



IEEE Recommended Practice for Distributed Simulation Engineering and Execution Process Multi-Architecture Overlay

IEEE Computer Society

Developed by the
Simulation Interoperability Standards Organization Standards Activity Committee
(SISO SAC)

IEEE Std 1730.1™-2023
(Revision of IEEE Std 1730.1-2013)



IEEE Recommended Practice for Distributed Simulation Engineering and Execution Process Multi-Architecture Overlay

Developed by the

**Simulation Interoperability Standards Organization Standards Activity Committee
(SISO SAC)**
of the
IEEE Computer Society

Approved 8 November 2023

IEEE SA Standards Board

THIS IS A REDLINE DOCUMENT PRODUCED BY TECHSTREET, AN ASME COMPANY.

This document is intended to provide users with an indication of changes from one edition to the next. It includes a full-text version of the new document, plus an indication of changes from the previous version.

Redlines are designed to save time and improve efficiencies by using the latest software technology to find and highlight document changes. More professionals are using valuable new technologies like redlines, to help improve outcomes in a fast paced global business world.

Because it may not be technically possible to capture all changes accurately, it is recommended that users consult previous editions as appropriate. In all cases, only the current base version of this publication is to be considered the official document.

This Redline is produced solely by Techstreet. IEEE has not prepared this Redline and is not responsible for its accuracy.

Redline Processing Notes:

1. ~~Red Text~~ - Red strikethrough text denotes deletions.
2. Blue Text - Blue underlined text denotes modifications and additions.

Abstract: A recommended practice for applying the Distributed Simulation Engineering and Execution Process (DSEEP) to the development and execution of distributed simulation environments that include more than one distributed simulation architecture is described. The distributed simulation architectures to which the recommended practice applies include Distributed Interactive Simulation (DIS), High Level Architecture (HLA), and Test and Training Enabling Architecture (TENA). The DSEEP Multi-Architecture Overlay (DMAO) identifies and describes multi-architecture issues and provides recommended actions for simulation environment developers faced with those issues. The DMAO also augments the DSEEP lists of inputs, recommended tasks, and outcomes with additional inputs, recommended tasks, and outcomes that apply to multi-architecture simulation environments. This document is an overlay to the DSEEP, which is a separate recommended practice.

Keywords: DIS, Distributed Interactive Simulation, distributed simulation, Distributed Simulation Engineering and Execution Process, DMAO, DSEEP, High Level Architecture, HLA, IEEE 1730™, IEEE 1730.1™, multi-architecture, systems engineering methodology, TENA, Test and Training Enabling Architecture

The Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York, NY 10016-5997, USA

Copyright © 2024 by The Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 21 February 2024. Printed in the United States of America.

IEEE is a registered trademark in the U.S. Patent & Trademark Office, owned by The Institute of Electrical and Electronics Engineers, Incorporated.

PDF: ISBN 979-8-8557-0479-2 STD26746
Print: ISBN 979-8-8557-0480-8 STDPD26746

IEEE prohibits discrimination, harassment, and bullying.

For more information, visit <https://www.ieee.org/about/corporate/governance/p9-26.html>.

No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.

Important Notices and Disclaimers Concerning IEEE Standards Documents

IEEE Standards documents are made available for use subject to important notices and legal disclaimers. These notices and disclaimers, or a reference to this page (<https://standards.ieee.org/ipr/disclaimers.html>), appear in all standards and may be found under the heading “Important Notices and Disclaimers Concerning IEEE Standards Documents.”

Notice and Disclaimer of Liability Concerning the Use of IEEE Standards Documents

IEEE Standards documents are developed within IEEE Societies and subcommittees of IEEE Standards Association (IEEE SA) Board of Governors. IEEE develops its standards through an accredited consensus development process, which brings together volunteers representing varied viewpoints and interests to achieve the final product. IEEE standards are documents developed by volunteers with scientific, academic, and industry-based expertise in technical working groups. Volunteers are not necessarily members of IEEE or IEEE SA and participate without compensation from IEEE. While IEEE administers the process and establishes rules to promote fairness in the consensus development process, IEEE does not independently evaluate, test, or verify the accuracy of any of the information or the soundness of any judgments contained in its standards.

IEEE makes no warranties or representations concerning its standards, and expressly disclaims all warranties, express or implied, concerning this standard, including but not limited to the warranties of merchantability, fitness for a particular purpose and non-infringement. IEEE Standards documents do not guarantee safety, security, health, or environmental protection, or guarantee against interference with or from other devices or networks. In addition, IEEE does not warrant or represent that the use of the material contained in its standards is free from patent infringement. IEEE Standards documents are supplied “AS IS” and “WITH ALL FAULTS.”

Use of an IEEE standard is wholly voluntary. The existence of an IEEE standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard.

In publishing and making its standards available, IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity, nor is IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing any IEEE Standards document, should rely upon their own independent judgment in the exercise of reasonable care in any given circumstances or, as appropriate, seek the advice of a competent professional in determining the appropriateness of a given IEEE standard.

IN NO EVENT SHALL IEEE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO: THE NEED TO PROCURE SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE PUBLICATION, USE OF, OR RELIANCE UPON ANY STANDARD, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE AND REGARDLESS OF WHETHER SUCH DAMAGE WAS FORESEEABLE.

Translations

The IEEE consensus balloting process involves the review of documents in English only. In the event that an IEEE standard is translated, only the English version published by IEEE is the approved IEEE standard.

Official statements

A statement, written or oral, that is not processed in accordance with the IEEE SA Standards Board Operations Manual shall not be considered or inferred to be the official position of IEEE or any of its committees and shall not be considered to be, nor be relied upon as, a formal position of IEEE. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that the presenter's views should be considered the personal views of that individual rather than the formal position of IEEE, IEEE SA, the Standards Committee, or the Working Group. Statements made by volunteers may not represent the formal position of their employer(s) or affiliation(s).

Comments on standards

Comments for revision of IEEE Standards documents are welcome from any interested party, regardless of membership affiliation with IEEE or IEEE SA. However, **IEEE does not provide interpretations, consulting information, or advice pertaining to IEEE Standards documents.**

Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Since IEEE standards represent a consensus of concerned interests, it is important that any responses to comments and questions also receive the concurrence of a balance of interests. For this reason, IEEE and the members of its Societies and subcommittees of the IEEE SA Board of Governors are not able to provide an instant response to comments, or questions except in those cases where the matter has previously been addressed. For the same reason, IEEE does not respond to interpretation requests. Any person who would like to participate in evaluating comments or in revisions to an IEEE standard is welcome to join the relevant IEEE working group. You can indicate interest in a working group using the Interests tab in the Manage Profile & Interests area of the [IEEE SA myProject system](#).¹ An IEEE Account is needed to access the application.

Comments on standards should be submitted using the [Contact Us](#) form.²

Laws and regulations

Users of IEEE Standards documents should consult all applicable laws and regulations. Compliance with the provisions of any IEEE Standards document does not constitute compliance to any applicable regulatory requirements. Implementers of the standard are responsible for observing or referring to the applicable regulatory requirements. IEEE does not, by the publication of its standards, intend to urge action that is not in compliance with applicable laws, and these documents may not be construed as doing so.

¹ Available at: <https://development.standards.ieee.org/myproject-web/public/view.html#landing>.

² Available at: <https://standards.ieee.org/content/ieee-standards/en/about/contact/index.html>.

Data privacy

Users of IEEE Standards documents should evaluate the standards for considerations of data privacy and data ownership in the context of assessing and using the standards in compliance with applicable laws and regulations.

Copyrights

IEEE draft and approved standards are copyrighted by IEEE under U.S. and international copyright laws. They are made available by IEEE and are adopted for a wide variety of both public and private uses. These include both use, by reference, in laws and regulations, and use in private self-regulation, standardization, and the promotion of engineering practices and methods. By making these documents available for use and adoption by public authorities and private users, neither IEEE nor its licensors waive any rights in copyright to the documents.

Photocopies

Subject to payment of the appropriate licensing fees, IEEE will grant users a limited, non-exclusive license to photocopy portions of any individual standard for company or organizational internal use or individual, non-commercial use only. To arrange for payment of licensing fees, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; +1 978 750 8400; <https://www.copyright.com/>. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

Updating of IEEE Standards documents

Users of IEEE Standards documents should be aware that these documents may be superseded at any time by the issuance of new editions or may be amended from time to time through the issuance of amendments, corrigenda, or errata. An official IEEE document at any point in time consists of the current edition of the document together with any amendments, corrigenda, or errata then in effect.

Every IEEE standard is subjected to review at least every 10 years. When a document is more than 10 years old and has not undergone a revision process, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE standard.

In order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit [IEEE Xplore](#) or [contact IEEE](#).³ For more information about the IEEE SA or IEEE's standards development process, visit the IEEE SA Website.

Errata

Errata, if any, for all IEEE standards can be accessed on the [IEEE SA Website](#).⁴ Search for standard number and year of approval to access the web page of the published standard. Errata links are located under the Additional Resources Details section. Errata are also available in [IEEE Xplore](#). Users are encouraged to periodically check for errata.

³ Available at: <https://ieeexplore.ieee.org/browse/standards/collection/ieee>.

⁴ Available at: <https://standards.ieee.org/standard/index.html>.

Patents

IEEE standards are developed in compliance with the [IEEE SA Patent Policy](#).⁵

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken by the IEEE with respect to the existence or validity of any patent rights in connection therewith. If a patent holder or patent applicant has filed a statement of assurance via an Accepted Letter of Assurance, then the statement is listed on the IEEE SA Website at <https://standards.ieee.org/about/sasb/patcom/patents.html>. Letters of Assurance may indicate whether the Submitter is willing or unwilling to grant licenses under patent rights without compensation or under reasonable rates, with reasonable terms and conditions that are demonstrably free of any unfair discrimination to applicants desiring to obtain such licenses.

Essential Patent Claims may exist for which a Letter of Assurance has not been received. The IEEE is not responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patents Claims, or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility. Further information may be obtained from the IEEE Standards Association.

IMPORTANT NOTICE

Technologies, application of technologies, and recommended procedures in various industries evolve over time. The IEEE standards development process allows participants to review developments in industries, technologies, and practices, and to determine what, if any, updates should be made to the IEEE standard. During this evolution, the technologies and recommendations in IEEE standards may be implemented in ways not foreseen during the standard's development. IEEE standards development activities consider research and information presented to the standards development group in developing any safety recommendations. Other information about safety practices, changes in technology or technology implementation, or impact by peripheral systems also may be pertinent to safety considerations during implementation of the standard. Implementers and users of IEEE Standards documents are responsible for determining and complying with all appropriate safety, security, environmental, health, and interference protection practices and all applicable laws and regulations.

⁵ Available at: <https://standards.ieee.org/about/sasb/patcom/materials.html>.

Participants

At the time this recommended practice was completed, the Distributed Simulation Engineering and Executions Process Mult-Architecture Overlay Product Development Group/ Working Group had the following membership:

Robert Lutz, *Chair*
Katherine L. Morse, *Vice Chair*

Jane Bachman
Jack Borah
Charles Brooks
Don Brutzman
Dannie Cutts
David L. Drake
Don Gadomski
Michael Heaphy

Jean-Louis Igarza
Paul Lowe
Farid Mamaghani
Mark McCall
Dawn Moffat
Michael Montgomery
David Phillips
Laurent Prignac
Greg Quilliam

David Ronnfeldt
George Schleh
Roy Scrudder
Robert Siegfried
Clyde Smithson
Tom van den Berg
René Verhage
Eric Whittington

The following members of the individual Standards Association balloting group voted on this recommended practice. Balloters may have voted for approval, disapproval, or abstention.

Tom Berg
Curtis Blais
Shelly Blake-Plock
David Fuschi
Werner Hoelzl
Murray Little

Greg Luri
Robert Lutz
Mark McCall
Bjorn Moller
Katherine L. Morse
Spencer Ochsner

Sivaraman P
Bansi Patel
R. K. Rannow
Christopher Rouget
Thomas Starai
John Vergis

When the IEEE SA Standards Board approved this recommended practice on 8 November 2023, it had the following membership:

David J. Law, *Chair*
Ted Burse, *Vice Chair*
Gary Hoffman, *Past Chair*
Konstantinos Karachalios, *Secretary*

Sara R. Biyabani
Doug Edwards
Ramy Ahmed Fathy
Guido R. Hiertz
Yousef Kimiagar
Joseph L. Koepfinger*
Thomas Koshy
John D. Kulick

Joseph S. Levy
Howard Li
Johnny Daozhuang Lin
Gui Lin
Xiaohui Liu
Kevin W. Lu
Daleep C. Mohla
Andrew Myles

Paul Nikolich
Annette D. Reilly
Robby Robson
Lei Wang
F. Keith Waters
Karl Weber
Philip B. Winston
Don Wright

*Member Emeritus

Introduction

This introduction is not part of IEEE Std 1730.1-2023, IEEE Recommended Practice for Distributed Simulation Engineering and Execution Process Multi-Architecture Overlay.

Modeling and simulation (M&S) has long been recognized as a critical technology for managing the complexity associated with modern systems. In many industries, M&S is a key enabler of many core systems engineering functions. For instance, early in the systems engineering process, relatively coarse constructive models are generally used to identify capability gaps, define systems requirements, and examine/compare potential system solutions. As preferred concepts are identified, higher-fidelity models are used to evaluate alternative system designs and to support initial system development activities. As design and development continues, very high-fidelity models are used to support component-level design and development, as well as developmental test. Finally, combinations of live, virtual, and constructive M&S assets are frequently used to support operational test and training requirements.

The advent of modern networking technology and the development of supporting protocols and architectures have led to widespread use of distributed simulation. The strategy behind distributed simulation is to use networks and support simulation services to link existing M&S assets into a single unified simulation environment. This approach provides several advantages as compared to development and maintenance of large monolithic stand-alone simulation systems. First, it allows each individual simulation application to be co-located with its resident subject matter expertise rather than having to develop and maintain a large stand-alone system in one location. In addition, it facilitates efficient use of past M&S investments, as new, very powerful simulation environments can be quickly configured from existing M&S assets. Finally, it provides flexible mechanisms to integrate hardware and/or live assets into a unified simulation environment for test or training, and it is much more scalable than stand-alone systems. Examples of hardware and/or live assets are stimulators, live platforms, operational (command and control) systems, field instrumentation, and tracking devices.

There are also some disadvantages of distributed simulation. Many of the issues related to distributed simulation are related to interoperability concerns. Interoperability refers to the ability of disparate simulation systems and supporting utilities (e.g., viewers, loggers) to interact at runtime in a coherent fashion. There are many technical issues that affect interoperability, such as consistency of time advancement mechanisms, compatibility of supported services, data format compatibility, and even semantic mismatches for runtime data elements.

Distributed simulation environments may involve the integration of hardware and/or live assets. The integration of hardware and/or live assets creates additional concerns besides the earlier mentioned interoperability concerns between simulation systems. Examples of technical issues that are a consequence of integrating hardware and/or live assets include hard real time execution, mix of ground truth (simulation) data with non-ground truth (operational or live) data, and time space position information updates.

The capabilities provided by today's distributed simulation architectures are designed to address such issues and allow coordinated runtime interaction among participating simulations. Three distributed simulation architectures are in common use in the M&S community and will be explicitly mentioned in this recommended practice, although these guidelines are not limited to those architectures. The widely used architectures are Distributed Interactive Simulation (DIS) as defined by the IEEE 1278™ family of standards, High Level Architecture (HLA) as defined by the IEEE 1516™ family of standards, and the Test and Training Enabling Architecture (TENA) used on test and training ranges. Associated with each of these simulation architectures is a systems engineering process for developing distributed simulation systems using that architecture. These processes, while individually effective, have been developed separately with the technical features, supporting facilities, and user community requirements of a specific architecture in mind, and consequently are different from each other in both large and small ways.

In some situations, sponsor requirements may necessitate the selection of simulations, interfaces to live systems, and data collectors whose external interfaces are aligned with more than one simulation architecture. This is what is known as a multi-architecture simulation environment. When more than one simulation architecture is used in the same simulation environment, interoperability problems are compounded by the architectural differences. For such simulation environments to operate properly, reconciling middleware incompatibilities, dissimilar metamodels for data exchange, and differences in the nature of the services that are provided by the architectures is necessary. Developers have devised many different workarounds for these types of interoperability problems over the years. One possible solution is to choose a single architecture for the simulation environment and require all participants to modify the native interfaces of their applications to conform to it. While this solution is relatively straightforward and easy to test, it is usually impractical (particularly in large applications) because of the high cost and schedule penalties incurred. Another approach is the use of gateways, which are independent software applications that translate between the protocol used by one simulation architecture to that of a different simulation architecture (see Figure 1). While effective, gateways represent another potential source of error (or failure) within the simulation environment, can introduce undesirable latencies into the system, and add to the complexity of simulation environment testing. In addition, many gateways are legacy point solutions that provide support only for a very limited number of services and only for very specific versions of the supported simulation architectures. Thus, it may be difficult to find a suitable gateway that fully supports the needs of a given application. For the relatively small number of general-purpose gateways that are configurable, the effort required to perform the configuration function can be significant and can result in excessive consumption of project resources.

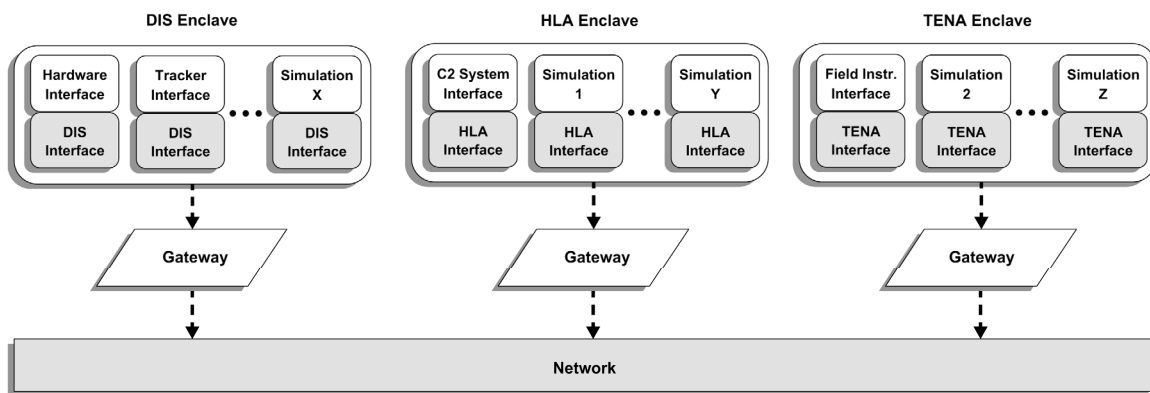


Figure 1—Gateway configuration

The use of middleware is a similar approach but provides the translation services in software directly coupled to the simulation instead of an independent application (see Figure 2).⁶ While middleware approaches are also effective, they introduce many of the same technical issues that are associated with gateways (e.g., source of error, possible latency penalties). In general, all of these “solutions” have limitations and cost implications that increase technical, cost, and schedule risk for multi-architecture developments.

⁶ Note that this use of the term “middleware” is different in some user communities, who may use this term to refer to the infrastructure elements that provide distributed simulation services [e.g., the HLA Runtime Infrastructure (RTI)].

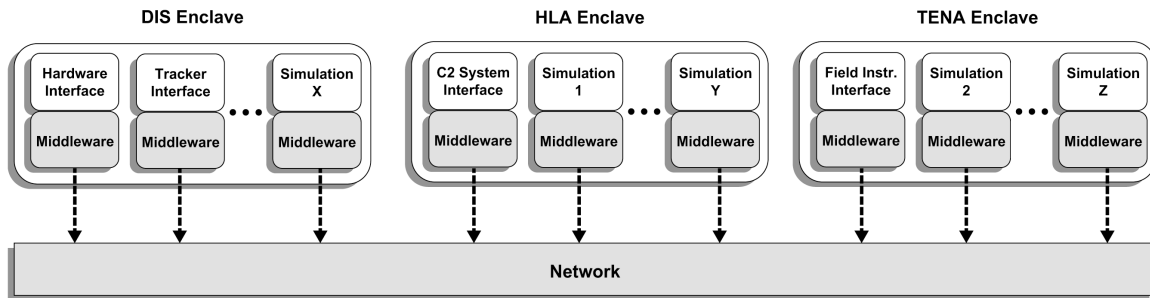


Figure 2—Middleware configuration

Many technical issues arise when multi-architecture simulation environments are being developed and executed. These issues tend to increase program costs and can increase technical risk and impact schedules if not resolved adequately. One widely reported and particularly vexing issue concerns the situation where users from different architecture communities are brought together to develop a single multi-architecture distributed simulation environment, but the differences in the development processes native to each architecture lead to misunderstandings, misinterpretations, and general confusion among team members. This impacts risk from many different perspectives, and creates a persistent barrier to effective collaboration. An effective solution to this problem is to establish a common systems engineering process for the development and execution of multi-architecture simulation environments.

Many process standards already exist in the systems and software engineering communities. Rather than develop an entirely new process standard for multi-architecture simulation environments, leveraging and extending an existing standard seems to be much more efficient and helps to avoid the creation of potentially competing and conflicting products. While the principles that underlie existing systems and software standards (e.g., Electronic Industries Alliance EIA-632, International Organization for Standardization/International Electrotechnical Commission ISO/IEC 15288) are certainly applicable, direct reuse of any process standard outside of the M&S domain would require a significant degree of tailoring. A more effective choice would be an existing distributed simulation engineering process standard, preferably one that is independent of any individual simulation architecture. The IEEE 1730 Distributed Simulation Engineering and Execution Process (DSEEP) fits that requirement and provides a suitable choice for the desired multi-architecture process framework.

The DSEEP is a generalized systems engineering process for building and executing distributed simulation environments, independent of the underlying simulation architecture. Intended as a high-level framework for simulation environment construction and execution, the DSEEP is the successor of architecture-dependent engineering processes, e.g., concerning HLA (IEEE Std 1516.3™) or DIS (IEEE Std 1278.3™).

The DSEEP represents a tailoring of best practices in the systems and software engineering communities to the M&S domain, and in particular, to the development and execution of distributed simulation environments. The DSEEP is simulation architecture-neutral, but it does contain annexes that map this architecture-neutral view to DIS, HLA, and TENA terminology. A top-level view of the DSEEP is provided in Figure 3.

A short description of each of these seven major steps follows:

Step 1: Define Simulation Environment Objectives. The user, the sponsor, and the development/integration team define and agree on a set of objectives and document what is to be accomplished to achieve those objectives.

Step 2: Perform Conceptual Analysis. The development team performs scenario development and conceptual modeling and develops the simulation environment requirements based upon the characteristics of the problem space.

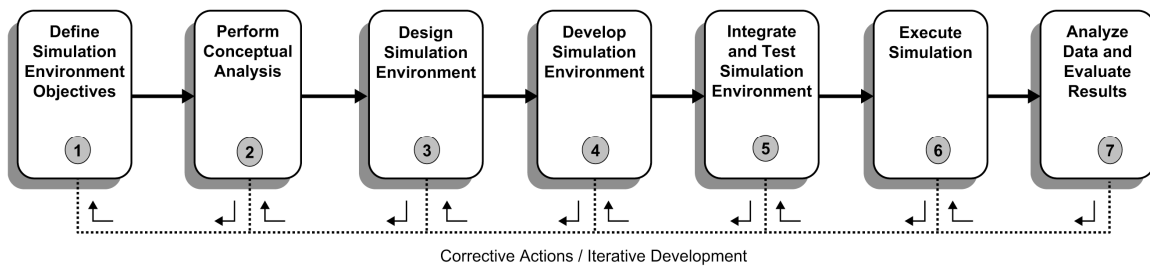


Figure 3—Distributed Simulation Engineering and Execution Process (DSEEP) top-level process flow view

Step 3: Design Simulation Environment. Existing member applications that are suitable for reuse are identified; design activities for member application modifications and/or new member applications are performed; required functionalities are allocated to the member applications; and a plan is developed for development and implementation of the simulation environment.

Step 4: Develop Simulation Environment. The simulation data exchange model is developed; simulation environment agreements are established; and new member applications and/or modifications to existing member applications are implemented.

Step 5: Integrate and Test Simulation Environment. All necessary integration activities are performed, and testing is conducted to verify that interoperability requirements are being met.

Step 6: Execute Simulation. The simulation environment is executed and the output data from the execution is pre-processed.

Step 7: Analyze Data and Evaluate Results. The output data from the execution is analyzed and evaluated, and results are reported back to the user/sponsor.

In the DSEEP document, each of these seven steps is further decomposed into a set of interrelated lower-level activities. Each activity is characterized according to a set of required activity inputs, one or more activity outcomes, and a list of recommended specific tasks. Although these activity descriptions are identified in a logical sequence, the DSEEP emphasizes that iteration and concurrency are to be expected, not only across activities within a step but across steps as well.

Although the DSEEP provides the guidance required to build and execute a distributed simulation environment, the implicit assumption within the DSEEP is that only a single simulation architecture is being used. The only reference to multi-architecture development in the DSEEP is provided in the following paragraph from DSEEP Activity 3.2 (Design Simulation Environment):

“In some large simulation environments, it is sometimes necessary to mix several simulation architectures. This poses special challenges to the simulation environment design, as sophisticated mechanisms are sometimes needed to reconcile disparities in the architecture interfaces. For instance, gateways or bridges to adjudicate between different on-the-wire protocols are generally a required element in the overall design, as well as mechanisms to address differences in simulation data exchange models. Such mechanisms are normally formalized as part of the member application agreements, which are discussed in Step 4.” (IEEE Std1730)⁷

Clearly, additional guidance is necessary to support the development of multi-architecture simulation environments. However, the major steps and activities defined in the DSEEP are generally applicable to either single- or multi-architecture development. Thus, the DSEEP provides a viable framework for the

⁷ Throughout the document, text quoted directly from the DSEEP (IEEE Std 1730) is both enclosed in quotation marks and highlighted with gray shading.

development of the desired process, but it should be augmented with additional tasks as necessary to address the issues that are unique to (or at least exacerbated by) multi-architecture development. Such augmenting documentation is often referred to as an *overlay*. This document constitutes such an overlay to the DSEEP.

In summary, the strategy implemented in this product is to augment the major DSEEP steps and activities with the additional tasks needed to address the issues that are unique to (or at least exacerbated by) multi-architecture development. These tasks collectively define a “how to” guide for developing and executing multi-architecture simulation environments, based on recognized best practices.

Contents

1. Overview	14
1.1 Scope	14
1.2 Purpose	14
1.3 Word usage	14
2. Normative references.....	15
3. Definitions, acronyms, and abbreviations	15
3.1 Definitions	15
3.2 Acronyms and abbreviations	16
4. Multi-architecture issues and solutions	19
4.1 Step 1: Define simulation environment objectives	21
4.2 Step 2: Perform conceptual analysis.....	27
4.3 Step 3: Design simulation environment.....	32
4.4 Step 4: Develop simulation environment.....	56
4.5 Step 5: Integrate and test simulation environment.....	71
4.6 Step 6: Execute simulation	77
4.7 Step 7: Analyze data and evaluate results.....	83
Annex A (informative) Bibliography	86

IEEE Recommended Practice for Distributed Simulation Engineering and Execution Process Multi-Architecture Overlay (~~DMAO~~)

~~IMPORTANT NOTICE: IEEE Standards documents are not intended to ensure safety, security, health, or environmental protection, or ensure against interference with or from other devices or networks. Implementers of IEEE Standards documents are responsible for determining and complying with all appropriate safety, security, environmental, health, and interference protection practices and all applicable laws and regulations.~~

~~This IEEE document is made available for use subject to important notices and legal disclaimers. These notices and disclaimers appear in all publications containing this document and may be found under the heading "Important Notice" or "Important Notices and Disclaimers Concerning IEEE Documents." They can also be obtained on request from IEEE or viewed at <http://standards.ieee.org/IPR/disclaimers.html>.~~

1. Overview

~~A recommended practice for applying the Distributed Simulation Engineering and Execution Process (DSEEP) to the development and execution of distributed simulation environments that include more than one distributed simulation architecture is described. Distributed simulation architectures covered include Distributed Interactive Simulation (DIS), High Level Architecture (HLA), and Test and Training Enabling Architecture (TENA). This document identifies and describes multi-architecture issues; provides recommended actions for simulation environment developers faced with those issues; and augments the DSEEP lists of inputs, recommended tasks, and outcomes.~~

~~The DSEEP Multi-Architecture Overlay (DMAO) (IEEE Std 1730.1) is intended as a companion guide to the DSEEP (IEEE Std 1730™-2010).[†] The simulation environment user/developer should assume that the guidance provided by the DSEEP is applicable to both single and multi-architecture developments. The DMAO provides the additional guidance needed to address the special concerns of the multi-architecture user/developer. In other words, the DMAO does not replace the DSEEP for multi-architecture simulation environment development; it augments the DSEEP.~~

1.1 Scope

This [recommended practice document](#) defines the issues that are either unique to or exacerbated by the use of multiple simulation architectures in the same simulation environment, along with recommended actions for properly addressing these issues. The [Distributed Simulation Engineering and Execution Process \(DSEEP\)](#) (IEEE Std 1730⁸) provides the overarching process framework. The alignment of DSEEP activities with those additional tasks necessary to address the multi-architecture concerns collectively define a “how to” guide for developing and executing multi-architecture simulation environments, based on industry best practices.

1.2 Purpose

[The purpose of this recommended practice is to identify DSEEP activities requiring special attention in multi-architecture simulation environments and to provide additional guidance on multi-architecturespecific inputs, tasks, and outcomes to these issues.](#)

1.3 Word usage

[The word *shall* indicates mandatory requirements strictly to be followed in order to conform to the standard and from which no deviation is permitted \(*shall* equals *is required to*\).^{9, 10}](#)

^{8†} Information on references can be found in Clause 2.

⁹ [The use of the word *must* is deprecated and cannot be used when stating mandatory requirements; *must* is used only to describe unavoidable situations.](#)

¹⁰ [The use of *will* is deprecated and cannot be used when stating mandatory requirements; *will* is only used in statements of fact.](#)

The word *should* indicates that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required (*should* equals *is recommended that*).

The word *may* is used to indicate a course of action permissible within the limits of the standard (*may* equals *is permitted to*).

The word *can* is used for statements of possibility and capability, whether material, physical, or causal (*can* equals *is able to*).

2. Normative references

The following referenced documents are indispensable for the application of this document (i.e., they must be understood and used, so each referenced document is cited in text and its relationship to this document is explained). For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments or corrigenda) applies.

IEEE Std 1730TM ~~2010~~, IEEE Recommended Practice for Distributed Simulation Engineering and Execution Process (DSEEP).^{11, 12 2,3}

¹¹ The IEEE standards or products referred to in Clause 2 are trademarks owned by The Institute of Electrical and Electronics Engineers, Incorporated.

¹² IEEE publications are available from The Institute of Electrical and Electronics Engineers, ~~445 Hoes Lane, Piscataway, NJ 08854, USA~~ (<http://standards.ieee.org/>).

³ ~~This IEEE document is a trademark of The Institute of Electrical and Electronics Engineers, Inc.~~

3. Definitions, acronyms, and abbreviations~~, and acronyms~~

3.1 Definitions

For the purposes of this document, the following terms and definitions apply. The *IEEE Standards Dictionary Online* should be consulted for terms not defined in this clause. ^{13 4}

“conceptual model: An abstraction of what is intended to be represented within a simulation environment, which serves as a frame of reference for communicating simulation-neutral views of important entities and their key actions and interactions. The conceptual model describes what the simulation environment will represent, the assumptions limiting those representations, and other capabilities needed to satisfy the user’s requirements. Conceptual models are bridges between the real world, requirements, and simulation design.” ~~(IEEE Std 1730)~~⁵

constructive simulation: Models and simulations that involve simulated people operating simulated systems. Real people stimulate (make inputs) to such simulations but are not involved in determining the outcomes.

issue: A concern, such as a situation within a development process or a technical element of an architecture, from which obstacles to achieving the objectives of the simulation environment may arise.

live simulation: A simulation involving real people operating real systems.

^{13 4} IEEE Standards Dictionary Online ~~subscription~~ is available at: <http://dictionary.ieee.org>. An IEEE account is required for access to the dictionary, and one can be created at no charge on the dictionary sign-in page. ~~at~~http://www.ieee.org/portal/innovate/products/standard/standards_dictionary.html.

⁵ ~~Throughout the document, text quoted directly from the DSEEP (IEEE Std 1730-2010) is both enclosed in quotation marks and highlighted with gray shading.~~

live, virtual, and constructive (LVC) simulation: A broadly used taxonomy describing a mixture of live simulation, virtual simulation, and constructive simulation. Note that live, virtual, and constructive simulations always include a real or synthetic person in the simulation as contrasted with a science-based simulation, which models a phenomenon or process only.

NOTE—See (Department of Defense [B26],^{14, 15})⁶

member application: An application that is serving some defined role within a simulation environment. This can include live, virtual, or constructive (LVC) simulation assets or can be supporting utility programs such as data loggers or visualization tools.”

“**objective:** The desired goals and results of the activity to be conducted in the distributed simulation environment expressed in terms relevant to the organization(s) involved.”

“**requirement:** A statement identifying an unambiguous and testable characteristic, constraint, process, or product of an intended simulation environment.”

“**simulation data exchange model (SDEM):** A specification defining the information exchanged at runtime to achieve a given set of simulation objectives. This includes class relationships, data structures, parameters, and other relevant information. (IEEE Std 1730) ”

“**simulation environment:** A named set of member applications along with a common simulation data exchange model (~~SDEM~~) and set of agreements that are used as a whole to achieve some specific objective. Also referred to as a “(~~IEEE Std 1730~~) Syn: distributed simulation environment. ~~distributed simulation environment.~~

¹⁴ ⁶The numbers in brackets correspond to those of the numbers in the bibliography in Annex A.

¹⁵ Notes in text, tables, and figures of a standard are given for information only and do not contain requirements needed to implement this standard.

stimulation: [Stimulation is](#) the use of simulations to provide an external stimulus to a system or subsystem. An example is the use of a simulation representing the radar return from a target to drive (stimulate) the radar of a missile system within a hardware/software-in-the-loop simulation.

[NOTE—See](#) (Department of Defense [B27].)

virtual simulation: A simulation involving real people operating simulated systems. Virtual simulations inject human-in-the-loop ([HITL](#)) ~~factors~~ in a central role by exercising motor control skills (e.g., flying an airplane), decision skills (e.g., committing fire control resources to action), or communication skills [e.g., as members of a command, control, communications, computers, and intelligence (C4I) team].

3.2 Acronyms and abbreviations

~~API~~	~~application programming interface~~
C2	command and control
C4I	command, control, communications, computers, and intelligence
C4ISR	command, control, communications, computers, intelligence, surveillance, and reconnaissance
C-BML	Coalition Battle Management Language
CDS	cross-domain solution
COTS	commercial off-the-shelf
CTIA	Common Training Instrumentation Architecture
CTPS	Combat Trauma Patient Simulation
DCAT	Data Collection and Analysis Tool
DDM	data distribution management

DIF	Data Interchange Format
DIS	Distributed Interactive Simulation
DNS	Domain Name System
DoD	Department of Defense
DMAO	DSEEP Multi-Architecture Overlay
DSEEP	Distributed Simulation Engineering and Execution Process
DSSRT	Distributed Simulation Synchronization Replay Tool
ESTEEM	Embedded Simulation Test Evaluation Monitor
FDK	Federated-Simulation Development Kit
FOM	federation object model
GOTS	government off-the-shelf
GWB	gateway builder
HITL	human-in-the-loop¹⁶
HLA	High Level Architecture
HWIL	hardware-in-the-loop
IME	Integrated Middleware Environment
IP	internet protocol
JBD2	Joint Battlespace Dynamic Deconfliction
JCA	Joint Capability Area

¹⁶ [The acronym HITL is also used in some sources for “hardware-in-the-loop.”](#)

<u>JDEP</u>	<u>Joint Distributed Engineering Plant</u>
<u>JLVC</u>	<u>Joint Live Virtual and Constructive</u>
JMETC	Joint Mission Environment Test Capability
<u>INTC</u>	<u>Joint National Training Capability</u>
<u>ISAF</u>	<u>Joint Semi-Automated Forces</u>
<u>ISIMS</u>	<u>Joint Simulation System</u>
<u>ISSEO</u>	<u>Joint Single Integrated Air Picture System Engineering Office</u>
<u>IWARS</u>	<u>Joint Warfare System</u>
<u>LAN</u>	<u>local area network</u>
LROM	logical range object model
<u>LVC</u>	<u>live-virtual-constructive</u>
<u>LVCAF</u>	<u>LVC Architecture Framework</u>
<u>LVCAR</u>	<u>LVC Architecture Roadmap</u>
M&S	modeling and simulation
<u>ModSAF</u>	<u>Modular Semi-Automated Forces</u>
<u>MSIS</u>	<u>Modeling and Simulation Information System</u>
<u>MSRR</u>	<u>Model and Simulation Resource Repositories</u>
<u>MTMD</u>	<u>Maritime Theater Missile Defence</u>
<u>NATO</u>	<u>North Atlantic Treaty Organization</u>
NCOIC	Network-Centric Operations Industry Consortium

NTP	Network Time Protocol
OMT	Object Model Template
OpenMSA	Open Modeling and Simulation Architecture
PDU	Protocol Data Unit
RPR FOM	Real-time Platform-level Reference Federation Object Model
RTI	run-time infrastructure
SCOPE	Systems, Capabilities, Operations, Programs, and Enterprises
SDM	Simulation Data Exchange Model
SEIT	Synthetic Environments Integrated Test Bed
SIMPLE	SIMC4I Interchange Module for Plans, Logistics, and Exercises
SISO	Simulation Interoperability Standards Organization
SME	subject matter expert
SOAR	Special Operations Aviation Regiment
STOW-A	Synthetic Theater of War-Architecture
TACLANE	Tactical Local Area Network Encryption
TADIL	Tactical Digital Information Link
TCP	Transmission Control Protocol
TENA	Test and Training Enabling Architecture
TRADOC	Training and Doctrine Command
TSPI	Time, Space, and Position Information

UAV	unmanned aerial vehicle
UDP	User Datagram Protocol
UDP/IP	UDP Packets over IP
UJTL	Universal Joint Task List
V&V	verification and validation
VMF	Variable Message Format
VV&A	Verification, Validation, and Accreditation/Acceptance
WAN	wide area network
~~XML~~	~~Extensible Markup Language~~

4. Multi-architecture issues and solutions

This clause describes a set of distributed simulation environment development issues that are either specific to, or exacerbated by, the need to develop a multi-architecture simulation environment. Each issue is aligned with the DSEEP activity for which it has the greatest impact. For each issue, a description of the issue and recommended action(s) for addressing it are provided. The DSEEP defines sets of inputs, recommended tasks, and outcomes for each activity. [This product](#) ~~The DMAO~~ augments those DSEEP sets with additional inputs, recommended tasks, and outcomes that apply to multi-architecture simulation environments.

~~For each step, a figure adapted from the DSEEP is included to illustrate the additional discussion in this document. Data flow diagram notation is used in these figures to represent activities (rounded rectangles), data stores (cylinders), and information flows and products (arrows) (Scrudder et al. [B96]). In the figures, inputs and outputs specific to multi-architecture simulation environments are indicated with italic text and dashed lines.~~

Table 1 summarizes the content of this clause. It shows, for each DSEEP step and activity, the multi-architecture issues associated with that step and activity. Note that there may be zero, one, or more than one issue for a step and activity.

Table 1 — Multi-architecture issues by DSEEP step and activity

Step	Activity	Issue(s)
1 Define simulation environment objectives	1.1 Identify user/sponsor needs	—
	1.2 Develop objectives	—
	1.3 Conduct initial planning	1.3.1 Multi-architecture initial planning
		1.3.2 Required multi-architecture simulation environment expertise
		1.3.3 Inconsistent development and execution processes
		1.3.4 VV&A for multi-architecture applications
2 Perform conceptual analysis	2.1 Develop scenario	—
	2.2 Develop conceptual model	—
	2.3 Develop simulation environment requirements	2.3.1 Requirements for multi-architecture simulation environment
		2.3.2 Member application requirement incompatibility

(New)

Step	Activity	Issue(s)
1 Define simulation environment objectives	1.1 Identify user/sponsor needs	
	1.2 Develop objectives	
	1.3 Conduct initial planning	1.3.1 Multi-architecture initial planning
		1.3.2 Required multi-architecture simulation environment expertise
		1.3.3 Inconsistent development and execution processes
		1.3.4 VV&A for multi-architecture applications
2 Perform conceptual analysis	2.1 Develop scenario	
	2.2 Develop conceptual model	
	2.3 Develop simulation environment requirements	2.3.1 Requirements for multi-architecture simulation environment
		2.3.2 Member application requirement incompatibility

(Deleted)

Step	Activity	Issue(s)
3 Design simulation environment	3.1 Select member applications	3.1.1 Member application selection criteria for multi-architecture Simulation Environments
		3.1.2 Non-conforming member applications
	3.2 Design simulation environment	3.2.1 Gateway usage and selection decisions
		3.2.2 Object state update contents
		3.2.3 Object ownership management
		3.2.4 Time management in multi-architecture simulation environments
		3.2.5 Interest management capability differences
		3.2.6 Gateway translation paths
		3.2.7 DIS heartbeat translation
		3.2.8 Multi-architecture and inter-architecture performance
		3.2.9 Translating non-ground-truth network data
		3.2.10 Object identifier uniqueness and compatibility
		3.2.11 Cross domain solutions in multi-architecture simulation environments
		3.2.12 Multi-architecture save and restore
	3.3 Design member applications	3.3.1 New member application architecture
	3.4 Prepare detailed plan	3.4.1 Cost and schedule estimating for multi-architecture development
4 Develop simulation environment	4.1 Develop simulation data exchange model	4.1.1 Metamodel incompatibilities
		4.1.2 SDEM content incompatibilities
	4.2 Establish simulation environment agreements	4.2.1 Agreements to address multi-architecture development
		4.2.2 Tool availability and compatibility
		4.2.3 Initialization sequencing and synchronization
	4.3 Implement member application designs	4.3.1 Non-standard algorithms
5 Integrate and test simulation environment	5.1 Plan execution	5.1.1 Integration and test planning for multi-architecture simulation environments
		5.1.2 Multi-architecture execution planning considerations
	5.2 Integrate simulation environment	5.2.1 Live entity time, space, and position information updates
	5.3 Test simulation environment	5.3.1 Complexities of testing in a multi-architecture simulation environment
6 Execute simulation	6.1 Execute simulation	6.1.1 Monitoring and Controlling multi-architecture simulation environment execution
		6.1.2 Multi-architecture data collection
	6.2 Prepare simulation environment outputs	—
7 Analyze data and evaluate results	7.1 Analyze data	—
	7.2 Evaluate and feedback results	7.2.1 Multi-architecture simulation environment assessment

(New)

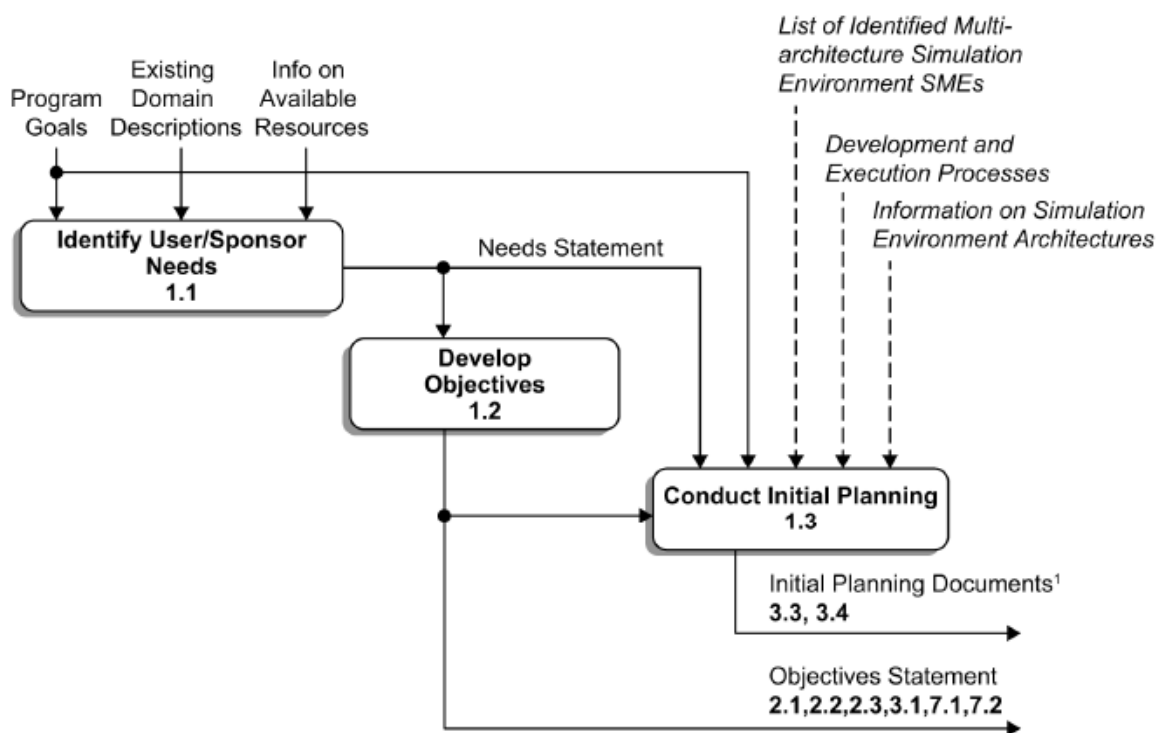
Step	Activity	Issue(s)
3 Design simulation environment	3.1 Select member applications	3.1.1 Member application selection criteria for multi-architecture simulation environments
		3.1.2 Non-conforming member applications
	3.2 Design simulation environment	3.2.1 Gateway usage and selection decisions
		3.2.2 Object state update contents
		3.2.3 Object ownership management
		3.2.4 Time management in multi-architecture simulation environments
		3.2.5 Interest management capability differences
		3.2.6 Gateway Translation Paths
		3.2.7 DIS heartbeat translation
		3.2.8 Multi-architecture and inter-architecture performance
		3.2.9 Translating non-ground-truth network data
		3.2.10 Object identifier uniqueness and compatibility
		3.2.11 Cross-domain solutions (CDSs) in multi-architecture simulation environments
		3.2.12 Multi-architecture save and restore
4 Develop simulation environment	3.3 Design member applications	3.3.1 New member application architecture
	3.4 Prepare detailed plan	3.4.1 Cost and schedule estimating for multi-architecture development
		4.1.1 Metamodel incompatibilities
	4.1 Develop simulation data exchange model	4.1.2 SDEM content incompatibilities
		4.2.1 Agreements to address multi-architecture development
		4.2.2 Tool availability and compatibility
	4.2 Establish simulation environment agreements	4.2.3 Initialization sequencing and synchronization
		4.3.1 Nonstandard algorithms
5 Integrate and test simulation environment	4.3 Implement member application designs	4.4.1 Network configuration
		5.1.1 Integration and test planning for multi-architecture simulation environments
	5.1 Plan execution	5.1.2 Multi-architecture execution planning considerations
		5.2.1 Live entity time, space, and position information (TSPI) updates
6 Execute simulation	5.2 Integrate simulation environment	5.3.1 Complexities of testing in a multi-architecture simulation environment
	5.3 Test simulation environment	6.1.1 Monitoring and controlling multi-architecture simulation environment execution
		6.1.2 Multi-architecture data collection
7 Analyze data and evaluate results	6.1 Execute simulation	6.2 Prepare simulation environment outputs
	7.1 Analyze data	7.2.1 Multi-architecture simulation environment assessment
	7.2 Evaluate and feedback results	

(Deleted)

4.1 Step 1: Define simulation environment objectives

“The purpose of Step 1 of the DSEEP is to define and document a set of needs that are to be addressed through the development and execution of a simulation environment and to transform these needs into a more detailed list of specific objectives for that [simulation](#) environment.” (IEEE Std 1730)

Figure 4-1 illustrates the activities, inputs, and outputs within Step 1. [Data flow diagram notation is used in Figure 4 and throughout this document to represent activities \(rounded rectangles\), data stores \(cylinders\), and information flows and products \(arrows\)](#) (Scrudder, et al. [B94]). In the figure, inputs and outputs specific to multi-architecture simulation environments are indicated with italic text and dashed lines.



¹See 4.1.3.3-2 for additional items included in Initial Planning Documents

Figure 4 — Step 1

4.1.1 Activity 1.1: Identify user/sponsor needs

4.1.1.1 Purpose

“The primary purpose of this activity is to develop a clear understanding of the problem to be addressed by the simulation environment. The needs statement may vary widely in terms of scope and degree of formalization. It should include, at a minimum, high-level descriptions of critical systems of interest, initial estimates of required fidelity and required behaviors for simulated entities, key events and environmental conditions that must be represented in the scenario, and output data requirements. In addition, the needs statement should indicate the resources that will be available to support the simulation environment (e.g., funding, personnel, tools, facilities) and any known constraints that may affect how the simulation environment is developed (e.g., required member applications, due dates, site requirements, and security requirements). In general, the needs statement should include as much detail and specific information as is possible at this early stage of the DSEEP.” (IEEE Std 1730)

4.1.1.2 Issues

No multi-architecture issues have been identified for this activity.

4.1.2 Activity 1.2: Develop objectives

4.1.2.1 Purpose

“The purpose of this activity is to refine the needs statement into a more detailed set of specific objectives for the simulation environment. The objectives statement is intended as a foundation for generating explicit simulation requirements (i.e., translating high-level user/sponsor expectations into more concrete, measurable goals). This activity requires close collaboration between the user/sponsor of the simulation environment and the development/integration team to verify that the original needs statement is properly analyzed and interpreted correctly, and that the resulting objectives are consistent with the stated needs. Early assessments of feasibility and risk should also be performed as part of this activity. In particular, certain objectives may not be achievable given practical constraints (such as cost, schedule, and availability of personnel or facilities) or even limitations on the state-of-the-art of needed technology. Early identification of such issues and consideration of these limitations and constraints in the objectives statement will set appropriate expectations for the development and execution effort.” (IEEE Std 1730)

4.1.2.2 Issues

No multi-architecture issues have been identified for this activity.

4.1.3 Activity 1.3: Conduct initial planning

4.1.3.1 Purpose

“The purpose of this activity is to establish a preliminary simulation environment development and execution plan. The intent is to translate the objectives statement, along with the associated risk and feasibility assessments, into an initial plan with sufficient detail to effectively guide early design activities. The plan may effectively include multiple plans, and should cover such considerations as verification and validation (V&V), configuration management, and security. The plan should also address supporting tools for early DSEEP activities, based on factors such as availability, cost, applicability to the given application, ability to exchange data with other tools, and the personal preferences of the development/integration team. The plan should also define a high-level schedule of key development and execution events, and provide additional scheduling detail for all pre-development (i.e., prior to Step 4) activities. Note that the initial plan will be updated and extended as appropriate in subsequent development phases as additional information is accumulated throughout the evolution of the distributed simulation design (see Activity 3.3).” (IEEE Std 1730)

4.1.3.2 Issues

4.1.3.2.1 Multi-architecture initial planning

This issue consists of the following:

- a) **Description.** During initial planning, work breakdown structures are typically developed to define the required project tasks, outline the overall project schedule, and provide a framework for estimating resource requirements for each stage of the project. However, the identity of several participating member applications may be unknown this early in the process, and thus, the requirement for a multi-architecture simulation environment design may be unknown. In the absence of better information, project managers frequently just assume single-architecture operation, ~~which and this assumption~~ underestimates the time and resources necessary to establish the simulation environment. ~~This and~~ increases project risk from several perspectives.

- b) **Recommended action(s).** The scope of the multi-architecture simulation environment effort should be established. The questions of what needs to be done and who needs to participate should be identified early in the development process. Although such considerations can be added during later development phases, omissions made during planning may increase the technical and schedule risk of the simulation development.

In general, planners should use their best judgment as to ~~about~~ what will be needed, based on the information available to them. If the initial plan assumes that the simulation environment development will be single-architecture, the sponsor should be made aware very early of the potential ~~need~~ for significant rework of the plan and the potential need for additional resources if the assumption is later found to be false. If the initial plan assumes that the simulation environment development will be multi-architecture, the relatively high level of resources required should be communicated very early to the sponsor. In that way, certain objectives can be relaxed as appropriate if resource demands are considered excessive. Another system development approach may be to plan for two simulation environments, ÷ one implemented as a single-architecture simulation environment and a second implemented as a multi-architecture simulation environment.

Multi-architecture simulation environments often involve more complex developments and may have additional technical, financial, schedule, and programmatic issues. Single-architecture simulation environments should, therefore, be preferred unless the benefits brought by a specific multi-architecture implementation outweigh the issues. For example, a benefit may be the ability to use the most appropriate applications for a given task when these may only support different architectures. Sufficiently analyzing the benefits, feasibility, limitations, constraints, trade-offs, and risks of multi-architecture engineering issues improves successful planning of a multi-architecture simulation environment. If the initial planning documents fail to reflect the additional developmental considerations required by a multi-architecture simulation environments, then the result will be major omissions in terms of what will eventually need to be integrated into a multiarchitecture simulation environment, both with respect to actual applications (e.g., gateways) and overarching requirements in the areas of performance, execution management, networking, and required complementary development activities [e.g., security and verification, validation, and accreditation/acceptance (VV&A)].

4.1.3.2.2 Required multi-architecture simulation environment expertise

[This issue consists of the following:](#)

- a) **Description.** In the event that the user/sponsor requires the use of certain member applications, and those member applications have existing interfaces that cut across more than one architecture, lack of personnel experienced in the development of multi-architecture simulation environments on the initial development team may result in unachievable cost and/or schedule objectives, which will adversely affect the planning process.

- b) **Recommended action(s).** Expertise in engineering and executing multi-architecture federations can be located through a variety of sources. Each architecture discussed in this document has a support structure, and those capabilities can provide both lessons learned and pointers to expertise in integrating the specific architecture with other architectures. Both the DIS architecture^{17 7} and the HLA^{18 8} have product support groups with open discussion reflectors managed by the Simulation Interoperability Standards Organization (SISO). TENA maintains an active help desk.^{19 9} Commercial and government organizations that provide gateways that address multi-architecture interoperability can be a source for expertise. Likewise, many developers of ~~the~~ architecture middleware, such as the HLA Runtime Infrastructure (RTI) and the TENA middleware, have extensive experience in multi-architecture integration. Organizations, such as the Network Centric Operations Industry Consortium (NCOIC), have experience in the integration of net-centric systems that may be useful in the integration of live components. Finally, the authors of the references cited in Annex A and the publication archives for the organizations that published those references are a source of expertise.

After identifying sources of expertise, there are two paths to consider — ÷ adding the appropriate experienced personnel to the team permanently or adding them temporarily. Both approaches are valid, and the specific situation should dictate the action taken.

Temporarily adding multi-architecture simulation environment expertise is typically done by using a consultant or team of consultants. The term consultant, as used in this [standard document](#), refers to a person temporarily assigned to the team to provide professional or technical advice to help ensure successful execution of the required activity. This added expertise can come from inside or from outside the current company or program. The addition of permanent subject matter experts (SMEs) in multi-architecture simulation environments can have a substantial long-term positive impact on the ability of a team to execute multi-architecture simulation events.

¹⁷ See ⁷<http://www.sisostds.org/StandardsActivities/SupportGroups/DISPSGDistributedInteractiveSimulation.aspx>.

¹⁸ ⁸ See <http://www.sisostds.org/StandardsActivities/SupportGroups/HLAEvolvedPSGHighLevelArchitectureEvolved.aspx>.

¹⁹ See ⁹<https://www.tena-sda.org/>. ~~display/intro/Home.~~

Both of the ~~previous~~ ~~above~~ approaches are valid even when multi-architecture expertise exists on a team, but specific architecture expertise is missing. For example, experience exists in HLA-to/from-DIS multi-architecture simulation environments, ~~but~~ ~~However~~, the requirement is for HLA to/from TENA, and no TENA expertise exists on the team. In this case, the addition of expertise is constrained to the unfamiliar architecture.

4.1.3.2.3 Inconsistent development and execution processes

This issue consists of the following:

- a) **Description.** Well-defined and understood processes and procedures for building simulation environments exist within a given architecture community. What does not exist is consistency of those processes and procedures across architecture communities in a multi-architecture simulation environment. In addition, communication between these distinct architecture communities can be a significant problem because of differences in terminology and can lead to misunderstandings that can require significant rework at some point later in the process.
- b) **Recommended action(s).** The key to allowing teams from different architecture communities to work together successfully in a multi-architecture simulation environment is to provide a mechanism for understanding unique processes and procedures and to correlate the unique terminology.

Experts from each of the architecture communities involved should coordinate and correlate their individual processes and procedures so that expectations can be managed and schedules can be coordinated. For instance, one community may take extra time working on the details of the ~~simulation data exchange model (SDEM)~~ (e.g., HLA and TENA) while others may be more concerned about the details of how the data flows on the network (e.g., DIS). This area is ultimately less of a technical issue and more of a management issue.

A documented common set of terminology that correlates terms across the architectures used in the multi-architecture simulation environment will allow the individual development teams to clearly understand the other's intent when working together to design and implement the requisite architecture. The DSEEP provides annexes for DIS, HLA, and TENA that can be used as a starting point for relating terminology across architectures. This correlation of terminology is critical as it [allows](#) ~~gives~~ teams with their own "language" the ability to communicate effectively throughout the development process.

Another way to help bridge the knowledge gap across teams' expertise is to have focused training sessions where the participating teams can learn about the other architectures and build a rapport. While ideally informal in format, each architecture team can have the opportunity to present the basics of what their architecture is and how it works. For example, the TENA community has created a TENA Overview Course and a more in-depth TENA Technical Introduction Course [that](#) ~~,which~~ can be presented and discussed. While these are likely sufficient to start a fundamental understanding of the TENA architecture, a more in-depth programming class, the TENA Hands-On Training, is also available when necessary.

4.1.3.2.4 VV&A for multi-architecture applications

[This issue consists of the following:](#)

- a) **Description.** [Verification, validation, and accreditation/acceptance](#) (VV&A) are required activities in almost all simulation environment development projects. Verification is necessary to confirm that the implemented simulation environment is consistent with its design specification. Validation is necessary to determine [if](#) ~~whether~~ the models in the simulation environment provide the level of accuracy and realism needed for the simulation environment's intended uses; [the latter](#) ~~,and validation~~ often involves SMEs in the domain of interest. In many cases, accreditation/acceptance is also required prior to production use of the simulation environment. VV&A activities should occur concurrently with implementation activities to the extent possible.

A multi-architecture simulation environment adds to the challenge of VV&A, for several reasons: (1) The time and resources needed to support VV&A in a multi-architecture simulation environment may be more than program resources allow, thereby increasing the risk of accepting and using a simulation environment that does not meet its design specification or produces invalid results. (2) Failure to plan for the VV&A activities required by a multi-architecture simulation environment can result in costly problems later in the multi-architecture engineering and execution process. (3) Multi-architecture simulation environment developers and users are likely to need additional information and expert assistance to perform verification and validation V&V (especially the former verification) because of their lack of familiarity with the implementation details of other architectures. (4) Tools that can monitor event execution and support V&V testing across a multi-architecture simulation environment may not be available. (5) The performance of the multi-architecture simulation environment should be tested to verify that information exchange among the participating members takes place as designed. (6) Multi-architecture simulation environment infrastructure functions need to be examined to identify and test architecture elements (such as gateways) that could have adverse impacts upon the validity of the simulation environment.

- b) **Recommended action(s).** VV&A in general is a broad topic described in a body of research literature whose large size is consistent with its importance in the practice of ~~modeling and simulation~~-(M&S) (for example surveys, see Balci [B3], Petty [B79 B81], or Youngblood [B117 B119]). VV&A of single-architecture distributed simulation systems is described in two IEEE standards (IEEE Std 1278.4 [B42] TM for DIS [B43] and IEEE Std 1516.4 [B45] TM for HLA-[B48]). Verification in a multi-architecture simulation environment should include the testing of all design decisions and resulting implementation aspects that support multi-architecture operation, and validation should consider the possible impact on validity of architectural elements specific to the multi-architecture simulation environment. Some important VV&A considerations specific to multi-architecture simulation environments, include the following:

- 1) *Multi-architecture-specific components.* The simulation environment may include components specific to its multi-architecture design, such as gateways and middleware. These “additional” components should be verified, to confirm that they are operating as specified and intended (e.g., verification that a gateway is translating all of the message types it is expected to ~~encounter~~), and validated, to confirm that they are not degrading the simulation’s validity (e.g., validation of real-time response in a training simulation including a gateway that imposes latencies) (Harkrider and Petty [B37]).
- 2) *Distributed simulation support operations.* Executing a distributed simulation typically requires a set of specialized architecture services that provide functionality that is part of the simulation’s infrastructure rather than part of its modeling. Examples include object naming, distributed logging, enumeration control, and pause and resume. In a multi-architecture simulation environment, these operations are more complex because of the need to coordinate and reconcile them across multiple architectures. Ensuring that these services operate correctly in a multi-architecture simulation environment will require additional specialized verification (Williams and Smith [B111 B113]).
- 3) *Correlation.* Because of the different design heritage and development histories of its separate architectures, a multi-architecture simulation environment is arguably more likely than a single-architecture simulation environment to have multiple representations of the same object, e.g., multiple terrain databases in different formats or multiple dynamics models of the same entity. Correlation (agreement) between multiple representations of the same object is a VV&A consideration in any simulation environment, but it is exacerbated in the multiarchitecture context. Additional V&V testing focused on correlation is likely to be required, e.g., terrain correlation in the Platform Proto-Federation (Petty, et al. [B81 B83]) and the Urban Resolve 2015 experiment (Williams and Smith [B111 B113]).

- 4) *Validation of architecture elements.* Some architectures include standard models as part of the architecture itself (e.g., dead reckoning in the DIS specification or coordinate conversion in the TENA middleware). When member applications that use such architectures are linked into a multi-architecture simulation environment, it may be necessary to validate those architecture-embedded models in the context of their interactions with other parts of the overall simulation environment. For example, validation issues resulting from mismatches among dead reckoning algorithms used by member applications and gateways are elaborated in Valle_z et al. [B102 B104], where three alternative resolutions are mentioned.
- 5) *Multiple accreditors.* Because the simulation environment is multi-architecture, there may be multiple accreditors (~~i.e.,~~ organizations responsible for approving the use of the simulation environment for the intended application). Additional V&V testing_z and additional documentation of those tests_z may be needed to meet the different testing and documentation requirements of the multiple accreditors. Simulation designers should allow for this effort during planning.
- 6) *Multiple architecture communities.* The communities associated with the different architectures may have different expectations for V&V testing and for documentation of those tests, thus adding ~~and these differences may add~~ to the VV&A effort for a multi-architecture simulation environment. Simulation designers should allow for this effort during planning.

~~The process of~~ Planning and ~~execution of executing~~ VV&A is a continuous ~~process~~ throughout the development of the multiarchitecture simulation environment. By execution time, conceptual model validation should be complete, and the test plans ~~that verify~~ ~~verifying~~ that the design solutions to the multi-architecture concerns (as defined by the Simulation Environment Agreements and other design documentation) are effective should have been properly executed. During execution, appropriate tools should be employed to verify that the operation of the multi-architecture simulation environment is consistent with previous test results, that performance is within acceptable bounds, and that any anomalies that occur do not adversely affect the validity of simulation results. If incompatibilities among the execution management tools from the different architectures provide an uneven picture of the state of the execution, it can be necessary to monitor the event execution across the architectures and compare the observations in a timely manner to verify that needed corrective actions are taken. Finally, in the post-execution phase, results validation activities should be conducted to determine whether the multi-architecture design produced the desired outputs.

4.1.3.3 Multi-architecture-specific inputs, tasks, and outcomes for Activity 1.3 ~~(conduct initial planning)~~

The inputs, tasks, and outcomes are as follows:

- a) Multi-architecture-specific activity inputs
 - 1) List of identified multi-architecture simulation environment SMEs (Issue 4.1.3. ~~2~~ ~~1.2~~)
 - 2) Development and execution processes for the selected architectures (Issue 4.1.3. ~~2~~ ~~1.3~~)
 - 3) Information on simulation environment architectures such as TENA, DIS, or HLA (Issue 4.1.3. ~~2~~ ~~1.3~~)
- b) Multi-architecture-specific tasks
 - 1) Plan for single-and multi-architecture simulation environments alternatives. (Issue 4.1.3. ~~2~~ ~~1.1~~)

- 2) Consult with SMEs in multi-architecture development^{20 40} (Issue 4.1.3. 2 ~~1.2~~)
 - 3) Select approach for adding personnel with multi-architecture experience—either through temporary or permanent staff augmentation^{20 40} (Issue 4.1.3. 2 ~~1.2~~)
 - 4) Document agreed-upon common terminology across architectures within the simulation environment^{20 40} (Issue 4.1.3. 2 ~~1.3~~)
 - 5) Extend the ~~following~~ existing DSEEP Task to include multi-architecture V&V concerns^{20 40}:
 - i) “Revise V&V plan and test plan (based on simulation environment test criteria), including multi-architecture concerns—V&V of gateways, implementation of support functions, multiple representations, and representations included as part of the architectures, as well as issues with multiple accreditors.” ^{20 40} (Issue 4.1.3. 2 ~~1.4~~)
- c) Multi-architecture-specific activity outcomes
- 1) Initial planning documents
 - i) Contingency plans for single-or multi-architecture simulation environments (Issue 4.1.3. 2 ~~1.1~~)
 - ii) Staffing plan to account for multi-architecture concerns (Issue 4.1.3. 2 ~~1.2~~)
 - iii) Common terminology (Issue 4.1.3. 2 ~~1.3~~)

^{20 40} Quoted text is from the DSEEP (IEEE Std 1730) §4.3.4.2; underlined material added to reflect multi-architecture extensions.

4.2 Step 2: Perform conceptual analysis

“The purpose of this step of the DSEEP is to develop an appropriate representation of the real-world domain that applies to the defined problem space and to develop the appropriate scenario. It is also in this step that the objectives for the simulation environment are transformed into a set of highly specific requirements that will be used during design, development, testing, execution, and evaluation.” (IEEE Std 1730)

Figure 5 2 illustrates the activities, inputs, and outputs within Step 2. ~~In the figure, inputs and outputs specific to multi-architecture simulation environments are indicated with italic text and dashed lines.~~

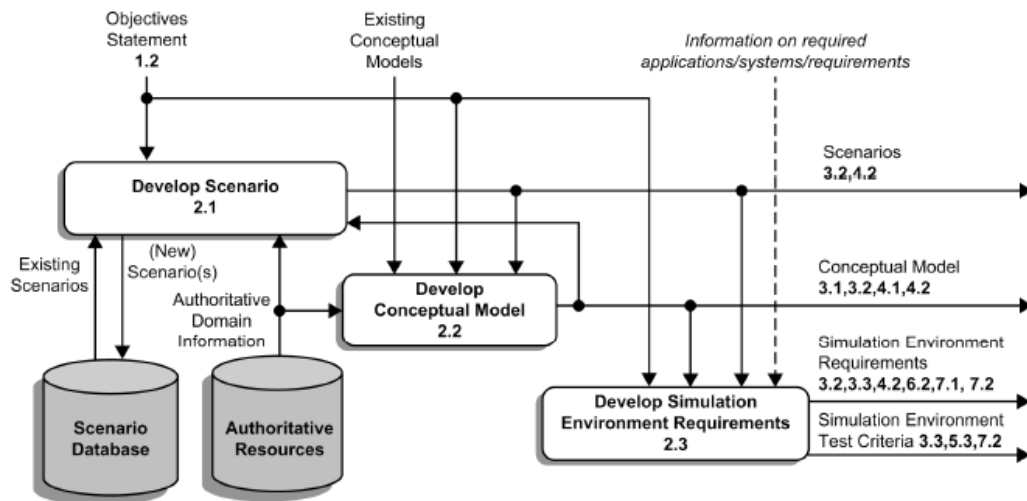


Figure 5 — Step 2

4.2.1 Activity 2.1: Develop scenario

4.2.1.1 Purpose

“The purpose of this activity is to develop a functional specification of the scenario. Depending on the needs of the simulation environment, the scenario may actually include multiple scenarios, each consisting of one or more temporally ordered sets of events and behaviors (i.e., vignettes). The primary input to this activity is the domain constraints specified in the objectives statement (Step 1), although existing scenario databases may also provide a reusable starting point for scenario development. Any additional input is provided by the conceptual model, which may be developed in parallel with the scenario. Where appropriate, authoritative sources for descriptions of major entities and their capabilities, behavior, and relationships should be identified prior to scenario construction. A scenario includes the types and numbers of major entities that must be represented within the simulation environment, a functional description of the capabilities, behavior, and relationships between these major entities over time, and a specification of relevant environmental conditions (such as urban terrain versus natural area, type of terrain, day/night, climate, etc.) that impact or are impacted by entities in the simulation environment. Initial conditions (e.g., geographical positions for physical objects), termination conditions, and specific geographic regions should also be provided. The product of this activity is a scenario or set of scenarios, which provides a bounding mechanism for conceptual modeling activities.

The presentation style used during scenario construction is at the discretion of the simulation environment development/integration team. Textual scenario descriptions, event-trace diagrams, and graphical illustrations of geographical positions for physical objects and communication paths all represent effective means of conveying scenario information. Software tools that support scenario development can generally be configured to produce these presentation forms. Reuse of existing scenario databases may also facilitate the scenario development activity.” (IEEE Std 1730)

4.2.1.2 Issues

No multi-architecture issues have been identified for this activity.

4.2.2 Activity 2.2: Develop conceptual model

4.2.2.1 Purpose

“During this activity, the development/integration team produces a conceptual representation of the intended problem space based on their interpretation of user needs and sponsor objectives. The product resulting from this activity is known as a conceptual model. The conceptual model provides an implementation-independent representation that serves as a vehicle for transforming objectives into functional and behavioral descriptions for system and software designers. The model also provides a crucial traceability link between the stated objectives and the eventual design implementation. This model can be used as the structural basis for many design and development activities (including scenario development) and can highlight correctable problems early in the development of the simulation environment when properly validated by the user/sponsor.

The early focus of conceptual model development is to identify relevant entities within the domain of interest, to identify static and dynamic relationships between entities, and to identify the behavioral and transformational (algorithmic) aspects of each entity. Static relationships can be expressed as ordinary associations, or as more specific types of associations such as generalizations (“is-a” relationships) or aggregations (“part-whole” relationships). Dynamic relationships should include (if appropriate) the specification of temporally ordered sequences of entity interactions with associated trigger conditions. Entity characteristics (attributes) and interaction descriptors (parameters) may also be identified to the extent possible at this early stage of the process. While a conceptual model may be documented using differing notations, it is important that the conceptual model provides insight into the real-world domain and includes an explanatory listing of the assumptions and limitations to properly bound the model.

The conceptual model needs to be carefully evaluated before the next step (Design Simulation Environment) is begun, including a review of key processes and events by the user/sponsor to confirm the adequacy of the domain representation. Revisions to the original objectives and conceptual model may be defined and implemented as a result of this feedback. As the conceptual model evolves, it is transformed from a general representation of the real-world domain to a more specific articulation of the capabilities of the simulation environment as constrained by the member applications of the simulation environment and available resources. The conceptual model will serve as a basis for many later development activities such as member application selection and simulation environment design, implementation, test, evaluation, and validation.” (IEEE Std 1730)

4.2.2.2 Issues

No multi-architecture issues have been identified for this activity.

4.2.3 Activity 2.3: Develop simulation environment requirements

4.2.3.1 Purpose

“As the conceptual model is developed, it will lead to the definition of a set of detailed requirements for the simulation environment. These requirements, based on the original objectives statement (Step 1), should be directly testable and should provide the implementation level guidance needed to design and develop the simulation environment. The requirements should consider the specific execution management needs of all users, such as execution control and monitoring mechanisms, data logging, [and so on](#) etc. Such needs may also impact the scenario developed in Activity 2.1. The simulation environment requirements should also explicitly address the issue of fidelity, so that fidelity requirements can be considered during selection of simulation environment member applications. In addition, any programmatic or technical constraints on the simulation environment should be refined and described to the degree of detail necessary to guide implementation activities.” (IEEE Std 1730)

4.2.3.2 Issues

4.2.3.2.1 Requirements for multi-architecture simulation environment

[This issue consists of the following:](#)

- a) **Description.** The initial simulation environment requirements can be derived from several sources, including the customer use cases, Joint Capability Areas (JCAs), Mission Threads, Universal Joint Task List (UJTL), and other operationally representative sources. During this requirement definition phase, the simulation environment design has typically not been completely determined and, therefore, potential multi-architecture design, development, integration, test, and execution requirements may be unknown. The selection of some specific member applications may, however, be directed by the sponsor and would require a multi-architecture simulation environment as a result.
- b) **Recommended action(s).** In general, it is unreasonable to expect that requirements for a multiarchitecture simulation development can be defined completely and unambiguously in a single, upfront requirements development activity. “Predictive” or “plan-driven” methods that assume this ability have an unfortunate history of failure when developing new software systems. Rather, “agile” or “adaptive” techniques are typically better suited for situations where requirements are uncertain early in development. As requirements for integration of multiple simulation architectures will generally be unknown until design time, multi-architecture developments generally have a high degree of requirements uncertainty early in the development process, and, therefore, are well-suited to agile methods that support incremental build strategies. However, since predictive methods tend to provide stronger project control mechanisms (needed for large projects), hybrid or balanced methods may work best for some multi-architecture developments (Boehm and Turner [B9]).

Developers should not assume that the requirements for a multi-architecture simulation environment can be fully described in the initial development project phase. In the initial phase, the focus should be on the functional requirements needed to drive conceptual modeling activities and member application selection. In subsequent project phases, as the roles and responsibilities of participating member applications become better defined, the need to bridge across different simulation architectures will become much clearer, and explicit requirements for interface modifications and bridging mechanisms can be integrated into the evolving set of system requirements. If these [requirements change](#) ~~requirement changes~~ impact the scope of the development effort to the degree that the availability of sufficient funding resources may be a concern, the sponsor should be consulted before implementing changes to program schedules and other supporting planning documents (e.g., VV&A, test).

4.2.3.2.2 Member application requirement incompatibility

[This issue consists of the following:](#)

- a) **Description.** When the set of stated requirements mandates the use of member applications that span different architectures, the potential for incompatible requirements exists. By virtue of their fundamental design intent and implementation assumptions, different distributed simulation architectures are generally better suited for satisfying certain application requirements than they are for others. Member applications developed for different architectures often conform to and exhibit the design intent and assumptions of those architectures. Thus, incompatibilities in requirements may be introduced into the simulation environments as a result of inherent architectural differences between member applications from different architectures.

Technical incompatibilities introduced by the incompatibilities in requirements can manifest themselves in many ways. For example, by virtue of DIS's exploitation of specific network services and its protocol-embedded SDEM, member applications developed for DIS are typically well suited for requirements related to virtual entity-level real-time training applications. However, a requirement for repeatability is potentially problematic for a DIS member application because of the architecture's unconstrained time, best effort delivery of User Datagram Protocol (UDP) Packets over Internet Protocol (IP) (UDP/IP), and typical model sensitivity to slight differences in Protocol Data Unit (PDU) arrival time. For another example, TENA focuses on disparate live and virtual range member applications. Consequently, member applications designed for TENA typically have difficulty supporting a non-real-time unit-level constructive simulation. Therefore, when member applications developed for different architectures are linked into a single multiarchitecture simulation environment, some of the requirements for the multi-architecture simulation environment may be incompatible with the requirements that any a particular member application can readily support.

- b) **Recommended action(s).** The goal is to understand the differences and to start addressing the technical incompatibilities at this early stage of the process and, where necessary, make decisions on how to resolve requirements incompatibilities.

The first step in addressing the incompatibilities among potential member applications is to clearly document where they exist. Many "wedge issues," which characterize the basic differences among distributed simulation architectures (Richbourg and Lutz [B88 B90]), are manifested in incompatibilities between potential member applications, including the following:

- 1) Differences in coordinate reference frames and the ability to support multiple reference frames
- 2) Differences in time management, e.g., ~~the ability of~~ member applications' ability ~~applications~~ to support event ordering, rate of time advancement, and flexibility of rate of time advancement