanishingly small. DARWARS uses the uniqueness of the digests to detect files on different computers having the same content without transferring the complete file across the network.

The DARWARS web-service includes methods to perform the functions of creating, writing, and closing a new file and opening, reading, and closing an existing file: `createDataFile, appendDataFile, closeDataFile, openDataFile, and readDataFile`.

Other methods exist to fetch the description and digest of an existing file (`getFileDescription`, `getFileDigest`). The description is useful for labeling the file and can be viewed as the equivalent of the file name in a conventional file system. The digest is useful for comparing the content of the remote file with the content of a local file without needing to transfer the content. The `DataFileIds` corresponding to the digests of multiple files can be obtained in a single web-service call (`getDataFileIdsOfDigests`).

4.4 User Profile Info

DARWARS maintains profiles for all its users. Profiles include common information such as name, and email, as well as DARWARS specific information, such as user class within DARWARS, training objectives assigned, etc. Specifically, the user profile includes: name, password, nickname, email, user clases, pending roles and objective assignments. These are explained in more detail below.

The user class is one of: developer, admin, instructor, or trainee. Additional user classes may be added as they become necessary. User classes exist primarily for security reasons; for example, only an instructor or the trainee themselves can assign training objectives to a trainee. Pending user classes may be assigned by anyone, but must be approved by an admin.

Objective assignments are training objectives assigned to the user. Training packages contain one or more *training roles*. Each training role is supported by a training system and fulfills one or more training objectives. The user fulfills one of their training objectives by selecting a training role (or being assigned a role) that fulfills that objective.

A simplistic example illustrates these concepts. Suppose that the user has an objective "Learn to Fly" and there is a training package that has a role "pilot" that fulfills the objective "Learn to Fly". By running the training package in the role "pilot", the user fulfills their "learn to fly" objective.

In addition to the information stored in the user profile, there are SOAP API methods that support accessing training information for the user, including:

- Training packages that match the user's objectives
- Training events that match the user's objectives, i.e. scheduled or ad hoc training events that will use training packages that match the user's objectives
- Training sessions, i.e. completed training events, that match the user's objectives

5 Client-Side Services

5.1 Introduction

We anticipate that the DARWARS infrastructure will provide client-side services that have general applicability across training systems. Such services will be accessible to the user from the DARWARS interface prior to, between, and after training sessions. Additionally, training systems will be able to access these services for use during training sessions. The currently envisioned client-side services are services such as instant messaging, email, or voice over IP which support communications and building a virtual learning community.

5.2 Instant Messaging (IM)

DARWARS has integrated an instant message service allowing users to communicate in real-time through typed text chat. The DARWARS instant messaging infrastructure is based on the Jabber [JABBER] protocols and server. DARWARS uses a modified version of the JBother [JBOTHER] Jabber client. While other Jabber clients are expected to work, none have been tested with the DARWARS infrastructure. DARWARS automatically organizes contact lists on behalf of users. For example, contact lists are dynamically created for each multi-user training event. Users may also create their own contact lists, which will be stored on the DARWARS server (cf. JBother documentation, IM "buddy" lists). When users access DARWARS from another computer, their contact lists are available as a result.

The DARWARS instant message service is available from the DARWARS web interface. Users can launch the client from any web page using the "Chat" link. Each training event also offers links to chat with other people signed up for the event. These links launch the instant message client and automatically start a session with the selected user.

5.3 Voice over IP (VoIP)

5.3.1 Overview

As of the release of this documentation a functional Voice over Internet Protocol (VoIP) application has been developed. The current design provides VoIP as an additional background stand-alone application (required resource) to a main program such as a game or training system, although it can be used on its own. The application itself is based on a client/server topology in which a VoIP client searches for available voice sessions to join. The voice sessions are hosted by the VoIP server, which in turn controls the flow of voice traffic to clients. Creating groups within the voice session on the server and allowing clients to register with many group(s) establishes independent communication networks (COMMS). Once clients are in a session a recording client can be introduced to record specific clients or COMMS. Currently, both Push-To-Talk and voice activated schemes are implemented so that a client can begin voice transmission.

The VoIP application is based on Microsoft's DirectX [DIRECTX] technologies with a C++ graphics device interface (GDI) implementation. Specifically, DirectPlay is used to handle the creation of the clients and servers as well as the network interactions, while DirectVoice is used to create the voice session. DirectX 9 must be installed to run the client or server. Currently, neither application checks for the presence of DirectX.

5.3.2 Usage

A detailed usage guide for VoIP is referenced in section 6.1 "Related DARWARS documents".

6 References

6.1 Related Documents

- DARWARS Architecture Document
- DARWARS Integration and Transition Plan
- DARWARS Ambush! Setup Guide, Appendix B: VoiceComm User's Manual
- Designing a DARWARS Architecture that Supports Learning: Progress Report of the DARWARS Training Impact Group
- DARWARS Administrator's Guide (not yet released)
- DARWARS Instructor's Guide (not yet released)
- DARWARS User's Guide (not yet released)
- MÄK DIS/HLA Game-Link Developer's Guide

6.2 Related Links

[AXIS] Apache Axis SOAP implementation (http://ws.apache.org/axis/)

[GSOAP] gSoap (http://gsoap2.sourceforge.net/)

[SOAP] Simple Object Access Protocol (http://www.w3.org/TR/SOAP)

[HTTP] Hypertext Transport Protocol (http://www.w3.org/Protocols/rfc2616/rfc2616.html)

[HTTPS] Hypertext Transport Protocol with secure socket layer (SSL) encryption

[J2SE] Java 2 Standard Edition (http://java.sun.com/j2se)

[J2EE] Java 2 Platform, Enterprise Edition (http://java.sun.com/j2ee)

[JBOSS] The JBoss Application Server (http://www.jboss.org/)

[MYSQL] The MySQL Relational Database (http://www.mysql.com/)

[JABBER] Jabber Instant Messaging (http://www.jabber.org/)

[JBOTHER] A pure Java Jabber client (http://www.jbother.org/)

[DIRECTX] Microsoft DirectX (www.microsoft.com/windows/directx/default.aspx)

Appendix A – Client Host Configuration

At the present time, DARWARS requires all client computers that will run training systems to have a single configuration file available. This file is known as the application map, or appMap. The contents of the file tell the DARWARS launcher where the training system executables are installed on the local machine.

The DARWARS launcher is responsible for locating the appMap file and reading its contents. On Windows, the launcher looks in the following places:

%DARWARS_HOME%\bin\appMap.txt
C:\darwars\bin\appMap.txt
D:\darwars\bin\appMap.txt
C:\darwars\darworld\bin\appMap.txt

On Unix systems the launcher looks in the following places:

$DARWARS_HOME/bin/appMap.txt
/usr/local/darwars/bin/appMap.txt
/opt/darwars/bin/appMap.txt

Application Map File Format

Each line of the appMap file describes a training system. The format of the line is:

```
<Training System Name>=<Start Directory>|<Executable>
```

The Training System Name must match the name of the Training System as defined in the DARWARS Training System Editor. The Start Directory is the directory from which the executable will be run. The Executable file is executed with the appropriate command line arguments as specified in the Training System profile.

A sample appMap file follows. Note that executables containing spaces must be quoted, while directory names with spaces must not be quoted.

```
MAGTF\ XXI=%MAGTF_HOME%|%MAGTF_HOME%\\MAGTF.exe
BC2010=%BC2010_HOME%|%BC2010_HOME%\\BC2010.exe
Fleet\ Command=%FLEETCMD_HOME%|'%FLEETCMD_HOME%\\Fleet Command.exe'
TicTacToe=C:\\Program Files\\Tic Tac Toe|'C:\\Program Files\\Tic Tac Toe\\bin\\tictactoe.bat'
Test\ LMTS=%TESTLMTS_HOME%|%TESTLMTS_HOME%\\bin\\testLMTS.bat
Tactical\ Language\ Trainer=d:\\UT2003\\System|d:\\UT2003\\System\\UT2003.exe
Exodus=%EXODUS_HOME%|%EXODUS_HOME%\\Exodus.exe
```

The example file makes use of environment variables to denote directory locations. This technique can make it easier for system administrators to install software wherever necessary, but have a common appMap file across all client systems.

Appendix B - SOAP API

This appendix documents the C SOAP API. The Java SOAP API is not documented here, but further details are available on request from BBN. The DARWARS editors all use the Java SOAP API extensively, but none of the early training systems are expected to use the Java API. Thus the C API was chosen for inclusion in this document.

DARWARS manages the SOAP connection through a `darwars_t` struct. This struct contains pertinent information for the current SOAP connection, and can be used to determine underlying SOAP status and fault information. The `darwars_t` struct is initialized by a call to `dw_init_darwars` with a previously allocated `darwars_t` struct, the SOAP endpoint, and the JSessionId.

```
int dw_init_darwars(darwars_t *darwars, char *endpoint, char *jsession);
```

When a program is done using the DARWARS SOAP connection it should free the associated resources by calling `dw_destroy_darwars`.

```
void dw_destroy_darwars(darwars_t *darwars);
```

Each SOAP call returns a status code. If the status code is not `SOAP_OK`, the error can be printed using `dw_print_fault`.

```
void dw_print_fault(darwars_t *darwars, FILE *stream);
```

A program can get information about the current user with `dw_getCurrentUser`. The user will be allocated by the SOAP infrastructure.

```
int dw_getCurrentUser(darwars_t *darwars, User **user);
```

The current user's name can be obtained with a call to dw_getCurrentUsername.

```
int dw_getCurrentUsername(darwars_t *darwars, char **username);
```

Training records are accessible to a program by calling `dw_getTrainingRecords`. The username argument indicates which user's training record will be examined. An array of objective ids and a count of the objectives provide a means of filtering the training record to those records that involve the given objectives. A specific training session can be specified as well. The data is returned in the records argument, and a count of the number of training records will be stored in rcount.

```
int dw_getTrainingRecords(darwars_t *darwars, char *username,
                                 int objectives[], int ocount, int session,
                                 TrainingRecord ***records, int *rcount);
```

Training record objects can be allocated using `dw_allocTrainingRecord`. This is how a program can create a new training record to be added to a user's permanent record through `dw_addTrainingRecord`.

```
TrainingRecord *dw_allocTrainingRecord(char *username,
                                        int objective_id,
                                        int session_id,
                                        int score,
                                        char *descriptor);
```

Training records allocated with `dw_allocTrainingRecord` should be freed using `dw_freeTrainingRecord`.

```
void dw_freeTrainingRecord(TrainingRecord *record);
```

Training records are added to the current user's permanent record with `dw_addTrainingRecord`.

```
int dw_addTrainingRecord(darwars_t *darwars, TrainingRecord *record);
```

Appendix C – DARWARS Installation

1. Install Java SDK 1.4.2.

2. Create a DARWorld folder (e.g. /opt/darworld on Linux or C:\darworld on Windows).

3. Install MySQL 4.0.17 (http://www.mysql.com)

4. Run MySQL as a service. Assuming that MySQL has been installed in C:\mysql:

    a. cd c:\mysql\bin

    b. mysqld --install

    c. net start mysql

5. Install JBoss 3.2.3 (http://www.jboss.org).

6. Install the Instant Messenger server, jabberd 1.4.3 (http://www.jabberd/jabberstudio.org/1.4).

7. Define the following environment variables:

    a. JAVA_HOME – The directory where you installed the Java SDK in step 1

    b. JBOSS_HOME – The directory where you installed Jboss in step 5

    c. JABBER_HOME – The directory where you installed Jabber in step 6

    d. DARWORLD_HOME –The DARWorld folder created in step 2

8. Add the following to the PATH:

    a. $JAVA_HOME\bin

    b. C:\mysql\bin\ (or the location of the installed MySQL)

9. Obtain the DARWARS distribution (dist.zip) from BBN. Unzip the distribution and install the files as follows:

    a. Move darworld.ear, clientjars.war and demoprebrief.war to $JBOSS_HOME/server/all/deploy.

    b. Move mysql-ds.xml to $JBOSS_HOME/server/all/deploy.

    c. Move mysql.jar to $JBOSS_HOME/server/all/lib.

    d. Configure JBoss login by running:

    ```
    java –jar jbossconf.jar $JBOSS_HOME/server/all/conf –
    add login-config.xml
    ```

    e. Setup the demonstration MySQL database by running the following commands:

i. `mysql –u root –p < create_db.sql`

ii. `mysql –udarwars –pdarwars “—execute=source make_tables.sql” darwars`

iii. `mysql –udarwars –pdarwars “—execute=source insert_tables.sql” darwars`

10. Run JBoss:

    a. cd $JBOSS_HOME\bin

    b. run.bat –c all

11. When JBoss has finished initialization (the last message printed to the standard output is “Started in mm:ss”), start Internet Explorer (or another browser such as FireFox) and display the DARWARS login page at: http://localhost:8080.

Appendix D – DARWARS Administration

Several services are available to manage Training Systems, Packages, and Events as well as user information. These services are available via the DARWARS web interface. These operations irrevocably alter the underlying DARWARS database; confirmation is required on all operations. Log in as "administrator" (default password "admin"). The administrator's start page provides the following services:

- **Pending Member Approval:** List new users and requested roles, This service allows the administrator to approve or deny each role for each user.
- **Reset Database and Jabber:** Delete all training (systems, packages, and events) and user information (users and objectives) that was added or modified since the DARWARS system was installed. This service re-initializes the underlying database to its state at the time that the DARWARS system was installed. Instant Messenger (Jabber) buddy lists are also reset to include only those users who are in the same Training Events. (If there are no Training Events defined, then the buddy lists will be empty.)
- **Manage Users:** Delete users or delete roles for a user.
- **Manage Events:** Edit existing Training Events (modify user assignments and event time), delete events, and create new events.
- **Delete All Training Events and Trainees:** Delete all Training Events and all users whose only role is "Trainee" (i.e. does not delete instructors and administrators).
- **Delete All Training Packages:** Delete all Training Packages.
- **Delete All Unmarked Events and Assignments:** Delete all Unmarked Training Events and User Objective Assignments.
- **Mark All Events and Assignments:** Mark all Training Events and User Objective Assignments so that they cannot be deleted. This and the previous service are provided primarily to support demonstrations. Training Events and User Objective Assignments can be created and then marked. During a demonstration, additional Training Events and User Objective Assignments can be created. Using the previous "Delete Unmarked" service, the underlying DARWARS database can be "rolled back" to its pre-demonstration, marked state.

Import and Export Training Packages: Import and export Training Packages. This service supports exchanging Training Packages with other DARWARS installations. Individual Training Packages or all Training Packages can be exported. Exported Training Packages include all associated information, including related[2] Training Packages (for example, After Action Reviews), Training Roles, Training Systems and Resources, and Objectives. This service supports importing Training Packages based on new Training Systems (select "Create New Resources and Objectives") or based on existing Training Systems (select "Use Existing Resources and Objectives"). In the first case, the associated Training Systems and Resources and Objectives are imported and assigned unique names; in the second case, existing Training Systems, Resources and Objectives (i.e. those with matching names) are used.

[2] The training package relationships are specified in the Training Package Editor.

Appendix E - Glossary

Competency

One type of Objective, a competency is a type of thing to be learned within DARWARS; a desired end-state of training. Competency is measured by Proficiency.

Condition

The particular characteristics of the external environment (external to the learner) that is created by a training package and within which learning occurs during the training session. This is similar to Training Package in that a condition includes the specific values of parameters that can vary within a training system (i.e., weather conditions, number of players). However, the condition may also include characteristics that are fixed within a training system (i.e., air platform, language of instruction), if they have implications for training objectives.

Event

A training event is an instantiation of a training package for a set of users. Scheduled training events have a particular time at which they are scheduled to occur. Ad hoc training events are created immediately before they occur and often use training packages designed for individuals, but may use multi-user packages.

The participants of a training event may be completely specified in advance by the creator of the event (e.g. an instructor) or users may sign up for particular roles in the interval between the creation of the event and its initiation. The final stages of this process is called the training lobby. In the minutes prior to the initiation of a training session, users who have already signed up for the event check into the lobby and other users join the event to fill any of the open training roles. In some cases, competent users may have to be found to fill any unfilled roles.

Experience

One type of Objective, an Experience is a type of thing to be learned within DARWARS; a desired end-state of a training session. Experience is 'measured' by Exposure; it is gained by participating in various Training Packages.

An experience is a developmental event during training and/or career necessary to learn a knowledge or skill, or practice an essential competency under operational conditions.

Exposure

Indicator of movement towards an Experience Objective, in terms of the number and types of training packages/events that the user has had. Unlike Proficiency, exposure is largely experiential; users learn implicitly through exposure to varied events.

Knowledge

Information or facts that can be accessed quickly under stress.

Measure

A description of a behaviorally anchored, observable action within the training system that can be calculated and linked to a particular objective (or sub-objective) to demonstrate mastery of a task (or competency, knowledge, skill) in that simulation. Should include:

- A text definition.
- A scale (norms, criteria).
- The actual metric or algorithm.
- (ideally) Some criterion validity (productiveness of other similar performance, differentiates other unrelated performance).

Mission

See Condition.

Objective

What a person would want to learn within DARWARS, and what a training system is able to teach. An objective can have sub-objectives. It is synonymous with *goal*. The objective is to learn a thing or to participate in different situations; the Competency (or knowledge, skill) is the thing to be learned and the Experiences are the situations to participate in.

There are two kinds of objectives within the DARWARS architecture:

- Competency - Measured by Proficiency.
- Experience - Measured by Exposure.

A training objective is any skill, knowledge, experience, competency, task, etc. that a user might need or be required to have, know, or do (i.e., Learning Objective). A training objective that a user has is called an objective assignment, to give a name to the relationship.

A training objective is also any skill, knowledge, experience, competency, or task that a training system might teach or provide (i.e., Teaching Objective). The training objectives that a training system addresses are not necessarily the same as the training objectives that a user might have. However, the training objectives addressed by a training role must be related to the objectives of a user in some way for the training to be at all relevant.

Learning Objective

See above.

Teaching Objective

See above.

Objective Assignment

See above.

Objective Fulfillment

A training objective in which a user can gain proficiency or exposure by participating in a training role.

Package

A training package is a specification of a particular way of running one or more training systems to train one or more users. A training package is mainly a container for the specifications of how to run each of one or more training systems that are to be run concurrently to train one or more users operating cooperatively or antagonistically. These specifications are called training role bindings.

Proficiency

Indicates current level of achievement or degree of ability, in terms of the desired end state objective or competency. Unlike exposure, proficiency is more than experiential; it implies explicit measurement and evaluation of performance. It is synonymous with *Competency Level*.

Required Resource

A required resource also has a profile just like that of a training system that specifies in what ways the required resource can be run. Usually, the bindings that specify how to run a required resource are specified in the relationship between the requiring training system and the required resource. An example of a required resource would be a federation server in an HLA federation. Another example would be a speech recognition program servicing a language trainer. The former is a shared required resource and the latter is a private required resource.

Role

Any part that a trainee can take within a scenario or mission. Constrained by the training system designers.

A training role is a named role in a training session.

Related concept: Role Binding. The collection of training system parameter bindings that cause the training system to place the user in that role. That is, if the role is "pilot" the user will be placed in a cockpit and expected to fly a mission. The naming of a training role is arbitrary and specific to the training package in which it is used. However, most training package authors would want to choose meaningful and accurate names and use the same names to designate the same roles in similar training packages. The only actual requirement is that the names of different roles be distinct within a training package and that the names of equivalent or identical roles in a training package be identical. A training package specifies a minimum and maximum number of instances of each training role.

Training roles are associated with the training objectives that will be addressed during the training session for the user filling the role. For lack of a better term these relationships are called objective fulfillments.

Scenario

See Condition.

Session Log

The activity during a training session is recorded in a session log. Most of the content of a session log entry has a fixed form including a timestamp, source, and sequence. These uniquely identify the entry and provide a strict ordering of events within a given source and an approximate ordering between sources (within the accuracy of the clocks in the individual sources). The session log records significant events that occur during the training session. Many of the events are expected to be relevant to an after action review. An example of this would be comments by users. The session log is not a recording of the session that can be "played back". The data to support such a capability is far too voluminous. However, the session log is the right place to identify points in such a record that are significant for some reason.

Session Master

The session master is the user that is responsible for starting a training session. He may cancel the training event (for example, if all required training roles cannot be filled). In some cases, he might abort the training session (if, for example, some user or user's network connection fails). The session master may be designated in advance. Otherwise, the first user to enter the training lobby becomes the session master.

Skills

Compiled actions that can be carried out successfully under stress.

Training Package

See Package.

Training Session

A training session is the actual execution of a training event. A training session is created when all required training roles have been filled with checked-in users and the session master starts the session. The training session lasts until the training system(s) finish or the session master aborts the session.

See also Session Log, Session Master.

Training System

A Training System is an application which supports computer-based, experiential training.

Training System Profile

A training system profile is a specification of the ways in which a training system can be operated. The presumption is that these various ways will allow the training system to address different training objectives or the same objectives with varying emphasis. The training system profile does not specifically refer to training objectives though it may do so in general terms. The training system profile does specify names and permissible values for its operating parameters. In addition, each parameter has a description intended to convey to the training package author the rationale for choosing various parameter values.

For this discussion, a training system is a single executable program that provides a single user with a training experience. For multi-user training, multiple instances of various training systems and other resource applications must or may be run together. A given user has one training system that is primarily responsible for the user's experience. Other executable programs may be required to support one or more instances of training systems. These other programs are called required resources.

Appendix F - DARWorld Acronym List

AAR	After Action Review
ACL	Access Control List
ADL	Advanced Distributed Learning Initiative
AFRL	Air Force Research Laboratory
AGR	AICC Guidelines & Recommendations (see AICC)
AICC	Aviation Industry CBT (Computer-Based Training) Committee
API	Application-Program Interface
ARIADNE	A European Union standards organization.
BEEP	Block Extensible Exchange Protocol
BLUEFOR	Blue Forces (friendly forces)
C2	Command & Control
CA	Certificate Authority
CBT	Computer-Based Training (see AICC)
CGF	Computer Generated Forces
CGI	Common Gateway Interface (web scripting)
CMI	Computer Managed Instruction
CONOPS	Concept of Operations
COTS	Commercial off-the-shelf
CRL	Certificate Revocation List (a type of server)
CTC	Combat Training Center (U.S. military)
DARPA	Defense Advanced Research Projects Agency
DIS	Distributed Interactive Simulation
DoD	Department of Defense
EJBCA	Enterprise Java Beans Certificate Authority
GOTS	Government off-the-shelf
HLA	High Level Architecture
HTTPS	Hypertext Transport Protocol with secure socket layer (SSL) encryption
ICT	Institute for Creative Technology
IDA	Institute for Defense Analyses
IDE	Integrated Development Environment
IEEE	Institute of Electrical & Electronics Engineers
IMS	Instructional Management System
ISI	Information Sciences Institute
ISO	International Organization for Standardization
ISR	Intelligence, Surveillance, and Reconnaissance – is this in the doc???

ITS	Intelligent Tutoring System
J2EE	Java 2 Enterprise Edition
Jabber, Jabberd	Jabber is an open, XML based, software platform. Jabberd provides a server implementation of the Jabber protocols.
Jabberoo	An object-oriented library for the Jabber protocol.
JDBC	Java Database Connectivity
JFCOM	Joint Forces Command
JNI	Java Native Interface
JSAF	Joint Semi Automated Force
JSO	Java Shared Object
JVM	Java Virtual Machine
LDAP	Lightweight Directory Access Protocol
LCMS	Learning Content Management System
LMS	Learning Management System
LMTS	Last Meter Training System
LOM	Learning Objects Metadata
LTSA	Learning Technology Systems Architecture
LTSC	Learning Technology Standards Committee
MAV	Micro Air Vehicle
MMOG	Massive Multiplayer Online Game
MMP	Massive Multi Player
ModSAF	Modular Semi Automated Force
MOUT	Military Operations on Urbanized Terrain
NPC	Non Player Character
OneSAF	(Army's single) Semi Automated Force
OOTS	Open source Off The Shelf
OPFOR	Opposing Forces
OPORD	Operational Orders
OSD	Office of the Secretary of Defense
PEO STRI	Program Executive Office for Simulation, Training & Instrumentation
PKI	Public Key Infrastructure
PROMETEUS	PROmoting Multimedia Access to Education and Training in EUropean Society
RA	Registration Authority
ROE	Rules Of Engagement
SAF	Semi Automated Force
SCORM	Sharable Content Object Reference Model

SIMNET	Simulation Network
SME	Subject Matter Expert
SMTP	Simple Mail Transfer Protocol
SOAP	Simple Object Access Protocol
SSL	Secure Socket Layer
STRICOM	Simulation, Training, and Instrumentation Command
TLS	Transport Level Security
TSD	Training Session Descriptor – same as Training Package
UAV	Unmanned Aerial Vehicle
VoIP	Voice Over IP (Internet Protocol)
XML	eXtensible Markup Language
XMPP	Extensible Messaging and Presence Protocol
XMSF	Extensible Modeling and Simulation Framework
YAJA	Yet Another Jabber API