Master Test Plan for EMgine: A Computational Model of Emotion for Enhancing Non-Player Character Believability in Games

Geneva M. Smith

Version 1.0 (March 24, 2023)

Geneva M. Smith
Dept. of Computing and Software—McMaster University

Revision History

Date	Version	Notes
March 24, 2023	1.0	• Initial version

Contents

List of Tables

List of Figures

1 Symbols, Abbreviations and Acronyms

For EMgine's other symbols, abbreviations, and acronyms, see the:

- Software Requirement Specification (SRS) at https://github.com/GenevaS/EMgine/blob/main/docs/SRS/EMgine_SRS.pdf, and
- Module Guide (MG) at https://github.com/GenevaS/EMgine/blob/main/docs/Design/MG/EMgine_MG.pdf.

Abbrv.	Description
ATP	Acceptance Test Plan
ATR	Acceptance Test Report
CME	Computational Model of Emotion
CS	Computer Science
HCI	Human-Computer Interaction
IR	Issue Report
MG	Module Guide
MIS	Module Interface Specification
MTP	Master Test Plan
MTR	Master Test Report
NPC	Non-Player Character (Video Games)
SDA	Software Development Artifact
SE	Software Engineering
SIUTP	System, Integration, and Unit Test Plan
SIUTR	System, Integration, and Unit Test Report
SDLC	Software Development Life Cycle
SRS	Software Requirements Specification
V & V	Verification and Validation

2 Introduction

This document gives an overview of the test planning and management for EMgine. Its purpose is to describe the overall goals of the testing efforts, how they relate to EMgine's concept, and where they fit in EMgine's Software Development Life Cycle (SDLC). It describes the necessary testing activities from concept to validation, including what is required to do the activities and their expected outcomes and completion time.

The document's content and organization is loosely based on IEEE Std 829-2008 Clause 8: Master Test Plan (MTP) (IEEE Computer Society, 2008).

2.1 Summary of EMgine's Purpose and Design Goals

EMgine is a Computational Model of Emotion (CME) for Non-Player Characters (NPCs) to enhance their believability, with the goal of improving long-term player engagement. EMgine is for *emotion generation*, accepting user-defined information from a game environment to determines what emotion and intensity a NPC is "experiencing". How the emotion is expressed and what other effects it could have on game entities is left for game designers/developers to decide.

EMgine aims to provide a feasible and easy-to-use method for game designers/developers to include emotion in their NPCs, they perceive to be challenging with the current tools and restrictions (Broekens et al., 2016). EMgine should be modular and portable such that game designers/developers can use it in their regular development environment, and should not require knowledge of affective science, psychology, and/or emotion theories. Therefore, it is a library of components to maximize a game designer/developer's control over how and when EMgine functions.

2.2 Scope of Testing Efforts

The overall goals of EMgine's testing effort are:

1. Ensure traceability between EMgine's concept and each of its software development artifacts (SDAs)
2. Create a basis for building confidence in the grounding of EMgine's behaviours in affective science/psychology
3. Create a basis for building confidence in EMgine's functionality and usability (i.e learnability and amount of effort required to use it)

A test plan must be executed in full for each new major version of a software development artifact (SDA). Previous versions of that SDA do not need to be retested. An informal version of a test plan can be executed for new minor versions of a SDA to help reduce the effort required when testing the next major version.

The testing effort includes a verification and/or validation plan for each SDA that is expected from EMgine's SDLC stages (Table 1). Each plan has an associated report that documents the results of the testing activities:

- The Master Test Report (MTR) documents the results of *document* verification (i.e. SRS, MG, MIS, Test Plans) and references all other test reports

- The System, Integration, and Unit Test Report (SIUTR) documents the results of system, integration, and unit testing
- The Acceptance Test Report (ATR) documents the results of acceptance testing

Table 1: Expected SDAs for EMgine's SDLC Stages

Stage	**SDA**
Concept Definition	• Concept Summary • Master Test Plan • Master Test Report (Partial)
Requirements Analysis	• Software Requirements Specification • System Test Plan • Acceptance Test Plan • Master Test Report (Partial)
Architecture Definition	• Module Guide • Integration Test Plan • Master Test Report (Partial)
Design Definition	• Module Interface Specification • Unit Test Plan • Master Test Report (Partial)
Implementation	• Source Code • Source Code Documentation
Verification	• System, Integration, and Unit Test Report
Validation	• Acceptance Test Report

2.3 Project Organization

EMgine's development uses the general software development life cycle processes of requirements analysis, architecture and design definition, implementation, verification, and validation. Each of these processes outputs at least one Software Development Artifact (SDA) which serves as a description of the process's results and becomes an input to subsequent processes (Figure 1). Each SDA has an associated verification and/or validation plan (Table 2) to ensure that it adheres to the EMgine's concept.

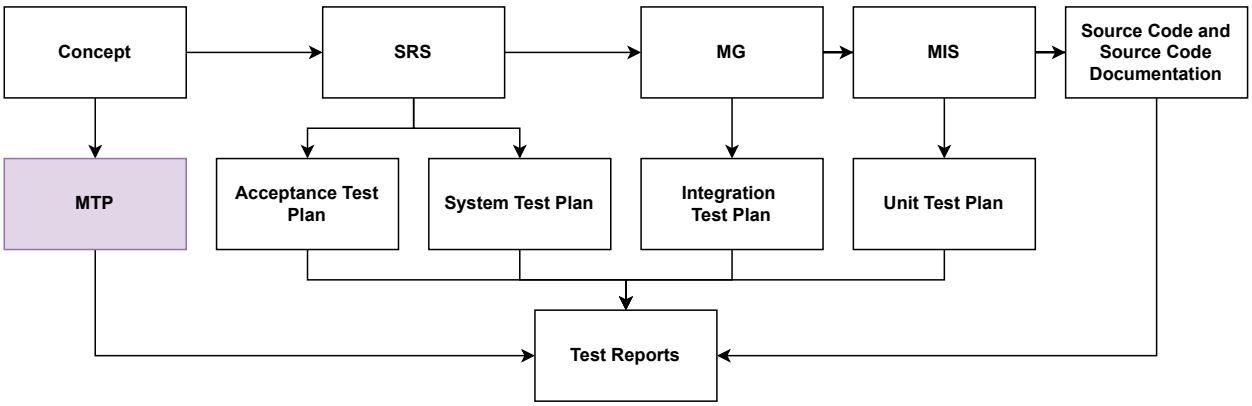


Figure 1: Dependencies Between EMgine's SDAs (MTP Highlighted)

Table 2: Location of V & V Plans for EMgine's SDAs

SDA	**Location of V & V Plan**
Concept Summary	–
Master Test Plan	Master Test Plan
Software Requirements Specification	Master Test Plan
Module Guide	Master Test Plan
Module Interface Specification	Master Test Plan
System, Integration, and Unit Test Plan	Master Test Plan
Acceptance Test Plan	Master Test Plan
Source Code and Source Code Documentation	System, Integration, and Unit Test Plan Acceptance Test Plan

2.4 Relevant Documentation

The Master Test Plan (MTP) refers to the following Software Development Artifacts (SDAs):

- **Title**: Concept Summary for EMgine: A Computational Model of Emotion for Enhancing Non-Player Character Believability in Games
  Location: https://github.com/GenevaS/EMgine/blob/main/docs/ConceptSummary/EMgine_ConceptSummary.pdf
  Description: Product of the concept definition process. An overview of EMgine purpose, design goals, and motivation.

- **Title**: Software Requirements Specification for EMgine: A Computational Model of Emotion for Enhancing Non-Player Character Believability in Games
  Location: https://github.com/GenevaS/EMgine/blob/main/docs/SRS/EMgine_SRS.pdf
  Description: Product of the requirements analysis process. EMgine's problem domain, purpose, underlying models, requirements (functional and nonfunctional), and likely changes.

- **Title**: Module Guide for EMgine: A Computational Model of Emotion for Enhancing Non-Player Character Believability in Games
  Location: https://github.com/GenevaS/EMgine/blob/main/docs/Design/MG/EMgine_MG.pdf
  Description: Product of the architecture definition process. An overview of EMgine's architecture and component modules.

- **Title**: Module Interface Specification for EMgine: A Computational Model of Emotion for Enhancing Non-Player Character Believability in Games
  Location: TDB
  Description: Product of the design definition process. Specifications of each module in EMgine such that they are readily implementable.

- **Title**: Source Code and Documentation for EMgine: A Computational Model of Emotion for Enhancing Non-Player Character Believability in Games
  Location: TDB
  Description: Product of the implementation process. A software-based realization of EMgine's requirements and design, accompanied by documentation of the resulting components and/or processes.

- **Title**: User Manual for EMgine: A Computational Model of Emotion for Enhancing Non-Player Character Believability in Games
  Location: TDB
  Description: Product of the implementation process. A document written for EMgine's users to help them learn about and use EMgine.

EMgine has additional test plan documents as follows:

- **Title**: System, Integration, and Unit Test Plan for EMgine: A Computational Model of Emotion for Enhancing Non-Player Character Believability in Games
  Location: TDB
  Description: Test specifications to evaluate EMgine's models with respect to its design specifications (see Section 3.5).

- **Title**: Acceptance Test Plan for EMgine: A Computational Model of Emotion for Enhancing Non-Player Character Believability in Games
  Location: TDB
  Description: Test specifications to evaluate the ability of EMgine's models to produce expected emotions and intensities (see Section 3.6).

This document also refers to the:

- IEEE Standard for Software and System Test Documentation (IEEE Std 829-2008) (IEEE Computer Society, 2008) to inform the creation of this document, the System, Integration, and Unit Test Plan, and the Acceptance Test Plan
- IEEE Standard for System, Software, and Hardware Verification and Validation (IEEE Std 1012-2016) (IEEE Computer Society, 2017) to inform the creation of this document and the System, Integration, and Unit Test Plan
- IEEE Recommended Practice for Software Requirements Specifications (IEEE Std 830-1998 (R2009)) (IEEE Computer Society, 2009) to inform the evaluation of EMgine's Software Requirements Specification (SRS)
- ISO/IEC/IEEE International Standard - Systems and software engineering – System life cycle processes (ISO/IEC/IEEE 15288:2015(E)) (ISO, IEC, and IEEE, 2015) to inform the evaluation of EMgine's design

2.5 Verification and Validation Team and Responsibilities

These are individuals who contribute to the testing of EMgine's Software Development Artifacts (SDAs):

- *Geneva M. Smith* is the primary designer/developer of EMgine and has a background in Software Engineering (SE) and Human-Computer Interactions (HCI)
- *Dr. Jacques Carette* is the direct supervisor of Geneva M. Smith, has a background in Mathematics and Computer Science (CS), and a research interest in type theory
- *Dr. Spencer Smith* is a member of Geneva M. Smith's supervisory committee, has a background in SE and Civil Engineering, a research interest in applying software engineering to research applications, and is a co-creator of the documentation templates that EMgine uses (Smith and Yu, 2009)
- *Dr. Alan Wassyng* is a member of Geneva M. Smith's supervisory committee, has a background in SE, and a research interest in rigorous software development
- *Dr. Denise Geiskkovitch* is a member of Geneva M. Smith's supervisory committee and has a background in HCI with emphasis on psychology
- *G-ScalE Lab Members* are aware of EMgine but are otherwise not involved in its design/development and have backgrounds in SE, CS, HCI, classical engineering, and game design

Their assigned levels denote their involvement in EMgine's testing:

1. *Primary* members must participate in all stages and activities
2. *Secondary* members are expert reviewers and must participate during major revisions
3. *Tertiary* members are encouraged, but not obligated, to review EMgine at major revisions

Table 3: Team Roles for Test Plan Verification

Role	Name	Goal
Primary	Geneva M. Smith	• Coordinate team activities • Document results of Test Plan Verification process
Secondary	Drs Jacques Carette and Alan Wassyng	• Evaluate traceability to SRS, MG, and MIS • Evaluate traceability between test plan documents • Evaluate feasibility, correctness, completeness, and consistency of test plans • Evaluate readability of the documents
Tertiary	Drs Spencer Smith and Denise Geiskkovitch, G-ScalE Lab Members	

Table 4: Team Roles for Requirements Verification

Role	Name	Goal
Primary	Geneva M. Smith	• Coordinate team activities • Document results of Requirements Verification process
Secondary	Dr. Jacques Carette	• Evaluate models for mathematical correctness
	Dr. Spencer Smith	• Evaluate adherence to the SRS template described in Smith and Yu (2009) • Evaluate overall adherence to the characteristics of a good SRS as described by IEEE Standard 830-1998 (R2009) (IEEE Computer Society, 2009)
	Dr. Denise Geiskkovitch	• Evaluate models for logical use of affective science/psychology literature
Tertiary	Dr. Alan Wassyng	• Evaluate overall adherence to the characteristics of a good SRS as described by IEEE Standard 830-1998 (R2009) (IEEE Computer Society, 2009)

Table 5: Team Roles for Architecture Definition Verification

Role	Name	Goal
Primary	Geneva M. Smith	• Coordinate team activities • Document results of Architecture Definition Verification process
Secondary	Drs Jacques Carette and Spencer Smith	• Evaluate traceability between architecture, modules, and requirements from SRS • Evaluate traceability between document elements • Evaluate overall consistency, completeness, and understandability in the documentation
Tertiary	Dr. Alan Wassyng	• Evaluate traceability between architecture, modules, and requirements from SRS • Evaluate traceability between document elements • Evaluate overall consistency, completeness, and understandability in the documentation

Table 6: Team Roles for Design Definition Verification

Role	Name	Goal
Primary	Geneva M. Smith	• Coordinate team activities • Document results of Design Definition Verification process
Secondary	Drs Jacques Carette and Spencer Smith	• Evaluate traceability between module specifications and architecture from MG • Evaluate encapsulation of module components • Evaluate traceability between document elements • Evaluate overall consistency, completeness, and understandability in the documentation
Tertiary	Dr. Alan Wassyng	• Evaluate traceability between module specifications and architecture from MG • Evaluate traceability between document elements • Evaluate overall consistency, completeness, and understandability in the documentation

Table 7: Team Roles for Implementation Verification

Role	Name	Goal
Primary	Geneva M. Smith	• Implement automated testing if applicable • Coordinate team activities • Document results of Implementation Verification process
Secondary	Dr. Jacques Carette	• Evaluate traceability to the MIS • Evaluate overall consistency, completeness, and understandability in the source code and source code documentation
Tertiary	Dr. Spencer Smith and G-ScalE Lab Members	• Evaluate overall consistency, completeness, and understandability in the source code and source code documentation

Table 8: Team Roles for Implementation Validation

Role	**Name**	**Goal**
Primary	Geneva M. Smith	• Implement automated testing if applicable • Run user studies, analyze collected data, and report findings • Coordinate team activities • Document results of Validation process
Secondary	Dr. Alan Wassyng	• Evaluate traceability to stakeholder requirements/needs • Evaluate overall correctness, completeness, consistency, and readability of testing results
	Dr. Denise Geiskkovitch	• Advise user study data analysis and reporting process • Evaluate overall correctness, completeness, consistency, and readability of testing results
Tertiary	Drs Jacques Carette and Spencer Smith, G-ScalE Lab Members	• Evaluate overall correctness, completeness, consistency, and readability of testing results

3 Details of the Master Test Plan

This section outlines the methodologies and tools for Verification and Validation (V & V), and the team responsible for executing the plan for each of EMgine's Software Development Artifacts (SDAs).

3.1 Test Plan Verification

The goal of Test Plan verification is to evaluate all test plan documents for correctness, consistency, completeness, readability, feasibility, and traceability.

Method The test plan verification plan relies on peer review/document inspection with the following stages:

1. Preparation: Participants review the test plan documents with respect to their assigned role and goals (Table 3)
2. Meeting: Participants meet to discuss findings, potential issues, and proposed action plans to address them
3. Rework: The test plan author(s) revise the document(s) to address raised issues, guided by the proposed action plans
4. Follow Up: Participants verify that raised issues have been addressed satisfactorily

A recording device might be used to capture meeting proceedings in place of physical note taking so that all participants can focus on the discussion.

Peer review/inspection begins when there is a new major version of a test plan document. Peer review/inspection need be done on those documents only.

Peer review/inspection ends when reviewers agree that there are no issues that will likely result in the inability to execute any element of the plans and verify that all plan components contribute to EMgine's overall verification and validation (V & V).

Roles and Responsibilities To assist in the achievement of their assigned goals (Table 3) using peer review/document inspection:

- Primary team members are responsible for ensuring that reviewers have the necessary materials, moderating the inspection process and reading through the test plan document(s) during the meeting
- Secondary and tertiary members are reviewers whom are responsible for reviewing the test plan document(s) prior to the meeting so that they are prepared to discuss it with the team

Inputs

- Concept Summary for EMgine: A Computational Model of Emotion for Enhancing Non-Player Character Believability in Games
- Master Test Plan for EMgine: A Computational Model of Emotion for Enhancing Non-Player Character Believability in Games

- Acceptance Test Plan for EMgine: A Computational Model of Emotion for Enhancing Non-Player Character Believability in Games
- System, Integration, and Unit Test Plan for EMgine: A Computational Model of Emotion for Enhancing Non-Player Character Believability in Games
- Review guide for EMgine's Master Test Plan (MTP) (Appendix A.1)
- Review guide for EMgine's Acceptance Test Plan (ATP) (Appendix A.2)
- Review guide for EMgine's System, Integration, and Unit Test Plan (SIUTP) (Appendix A.3)

Outputs

- Objective evidence to assess the verification of the MTP, ATP, and SIUTP
- Objective evidence that the test plans are able to determine if their associated SDAs conform to their requirements and satisfy their testing criteria
- Objective evidence that the test plans are able to determine if EMgine satisfies its allocated system requirements and its intended use and user needs
- Input to Master Test Report (MTR)

Estimated Completion Time Five (5) weeks

Verification of test plan documentation is divided into parts. Only one part is tested per week to reduce participant fatigue and to better coordinate with the corresponding stage in the Software Development Life Cycle (SDLC):

- Part 1: Master Test Plan
- Part 2: Acceptance Test Plan
- Part 3: System, Integration, and Unit Test Plan: Introduction/Preamble, System Test Plan
- Part 4: System, Integration, and Unit Test Plan: Integration Test Plan
- Part 5: System, Integration, and Unit Test Plan: Unit Test Plan

Week	Day						
	1	2	3	4	5	6	7
1	Preparation (Master Test Plan)			Meeting		Follow Up	
					Rework		
2	Preparation (Acceptance Test Plan)			Meeting		Follow Up	
					Rework		
3	Preparation (SIUTP Intro/Preamble, System Test Plan)			Meeting		Follow Up	
					Rework		
4	Preparation (Integration Test Plan)			Meeting		Follow Up	
					Rework		
5	Preparation (Unit Test Plan)			Meeting		Follow Up	
					Rework		

3.2 Requirements Verification

The goal of requirements verification is to evaluate the Software Requirements Specification (SRS) for correctness, consistency, completeness, readability, testability, and traceability. This corresponds to the requirements evaluation and traceability analysis tasks in Section 9.2 of IEEE Std 1012-2016 (IEEE Computer Society, 2017).

Method The SRS verification plan relies on peer review/document inspection with the following stages:

1. Preparation: Participants review the SRS document with respect to their assigned role and goals (Table 4)
2. Meeting: Participants meet to discuss findings, potential issues, and proposed action plans to address them
3. Rework: The SRS author(s) revise the document to address raised issues, guided by the proposed action plans
4. Follow Up: Participants verify that raised issues have been addressed satisfactorily

A recording device might be used to capture meeting proceedings in place of physical note taking so that all participants can focus on the discussion.

Peer review/inspection begins when there is a new major version of the SRS document.

Peer review/inspection ends when reviewers agree that there are no issues that will likely result in an extensive loss of confidence in EMgine (e.g. barrier to adoption by primary stakeholders).

Roles and Responsibilities To assist in the achievement of their assigned goals (Table 4) using peer review/document inspection:

- Primary team members are responsible for ensuring that reviewers have the necessary materials, moderating the inspection process and reading through the SRS document during the meeting(s)
- Secondary and tertiary members are reviewers whom are responsible for reviewing the SRS document prior to the meeting(s) so that they are prepared to discuss it with the team

Inputs

- Software Requirements Specification for EMgine: A Computational Model of Emotion for Enhancing Non-Player Character Believability in Games
- Concept Summary for EMgine: A Computational Model of Emotion for Enhancing Non-Player Character Believability in Games
- References used to generate primary stakeholder requirements
- Review guide for EMgine's SRS (Appendix A.4)

Outputs

- Objective evidence to assess the verification of the SRS
- Objective evidence that the requirements are complete, correct, accurate, and testable
- Objective evidence that the requirements are traceable to EMgine's concept summary and intended use and user needs
- Objective evidence that the SRS is readable
- Input to Master Test Report (MTR)

Estimated Completion Time Four (4) weeks

Due to the document's size, SRS peer review/inspection is divided into parts. Only one part is tested per week to reduce participant fatigue:

- Part 1: Introduction (Section 2), General System Description (Section 3)
- Part 2: Problem Description (Section 4.1), Solution Characteristics Specification: Emotion Theories and Models (Section 4.2.1), Assumptions (Section 4.2.2), Conceptual Models (Section 4.2.3), Theoretical Models (Section 4.2.4)
- Part 3: Solution Characteristics Specification: General Definitions (Section 4.2.5), Data Definitions (Section 4.2.6), Type Definitions (Section 4.2.7), Instance Models (Section 4.2.8), Data Constraints (Section 4.2.9), Properties of a Correct Solution (Section 4.2.10)
- Part 4: Requirements (Section 5), Future Changes (Section 6), Traceability Matrices and Graphs (Section 7)

Week	Day						
	1	2	3	4	5	6	7
1	Preparation (Sections 2, 3)			Meeting		Follow Up	
				Rework			
2	Preparation (Sections 4.1, 4.2.1, 4.2.2, 4.2.3, 4.2.4)			Meeting		Follow Up	
				Rework			
3	Preparation (Sections 4.2.5, 4.2.6, 4.2.7, 4.2.8, 4.2.9, 4.2.10)			Meeting		Follow Up	
				Rework			
4	Preparation (Sections 5, 6, 7)			Meeting		Follow Up	
				Rework			

3.3 Architecture Definition Verification

The goal of architecture definition verification is to evaluate EMgine's Module Guide (MG) for correctness, consistency, completeness, readability, testability, and traceability. This corresponds to the design evaluation and traceability analysis tasks in Section 9.3 of IEEE Std 1012-2016 (IEEE Computer Society, 2017).

Method The MG verification plan relies on peer review/document inspection with the following stages:

1. Preparation: Participants review the MG document with respect to their assigned role and goals (Table 5)
2. Meeting: Participants meet to discuss findings, potential issues, and proposed action plans to address them
3. Rework: The MG author revises the document to address raised issues, guided by the proposed action plans
4. Follow Up: Participants verify that raised issues have been addressed satisfactorily

A recording device might be used to capture meeting proceedings in place of physical note taking so that all participants can focus on the discussion.

Peer review/inspection begins when there is a new major version of the MG document.

Peer review/inspection ends when reviewers agree that there are no issues that will likely result in an extensive loss of confidence in EMgine (e.g. an unaddressed requirement).

Roles and Responsibilities To assist in the achievement of their assigned goals (Table 5) using peer review/document inspection:

- Primary team members are responsible for ensuring that reviewers have the necessary materials, moderating the inspection process and reading through the MG document during the meeting(s)
- Secondary and tertiary members are reviewers whom are responsible for reviewing the MG document prior to the meeting so that they are prepared to discuss it with the team

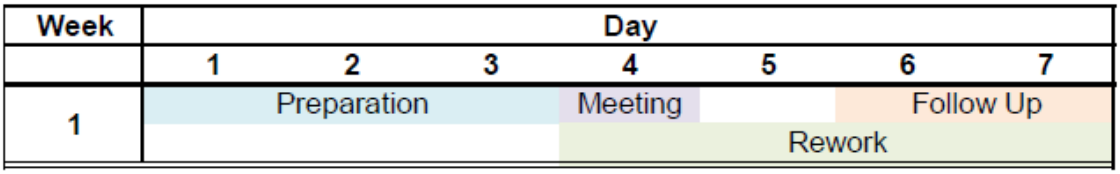
Inputs

- Module Guide for EMgine: A Computational Model of Emotion for Enhancing Non-Player Character Believability in Games
- Software Requirements Specification for EMgine: A Computational Model of Emotion for Enhancing Non-Player Character Believability in Games
- Review guide for EMgine's MG (Appendix A.5)

Outputs

- Objective evidence to assess the verification of the MG
- Objective evidence that the architecture design is complete, correct, accurate, and testable
- Objective evidence that the architecture design is realizable
- Objective evidence that the architecture design satisfies the requirements described in EMgine's Software Requirements Specification (SRS)
- Objective evidence that the MG is readable
- Input to Master Test Report (MTR)

Estimated Completion Time One (1) week



3.4 Design Definition Verification

The goal of design definition verification is to evaluate EMgine's Module Interface Specification (MIS) for correctness, consistency, completeness, readability, testability, and traceability. This corresponds to the design evaluation and traceability analysis tasks in Section 9.3 of IEEE Std 1012-2016 (IEEE Computer Society, 2017).

Method The MIS verification plan relies on peer review/document inspection with the following stages:

1. Preparation: Participants review the MIS document with respect to their assigned role and goals (Table 6)
2. Meeting: Participants meet to discuss findings, potential issues, and proposed action plans to address them
3. Rework: The MIS author revises the document to address raised issues, guided by the proposed action plans
4. Follow Up: Participants verify that raised issues have been addressed satisfactorily

A recording device might be used to capture meeting proceedings in place of physical note taking so that all participants can focus on the discussion.

Peer review/inspection begins when there is a new major version of the MIS document.

Peer review/inspection ends when reviewers agree that there are no issues that will likely result in an extensive loss of confidence in EMgine (e.g. insufficient information hiding).

Roles and Responsibilities To assist in the achievement of their assigned goals (Table 6) using peer review/document inspection:

- Primary team members are responsible for ensuring that reviewers have the necessary materials, moderating the inspection process and reading through the document(s) during the meeting(s)
- Secondary and tertiary members are reviewers whom are responsible for reviewing the MIS document prior to the meeting so that they are prepared to discuss it with the team

Inputs

- Module Interface Specification for EMgine: A Computational Model of Emotion for Enhancing Non-Player Character Believability in Games
- Module Guide for EMgine: A Computational Model of Emotion for Enhancing Non-Player Character Believability in Games
- Review guide for EMgine's MIS (Appendix A.6)

Outputs

- Objective evidence to assess the verification of the MIS
- Objective evidence that the module design specifications are complete, correct, accurate, and testable
- Objective evidence that no unintended features were introduced into the design
- Objective evidence that the module design specifications represents a complete transformation of the requirements described in EMgine's Software Requirements Specification (SRS)
- Objective evidence that the MIS is readable
- Input to Master Test Report (MTR)

Estimated Completion Time Four (4) weeks

Due to the number and relative complexity of the modules, MIS verification is divided into parts by level in the use hierarchy. This ensures that modules are verified before those that use them so that any necessary changes are made before their verification. Only one part is tested per week to reduce participant fatigue:

- Part 1: Emotion Intensity Type (M1), Emotion Intensity Decay Rate Type (M5), PAD Type (M8), Social Attachment (M15), Time (M16) World State (M17)
- Part 2: Emotion State Type (M3), Goal (M12), Plan (M13), Attention (M14)
- Part 3: Emotion Intensity Decay State (M6), Emotion Decay Function (M7), PAD Function (M9), Emotion Type (M10)
- Part 4: Emotion Intensity Function (M2), Emotion Generation Function (M4), Emotion Function (M11)

Week	Day						
	1	2	3	4	5	6	7
1	Preparation (Modules 1, 5, 8, 15, 16, 17)			Meeting		Follow Up	
				Rework			
2	Preparation (Modules 3, 12, 13, 14)			Meeting		Follow Up	
				Rework			
3	Preparation (Modules 6, 7, 9, 10)			Meeting		Follow Up	
				Rework			
4	Preparation (Modules 2, 4, 11)			Meeting		Follow Up	
				Rework			

3.5 Implementation Verification

The goal of implementation verification is to evaluate EMgine's source code and its documentation for correctness, consistency, completeness, accuracy, readability, testability, and traceability. This corresponds to the source code and source code documentation evaluation and traceability analysis tasks in Section 9.4 of IEEE Std 1012-2016 (IEEE Computer Society, 2017).

Method The implementation verification plan uses both static and dynamic testing methods:

1. Static Testing Method
   The source code and its documentation is statically tested by peer review. A peer review tests: the source code for traceability to Module Interface Specification (MIS) document and consistent use of coding standards and quality; and the source code's documentation for correctness, completeness, consistency, and readability.

   The peer review has the following stages:

   (a) Preparation: Participants review the source code and its documentation with respect to their assigned role and goals (Table 7)

   (b) Meeting: Participants meet to discuss findings, potential issues, and proposed action plans to address them

   (c) Rework: The source code/documentation author revises the code/documentation to address raised issues, guided by the proposed action plans

   (d) Follow Up: Participants verify that raised issues have been addressed satisfactorily

   A recording device might be used to capture meeting proceedings in place of physical note taking so that all participants can focus on the discussion.

   Peer review begins when there is a new major version of the source code and/or its documentation.

   Peer review ends when reviewers agree that there are no issues that will likely result in an extensive loss of confidence in EMgine (e.g. inconsistent naming conventions).

2. Dynamic Testing Method
   Test cases dynamically test the source code for its adherence to the behaviours specified in the functional requirements in the Software Requirements Specification (SRS) and the characteristics and/or properties specified in the nonfunctional requirements in the SRS.

   For information about test case types, specifications, and traceability to EMgine's requirements, see the "System, Integration, and Unit Test Plan for EMgine: A Computational Model of Emotion for Enhancing Non-Player Character Believability in Games" document.

Roles and Responsibilities To assist in the achievement of their assigned goals (Table 7):

1. Static Testing

   - Primary team members are responsible for ensuring that reviewers have the necessary materials, moderating the peer review process and reading through the document(s) during the meeting(s)

- Secondary and tertiary members are reviewers whom are responsible for reviewing the source code and source code documentation prior to the meeting so that they are prepared to discuss it with the team

2. Dynamic Testing

   - Primary team members are responsible for generating and implementing test cases and automated tools as described in the "System, Integration, and Unit Test Plan for EMgine: A Computational Model of Emotion for Enhancing Non-Player Character Believability in Games" document
   - Secondary and tertiary members do not participate in dynamic testing

Inputs

- Source Code and Documentation for EMgine: A Computational Model of Emotion for Enhancing Non-Player Character Believability in Games
- Module Guide for EMgine: A Computational Model of Emotion for Enhancing Non-Player Character Believability in Games
- Module Interface Specification for EMgine: A Computational Model of Emotion for Enhancing Non-Player Character Believability in Games
- Coding standards for the source code's implementation language
- Review guide for EMgine's source code and its documentation (Appendix A.7)
- System, Integration, and Unit Test Plan for EMgine: A Computational Model of Emotion for Enhancing Non-Player Character Believability in Games

Outputs

- Objective evidence to assess the verification of the source code and its documentation
- Objective evidence that the source code and its documentation are correct, accurate, and complete
- Objective evidence that the source code correctly implements the design specification in EMgine's Module Guide (MG) and MIS
- Objective evidence that the source code and its documentation are readable
- Input to Master Test Report (MTR)
- Input to the System, Integration, and Unit Test Report (SIUTR)

Estimated Completion Time Four (4) weeks

Due to the number and relative complexity of the modules, source and associated documentation verification is divided into parts by level in the MIS use hierarchy. This ensures that a piece of code/documentation is verified before integration testing with other code units. Only one part is tested per week to reduce participant fatigue:

- Part 1: Emotion Intensity Type (M1), Emotion Intensity Decay Rate Type (M5), PAD Type (M8), Social Attachment (M15), Time (M16) World State (M17)
- Part 2: Emotion State Type (M3), Goal (M12), Plan (M13), Attention (M14)
- Part 3: Emotion Intensity Decay State (M6), Emotion Decay Function (M7), PAD Function (M9), Emotion Type (M10)
- Part 4: Emotion Intensity Function (M2), Emotion Generation Function (M4), Emotion Function (M11)

Week	Day						
	1	2	3	4	5	6	7
1	Peer Review Preparation (Code Modules 1, 5, 8, 15, 16, 17)			Meeting		Follow Up	
				Rework			
	Test Case Implementation (Code Modules 1, 5, 8, 15, 16, 17)			Test Case Execution + Reporting			
2	Peer Review Preparation (Code Modules 3, 12, 13, 14)			Meeting		Follow Up	
				Rework			
	Test Case Implementation (Code Modules 3, 12, 13, 14)			Test Case Execution + Reporting			
3	Peer Review Preparation (Code Modules 6, 7, 9, 10)			Meeting		Follow Up	
				Rework			
	Test Case Implementation (Code Modules 6, 7, 9, 10)			Test Case Execution + Reporting			
4	Peer Review Preparation (Code Modules 2, 4, 11)			Meeting		Follow Up	
				Rework			
	Test Case Implementation (Code Modules 2, 4, 11)			Test Case Execution + Reporting			

3.6 Implementation Validation

The goal of implementation validation is to evaluate EMgine's ability to meet its stakeholders' expectations with respect to its behaviour and characteristics. This corresponds to the software acceptance test procedure V & V and traceability analysis tasks in Section 9.7 of IEEE Std 1012-2016 (IEEE Computer Society, 2017).

Method Validation relies on test cases and user studies:

- Test cases evaluate EMgine's models to see if they produce the "correct" emotion type and intensity given an emotion-eliciting context. This involves the creation of test cases from scenarios where "inputs" are deducible and "outputs" are readily observable.
- User studies evaluate meets the needs of EMgine's stakeholders by involving them directly in the testing process. Their design is based on methods and techniques from Human-Computer Interaction (HCI).

For information about test case specifications, user study designs, and traceability to EMgine's requirements, see the "Acceptance Test Plan for EMgine: A Computational Model of Emotion for Enhancing Non-Player Character Believability in Games" document.

Role Assignments To assist in the achievement of their assigned goals (Table 8):

- Primary team members are responsible for preparing testing reports and providing them to other members of the validation team
- Secondary team members are responsible for advising primary members during test report preparation in accordance with their expertise
- Tertiary team members are responsible for evaluating test reports for clarity and comprehensiveness

Inputs

- Source Code and Documentation for EMgine: A Computational Model of Emotion for Enhancing Non-Player Character Believability in Games
- User Manual for EMgine: A Computational Model of Emotion for Enhancing Non-Player Character Believability in Games
- Acceptance Test Plan for EMgine: A Computational Model of Emotion for Enhancing Non-Player Character Believability in Games
- Master Test Report (MTR)
- System, Integration, and Unit Test Report (SIUTR)

Outputs

- Objective evidence that EMgine satisfies all allocated system requirements and its intended use and user needs
- Input to MTR
- Input to the Acceptance Test Report (ATR)

Estimated Completion Time Two (2) weeks + X × four (4) weeks, where X is the number of user studies

This value is derived from estimates pertaining to the quantity of required test cases and the number and nature of the user studies.

It is difficult to know how many test cases are necessary to validate a CME. One report estimates that researchers created approximately 600 different scenarios for a CME with 24 emotion kinds (Elliott, 1998). For EMgine, this is approximately 200 scenarios for eight emotion kinds. Accounting for time to analyze and reason about tests the EMgine fails, the expected completion time for test case validation is two weeks if roughly 20 test cases are completed per day.

Due to the involvement of human participants, completing user studies is an extended process. It is estimated that—per user study—data collection will take approximately two and a half weeks and data analysis one and a half weeks for a total of four weeks.

4 Test Reporting Requirements

This section describes the type and contents of reports that must be generated to summarize the results of all Verification and Validation (V & V) efforts.

4.1 Master Test Report

The Master Test Report (MTR) provides an overview of all V & V efforts, including those documented in other test reports. The document's content and organization is based on IEEE Std 829-2008 Clause 17: Master Test Report (IEEE Computer Society, 2008). The MTR must include:

1. Document Revision History
2. Reference Material (e.g. symbols, acronyms, definitions)
3. Introduction
   Identify who prepared the report, its purpose, and its current status (e.g. Draft, Final)
   (a) Scope
   Summarize the components of EMgine tested, which might include a reference/addition/changes to the Master Test Plan (MTP) and/or a component's testing history
   (b) Relevant Documentation
   Provide references to documents necessary for compiling the MTR, including the: Acceptance Test Report (Section 4.2), System, Integration, and Unit Test Report (Section 4.3), and all other test reports created during V & V; and Software Development Artifacts (SDAs) and standards (e.g. IEEE Std 829-2008)
4. Details of the MTR
   "This section describes the overview of all aggregate test results, rational for any decisions, and the final conclusions and recommendations." (IEEE Computer Society, 2008, p. 67)
   (a) Overview of Aggregated Test Results
   Provide executive-level summaries of testing activities and tasks with references to the MTP (i.e. Section 3), categorized summaries of resolved and unresolved issues with references to reports (Section 4.4), summaries of collected metrics, and an overall assessment of EMgine's quality with justification
   (b) Decision Rationale
   Explain if EMgine passes or fails its V & V; if it conditionally passes, describe the qualifying restrictions and/or issues that must be resolved for EMgine to pass
   (c) Conclusions and Recommendations
   Describe the overall evaluation of EMgine, recommendations and circumstances for use, recommendations concerning its readiness for release, lessons learned during testing which might include the identification of issue categories and/or root cause analysis, and how to handle deferred issues

4.2 Acceptance Test Report

The Acceptance Test Report (ATR) provides an overview of the testing efforts made towards executing the "Acceptance Test Plan for EMgine: A Computational Model of Emotion for Enhancing Non-Player Character Believability in Games". The document's content and organization is a combination of IEEE Std 829-2008 Clause 15: Level Interim Test Report and 16: Level Test Report (IEEE Computer Society, 2008). The ATR must include:

1. Document Revision History
2. Reference Material (e.g. symbols, acronyms, definitions)
3. Introduction
   Identify who prepared the report, its purpose, and its current status (e.g. Draft, Final)
   (a) Scope
   Describe the contents and organization of the ATR and references to information captured by automated tools that are not contained in the ATR
   (b) Relevant Documentation
   Provide references to documents necessary for understanding the ATR, including the: Master Test Plan (MTP), Acceptance Test Plan (ATP), relevant Issue Reports (Section 4.4), and any additional relevant V & V documentation; and Software Development Artifacts (SDAs) and standards (e.g. IEEE Computer Society (2008))
4. Details of the ATR
   Preamble is "This section provides an overview of the test status and results, detailed test results, the status of issues related to these testing efforts, and any changes made from the Acceptance Test Plan (ATP).". If all testing is complete, append "...rationale for all decisions, and the final conclusions and recommendations."
   (a) Overview of Test Status and Results
   Provide a summary of the status of planned tests and results of testing efforts with references to the ATP, record the version and components of tested SDAs and the impact that the testing environment might have had on the results
   (b) Detailed Test Results
   Provide a summary of the testing activities and events, relevant metrics collected, variances of test items from their specifications and from tests to their documentation with explanations for the variances, and an evaluation of the comprehensiveness of testing efforts (e.g. coverage metrics)
   (c) Status of Related Issues
   Provide a summary of all related issues with references to their associated report (Section 4.4) with a description of their desired and actual status:
   - Resolved with a summary of their resolution
   - Unresolved ("Open" or "In Process" with a progress summary)
   - Deferred with explanations to address them

(d) Changes From Plans
Provide a summary of test activities that have not yet been executed with justification and describe any changes to testing activities such as tests that will not be done and tests that must be rerun

(e) Decision Rationale
When all testing activities are complete, Describe any issues considered during decision-making and justify the conclusions drawn from test results

(f) Conclusions and Recommendations
When all testing activities are complete, describe the overall evaluation of each test activity and its limitations, recommendations concerning their readiness for use and under what circumstances, and lessons learned during testing which might include the identification of issue categories and/or root cause analysis

4.3 System, Integration, and Unit Test Report

The System, Integration, and Unit Test Report (SIUTR) provides an overview of the testing efforts made towards executing the "System, Integration, and Unit Test Plan for EMgine: A Computational Model of Emotion for Enhancing Non-Player Character Believability in Games". The SIUTR must include all of the sections described for the Acceptance Test Report (ATR, Section 4.2) and relate its contents to the System, Integration, and Unit Test Plan (SIUTP) instead of the Acceptance Test Plan (ATP).

4.4 Issue Report

The Issue Report (IR) is for documenting any event that happens during testing that must be investigated. The document's content and organization is a combination of IEEE Std 829-2008 Clause 14: Anomaly Report (IEEE Computer Society, 2008). The ATR must include:

1. Document Revision History
2. Introduction
   Identify who prepared the report, its purpose, and its current status (e.g. Draft, Final)
   (a) Issue Identification Code/Number
   (b) Scope
   Describe any contextual information necessary for understanding the IR
   (c) Relevant Documentation
   Provide references to documents necessary for understanding the IR, such as a reference to a test plan and/or report
3. Details of the IR
   "This section identifies the items contained in the [IR] including its status and corrective actions taken." (IEEE Computer Society, 2008, p. 61)
   (a) Summary
   Provide a summary of the issue
   (b) Status of the Issue
   State the current status of the issue as one of:
   - Open
   - Assigned
   - Fixed
   - Retested with Fix
   - Resolved

   (c) Description of Issue
   Provide a description of the issue, sufficient conditions to reproduce it, which might include inputs, expected and actual outputs, unexpected outcomes, procedure step, environment, attempts to repeat the test, and the involved testers and/or observers, and any observations and/or activities that could help isolate and correct the issue
   (d) Context
   Identify the version and components of the Software Development Artifact (SDA) and/or Software Development Life Cycle (SDLC) stage the issue was discovered in; and the associated test level and items with their version level involved in the discovery
   (e) Date Discovered
   Record the date that the issue was first discovered
   (f) Impact
   If known, describe the breadth and depth of the issue's impact on affected SDAs and/or stakeholder needs, and any known workarounds for the issue

(g) Description of Corrective Action
Provide a summary of the activities taken to resolve the issue, which can include deferral or retirement of duplicate issue

(h) Conclusions and Recommendations
Describe the source/cause of the issue and/or provide recommendations for changes to development and/or testing processes to help prevent this issue in the future

5 References

Joost Broekens, Eva Hudlicka, and Rafael Bidarra. 2016. Emotional Appraisal Engines for Games. In *Emotion in Games: Theory and Praxis*, Kostas Karpouzis and Georgios N. Yannakakis (Eds.). Socio-Affective Computing, Vol. 4. Springer International Publishing, Cham, Switzerland, Chapter 13, 215–232. https://doi.org/10.1007/978-3-319-41316-7_13

Clark Elliott. 1998. Hunting for the Holy Grail with "Emotionally Intelligent" Virtual Actors. *ACM SIGART Bulletin* 9, 1 (June 1998), 20–28. https://doi.org/10.1145/294828.294831

IEEE Computer Society. 2008. *IEEE Standard for Software and System Test Documentation*. Technical Report IEEE Std 829-2008. IEEE, New York, NY, USA. https://doi.org/10.1109/IEEESTD.2008.4578383

IEEE Computer Society. 2009. *IEEE Recommended Practice for Software Requirements Specifications*. Technical Report IEEE Std 830-1998 (R2009). IEEE, New York, NY, USA. https://doi.org/10.1109/IEEESTD.1998.88286

IEEE Computer Society. 2017. *IEEE Standard for System, Software, and Hardware Verification and Validation*. Technical Report IEEE Std 1012-2016. IEEE, New York, NY, USA. https://doi.org/10.1109/IEEESTD.2017.8055462

ISO, IEC, and IEEE. 2015. *ISO/IEC/IEEE International Standard - Systems and software engineering – System life cycle processes*. Technical Report ISO/IEC/IEEE 15288:2015(E). IEEE, New York, NY, USA. https://doi.org/10.1109/IEEESTD.2015.7106435

Spencer Smith and Wen Yu. 2009. A document driven methodology for developing a high quality Parallel Mesh Generation Toolbox. *Advances in Engineering Software* 40, 11 (Nov. 2009), 1155–1167. https://doi.org/10.1016/j.advengsoft.2009.05.003

A Peer Review/Inspection Guides

These documents are intended as guides for reviewers in the peer review/inspection of EMgine's Software Development Artifacts (SDAs). Moderators are responsible for providing the guides to reviewers.

A.1 Master Test Plan Inspection Guide

This list is a guide for reviewers in the peer review of the Master Test Plan (MTP) document (Section 3.1). The list organizes prompts by MTP section in the order that they appear in the document. Reviewers are encouraged to include their own checks to this guide.

Introduction

- Does the introduction describe the purpose of EMgine's MTP?
- Does the summary of EMgine describe its intended purpose?
- Does the summary of EMgine list all of its required characteristics?
- Are EMgine's required characteristics consistent with its intended purpose?
- Are the aspects of EMgine and its software development artifacts (SDA) that are to be tested unambiguous?
- Are the aspects of EMgine and its SDAs that are *not* to be tested unambiguous?
- Are the testing objectives consistent with EMgine's intended purpose and characteristics?
- Are the objectives achievable?
- Is it clear how each plan relates to EMgine's Software Development Life Cycle (SDLC)?
- Does the relevant documentation list all of EMgine's documentation-based SDAs?
- Does the relevant documentation list all standards used to inform the MTP?

Plan Overview

- Is the list of team members complete?
- Are team member descriptions accurate with respect to their professional skills and proficiencies?
- Are the high-level goals assigned to each team member unambiguous?
- Are the high-level goals assigned to each team member consistent with their professional skills and proficiencies?
- Are the high-level goals assigned to each team member reasonable for each software development stage?
- Is there a plan for each of EMgine's SDAs?

- Are the verification methods reasonable for each SDA?
- Are the roles assigned to team members reasonable with respect to their professional skills, proficiencies, and schedule?
- Are the SDAs listed as necessary inputs for each plan unambiguous?
- Are the inputs to each verification method sufficient to complete it?
- Are the expected outcomes of each plan unambiguous?
- Are the outputs of each verification plan reasonable expectations?
- Is the expected completion time for each plan reasonable?

Readability

- Is the document well structured?
- Is the writing clear and cohesive?
- Is the writing concise?
- Do all abbreviations appear in "Symbols, Abbreviations and Acronyms" (Section 1)?
- Is the document free of spelling and grammar errors?

A.2 Acceptance Test Plan Inspection Guide

This list is a guide for reviewers in the peer review of the Acceptance Test Plan (ATP) document (Section 3.1). The list organizes prompts by ATP section in the order that they appear in the document. Reviewers are encouraged to include their own checks to this guide.

Introduction

- Does the introduction describe the purpose of EMgine's ATP?
- Does the summary of EMgine match that of Master Test Plan (MTP)?
- Are the aspects of EMgine that are to be tested by the ATP unambiguous?
- Are the aspects of EMgine that are *not* to be tested by the ATP unambiguous?
- Are the testing objectives consistent with the description in the MTP?
- Are the objectives achievable?
- Is it clear how the ATP impacts EMgine's development?
- Does the relevant documentation list all EMgine documentation relevant to the ATP?
- Does the relevant documentation list all standards used to inform the ATP?

Test Case Descriptions

- Is the purpose of the test cases unambiguous?
- Is the purpose of the test cases traceable to EMgine's concept summary?
- Is the purpose of the test cases traceable to EMgine's users and user needs?
- Are the inputs to each test case specific?
- Are the expected outputs of each test case explicitly defined?
- Is the purpose of each test case unambiguous?
- Are the methods for quantifying errors unambiguous?
- Are the methods for measuring qualitative data unambiguous?
- Are the test cases traceable to EMgine's concept summary?
- Are the test cases traceable to EMgine's users and user needs?
- Do the number and type of test cases seem sufficient?

User Study Descriptions

- Is the purpose of the user study unambiguous?
- Is the purpose of the user study traceable to EMgine's concept summary?
- Is the purpose of the user study traceable to EMgine's users and user needs?
- Are descriptions of the anticipated number and type of study participants unambiguous?
- Are the descriptions of study apparatuses complete?
- Are the descriptions of study apparatuses unambiguous?
- Are the descriptions of study procedures complete?
- Are the descriptions of study procedures unambiguous?
- Are the descriptions of independent variables complete?
- Are the descriptions of independent variables unambiguous?
- Are the descriptions of dependent variables complete?
- Are the descriptions of dependent variables unambiguous?
- Are descriptions of anticipated control, random, and confounding variables unambiguous?
- Is the user study replicable?
- Do the number and type of user studies seem sufficient?

A.3 System, Integration, and Unit Test Plan Inspection Guide

This list is a guide for reviewers in the peer review of the System, Integration, and Unit Test Plan (SIUTP) document (Section 3.1). The list organizes prompts by SIUTP section in the order that they appear in the document. Reviewers are encouraged to include their own checks to this guide.

Introduction

- Does the introduction describe the purpose of EMgine's SIUTP?
- Does the summary of EMgine match that of Master Test Plan (MTP)?
- Are the aspects of EMgine that are to be tested by the SIUTP unambiguous?
- Are the aspects of EMgine that are *not* to be tested by the SIUTP unambiguous?
- Are the testing objectives consistent with the description in the MTP?
- Are the objectives achievable?
- Is it clear how the SIUTP impacts EMgine's development?
- Does the relevant documentation list all EMgine documentation relevant to the SIUTP?
- Does the relevant documentation list all standards used to inform the SIUTP?
- Is the list of tools for automated testing and verification complete?
- Is the list of tools for automated testing and verification unambiguous?

System Test Description

- Are the inputs to each test case specific?
- Are the expected outputs of each test case explicitly defined?
- Is the purpose of each test case unambiguous?
- Are the methods for quantifying errors unambiguous?
- Are the methods for measuring qualitative data unambiguous?
- Are the test cases traceable to EMgine's requirements as described in the Software Requirements Specification (SRS)?
- Do the test cases represent a complete coverage of EMgine's requirements as described in the SRS? If not, does it explain why?

Integration Test Description

- Are the inputs to each test case specific?
- Are the expected outputs of each test case explicitly defined?
- Is the purpose of each test case unambiguous?
- Are the methods for quantifying errors unambiguous?
- Are the methods for measuring qualitative data unambiguous?
- Are the test cases traceable to the relationships between EMgine's modules as described in the the Module Guide (MG)?
- Do the test cases represent a complete coverage of the relationships between EMgine's modules as described in MG? If not, does it explain why?

Unit Test Description

- Are the inputs to each test case specific?
- Are the expected outputs of each test case explicitly defined?
- Is the purpose of each test case unambiguous?
- Are the methods for quantifying errors unambiguous?
- Are the methods for measuring qualitative data unambiguous?
- Are the test cases traceable to the constants, types, and/or access routines in EMgine's modules as described in the the Module Interface Specification (MIS)?
- Do the test cases represent a complete coverage of the constants, types, and/or access routines in EMgine's modules as described in the the MIS? If not, does it explain why?

Readability

- Is the document well structured?
- Is the writing clear and cohesive?
- Is the writing concise?
- Do all symbols, abbreviations, and acronyms in the SIUTP appear in the Reference Material?
- Is the document free of spelling and grammar errors?

A.4 SRS Inspection Guide

This list is a guide for reviewers in the peer review of the Software Requirements Specification (SRS) document (Section 3.2). The list organizes prompts by SRS section in the order that they appear in the document. Reviewers are encouraged to include their own checks to this guide.

Introduction

- Does the introduction describe the problem domain?
- Is the list of stakeholders complete?
- Is the description for each stakeholder accurate?
- Are the interests of each stakeholder accurate?
- Is the document's purpose described unambiguously?
- Is the document's purpose described accurately?
- Are the characteristics of the intended reader describing the readers of the SRS?
- Are the characteristics of the intended reader complete?
- Is the document accessible to its intended readers?
- Does the scope clearly state what EMgine will do?
- Does the scope state clearly what EMgine will not do?
- Is the description of the SRS's document structure accurate?

General System Description

- Is the list of entities in the system context clear?
- Is the list of entities in the system context complete?
- Are the responsibilities for each entity in the system context clear?
- Are the responsibilities for each entity in the system context complete?
- Is the system context diagram accurate?
- Are the user characteristics unambiguous?
- Are the user characteristics specific?
- Are the system constraints accurate?
- Are the system constraints complete?

Specific System Description: Problem Description

- Is the terminology and definitions list complete?
- Is the physical system description complete?
- Is the physical system description accurate?
- Are the goal statements abstract?
- Are the goal statements a refinement of the "Scope of the Requirements" section and/or stakeholder needs?
- Are the goal statements understandable by non-experts of the domain?

Specific System Description: Solution Characteristics Description

- Is the choice of affective theories/models traceable to the goal statements?
- Does each assumption address a single concern?
- Are the assumptions traceable to the "Scope of the Requirements" section and/or stakeholder needs?
- Is each assumption referenced at least once?
- Is the purpose of the Conceptual Models clear?
- Are the Theoretical Models refinements of the Conceptual Models?
- Is the list of General Definitions complete?
- Are derivations of General Definitions clear?
- Is the list of Data Definitions complete?
- Are derivations of Data Definitions clear?
- Are the Type Definitions abstract?
- Are the Type Definitions unambiguous?
- Are the Type Definitions referenced by at least one other model/type definition?
- Is it clear which Type Definitions users are responsible for defining?
- Are the Instance Models refinements of other models?
- Are the Instance Models abstract?
- Are the Instance Models clear?
- Are there links between each model/type definition and the models/type definitions they are referenced by?

- Are there links between each model/type definition and the models/type definitions they depend on?
- Are the Data Constraints complete?
- Are the Data Constraints unambiguous?
- Are the Properties of a Correct Solution complete?
- Are the Properties of a Correct Solution unambiguous?

Requirements

- Are the functional requirements abstract?
- Are the functional requirements testable?
- Are all functional requirements traceable to a stakeholder requirement?
- Do the nonfunctional requirements have objective fit criteria?
- Are the nonfunctional requirement fit criteria testable?
- Are all nonfunctional requirements traceable to a stakeholder requirement and/or user characteristic?

Future Changes

- Are the likely changes traceable to some component of the SRS?
- Is it feasible to hide the likely changes in a design?

Traceability Matrices and Graphs

- Are all necessary traceability matrices and graphs present?
- Are all necessary traceability matrices and graphs accurate?

Readability

- Does the document describe the problem domain?
- Is the document well structured?
- Is the writing clear and cohesive?
- Is the writing concise?
- Do all symbols, abbreviations, and notation in the SRS appear in the Reference Material?
- Is the document free of spelling and grammar errors?

A.5 Architecture Inspection Guide

This list is a guide for reviewers in the peer review of the Module Guide (MG) document (Section 3.3). The list organizes prompts by MG section in the order that they appear in the document. Reviewers are encouraged to include their own checks to this guide.

Introduction

- Does the introduction summarize the problem domain described in the SRS?
- Is the document's purpose described unambiguously?
- Is the document's purpose described accurately?
- Are the characteristics of the intended reader describing the readers of the MG?
- Are the characteristics of the intended reader complete?
- Is the document accessible to its intended readers?
- Is the description of the MG's document structure accurate?

Anticipated and Unlikely Changes

- Are the likely changes from the SRS associated with at least one anticipated change?
- Are the unlikely changes justified?

Connection Between Requirements and Design

- Is the justification for the proposed architecture unambiguous?
- Is the justification for the proposed architecture traceable to stakeholder needs as described in the SRS?
- Is the justification for the proposed architecture traceable to one or more functional and/or nonfunctional requirements from the SRS?

Module Hierarchy and Decomposition

- Do the modules satisfy the principle of information hiding?
- Is there one secret per module? If not, is it justified?
- Are the behaviour-hiding modules related to requirements in the SRS?
- Are the software decision modules related to concepts that are not in the SRS?
- Are the software decision modules necessary to the design?

Traceability Matrices

- Is every functional requirement from the SRS satisfied by at least one module?
- Is every module used to satisfy at least one functional requirement from the SRS?
- Is there one anticipated change per module? If not, is it justified?

Use Hierarchy Between Modules

- Is the use hierarchy complete?
- Is the use hierarchy accurate?
- Is the use hierarchy hierarchical?
- Does the hierarchy satisfy the principle of low coupling between modules?

Readability

- Is the document well structured?
- Is the writing clear and cohesive?
- Is the writing concise?
- Do all symbols, abbreviations, and notation in the MG appear in the Reference Material?
- Is the document free of spelling and grammar errors?

A.6 Design Inspection Guide

This list is a guide for reviewers in the peer review of the Module Interface Specification (MIS) document (Section 3.4). The list organizes prompts by MIS section in the order that they appear in the document. Reviewers are encouraged to include their own checks to this guide.

Introduction

- Does the introduction summarize the problem domain described in the SRS?
- Does the introduction have references/links to the SRS and MG?

Notation

- Is the notation unambiguous?
- Is the notation complete?
- Is the notation used consistently throughout the MIS?

Module Hierarchy

- Does the module hierarchy match the MG?
- Is there a specification for each module in the module hierarchy?

Module Specifications

- Does the introductory content describe the purpose and key characteristics of the module?
- Are the modules classified correctly (e.g. Interface, ADT, Generic)?
- Are all exported constants defined with a literal value?
- Are the assumptions justified?
- Do all access routines do something (i.e. output or state transition)?
- Are all operations type safe?
- Are all state invariants initially satisfied?
- Are all state invariants satisfied when an access routine terminates?
- Do local functions make the specification easier to understand?
- Does each module satisfy the principle of high cohesion?

Readability

- Is the document well structured?
- Is the writing clear and cohesive?
- Is the writing concise?
- Do all symbols, abbreviations, and notation in the MIS appear/have a reference to a definition in the Reference Material?
- Is the document free of spelling and grammar errors?

A.7 Implementation Inspection Guide

This list is a guide for reviewers in the peer review of the source code and source code documentation (Section 3.5). The list includes prompts for reviewing test case documentation, source code, and source code documentation. Reviewers are encouraged to include their own checks to this guide.

Test Case Documentation

- Is the documentation's purpose described unambiguously?
- Is the documentation's purpose described accurately?
- Is the scope of the test case verification accurate?
- Are the types of test cases used accurate?
- Are the types of test cases used sufficient?
- Are automated testing and/or verification tools documented?
- Do the test cases have specific inputs?
- Do the test cases have explicitly defined expected outputs?
- Is every functional requirement from the SRS satisfied by at least one system-level test case?
- Is there at least one test case where a set of modules is tested as an integrated unit such that only one new module is added to the set in a single test case?
- Are there sufficient test cases for each module?

Source Code and Documentation

- Does each source code file have a header listing the:
  - Last modified date
  - Programming language and its version
  - Imported libraries and/or components and their version
  - Operating System and its version
  - License
  - Author(s)
- Does each function have a header describing its purpose, inputs, and outputs?
- Are the coding standards of the implementation language used consistently throughout the source code and its documentation?
- Are constants named consistently?
- Are constants given symbolic names?

- Are variables named consistently?
- Are functions named consistently?
- Are source code files organized consistently with respect to each other?
- Are source code files formatted consistently with respect to each other?
- Is the source code as simple as possible?
- Are modifications to the source code clearly documented?
- Is the source code documentation complete/traceable to the source code?
- Is the source code documentation unambiguous?

Readability

- Is the code/document well structured?
- Is the writing clear and cohesive?
- Is the writing concise?
- (If applicable) Do all symbols, abbreviations, and notation in the document appear/have a reference to a definition in the Reference Material?
- Is the code/document free of spelling and grammar errors?