

Auditory Hazard Assessment Algorithm for Humans (AHAAH) Configuration Management Plan Version 1.0.0

1 Introduction

AHAAH is the Auditory Hazard Assessment Algorithm for Humans model that permits users to predict the hazard to the human auditory system from any impulsive pressure event. This mathematical model of the ear created to provide risk evaluation for impulsive noise is implemented in software. AHAAH is used to compute auditory risk units (ARUs) which quantifies hazards for warned and unwarned exposures for three levels of hearing protection. These three levels are exposures with no hearing protection devices, exposures with a single hearing protection device (earplugs or muffs), and exposures with double hearing protection devices (earplugs and muffs). AHAAH, as implemented for use with MIL-STD-1474E, includes eleven default hearing protection configurations. The U.S. Army Research Laboratory (ARL), Human Research and Engineering Directorate (HRED) and its predecessor organization, the U.S. Army Human Engineering Laboratory (HEL) developed AHAAH over the course of 40 years. AHAAH is currently maintained, developed, and managed by HRED and by the Ballistics Vulnerability and Lethality Division (BVLVD) of the Survivability/Lethality Analysis Directorate (SLAD) of ARL.

AHAAH is fully implemented in the Delphi programming language as a stand-alone code. Less complete implementations are available in the C++ and Java programming languages. It is anticipated that a full implementation will be made available for the C++ language both as a stand-alone code and as an embeddable library. Other programs that need to assess auditory hazards may call the embedded AHAAH library. AHAAH is currently embedded in the U.S. Army Operational Requirement-based Casualty Assessment (ORCA) model used for assessment of personnel.

This AHAAH configuration management plan (CMP) has been developed to control and manage AHAAH configuration items (CIs). An AHAAH CI is any code, data, document, image, script, or other product over which the AHAAH Configuration Control Board (CCB) requires configuration control.

The oversight authority for changes to AHAAH is the AHAAH Configuration Control Board (CCB). The process to manage the configuration control, maintenance, and development of the code is specified in this CMP. All changes to AHAAH are under the control of the AHAAH CCB including this CMP. The CMP is a living document, subject to changes directed by the AHAAH CCB, and accepted and approved by the ARL

management team. This CMP establishes the configuration management (CM) processes developed to manage the configuration control (CC), maintenance, and development of AHAAH CIs. The primary focus of this CMP is to document the CM and CC processes as established to control AHAAH releases after the baseline version 2.1.

Configuration Management (CM) is “A discipline applying technical and administrative direction and surveillance to: identify and document the functional and physical characteristics of a configuration item, control changes to those characteristics, record and report change processing and implementation status, and verify compliance with specified requirements” [1].

The development of this CMP is based on a number of sources. Existing CMPs for other models were reviewed and sections applied and adapted where appropriate. Included in particular are the CM procedures defined by Army Regulation 5-11, “Management of Army Models and Simulations”, dated 10 July 1997 [2]. The MUVES-S2 CMP and various draft versions of the BRL-CAD and ORCA CMP have influenced and contributed heavily to the AHAAH CMP.

2 AHAAH Configuration Control

This section defines the applicable CM policies and directives, roles and functions of the organizations responsible for AHAAH CM, the CCB Charter and the inter-relationship between AHAAH and the community. The AHAAH Development Team currently consists of the ARL HRED Perceptual Sciences Branch (PSB) auditory research team and ARL SLAD Warfighter Survivability Branch (WSB) methodology team.

Changes to AHAAH are created in the following sequence:

1. Software Defect Report/Request For Enhancement (optional, depending upon level of effort)
 - Performed by: End user
2. Project evaluation
 - Performed by: Initially screening by AHAAH Development Team followed by review by CCB
3. Prioritization (for non-trivial effort)
 - Performed by: CCB Chair and the AHAAH Model Manager
4. Staff allocation (for non-trivial effort)
 - Performed by: ARL management team
5. Task completion, revision control system commit
 - Performed by: AHAAH Development Team
6. Technical peer review
 - Performed by: AHAAH Development Team
7. CCB review & possible roll-back or back-out of changes
 - Performed by: CCB, AHAAH Model Manager
 - Release approval granted/not granted by CCB

8. Release of CCB approved product
 - Performed by: AHA AH Development Team.

2.1 Configuration Control Board

The principle function of the CCB is to conduct the review and decision process of step 7 above.

2.1.1 Representation

Representation on the AHA AH CCB is comprised of the principle stakeholders in AHA AH. The present AHA AH CCB is comprised of these voting members:

- Chairperson (typically the Branch Chief of the AHA AH development and maintenance team)
- User representative(s) assigned by the CCB chairperson
- Public Health Command (PHC) representative
- Centers for Disease Control and Prevention, National Institute for Occupational Safety and Health (NIOSH) representative
- US Army Aeromedical Research Laboratory (USAARL) representative
- US Air Force Research Laboratory (AFRL) Human Effectiveness Directorate representative
- ORCA model manager

The AHA AH model manager will be present, and other interested parties will be invited as needed but not as voting members. Meeting attendance is open to all interested parties.

2.1.2 Responsibilities

The CCB will review the contents of the project tracker (software defect, request for enhancement (RFE) or patch) and associated documentation on a regular basis.

The CCB will meet at least once every quarter to review and approve/disapprove any changes to the package for release preparation.

CCB meetings may be “virtual” where all members review the documents and render their judgments via E-mail.

The CCB chairperson has final authority on all decisions of the CCB.

CCB decisions to do other than accept the changes stemming from the technical review will produce annotation of the rationale for the decisions in the project tracker.

2.2 Requests, Reporting and Patches

2.2.1 Software Maintenance

Two types of software changes take place, either changes that support a new feature or maintenance. Four categories of maintenance that following ISO/IEC 14764 are tracked for each maintenance software modification:

- Corrective maintenance: Reactive modification of a software product performed after delivery to correct discovered problems.
- Adaptive maintenance: Modification of a software product performed after delivery to keep a software product usable in a changed or changing environment.
- Perfective maintenance: Modification of a software product after delivery to improve performance or maintainability.
- Preventive maintenance: Modification of a software product after delivery to detect and correct latent faults in the software product before they become effective faults.

2.2.2 Software Defect Reporting

When a software defect or unexpected behavior is encountered, it needs to be documented. This means making a formal report via e-mail to:

usarmy.APG.arl.list.AHAAH-help@mail.mil

Software defect descriptions should contain enough information for the development team to reproduce the problem. When AHA AH is embedded in other software, it is the responsibility of the model manager for the embedding software to coordinate the submission of software defects to the AHA AH-help mail list.

2.2.3 Requests for Features and Enhancements

All requests for features and enhancements to AHA AH will be submitted and documented via e-mail submission to:

usarmy.APG.arl.list.AHAAH-help@mail.mil

Requests for enhancements should include information about the nature of the change/feature, and impact on existing system or additional capability that would result.

3 Developer Access

Access to the AHA AH repository is restricted and is controlled by the AHA AH MM. Once access is granted, developers can retrieve the CIs for AHA AH from the source repository. Commit access is also restricted. The AHA AH MM and development team review candidates for developer/commit access. The development team recommends individuals for commit access to the CCB Chairperson for approval. Commit access is revoked on a similar basis.

All commits to the source repository must include annotation indicating the nature / purpose of the changes made. The commit message of non-trivial code changes should contain a citation to citation pointing to a work-item tracker entry on the project tracking system that effectively describes the functional requirements of the change. Small

changes that are expected to have no impact on user experience either through the library or user interface may be sufficiently documented in the revision control commit message alone.

4 New Feature Development

New features being incorporated into the software package should meet a minimum standard for acceptance. This includes a software review and appropriate documentation for software changes. For documentation, a technical and editorial review is necessary.

A new software application\ or feature that does not have documentation may be accepted if it passes a technical review. However, it may be excluded from the build system or binary distribution until suitable documentation exists. Developers are required to document their code and update/provide necessary external documentation when necessary to ensure timely acceptance.

For library additions, a software technical review is required for acceptance.

5 Revision Control

The entirety of AHAAH is maintained under a revision control system to manage changes to the model. Therefore, all CIs reside within revision control. All changes to the package are made through revision control. All portions of AHAAH are maintained in the same revision control repository. This includes source code, documentation, test suites, change logs, developer notes, etc. Revision control is currently facilitated by the opensource tool, git, <http://git-scm.com>. The reference repository is located on shared ARL assets.

5.1 Tags

The revision control system shall allow for the application of names or “tags” for versions of the repository. The following set of tags is maintained on the repository.

- Each “Release” version of the package
- The latest “stable” version of the repository. This is the most recent version of the package to compile and pass automatic testing on all supported platforms.
- The latest “stable” version of the repository for each supported platform.

5.1.1 Stable Branch

The latest stable version of the repository is the tip of branch “stable”.

```
% git co stable
```

This denotes a version of the package that has passed all automated tests. Most of the time, this branch will refer to a version with an odd-numbered minor version. The exception would be the time immediately following a release, preceding any commit of changes.

5.2 Branch Utilization

The revision control system supports the notion of “branches” to the code. This allows multiple evolutionary paths to be managed simultaneously without affecting each other. Changes from one branch can be merged into another branch in a defined process. The use of “topic branches” for the development of enhancements or maintenance that is expected to span multiple commits is strongly encouraged. .

6 Testing and Quality Assurance

An essential element of configuration management for AHAAH is the regression test suite. This test suite exercises all portions of AHAAH that represent a critical component. The test suite is to be run nightly on all platforms that are supported by ARL. The source code is retrieved, compiled, and regression tests are run. On nights where the regression test succeeds, an installation is performed and it is flagged as **stable**. This permits it to be used by the ORCA regression test suite for that night.

The nightly regression test will include a testing application designed to process various insults for particular configurations. The result of this reference set is compared to previous results. Any difference will result in the test reporting failure until a human inspects the results, identifies the change, and resolves whether the change represents correct behavior or not. The reference set includes tests needed to confirm proper operation with ARL ORCA. Other tests may be instituted on an as-needed basis.

The regression test resides in the AHAAH repository in the “test” directory. In this way, all developers may perform the regression test on their changes immediately, without performing a commit. This behavior is strongly encouraged.

7 Releases

AHAAH targets a quarterly release cycle. This allows software defect fixes and feature enhancements to be delivered in a timely fashion to the user community. It establishes confidence that issues raised are addressed, and that the package continues to be viable and evolving, as opposed to becoming “dead code”.

7.1 Release Generation

To facilitate the release cycle, developers are expected to “settle down” around the last few days before a quarterly release so that the package may be verified, tested, and released. Significant changes to the source code should not occur during these last few

days of the iteration, only minor software defects and build fixes. A release should then be made within the first few days of the next quarter's iteration.

The testing infrastructure should perform most of the functions of release generation.

7.2 Release Numbers

AHAAH has adopted a three-digit version number convention for identifying and tracking future releases. AHAAH version 2.1 will become version 2.1.0 even though releases are commonly referred to using only the first two digits. This convention has been adopted from common practice in many other open-source projects including the Linux kernel. This number follows a common convention where the three numbers represent:

`{MAJOR_VERSION}.{MINOR_VERSION}.{PATCH_VERSION}`

The `MAJOR_VERSION` should only increment when it is deemed that major changes have accumulated, significant new features have been added, or significant backwards incompatibilities were added to make a release warrant a major version increment. In general, releases of AHAAH that differ in the `MAJOR_VERSION` are not considered compatible with each other.

The `MINOR_VERSION` is updated more frequently. Every production release that is backwards incompatible in some manner should increment at least the minor version number. A minor version update is generally issued after significant development activity (generally one to three months or more of activity) has been tested and deemed sufficiently stable. When a stable release is made, it is tagged as a maintenance branch and patch releases are performed on that branch.

The `PATCH_VERSION` may and should be updated as frequently as is necessary. Every *production* maintenance release should increment the patch version. Patch versions are typically to address software defects that needed more immediately than the quarterly release cycle and that do not introduce backward incompatibility.

Release versions of the software are tagged in git with a tag of the form: *rel-major-minor-patch*. For example, using the following git command:

```
git tag rel-1-2-4
```

8 Revisions

Initial revision of the AHAAH Configuration Management Plan.

9 References

1. *Guide to the Software Engineering Body of Knowledge SWEBOK™*, IEEE –Trial Version 1.00 – May 2001.

2. *Management of Army Models and Simulations*, Army Regulation 5–11, 10 July 1997.
3. ISO/IEC 14764:2006 Software Engineering — Software Life Cycle Processes — Maintenance ISO/IEC 14764