

A Proposed Distributed After Action Review (DAAR) Standard Based on the Joint Training Experimentation Program (JTEP) DAAR

Reginald Ford
John Shockley
SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025
650-859-4375, 650-859-4165
reginald.ford@sri.com, john.shockley@sri.com

Cris Kobryn
PivotPoint Technology Corporation
P. O. Box 2320
Fallbrook, CA 92088
760-728-9747
cris.kobryn@ptcorp.com

Keywords:

Distributed After Action Review; LVC Training Systems, model-driven development, UML

ABSTRACT: *The Joint Training Experimentation Program (JTEP), a joint effort of the National Guard Bureau and California National Guard to link live, virtual, and constructive training systems to improve overall Guard training and readiness, has developed a Distributed After Action Review (DAAR) capability to facilitate AARs among geographically distributed exercise participants.*

The JTEP DAAR consists of synchronized 2-D and 3-D displays of individual entity position, maneuver, engagement, and tactical voice data along with a video teleconference capability that provides for human interaction among sites. In the initial development, the challenge to provide lossless remote viewing while staying within the limited (700 kbits/s) bandwidth allocated to JTEP on a shared network led to the approach of recording all exercise data at each site, running local instances of the playback process at each site, and sending only control information over the network.

This paper shows how JTEP DAAR capabilities can form the basis for a generalized standard for high-resolution distributed AAR, applicable especially to bandwidth-limited environments. The DAAR protocol is specified using Unified Modeling Language (UML). The paper describes the protocol and presents representative diagrams specifying key aspects of logger, playback, and viewer synchronization.

1. Introduction

1.1 Background

The Joint Training Experimentation Program (JTEP) is a National Guard Bureau program managed by the California National Guard. JTEP's charter is to use live, virtual, and constructive (LVC) training systems to enhance overall Guard training and readiness. Prior to the initiation of JTEP in 2002, the Guard used advanced live, virtual, and constructive (LVC)

systems¹ to support training, but each system was stand-alone. JTEP was conceived to bring to the Guard the benefits of integrating existing or readily available training environments, and to enable LVC interaction over non-dedicated wide-area networks (WANs).

¹ A live "simulation" comprises real people, real vehicles, real environment, and simulated weapons. A virtual simulation comprises real people, simulated vehicles, simulated environment, and simulated weapons. A constructive simulation comprises some real people, some simulated people, simulated vehicles, simulated environment, and simulated weapons.

One of the goals of such integration is to enable training between geographically distributed sites. In one of JTEP's early demonstrations, LVC training systems located at Camp Roberts and Camp San Luis Obispo, CA, 50 miles away, interacted in real time over the shared California Army National Guard (CARNG) network. Early in the planning for this demonstration, the JTEP team realized that such a distributed training environment required a Distributed After Action Review (DAAR) capability to facilitate an AAR between sites. The first JTEP DAAR linked these two locations along with the Joint Forces Headquarters (JFHQ) in Sacramento. This configuration enabled troops being trained at both training sites to conduct a single AAR that was also accessible at JFHQ.

The initial DAAR network established the fundamental requirements and the basic design of the DAAR that is proposed as the starting point for a standard in this paper. The principal requirement is to provide synchronized viewing of exercise data and displays over a low bandwidth, shared network that is geographically distributed between exercise sites. Additionally, there is a requirement to provide participant interaction between DAAR sites so that a true Observer Controller-led AAR can be conducted. These requirements drove the basic DAAR design to consist of separate instances of all exercise data recorded at each DAAR site. The data are synchronized for viewing by sending control commands (e.g., play, pause, and viewpoint) over the network so that participants at each site see and hear the same data. Participant interaction is facilitated by a standard video teleconference (VTC) connection. This configuration has proven to be successful in both synchronizing the exercise data and facilitating the necessary human interaction between sites [1].

The DAAR files are also captured on a CDROM and distributed with a playback software installer. This Takehome Package is playable on standard desktop computers [2].

Since the initial demo, we have successfully applied the DAAR design to several actual training exercises. In May 2005, a satellite link to the internet was added to enable a transcontinental DAAR between the range at the Marine Corps Air Ground Combat Training Center at Twenty-nine Palms, CA and Washington, DC. The DAAR capability has also been deployed in

EXportable Combat Training Capability (XCTC)² exercises in Kentucky (2005) and Indiana (2006). In these exercises, DAAR connections were made to the respective state's JFHQs as well as National Guard Bureau (NGB) Headquarters in Washington, DC. As a result of these exercises, the JTEP DAAR has become an important and visible component to both XCTC and JTEP exercises.

1.2 From DAAR product to DAAR standard

The JTEP DAAR could be supported as a DAAR product available for others to use, e.g., the Joint National Training Capability (JNTC). However, there are many extant AAR capabilities, and it makes more sense to leverage them into a DAAR capability than to try to do the job with a unique product. We are proposing the core protocols of a standard, based on the JTEP DAAR design.

The DAAR control mechanisms described in this paper are few and simple. We believe that their simplicity is a key advantage recommending their use. The proposed protocols do not comprise a complete and finished standard for meeting all DAAR requirements of all possible LVC federations, but they are complete in the sense that have fully supported the conduct of actual DAARs in JTEP and XCTC exercises. We believe this makes a good starting point that a standards development committee can refine and elaborate.

1.3 Protocol development process and documentation

A secondary goal of this paper is to recommend the Unified Modeling Language v. 2.1 (UML) based model-driven development (MDD) process we have used to document the proposed protocols. Section 8 describes the MDD process. Space constraints restrict us to showing only a few of the UML diagrams. It is important to understand that the diagrams are not

² XCTC is a National Guard Bureau program to provide a combat training center-like experience to National Guard soldiers who are preparing for deployment. JTEP-developed technologies were critical enablers for initiating the XCTC program, and JTEP continues to develop new technologies for application in XCTC. XCTC, in turn, provides a valuable operational training environment for validating JTEP technologies.

simply drawings but are backed by a semantically rich structural and behavioral model.³

The key characteristics of the protocols are contained in the behavior model, for example, the exchange of messages shown on sequence diagrams. Structural aspects such as the contents of messages are lower level details, and are less important for understanding the proposed standards.

2. Requirements and use cases

2.1 Use cases

Figure 2 is a use case diagram illustrating the top-level user view of a DAAR federation. The agents (called “actors” in UML) who interact with DAAR federate systems are shown as stick figures on the left side of the diagram. The use cases are shown as ellipses, and lines show actor communications with use cases. On the right side of the diagram are references to the sequence diagrams that show the detailed behaviors that “realize” (implement) the use cases.⁴

The core DAAR protocol requirements are in *Log Exercise Data* (discussed in section 3), *Playback Exercise Data* (discussed in section 4), *View Exercise Data* (discussed in section 5), and *Manage Configuration* (discussed in section 6). *Control DAAR Systems* (discussed in section 7) is essential to a successful DAAR but this paper presents no recommendations for standardizing protocols.

2.1 Key requirements

The driving requirements are:

- The playback controls and viewpoint manipulation at the master site(s) must be reflected instantly at remote sites.

- All sites must be at the same “place” in the playback at the same time.⁵
- Map and video display quality at remote sites should preserve the clarity and responsiveness they have at the originating site.
- The DAAR must be capable of faster-than-real time replay (e.g., 60×).⁶

Some capabilities need to be supported by the DAAR protocol but may be optional for some implementations including:

- Live and post-exercise event annotations should be collected from multiple sources and sites and made accessible to the playback controller(s) or presenter(s).
- Playback and viewpoint control should be transferable among sites.

3. Logging

3.1 Proposed logging requirements

The DAAR playback protocols are designed to work with heterogeneous and distributed log files. Any logging method is satisfactory as long as the resultant data stores can be accessed in a way that is consistent with implementing the playback control interface protocol described in section 4. Specific performance and architectural requirements will vary for different federations and DAAR objectives.

The primary common requirement is that all data are consistently and accurately tagged with the exercise time. Data time tags and the wallclock time of logging may not be the same, for example, due to a lag in receipt of data by the logger, or because the exercise is conducted in virtual rather than real time. Time tag accuracy and resolution requirements are driven by the need to synchronize playback across heterogeneous sources so that (1) events are correctly sequenced, and (2) the AAR audience at distributed locations see the same situation and events simultaneously. In our

³ The model was created using Enterprise Architect, version 6.5.

⁴ The names of the use cases and the behavioral diagrams that realize them are not required to be the same by UML. However, the convention of applying similar names to semantic constructs that cross diagram types reinforces architectural integrity guidelines and makes the notation more self-documenting.

⁵ As discussed in sections 4 and 5, this does not necessarily imply that all sites are seeing exactly the same thing at the same time.

⁶ It is not necessarily required that all federates are capable of processing and displaying all entities and all updates. Differential filtering may be consistent with the goals of a particular DAAR.

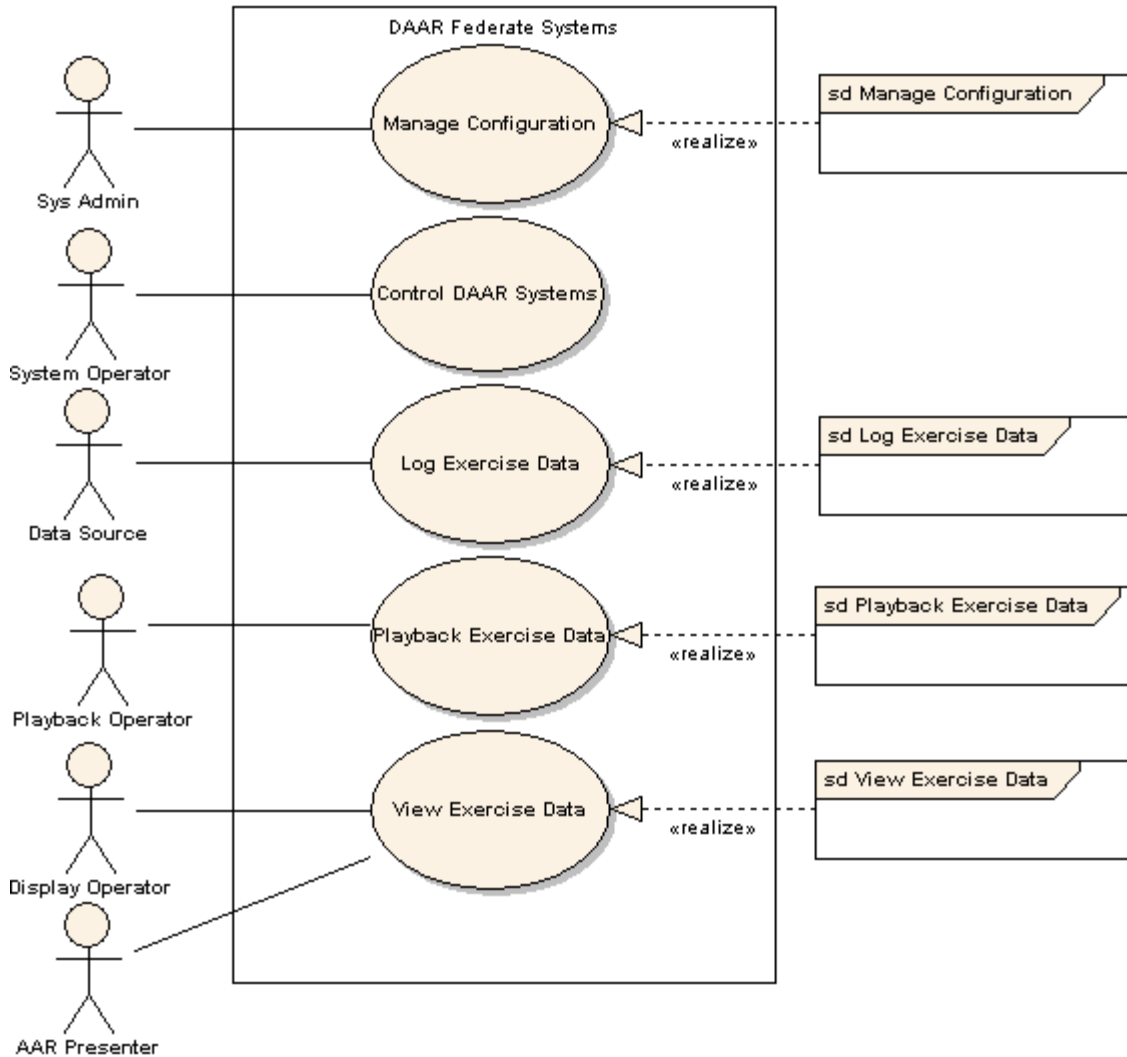


Figure 2.1 DAAR use cases.

experience, one second resolution and accuracy is generally sufficient for human perception, but resolution and accuracy for correct sequencing depends on the type of data displayed (e.g., correct direct fire ground force engagement sequencing may have a tighter time accuracy and resolution requirement).

Other requirements for logging depend primarily on network constraints and live exercise data distribution requirements. Here is a typical scenario that calls for distributed logging:

- There is insufficient bandwidth among AAR sites for each to stream playback at requisite playback speeds (e.g., 60x for an overview of

events that is scaled one hour actual time to one minute viewing time).

- Each AAR site receives the requisite data for playback during the live exercise.
- There is insufficient time to distribute data after the exercise from a master recording site to remote sites.
- Sites do not all have the same loggers.
- Security or firewall considerations limit distribution of data.

When logging is distributed, there may be discrepancies in the data recorded at different sites, for example, due to filtered real-time data distribution or differences in logging start-stop intervals. Protocols

are needed to determine if gaps in recorded data may compromise the integrity of a distributed playback, and to fill gaps as required. We defer recommendations for gap rectification to future work.

3.2 JTEP distributed logging implementation

To support direct access to time-series data during playback, the logger captures the raw data in one file and stores indexing information in another. The index references messages in the log files to (1) the time tag in the message if synced to an accurate time source, or (2) the system time on the logger machine at the time of capture. Because loggers at the remote sites do not receive protocol data units (PDUs) at the same instant as the master logger, times indexed by method 2 will have slight discrepancies, but these differences are too small to make any perceptible difference in a DAAR.

4. Playback

4.1 Proposed distributed playback protocol

Figure 4.1 shows a UML interaction overview diagram for distributed playback. Each box is a reference to the

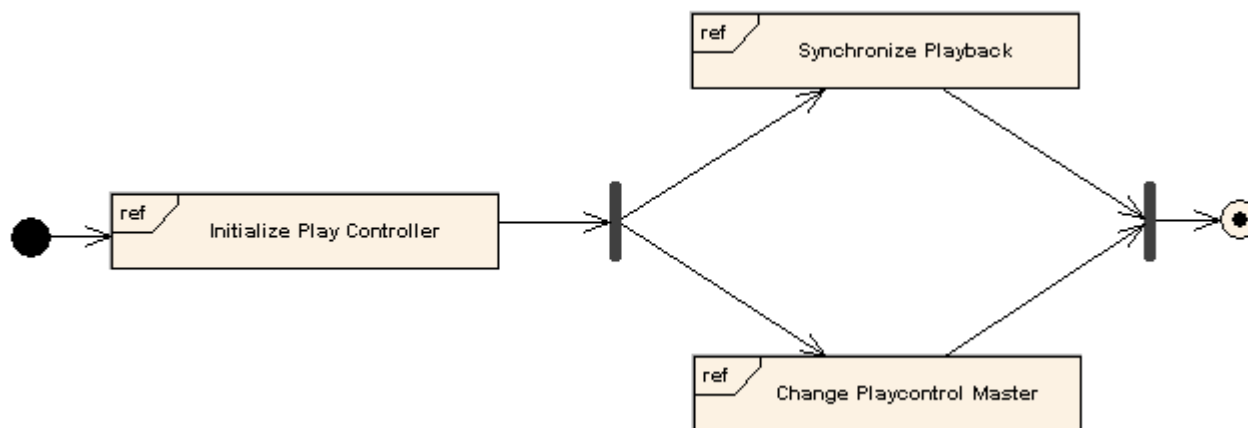


Figure 4.1 Playback interaction overview.

In the upper left of Figure 4.2 is a UML *alt* (i.e., alternative) “combined fragment” that shows two methods by which an operator can select a playback time: (1) directly by time entry, forward/back buttons, and the like; or (2) indirectly by selecting a time-stamped event annotation.⁷ When the operator selects a time, the playback control process synchronizes the players to a new time reference by sending a *startTime*

⁷ There may be additional ways to choose time, but the synchronization protocol does not depend on enumerating them.

sequence diagram that shows the specific protocol interactions among actors and federate applications. The first activity, *Initialize Play Controller*, is not addressed in this paper. It includes an optional protocol for distributing annotations entered at different sites. After initialization, *Synchronize Playback* and *Change Playback Master* are executed concurrently and asynchronously.

Synchronize Playback, shown in Figure 4.2, realizes the *Synchronize Playback* use case shown in Figure 2.1. Four object types participate in the transactions: a playback operator, a play controller, log data players, and media players. Interactions between the playback operator and the playback controller are considered to be integral to a single application. The specific characteristics of these interactions are not within scope of the proposed standards, but their causal and sequential relationships with interactions among the play controller, log players, and viewers/audio are part of the protocol. These applications can represent any combination of federates and sites, and therefore the messages exchanged among them are the core of the distributed playback synchronization protocol.

message.⁸ This paper does not propose the format and contents of messages, but it will be necessary to standardize methods and object models to achieve synchronized DAAR play among an arbitrary collection of loggers, players, and viewers.

⁸ It is desirable for players to play back persistent data that begins after the last played time and persists after the new start time. Although this might be specified as a recommended practice, it is probably out of scope to require that it be supported by all players.

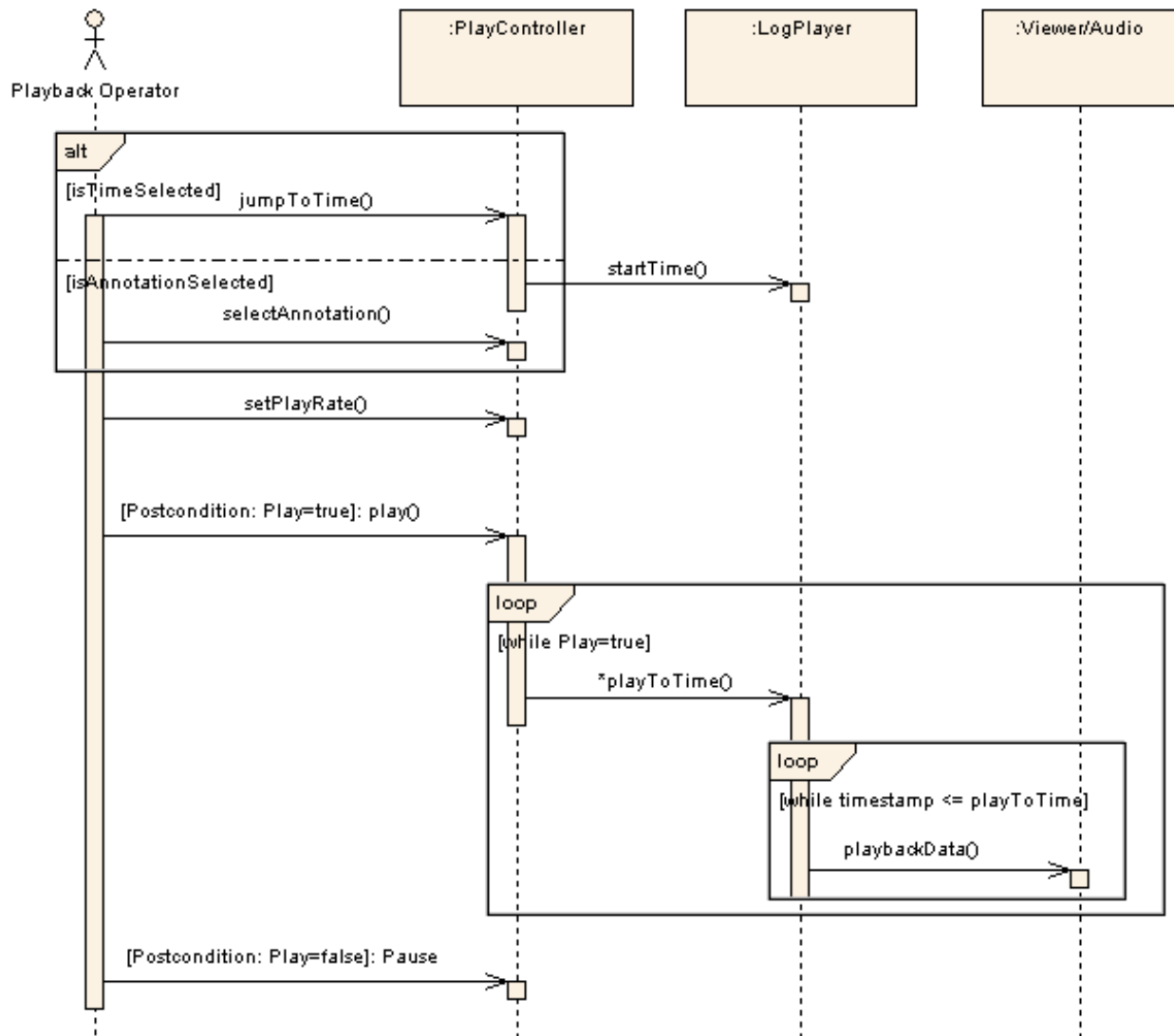


Figure 4.2 Playback synchronization.

When possible, the *startTime* should be sent by a reliable transport protocol. However, because a reliable protocol may not be available for some federates, a method for compensating for a lost *startTime* message is to redundantly include the referenced start time within each of the *playToTime* messages described below.

After the operator signals “play”, and until “pause”⁹ is selected, synchronization is driven by a sequence of *playToTime* messages, as shown in the outer *loop* combined fragment in Figure 4.2. Players interpret each *playToTime* as a directive to play all data in the

interval between the last time received (or the initial time) and the current time, as shown in the inner *loop* combined fragment in Figure 4.2. Playback pauses or stops when the flow of *playToTime* messages stops.

The granularity of the intervals is configurable (see section 6). To achieve the smoothest possible data playback using the play-to time method, the interval should match the finest granularity/accuracy of indexed time stamps among the log files. For example, if data are stamped to 100 ms granularity, a play-to interval of once per second would result in all data in an integral second being played at once, whereas 10 times per second would meter out data in their proper 100 ms intervals.

⁹ A “stop” action is not needed by the protocol because it is functionally indistinguishable from “pause.”

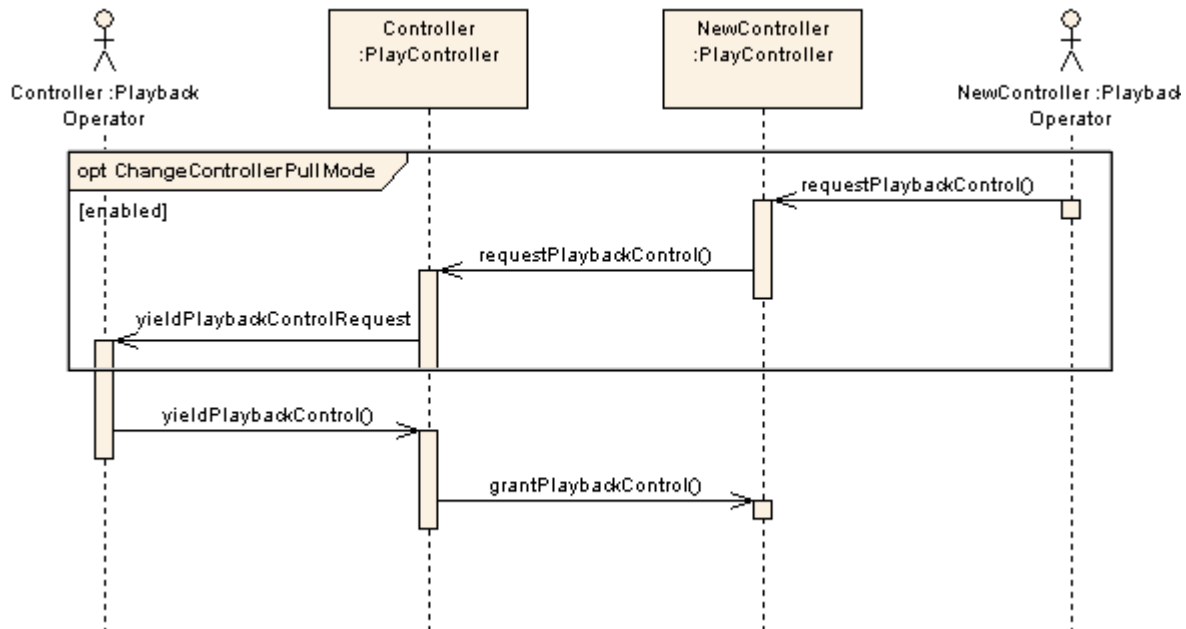


Figure 4.3 Change play controller.

It is not necessary to send playback rate information to players because fast playback is achieved simply by scaling the play-to time messages. For example, a 10× rate can be achieved by sending play-to messages at the same rate, but making the play-to time in two successive messages 10× as large as it is in normal speed playback.¹⁰

At any one time there is normally one operator and one playback control application, but the roles can be switched among sites, as shown in Figure 4.3. A “push” version of the protocol is initiated when the playback operator tells the play control application to switch to a new controller. The divesting play control application sends a *grantPlaybackControl* message to the acquiring application.¹¹ A “pull” version of the

¹⁰ An alternative method would be to send the same messages as with 1× rate, but reduce the spacing between the messages by a factor of 10. However, this method could result in excessive control message traffic when high-speed playback rates (e.g., 60×) are combined with fine granularity play-to intervals (e.g., 10 ms).

¹¹ Although it is not shown in Figure 4.3, the protocol should be elaborated to ensure reliability of the transfer, and recovery if the transfer fails (see, for example [3]).

protocol, as shown in the *opt* (i.e., optional) combined fragment, allows the operator of a potential controller to request transfer of control.

4.2 JTEP distributed playback implementation

At each site, a separate data or media player instance reads and plays each file, under the control of a single play control process located at the master site. We first used this method of separating file players and play control in the Deployable Force on Force Instrumented Range System (DFIRST), and found that it scales well and promotes software simplicity. JTEP extended this method to achieve playback composition in which players from heterogeneous systems are used together in synchrony.

The control messages sent to JTEP’s Distributed Interactive Simulation (DIS) log players are formatted as DIS SetData PDUs. Control messages sent to DFIRST players are in a native DFIRST message format. Because DIS uses best effort UDP rather than reliable TCP, we include a redundant copy of the current referenced start time in all play-to messages.

Because audio playback is heard by remote sites over the same audio channel that carries the presenter’s voice, JTEP does not distribute audio playback.

5. View Control

5.1 Proposed view control protocols

Figure 5.1 illustrates the generic DAAR protocol to synchronize views among distributed display viewer instances. The *Master* viewer application responds directly to *DisplayOperator* view control actions. The *Master* publishes its view parameters, and the remote *Slave* viewers synchronize to these parameters.

The protocols shown in Figure 5.1 synchronize views on the same type of viewer, for example, 2-D to 2-D or 3-D to 3-D. The same protocols could also be applied for synchronizing 2-D to 3-D displays or *vice versa*.

Figure 5.1 also shows synchronization of unspecified attributes. Examples include filter and declutter parameters, and show and hide commands for overlay graphics. This kind of distributed control requires a common way to reference information about displayed entities, which is typically in the domain of interoperability enablers such as HLA, TENA, and DIS. We defer discussion of a general protocol to a future standards development working group

The proposed protocol does not require that all sites have the same viewers, maps, terrain databases, etc. A disadvantage of such heterogeneity is the possibility that the presenter may refer to something on a local

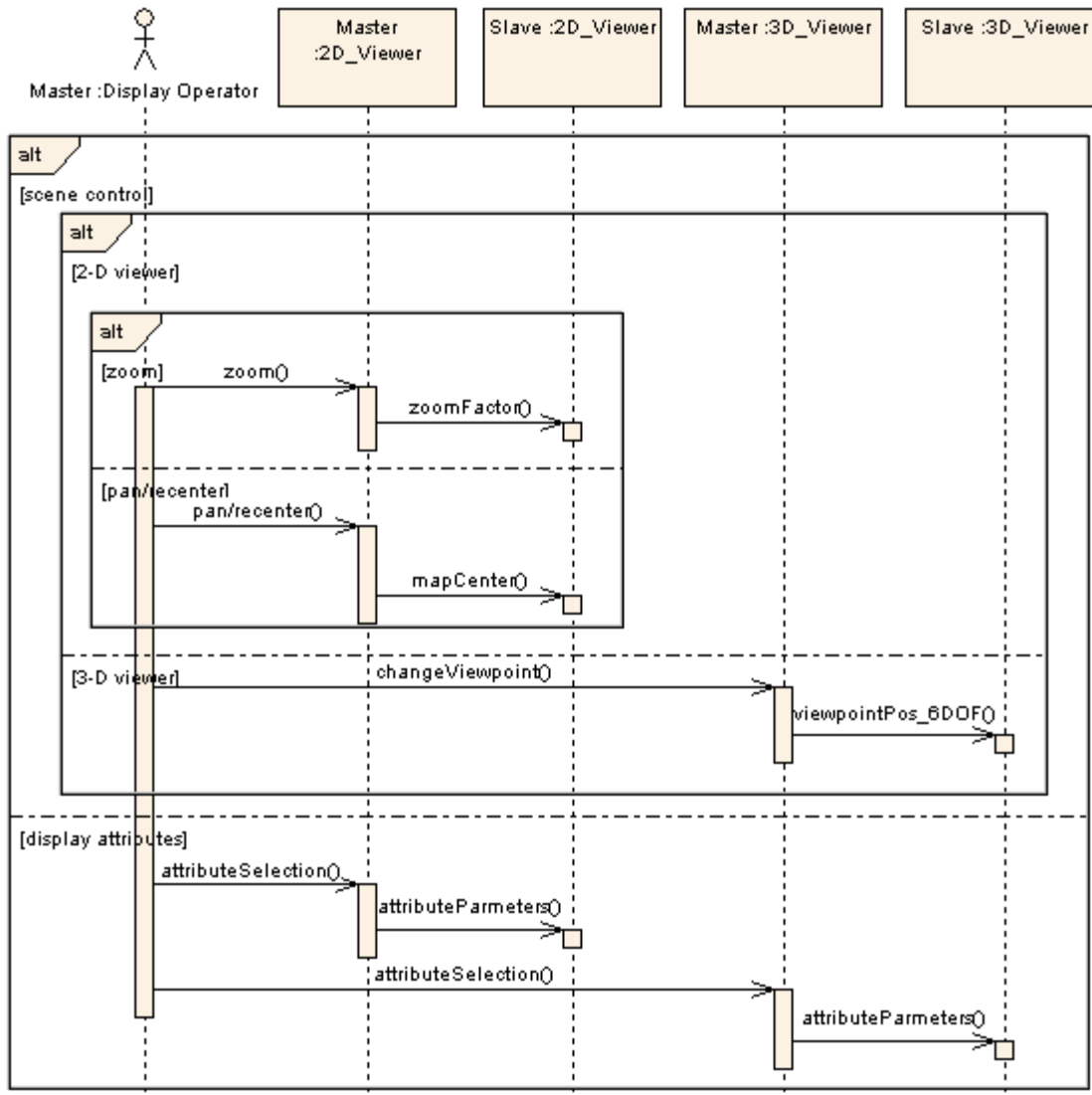


Figure 5.1 Viewer synchronization.

display that differs from what audiences at remote locations are seeing. However, it is sometimes impractical for all DAAR sites to have the same viewers, for example, due to licensing costs, security restrictions on software installation, application familiarity and support, or the need to quickly establish an improvisational DAAR federation. It is a matter of judgment whether the similarity of the view presented by heterogeneous viewers, maps, and terrain is sufficient to meet the objectives of a particular DAAR. Streaming video media and VTC are alternatives, but

may not be satisfactory unless remote sites have a very high bandwidth connection to the source site.

AAR presenters often use a laser pointer to call attention to particular areas or events on a screen. Figure 5.2 shows a protocol using master and slave *PointerTracker* applications to replicate local pointer movement on remote displays. This approach requires the presenter to use a mouse or gyro pointer instead of a laser device. Alternatively, the display operator might “shadow” the presenter’s laser pointing with mouse pointing on the master display.

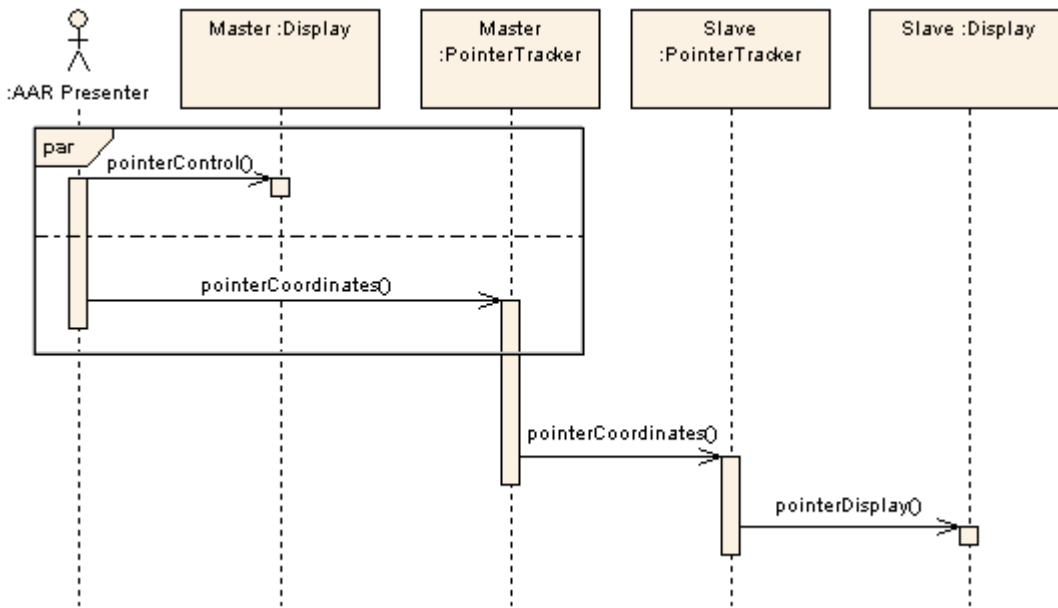


Figure 5.2 Synchronize pointer.

Figure 5.3 shows the protocol for transferring master view control. It is similar to the protocol for transferring play control, but it also includes coordinated transfer of pointer control.

5.2 JTEP view control implementation

Remote site 2-D and 3-D displays and mouse pointers are slaved to masters located at the presentation site. We adapted our 2-D display software to send and receive messages reporting map center and zoom factor. To synchronize MetaVR Virtual Reality Scene Generator (VRSG) 3-D displays, we configure the master 3-D display to export its “camera” viewpoint, which the operator manipulates using a “spaceball” controller device. DIS entity state PDUs report the eyepoint as position and orientation. The remote displays are configured to set their eyepoint to mimic

the fictitious entity that represents the master’s eyepoint.

Because the 3-D eyepoint is sent as a standard entity state PDU, it will be displayed on the 2-D display unless specifically hidden. Although it can sometimes be useful to operators to see the location of the 3-D eyepoint represented on the 2-D display, the 3-D operator’s control actions may cause it to move rapidly across the map. We assigned the entity a special DIS “enumeration” that we could categorically “hide” on the display so as not to distract the audience.

6. Configuration

6.1 Proposed configuration protocols

Configuration decisions include:

- Distribute logging or localize it to a central site. Different methods may be selected for different logs, sites, or federates, as appropriate for DAAR data needs, network constraints, etc.
- Send play control signals only to remote sites (who will play data from their own files), or distribute data played at one site to remote sites. Different methods may be selected for different players, sites, or federates.

- Stream video signals from one site to remote sites, or synchronize views of local instances of map displays or video running at each site. Different methods may be selected for different viewer types, sites, or federates.
- The play-to interval can be tailored to specific data and network requirements, as discussed in section 4.1.

This paper does not include recommendations for

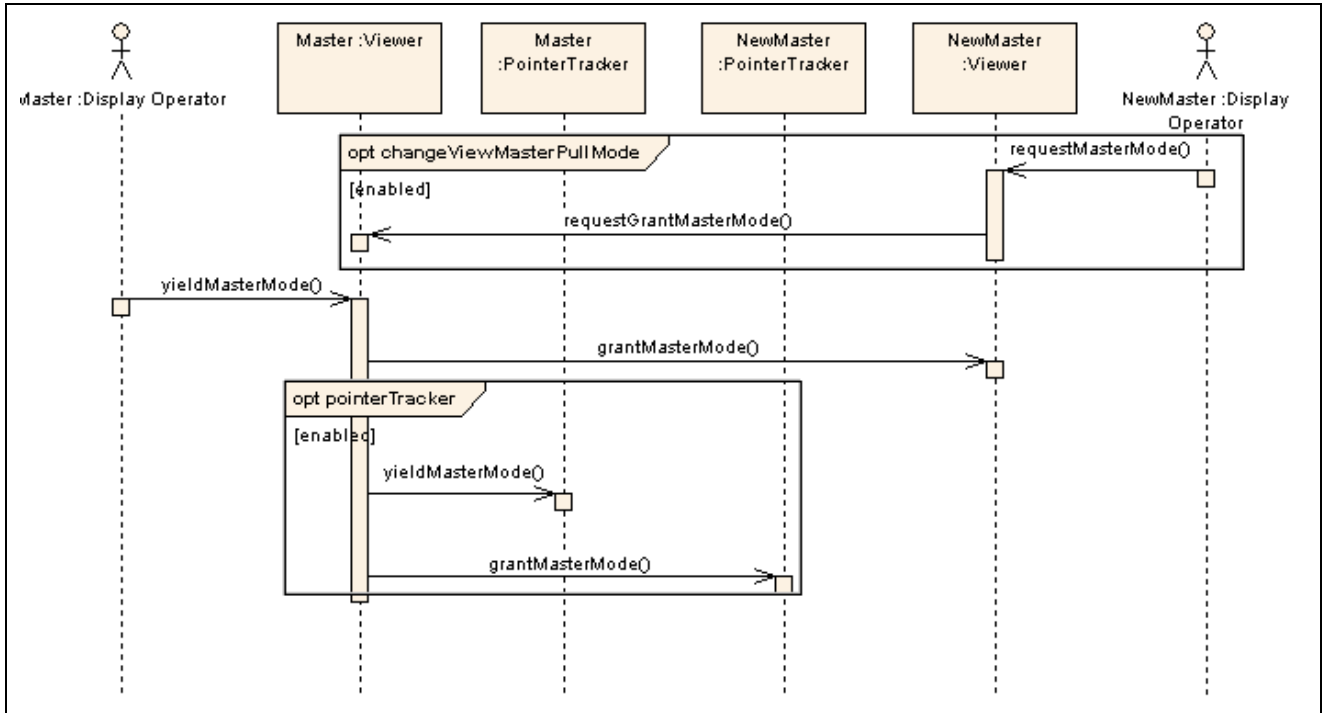


Figure 5.3 Change view master.

protocols to synchronize configuration. However, we think they should be included in a DAAR standard because they will lower the effort required to set up and maintain a DAAR federation and will reduce the incidence of errors and miscues.

6.2 JTEP configuration implementation

When JTEP operates in a severely bandwidth-constrained environment, logging, playback, and viewing are configured so that only control information is sent over the network. When bandwidth is more generous, players are sometimes run only at the master site, and data is steamed to the viewers at remote sites.

To synchronize configuration among sites, JTEP uses the services of its control executive software, which is discussed in section 7.2. The JTEP executive starts,

stops, and configures local and remote applications, which can effect any combination of centralized or distributed logging, playback, and viewing.

7. Control

7.1 Control protocols

We do not include a proposal for standardizing exercise control of federates. In general, mechanisms provided by HLA, TENA and DIS, and processes routinely practiced by LVC federations may be sufficient.

7.2 JTEP control implementation

JTEP has a central executive that configures, controls, and monitors heterogeneous loggers, players, and viewers in a distributed federation. However, it is not

necessary for a federate to be under JTEP control to participate in a JTEP DAAR.

8. The model-driven development process

In order to specify the DAAR protocols precisely and concisely, we determined that we needed an architectural description language with robust commercial software tool support. We chose UML 2.1 for this purpose, since this major revision of the software industry's standard modeling language offers many features that improve UML's precision and scalability [4]. We chose Sparx Enterprise Architect (EA) as our UML modeling tool because its user interface is intuitive and straightforward to use, and it offers many enhancements to UML that support the specification of enterprise architectures (e.g., requirements modeling, data modeling).

UML is a modeling language and not a method. To take full advantage of UML's precision and expressive power, we adopted a Model-Driven Development (MDD) process. The MDD process that we use to specify the DAAR protocols is a variation of the Unified Process [5], offering several pragmatic improvements which are summarized below.

Requirements-driven. Our MDD process specifies all system requirements using a UML model during the initial phase of the MDD lifecycle, and subsequently ensures that the rest of the UML model satisfies those requirements. Whereas the Unified Process uses Use Case diagrams for this purpose, we chose to augment Use Case diagrams with Enterprise Architect's custom extensions to UML for requirements modeling because they are better suited for specifying performance, user interface and testing requirements.¹²

By systematically relating all new model elements to the requirements that they satisfy throughout the MDD lifecycle, we can incrementally generate a verification & validation (V&V) audit trail with modest additional effort. The advantage of this incremental approach to V&V is that we are always sure that we are "modeling the system right" and "modeling the right system" [6].

Architecture-centric. Our MDD process emphasizes the importance of a precise and concise architectural model throughout the MDD lifecycle. The architectural guidelines that we have applied to the DAAR model are designed to address pragmatic model integrity and consistency issues, such as ensuring that all Use Cases

¹² This paper does not include an example requirements diagram, but they are part of the full DAAR model.

are realized by one or more behavioral diagrams (e.g., Activity or Sequence diagrams) in a straightforward and consistent manner, and that all messages on sequence diagrams are associated with operations or signals supported by the classes of the communicating objects. We have found that following architectural guidelines to maintain the integrity of a model provides essential quality controls for incremental, iterative development.

Agility. One of the most common criticisms of UML 2.x is that it is gratuitously large and complex. For our project, we identified a parsimonious subset of UML 2.x diagrams and features that are sufficient to define the JTEP DAAR protocols, and we avoided other language features that were unnecessary. Our subset includes the following seven diagrams selected from the thirteen diagrams that UML 2.0 offers: Use Cases, Classes, Sequences, Interaction Overview, Activities, Deployment, and Package diagrams. We augmented this "UML Lite" with the custom EA Requirements diagrams, since we needed to capture non-functional requirements more precisely than Use Cases allow.¹³

Language neutrality. Although we use UML to specify the DAAR protocols, our MDD process is language independent. It may be also be used with other visual modeling languages, such as the SysML or "visual OWL" (i.e., the Web Ontology Language with visual notation that supports its semantics) [7-9].¹⁴

9. Conclusions and future work

This paper generalizes the JTEP DAAR to a standard that is sufficient to meet the needs of federations with similar requirements. However, we do not presume that all federations have the same requirements [10]. Therefore we propose that a SISO working group be formed to establish a more comprehensive set of requirements and consider using the proposal in this paper as a starting point.¹⁵

Our plans for future work include application of semantic-web based concepts developed in the Open Netcentric Interoperability Standards for Training and Testing (ONISTT) program. ONISTT knowledge bases

¹³ The emerging SysML (Systems Modeling Language) standard for systems engineering includes Requirement diagrams as extensions to UML.

¹⁴ We will discuss this topic in a future paper.

¹⁵ DAAR proposals presented in [11] and [12] have some similarities with our proposal and some differences. The current paper stays close to our own practical experience and does not attempt to synthesize the recommendations.

(KBs) and Analyzer software can be applied to validate proposed DAAR confederations and to auto-generate selected configuration artifacts [13].

9. References

- [1] Reginald Ford, Scott Oberg, Richard Giuli, and Vicky Lamar, "The Joint Training Experimentation Program Approach to Distributed After Action Review", 04E-SIW-063, Simulation Interoperability Workshop, June 2004.
- [2] Reginald Ford, Virginia Gallagher, John Shockley, Isobel Wadsworth, Scott Oberg, and Richard Giuli, "The JTEP Takehome Package: An AAR Tool for Distributed Training", 04F-SIW-112, Simulation Interoperability Workshop, September 2004.
- [3] F. Hill, "Transfer Control DIS Standards Document," Draft SISO Reference Document, December 14, 2004.
- [4] Cris Kobryn, "UML 3.0 and the Future of Modeling," *Software and Systems Modeling*, 3(1), 4-8.
- [5] Ivar Jacobson, et al., *The Unified Software Development Process*. Reading, MA: Addison-Wesley, 1999.
- [6] Barry Boehm, Verifying and validating software requirements and design specifications. *IEEE Software*, Vol. 1(1), 1984.
- [7] UML: Superstructure v. 2.1.1, OMG document formal/07-02-03, February 2007.
- [8] OMG Systems Modeling Language (OMG SysML) Specification (proposed available revision), OMG document ptc/07-02-03, February 2007
- [9] OWL Web Ontology Language Semantics and Abstract Syntax, W3C Recommendation, February 2004
- [10] BAE Systems Technical Services, Joint After Action Review (AAR) Tool Set Functional Needs Analysis (FNA), prepared for Joint National Training Capability, Joint Management Office (Instrumentation), US Joint Forces Command.
- [11] Virginia Travers, William Ferguson, and Timothy Langevin, "A Federating Protocol for Distributed

After Action Review," Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC), 2006 Paper No. 2927

- [12] Randy Pitz and Curtis Armstrong, "Advanced Distributed Debrief for Coalition Training," Interservice/Industry Training, Simulation, and Education Conference, 2007 Paper No. 7159
- [13] Reginald Ford, David Hanz, Mark Johnson, and Daniel Elenius, "Purpose-Aware Interoperability: The ONISTT Ontologies and Analyzer," 07F-SIW-088, Simulation Interoperability Workshop, September 2007.

Author Biographies

REGINALD FORD is a Software Development Manager at SRI International. He has a B.A. from Yale University and a Ph.D. from University of California, Berkeley. He has 27 years of experience in test and training range instrumentation systems for the Army, Navy, Air Force, and Marine Corps. He is a technical lead for the development of Open Netcentric Standards for Training and Testing (ONISTT). He also manages JTEP integration of LVC software systems.

JOHN SHOCKLEY is Program Manager at SRI International. He has 24 years of experience in test and training range instrumentation systems for the Army, Navy, and Air Force. He began working on modeling and simulation aspects of these systems and has since participated in DIS/HLA standards development activities for over 13 years, concentrating on integrating live and virtual systems. He is the JTEP project leader.

CRIS KOBRYN is the CEO and Founder of PivotPoint Technology Corporation. He has 25 years experience in applying advanced software technologies to solve tough problems at Fortune 500, midsize and startup companies. Cris is an internationally recognized expert in visual modeling languages and model-driven development. He is collaborating with the JTEP team to improve the precision, scalability and reusability of the protocol architecture.