MEDIATION TO IMPLEMENT FEEDBACK IN TRAINING

David Maluf, Gio Wiederhold, Ted Linden, Priya Panchapagesan

Department of Computer Science
Stanford University, Stanford CA 94305

1 Overview

The problem addressed by Mediation to Implement Feedback in Training (MIFT) is to customize the feedback from training exercises by exploiting knowledge about the training scenario, training objectives, and specific student/teacher needs. We achieve this by inserting an intelligent mediation layer into the information flow from observations collected during training exercises to the display and user interface. Knowledge about training objectives, scenarios, and tasks is maintained in the mediating layer. A designer constraint is that domain experts must be able to extend mediators by adding domain-specific knowledge that supports additional aggregations, abstractions, and views of the results of training exercises.

The MIFT mediation concept is intended to be integrated with existing military training exercise management tools and reduce the cost of developing and maintaining separate feedback and evaluation tools for every training simulator and every set of customer needs. The MIFT Architecture is designed as a set of independently reusable components which interact with each other through standardized formalisms such as the Knowledge Interchange Format (KIF) [Genesereth92] and Knowledge Query and Manipulation Language (KQML) [Finin94].

2 Mediation applied to military exercise management

The initial application of MIFT is the Exercise Analysis and Feedback phase of military exercise management as schematically shown in Figure 1. More precisely, the focus is on simulation-based army training exercises [Crissey94]. MIFT handles some of the information flows involved in training exercise management. The intent of MIFT is to supplement the flow of information from simulations to evaluation and review and complete a feedback loop by supplying information to plan and tailor future training exercises.

MIFT processes the data that is logged during training exercises and uses scenario information and domain knowledge to organize the data from the exercises in ways that are meaningful and useful for the Observer/Controllers (O/Cs) managing the exercises, trainees, commanders, exercise evaluators, and others interested in the results of training exercises. MIFT is designed to feed information to other software systems that generate training scenarios and help commanders plan future training exercises tailored to the needs of their trainees. The MIFT design is intended to integrate with other exercise management applications and achieve two key application goals for exercise feedback:

1. The software is easy to use by using domain-specific exercise concepts and terminology.
2. Domain experts are able to extend feedback software and tailor it to domain-specific and local needs.

MIFT will achieve the first goal by incorporating knowledge about the scenario objectives and the task and subtasks to be trained. MIFT uses this scenario knowledge to relate simulation results to the objectives and tasks to be trained so that O/Cs, trainees, and commanders can query the simulation results using scenario-based terminology. For example, rather than forcing the O/C to formulate a query to "select all enemy detections of Alpha company before

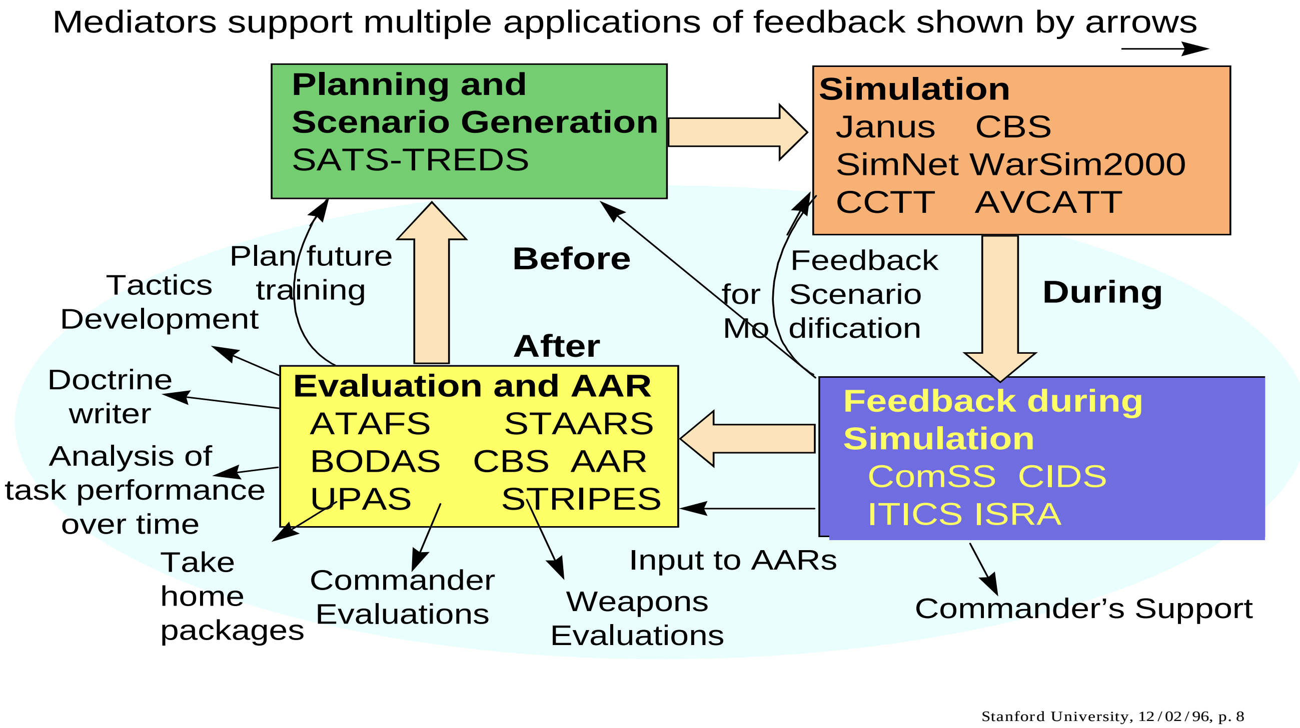

Figure 1: The application Mediation to Implement Feedback in Training (MIFT) is the Exercise Analysis and Feedback phase of military exercise management.

an assault," the O/C can simply ask whether Alpha company achieved its scenario subtask of remaining hidden until the beginning of the attack. The mediator will know that enemy detections before the attack are evidence that the unit was not successful in remaining hidden. In general, MIFT produces results tailored to the needs of exercise planners, weapons designers, and tactics developers. The second goal of a mediator-based architecture is to enable military training and support personnel to tailor and extend analysis and feedback software to meet their own local needs.

3 Mediation Technology

A mediator is a software module that exploits encoded knowledge about certain sets or subsets of data to create information for a higher layer of applications [Wiederhold92]. It should be small and simple, so that it can be maintained by one expert or, at most, a small and coherent group of experts. The first step in developing a mediation architecture for training feedback is to isolate the mediators from lower-level data sources and from higher-level user interface and application code. This will enable mediator to achieve the role of a reusable middleware. Mediators interact with each other through the standardized knowledge exchange and communications protocols. We have used standard knowledge exchange and communications protocols based on Knowledge Interchange Format (KIF) and Knowledge Query and Manipulation Language (KQML) so that mediators can work with data from multiple knowledge sources and supply information that is reusable in multiple roles. The MIFT mediation architecture combines plug-in components at three levels:

1. User interfaces that accept information from mediators and provide a standard set of display options.

2. Mediators that use scenario-based knowledge to analyze, transform, query, and present simulation results. A mediator supports numerous

modules which are relatively small components. A module is a collection of rules reflecting the domain knowledge functionalities. Domain experts extend the analysis functionality by adding domain knowledge to mediators or by plugging in additional modules.

3. Wrappers connect MIFT with the output formats of operational simulators. Currently wrappers are tailored for JANUS and SimNet/LEAF data.

4 Implementation and Functionalities

The current MIFT user interface is built on Web browsers, hence enabling a multiple platform execution. In other words, the MIFT user interface can run at any location that supports Web browsing; the user does not have to download the simulation data. An innovation of the user interface is that it is designed to display information received from a mediator. Users connect to MIFT and the underlying exercise results by using a Java-capable browser. Building the user interface in a browser has several advantages:

1. Users can access exercise results in the same way they access other information from local and remote sources. The user interface will be increasingly familiar to O/Cs and trainees.

2. The exercise data may be local or remote. Startup and initialization is simple. Users do not have to download and manage the exercise data.

A key benefit of mediators for military training applications is that they avoid the need for each simulation program having to build from scratch and maintain a separate set of analysis and feedback software packages.

The operations referenced by from the mediator can be layered in the direction of the data-to-knowledge aggregation as shown in Figure 2. For example, the first two levels in the mediator perform standard aggregations, selections, and analyses on the data sources. We have implemented these two levels to provide a basic level of functionality for higher levels. The third level in the mediator uses knowledge of the training scenario so that O/Cs and trainees can obtain feedback about how well specific scenario tasks have been performed. The mediator allows users to obtain specific feedback without having to understand the structure of the underlying data. A planned fourth level in the mediator will use domain-specific models about the exercise, the scenario, and causal relationships in the exercise to analyze the data for its probable significance and automatically call the users' attention to what it perceives as the more relevant exercise results. It is useful to think of the mediator as composed of three parts:

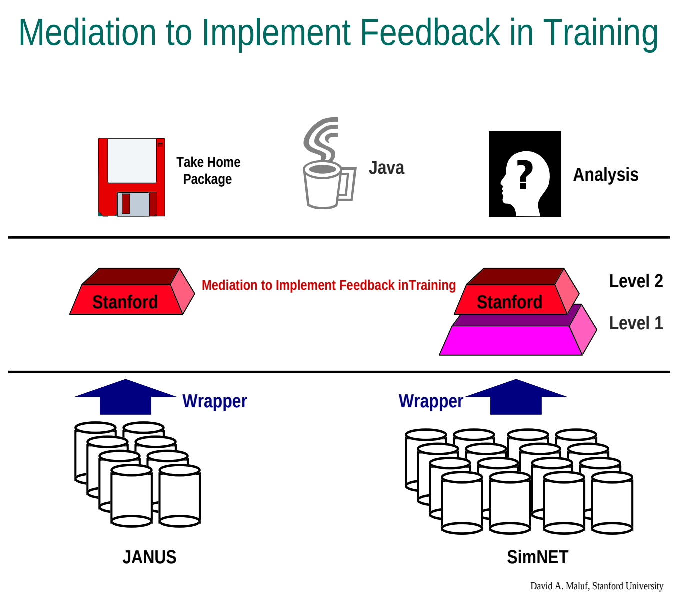

Figure 2: The operations envisaged from the mediator can be layered in the direction of the data-to-knowledge aggregation.

1. Data from disparate sources are converted into object instances over which inferences can be performed.

2. Knowledge about the application domain is maintained in declarative representations.

3. An inference engine processes the knowledge and data sources to produce higher level information

that is passed to other mediators or to the user interface in a standardized form.

One of the MIFT functionalities is that an Observer/Controller (O/C) will depend upon during an After Action Review (AAR) or that a trainee will use after the AAR. Similar MIFT functionality will be useful to commanders, exercise evaluators, weapons designers, and others, but each of these other users is likely to want a different user interface and additional mediator functionality.

MIFT uses wrappers to isolate the mediators from the specific data formats and other differences between simulator outputs. Wrappers are written in C++. When a mediator needs additional information, it calls the appropriate wrapper. The wrapper accesses the data and creates instances of the appropriate Clips objects. The current implementation includes wrappers that processes the outputs of Janus simulation runs, and LEAF formated data from SimNet results. We believe that MIFT functionality can be made available for additional simulators by writing the appropriate wrapper to process simulators outputs. Writing additional wrappers requires programming expertise, but it is not a major undertaking. Using MIFT on a different simulation may also require additional modules and/or user interfaces to provide new functionality appropriate for that simulation. For example, the mediator that creates force ratios is more useful for simulations at the battalion or higher level and might not have been developed for analysis of simulations at the company level.

MIFT is intended to allow analysis and evaluation software to be reused by all of the different consumers of simulation results. In addition to trainees, O/C, and commanders, others who need to analyze and evaluate simulation results include exercise planners, training managers, weapons designers, tactics developers, and doctrine writers. MIFT can also provide results to other software applications; for example, software used to assist in exercise planning and preparation can use MIFT analyses of previous exercises to identify the tasks and subtasks that need to be emphasized in additional training. Thus MIFT contributes to completing the feedback loop from the results of one simulation run into the planning and preparation for future training.

The Mediator is currently written in Clips 6.0 [Riley91], a widely-available and easily portable expert system shell. Since user interface functions and data access functions are separated out into other components, the module implementations are quite small. For example, the force ratio computation for any set/ and/or combination of units is only four rules for a total of 12 lines. Most other mediators at the current stage are smaller. We believe that some domain experts will be able to write modules in Clips.

5 Acknowledgments

This research was supported by a grant from DARPA under the CAETI-EXMAN program and under contract N66001-95-C08618; Kirstie Bellman is the Program Manager. Input from Julia Loughran from IDA has been particularly helpful.

6 References

[Crissey94] M. Crissey, G. Stone, D. Briggs and M. Mollaghasemi, “Training Exercise Planing: Leveraging Data and Technologies”; Proceedings of the 16th Interservice/Industry Training System and Education Conference. Washington, DC: national Security Industrial Association.

[Finin94] T. Finin, J. Weber, G. Wiederhold, M. Genesereth, R. Fritzson, J. McGuire, S. Shapiro, and C. Beck, “Specification of the KQML Agent-Communication Language”; Stanford February 1994.

[Genesereth92] M. Genesereth.”Knowledge Interchange Format”; Technical Report Logic-92-1, Stanford University, 1992.

[Riley91] Riley, G. “CLIPS: An Expert System Building Tool”; Proceedings of the Technology 2001 Conference, San Jose, CA, December 1991.

[Wiederhold92] G. Wiederhold, “Mediators in the Architecture of Future Information Systems”; IEEE Computer, March 1992, pages 38-49.