

FLUMINENSE FEDERAL UNIVERSITY

DOUGLAS PAULO DE MATTOS

**An Approach for Authoring Mulsemmedia Applications  
Based on Events**

NITERÓI

2021

FLUMINENSE FEDERAL UNIVERSITY

DOUGLAS PAULO DE MATTOS

# An Approach for Authoring Mulsemmedia Applications Based on Events

Thesis presented to the Computing Graduate Program of Fluminense Federal University in partial fulfillment of the requirements for the degree of Doctor of Science. Topic Area: Computer Science.

Advisor:

Profa. Débora Christina Muchaluat Saade, D.Sc.

Co-advisor:

Prof. Gheorghita Ghinea, PhD

NITERÓI

2021

Ficha catalográfica automática - SDC/BEE  
Gerada com informações fornecidas pelo autor

D278a De mattos, Douglas Paulo  
An Approach for Authoring Mulsemmedia Applications Based on  
Events / Douglas Paulo De mattos ; Débora Christina Muchaluat-  
Saade, orientadora ; Gheorghita Ghinea, coorientador.  
Niterói, 2021.  
174 f. : il.

Tese (doutorado)-Universidade Federal Fluminense, Niterói,  
2021.

DOI: <http://dx.doi.org/10.22409/PGC.2021.d.13640792750>

1. Sistemas multimidia. 2. Multimidia (Ciencia da  
computacao). 3. Multimidia interativa. 4. Produção  
intelectual. I. Muchaluat-Saade, Débora Christina,  
orientadora. II. Ghinea, Gheorghita, coorientador. III.  
Universidade Federal Fluminense. Instituto de Computação.  
IV. Título.

CDD -

Douglas Paulo de Mattos

An Approach for Authoring Multimedia Applications Based on Events

Thesis presented to the Computing Graduate Program of Fluminense Federal University in partial fulfillment of the requirements for the degree of Doctor of Science. Topic Area: Computer Science.

Approved in September, 2021.

APPROVED BY



Prof. D.Sc. Débora Christina Muchaluat Saade - Advisor, UFF



Prof. PhD Gheorghita Ghinea - Co-advisor, Brunel University London



Prof. PhD Célio Vinicius Neves de Albuquerque, UFF



Prof. D.Sc. Igor Monteiro Moraes, UFF



Prof. D.Sc. Marcelo Ferreira Moreno, UFJF



Prof. PhD Niall Murray, Athlone Institute of Technology

Niterói

2021



# Resumo

O conceito de mulsemídia (*MulSeMedia - Multiple Sensorial Media*) tem sido explorado para oferecer aos usuários novas sensações usando outros sentidos além da visão e audição. A demanda para a produção de tais aplicações tem motivado diversos estudos na fase de autoria mulsemídia. Alguns estudos propõem modelos mulsemídia que fornecem efeitos sensoriais em uma abstração de baixo nível por meio da identificação de sensores e atuadores. No entanto, esta abordagem não é adequada para sistemas mulsemídia, uma vez que não especifica o comportamento espacial e temporal do conteúdo multimídia e dos efeitos sensoriais em uma abstração de alto nível. Outros estudos que integram efeitos sensoriais com conteúdo audiovisual usam o paradigma baseado em linha do tempo como seu modelo de sincronização temporal, o qual possui várias limitações inerentes. Além disso, as aplicações mulsemídia raramente exploram conteúdo visual 360°.

Esta tese propõe uma abordagem para a autoria de aplicações mulsemídia baseada em eventos. Esta abordagem tem como objetivo aprimorar o desenvolvimento de aplicações mulsemídia e auxiliar o desenvolvimento de ambientes de autoria gráfica baseados em visão temporal para usuários sem conhecimentos de programação. Esta abordagem envolve a proposta de um modelo conceitual mulsemídia (MultiSEM - *Multimedia Sensory Effect Model*), uma linguagem baseada em XML (MultiSEL - *Multimedia Sensory Effect Language*), e um ambiente gráfico baseado em visão temporal para a autoria de aplicações mulsemídia (STEVE 2.0 - *Spatio - Temporal View Editor*) projetado para usuários sem conhecimento de programação.

MultiSEM representa os efeitos sensoriais como entidades de primeira classe (nós do documento). O modelo permite, assim, a integração espaço-temporal e a sincronização dos efeitos sensoriais com o conteúdo multimídia. Além disso, MultiSEM define entidades para especificar propriedades de apresentação de nós de mídia e características de renderização de efeitos sensoriais. O modelo proposto usa o paradigma de sincronização temporal baseado em eventos. Em relação ao MultiSEL, a linguagem define elementos baseados nas entidades do MultiSEM e visa facilitar a autoria de aplicações mulsemídia com conteúdo visual 360° para ambientes de realidade virtual. Também permite a especificação de aplicações tradicionais multimídia sem conteúdo 360° e/ou efeitos sensoriais. Além disso, o MultiSEL visa apoiar a interoperabilidade entre ferramentas de autoria mulsemídia.

Como prova de conceito, o modelo MultiSEM foi implementado no editor STEVE 2.0, que permite aos autores especificar aplicações mulsemídia 2D usando relações temporais baseadas em eventos para sincronizar a mídia tradicional e os efeitos sensoriais. STEVE 2.0 também permite a definição de propriedades de apresentação de mídia e propriedades de renderização de efeito sensorial. Além disso, ele se integra com o middleware Ginga-NCL estendido para efeitos sensoriais exportando projetos STEVE para documentos NCL 4.0.

Com os experimentos de STEVE, foi avaliada sua usabilidade, recursos específicos e

experiência do usuário. Além disso, demonstrou-se que STEVE permitiu que usuários sem habilidades de programação criassem suas aplicações mulsemídia. Foram apresentadas integrações do STEVE com outras tecnologias mulsemídia em direção de propostas de plataformas mulsemídia completas. Essas plataformas podem ser vistas como ferramentas fim-a-fim que atuam nas fases de produção, distribuição e renderização de aplicações mulsemídia.

**Palavras-chave:** Modelagem de Efeitos Sensoriais, Aplicações Mulsemídia, Modelo Conceitual, Ambientes de Autoria Mulsemídia, Aplicações Mulsemídia com conteúdo audiovisual 360°.

# Abstract

The mulsemmedia (MulSeMedia - Multiple Sensorial Media) concept has been explored to provide users with new sensations using other senses beyond sight and hearing. The demand for producing such applications has motivated various studies in the mulsemmedia authoring phase. Some studies propose mulsemmedia models that provide sensory effects at a low-level abstraction by identifying sensors and actuators. However, this approach is not suitable for mulsemmedia systems since it does not specify the spatial and temporal behavior of multimedia content and sensory effects at a high-level abstraction. Other studies integrating sensory effects with audiovisual content use the timeline-based paradigm as their temporal synchronization model, with several inherent limitations. In addition, mulsemmedia applications rarely explore 360° visual content.

This thesis proposes an approach for authoring mulsemmedia applications based on events. It aims to enhance the development of mulsemmedia applications and support the development of graphical authoring environments based on temporal views for users with no programming skills. This approach involves the proposal of a mulsemmedia conceptual model (MultiSEM - Multimedia Sensory Effect Model), an XML-based language (MultiSEL - Multimedia Sensory Effect Language), and a graphical environment based on temporal view for authoring mulsemmedia applications (STEVE 2.0 - Spatio-Temporal View Editor) designed for users with no programming skills.

MultiSEM represents sensory effects as first-class entities (document nodes). The model thus allows the spatio-temporal integration and synchronization of sensory effects with multimedia content. Additionally, MultiSEM defines entities to specify presentation properties of multimedia nodes and rendering characteristics of sensory effects. Moreover, the proposed model uses the event-based temporal synchronization paradigm. Regarding MultiSEL, the language defines elements based on MultiSEM's entities and aims at enhancing the authoring of mulsemmedia applications with 360° visual content for virtual reality environment. It also allows the specification of traditional multimedia applications with no 360° content and/or sensory effects. In addition, MultiSEL aims at supporting the interoperability between mulsemmedia authoring tools.

As proof of concept, we have implemented MultiSEM in STEVE 2.0, which allows authors to specify 2D mulsemmedia applications using event-based temporal relations to synchronize traditional media and sensory effects. STEVE 2.0 also allows the definition of media presentation properties and sensory effect rendering properties. Moreover, it integrates with the Ginga-NCL middleware extended for sensory effects by exporting STEVE projects to NCL 4.0 documents.

With STEVE's experiments, we analyzed its usability, specific features, and user experience. Also, we demonstrated that STEVE allowed users with no programming skills to create their mulsemmedia applications. Additionally, we present STEVE's integrations with other mulsemmedia technologies on the need for mulsemmedia platforms. Those platforms

can be seen as end-to-end tools that act in the production, distributing, and rendering phases of mulsemmedia applications.

**Keywords:** Modeling Sensory Effects, Mulsemmedia Applications, Conceptual Model, Mulsemmedia Authoring Environment, 360° mulsemmedia applications.

# List of Figures

3.1	Mulsemmedia Models Overview . . . . .	30
3.2	Mulsemmedia Tools Overview . . . . .	40
4.1	MultiSEM Node Representation . . . . .	47
4.2	Presentation Property Entity . . . . .	50
4.3	Fade-in and Fade-out Times in MultiSEM . . . . .	50
4.4	Sensory Effect Presentation Property . . . . .	51
4.5	MPEG-V Spatial Model . . . . .	52
4.6	Media Node Presentation Property . . . . .	54
4.7	Attributes of Crop, Position and Size Entities . . . . .	55
4.8	MultiSEM Events . . . . .	56
4.9	Event State Machine . . . . .	57
4.10	Relation and Connector Class Diagram . . . . .	58
4.11	Role Class Diagram . . . . .	60
4.12	Constraint Glue Class Diagram . . . . .	62
4.13	Causal Glue Class Diagram . . . . .	63
4.14	NCM 3.0 Node Diagram . . . . .	68
5.1	Intensity Spatial Attenuation Concept (scent device [56]) . . . . .	76
5.2	Scene 1 Structural View . . . . .	85
5.3	Body Structural View . . . . .	87
5.4	Scene 2 Structural View . . . . .	88
6.1	Structural View 1 [74] . . . . .	90
6.2	After User Interaction [74] . . . . .	92

6.3	Structural View 2 [74]	94
6.4	STEVE 2.0 architecture highlighting the modifications against STEVE 1.0	96
6.5	STEVE 2.0 Data Flow	97
6.6	Graphical Interface of STEVE 2.0	99
6.7	STEVE Event-based Timelines for Our Mulsemmedia Application	100
6.8	STEVE's <i>New User Interaction</i> Window	100
6.9	STEVE's Integrations	102
7.1	STEVE's Temporal View for Task 1 and 2	107
7.2	STEVE's Temporal View for Task 3	107
7.3	Directed Graph for our Goals, Questions and Metrics	113
7.4	Mean Value for $G1$ , $G2$ , and $G3$ Questions	114
7.5	Acceptability and Adjective Scales associated with raw SUS Score	116
7.6	Mean Value per UEQ Scales	117
7.7	Mean Value per UEQ Group	117
E.1	Mean Value per UEQ Item	157
E.2	Distribution of Answers per UEQ Item	158

# List of Tables

2.1	Features of Multimedia Authoring Tools . . . . .	24
3.1	Features of Mulsemedia Authoring Tools . . . . .	38
4.1	State Machine Transitions [116, 117] . . . . .	57
4.2	Temporal Relations of the Part 1 of MultiSEM's Predefined Connector Base	65
4.3	Modeling Difference between MultiSEM and NCM . . . . .	71
5.1	MultiSEL's Functional Areas and Modules . . . . .	73
5.2	<i>Structure</i> Functional Area . . . . .	79
5.3	<i>Metainformation</i> Functional Area . . . . .	79
5.4	<i>Components</i> Functional Area . . . . .	80
5.5	<i>Presentation and Rendering Specification</i> Functional Area . . . . .	81
5.6	<i>Interfaces</i> Functional Area . . . . .	81
5.7	<i>Relation</i> Functional Area . . . . .	82
7.1	STEVE's Experiment Goals . . . . .	105
7.2	Questions for G1 Goal . . . . .	108
7.3	Metrics for G1 Questions . . . . .	108
7.4	Questions for G2 Goal . . . . .	109
7.5	Metrics for G2 Questions . . . . .	109
7.6	Questions for G3 Goal . . . . .	109
7.7	Metrics for G3 Questions . . . . .	110
7.8	SUS Questionnaire for STEVE . . . . .	110
7.9	UEQ's Questionnaire . . . . .	112

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivation . . . . .	2
1.2	Thesis Statements . . . . .	3
1.3	Goals . . . . .	5
1.4	Thesis Contributions . . . . .	7
1.5	Thesis Structure . . . . .	7
<b>2</b>	<b>Multimedia Authoring</b>	<b>9</b>
2.1	Modeling: Temporal Synchronization Paradigms . . . . .	9
2.1.1	Script-based Paradigm . . . . .	10
2.1.2	Timeline-based Paradigm . . . . .	10
2.1.3	Graph-based Paradigm . . . . .	11
2.1.4	Hierarchy-based or Structure-based Paradigm . . . . .	12
2.1.5	Constraint-based Paradigm . . . . .	13
2.1.6	Event-based Paradigm . . . . .	13
2.2	Authoring GUI Approaches . . . . .	14
2.2.1	Textual View Editing . . . . .	14
2.2.2	Structural View Editing . . . . .	15
2.2.3	Layout View Editing . . . . .	15
2.2.4	Spatial View Editing . . . . .	15
2.2.5	Temporal View Editing . . . . .	16
2.2.6	Wizard Approach . . . . .	16



---

2.2.7	Form-based Interface . . . . .	16
2.3	Authoring Tools . . . . .	16
2.3.1	Academic Tools . . . . .	16
2.3.2	Commercial Tools . . . . .	20
2.4	Desirable Features for Multimedia Authoring Tools . . . . .	21
2.5	Comparison of Multimedia Authoring Tools . . . . .	23
<b>3</b>	<b>Related Work</b>	<b>25</b>
3.1	Mulsemmedia Modeling . . . . .	25
3.1.1	MPEG-V . . . . .	25
3.1.2	NCL 4.0 . . . . .	27
3.1.3	Guedes et al. . . . .	27
3.1.4	ASAMPL . . . . .	28
3.1.5	Comparison of Mulsemmedia Models . . . . .	29
3.2	Mulsemmedia Authoring Tools . . . . .	29
3.2.1	CrowdMuse . . . . .	30
3.2.2	H-Studio . . . . .	31
3.2.3	Sang-Kyun Kim et al. . . . .	31
3.2.4	SEVino . . . . .	32
3.2.5	RoSE . . . . .	32
3.2.6	SMURF . . . . .	32
3.2.7	Real 4D Studio . . . . .	33
3.2.8	MulSeMaker . . . . .	33
3.2.9	FeelReal . . . . .	34
3.3	Desirable Features for Mulsemmedia Authoring Tools . . . . .	34
3.4	Comparison of Mulsemmedia Authoring Tools . . . . .	37
3.5	Mulsemmedia Simulators and Players . . . . .	40

3.5.1	SESim - Sensory Effect Simulator . . . . .	41
3.5.2	SEMP - Sensory Effect Media Player . . . . .	41
3.5.3	Sensible Media Simulator . . . . .	41
3.5.4	Sensorama . . . . .	42
3.5.5	Multimedia Multisensorial 4D Platform . . . . .	42
3.5.6	Real 4D Studio Simulator . . . . .	42
3.5.7	PlaySEM SER 2 . . . . .	43
3.5.8	Josué et al. . . . .	44
3.6	Gaps of Authoring Tools . . . . .	44
<b>4</b>	<b>MultiSEM - Multimedia Sensory Effect Model</b>	<b>46</b>
4.1	Nodes . . . . .	47
4.2	Presentation Properties . . . . .	49
4.2.1	Sensory Effect Properties . . . . .	51
4.2.2	Media Properties . . . . .	53
4.3	Events . . . . .	55
4.4	Relations . . . . .	58
4.4.1	Role . . . . .	60
4.4.2	Glue . . . . .	62
4.4.3	Predefined Connector Base . . . . .	64
4.4.3.1	Part 1: Allen's Temporal Relations . . . . .	64
4.4.3.2	Part 2: Interactivity Relations . . . . .	66
4.4.3.3	Part 3: Synchronounous Relations with Attribution Action . . . . .	66
4.4.3.4	Part 4: Relations with Statement Assessment . . . . .	67
4.5	Comparison . . . . .	68
4.5.1	MultiSEM vs NCM . . . . .	68
4.5.1.1	Nodes . . . . .	68

4.5.1.2	Composition: <i>Context</i> . . . . .	69
4.5.1.3	Composition: <i>Switch</i> . . . . .	69
4.5.1.4	Content Anchors . . . . .	69
4.5.1.5	Presentation Properties . . . . .	70
4.5.2	MultiSEM vs MPEG-V . . . . .	70
<b>5</b>	<b>MultiSEL - An XML-based Language for 360° Mulsemmedia Applications</b>	<b>72</b>
5.1	<i>Structure</i> Functional Area . . . . .	73
5.2	<i>Metainformation</i> Functional Area . . . . .	73
5.3	<i>Components</i> Functional Area . . . . .	73
5.4	<i>Presentation and Rendering Specification</i> Functional Area . . . . .	74
5.5	<i>Interfaces</i> Functional Area . . . . .	76
5.6	<i>Relations</i> Functional Area . . . . .	76
5.7	Language Structure . . . . .	78
5.8	Case Study . . . . .	83
<b>6</b>	<b>STEVE 2.0 - An Authoring Environment for Mulsemmedia Applications</b>	<b>89</b>
6.1	Case Study . . . . .	89
6.2	STEVE 2.0 . . . . .	93
6.2.1	STEVE 2.0 Architecture . . . . .	95
6.2.2	STEVE 2.0 Data Flow . . . . .	97
6.2.3	Graphical User Interface . . . . .	98
6.2.4	Temporal View . . . . .	99
6.3	STEVE's Integrations: on the Need for Mulsemmedia Platforms . . . . .	101
6.4	Final Remarks . . . . .	102
<b>7</b>	<b>Evaluation</b>	<b>104</b>
7.1	Methodology . . . . .	104

7.1.1	Tasks . . . . .	106
7.2	Questionnaires . . . . .	107
7.2.1	G1 Questions . . . . .	107
7.2.2	G2 Questions . . . . .	108
7.2.3	G3 Questions . . . . .	109
7.2.4	SUS Questionnaire for G4 . . . . .	110
7.2.5	UEQ Questionnaire for G5 . . . . .	111
7.3	Results . . . . .	111
7.3.1	G1 Analysis . . . . .	113
7.3.2	G2 Analysis . . . . .	114
7.3.3	G3 Analysis . . . . .	115
7.3.4	SUS Score for G4 . . . . .	115
7.3.5	UEQ Analysis Tool for G5 . . . . .	115
7.4	Final Remarks . . . . .	118
<b>8</b>	<b>Conclusions</b>	<b>119</b>
8.1	Contributions . . . . .	119
8.2	Answering the Research Questions . . . . .	121
8.3	Limitations . . . . .	122
8.4	Publications . . . . .	123
8.5	Future Directions . . . . .	124
	<b>References</b>	<b>126</b>
	<b>Appendix A - MultiSEM's Connector Base</b>	<b>136</b>
A.1	Connectors for Allen's Temporal Relations . . . . .	136
A.2	Connectors for Interactivity Relations . . . . .	138
A.3	Connectors for Synchronounous Relations with Attribution Action . . . . .	140

---

A.4 Connectors for Relations with Statement Assessment . . . . .	141
<b>Appendix B - Complete UML Diagram of MultiSEM</b>	<b>143</b>
<b>Appendix C - MultiSEL's XML Schema</b>	<b>147</b>
C.1 Functional Areas . . . . .	147
C.2 <i>Structure</i> Functional Area . . . . .	147
C.3 <i>Metainformation</i> Functional Area . . . . .	148
C.4 <i>Components</i> Functional Area . . . . .	149
C.5 <i>Presentation and Rendering Specification</i> Functional Area . . . . .	150
C.6 <i>Interfaces</i> Functional Area . . . . .	150
C.7 <i>Relations</i> Functional Area . . . . .	151
<b>Appendix D - MultiSEL Document Example</b>	<b>153</b>
<b>Appendix E - UEQ Results</b>	<b>156</b>

# Chapter 1

## Introduction

Multimedia applications are available on different devices such as smartphones, computers, tablets, and TV sets. Not only a massive volume of media content is consumed, but ordinary users also produce it. That content has only audiovisual media, which involves only two human senses, sight and hearing.

However, most human communication is non-verbal, and most of us use all five senses (sight, hearing, touch, taste, and smell) to comprehend our world. In addition, we can feel internal body changes, which are called interoceptive capabilities [64]. We can classify those capabilities in the following categories [64]: equilibrioception, sense of balance; thermoception, sense of heat and cold; proprioception, awareness of the position of our body or part of it; nociception, sense of pain; and interoception, the capacity of feeling the internal organs.

All those sensory stimuli have been explored by the concept of MulSeMedia (Multiple Sensorial Media) [62]. This concept provides users with new sensations by exploring other senses beyond sight and hearing in interactive multimedia applications. In addition, the use of multiple sensory effects has also provided users with new immersive content increasing their quality of experience (QoE) [127, 102, 131, 90].

We can create mulsemedia applications using light, wind, and scent, for instance. As an example of an application with multiple sensory effects, we have 4D cinemas, where motion chairs synchronize with the movies' audiovisual content. Indeed, mulsemedia technology is moving from highly specialized niches to the mainstream, with devices becoming increasingly affordable [104]. Another domain that evidences this transition is that of digital games. In this respect, mulsemedia applications can provide an immersive environment to increase the game reality and players' QoE. In the entertainment indus-

try, sensory effects are also applied to simulators (e.g., flight and driving) to make the simulation experience more realistic.

Also, mulsemmedia applications has a great potential in the health area, specifically in therapeutic use for people with special needs (e.g., learning disabilities, autism, Alzheimer’s disease, and dementia) [62]. Several works have presented how sensory effects can improve memory, learning process [101], and selective and supportive attention in people with autism [55]. The work discussed in [124] presents an immersive environment that makes use of virtual reality to investigate neurological influence in healthy patients during physical activity and to compare with the technique of motion representation [121], used in therapies (e.g., *Snoezelen* multisensory rooms [32]). In [95], the authors present an interactive environment that provides visual, sonorous, and vibrational stimuli to promote creativity, exploration, and fun in children with autism. If the user experiences a negative behavior, the intelligent environment responds by reducing the production of stimuli. Otherwise, the environment increases its complexity by generating more sensory stimuli.

All of these applications aim at providing immersive environments to increase the user QoE in multimedia applications. Several studies [78, 48, 65, 34, 104] have been published in the literature discussing solutions that integrate, to varying degrees, sensory effects with multimedia applications.

## 1.1 Motivation

Concerning the workflow of mulsemmedia applications, we can subdivide it into three phases [41]: authoring (or production), distribution, and rendering of applications in the physical environment. In the authoring phase, which represents the focus of this thesis, sensory effects can be defined through digital capturing and processing of data obtained from sensors, automatic extraction of sensory effects from audiovisual contents, and manual specification by authors. Furthermore, we can combine the manual specification of authors with a crowdsourcing approach. In [44], users give suggestions about time intervals in which specific sensory effects could be activated according to audiovisual contents provided by mulsemmedia application authors. That approach supports authors in enhancing the specification of time intervals in which sensory effects should be rendered with audiovisual contents.

Regarding advances in mulsemmedia application modeling in the authoring phase, studies in [65, 105] provide sensory effects at a low-level abstraction by identifying sensors and

actuators. This approach is frequently used in IoT (Internet of Things) solutions supporting the inherent heterogeneity of devices in IoT environments. However, this approach is not suitable for mulsemmedia systems [104] since it does not allow the specification of the spatial and temporal behavior of multimedia content and sensory effects at a high-level abstraction as in traditional multimedia model approaches [114, 67].

Another solution for representing sensory effect metadata is the MPEG-V standard [78], which uses the timeline-based paradigm to synchronize sensory effects with existing multimedia content. Additionally, other proposals focus on enhancing the specification of sensory effect metadata and the annotation of multimedia content with sensory effects by using graphical tools as in [126, 34, 77]. However, these initiatives use the timeline-based paradigm, which has several inherent limitations [25]. Moreover, these solutions do not allow authors to specify an entire mulsemmedia application by defining media items and effects' spatial and temporal behavior. Another approach is that described in [50], which makes use of templates and wizards. However, that solution makes the tool less expressive by restricting authors to the set of predefined applications available in the tool. Moreover, the authors are not able to modify the behavior of these applications.

Investigating the challenge of modeling immersive environments with multiple sensory effects is essential for advancing mulsemmedia applications. In [41], authors highlight the demand for tools that enhance mulsemmedia application development. They also emphasize the importance of the temporal synchronization of sensory effects for supporting the effectiveness of mulsemmedia applications. In addition to multisensory-related studies, mulsemmedia applications rarely explore 360° visual content. However, new technologies such as HMDs (Head-Mounted Displays) and omnidirectional cameras [132] have enabled the large production, distribution and consumption of 360° content for virtual reality environments. Accordingly, the main research question we are addressing in this thesis is:

- *How can we enhance the authoring phase of interactive multimedia applications with multiple sensory effects?*

## 1.2 Thesis Statements

We make the following statements and present secondary questions in order to answer our main research question.

1. As the first step in enhancing the mulsemmedia application development, it is es-



essential to provide a mulsemmedia conceptual model to represent the spatio-temporal behavior of mulsemmedia documents in a structured fashion. In addition, that model should support the specification of document nodes (media and sensory effects), providing entities to represent them and their presentation characteristics. Moreover, conceptual models are crucial to define mulsemmedia languages and graphical authoring tools as in the multimedia domain. Chapter 2 supports that statement in the multimedia area, and, in Chapter 3, we keep arguing it for the mulsemmedia domain, giving a literature review of studies that explore the mulsemmedia modeling and graphical authoring tools. Indeed, in Chapter 4, we propose a novel mulsemmedia conceptual model to answer the following question derived from that statement.

- *What are the necessary entities a mulsemmedia conceptual model should define in order to support the development of graphical authoring environments based on temporal view for mulsemmedia applications?*
2. Defining a declarative mulsemmedia language, based on a conceptual model, allows authors to write mulsemmedia documents specifying their spatio-temporal behavior. Also, it allows distributing those applications in structured documents supporting the interoperability between mulsemmedia systems and formatters. That statement is also supported by our study of multimedia and mulsemmedia modeling and tools in Chapter 2 and 3 respectively. Additionally, we propose a new declarative mulsemmedia language supporting 360° visual content based on the proposed model in Chapter 5. Based on that statement, we also question the following. We aim to answer that question in Chapter 5.
- *What are the necessary elements a declarative mulsemmedia language should have to support the writing of mulsemmedia documents with 360° visual content with that can be exchanged between different mulsemmedia systems?*
3. We also state that providing an authoring tool enhances the development of mulsemmedia applications by abstracting not only the complexity of specifying mulsemmedia documents textually but also the challenges of synchronizing sensory effects in the physical environments, in which we have to deal with rendering and dissipation delays of sensory effects, for example, scent and temperature effects. We argue that statement giving a literature review of graphical authoring tools for multimedia and mulsemmedia applications in chapters 2 and 3. In addition, we propose a graphical authoring environment for mulsemmedia applications in Chapter 6 and evaluate its

usability and user experience in Chapter 7. We derive from that statement the following questions. Chapter 3 answers the first and second questions and Chapter 7 answers the third one.

- *What are the desirable features for mulsemedia authoring tools in order to enhance the development of mulsemedia applications even by users with no knowledge of multimedia or mulsemedia modeling and standard languages?*
- *What are the gaps of the mulsemedia authoring tools presented in our literature review?*
- *Can users with no knowledge of multimedia or mulsemedia modeling and standard languages create their mulsemedia applications using a graphical authoring tool?*

## 1.3 Goals

According to our statements and questions above, this thesis aims to enhance the development of mulsemedia applications and support the development of graphical authoring environments based on temporal views for users with no programming skills by proposing an approach for authoring mulsemedia applications based on events.

This approach involves three proposals. The first proposal is a mulsemedia conceptual model, called MultiSEM (Multimedia Sensory Effect Model) [83, 48], for representing mulsemedia applications, arguing for *Statement 1*. The second proposal, arguing for *Statement 2*, is an XML-based language, MultiSEL (Multimedia Sensory Effect Language), based on MultiSEM for specifying 360° mulsemedia applications. We also propose a graphical environment for authoring mulsemedia applications, named STEVE 2.0 [48, 83], based on MultiSEM and designed for users with little or no knowledge of programming. We argue for *Statement 3* proposing STEVE 2.0.

MultiSEM represents sensory effects as mulsemedia application nodes, i.e., the model represents them as first-class entities. Therefore, sensory effects can also be synchronized temporarily with other nodes, whether traditional media or sensory effects. This representation of sensory effects with a high-level abstraction was initially proposed in [74]. Our model is based on the NCM (Nested Context Model) [116, 117] model and the MPEG-V standard proposing a simplification of NCM and integrating MPEG-V's sensory effect vocabulary into MultiSEM's sensory effect representation. That NCM simplification aims

at enhancing the development of mulsemmedia authoring environments based on temporal views.

The model also defines entities to specify the presentation properties of multimedia nodes and rendering characteristics of sensory effects. Additionally, MultiSEM uses the event-based temporal synchronization paradigm. This paradigm is applied to several proposals of authoring tools and multimedia models [47, 66, 96, 116, 117, 52]. It also uses the concept of hypermedia connector [91] to represent spatial and temporal relations. Furthermore, our proposal defines a set of causal connector instances that represent causal temporal relations based on Allen's time interval relations [11].

Regarding MultiSEL, the language defines elements based on MultiSEM entities. It aims at enhancing the authoring of mulsemmedia applications with 360° visual content for virtual reality environment. MultiSEL uses a declarative approach, which follows the paradigm of other multimedia languages such as HTML [38], NCL (Nested Context Language) [103] and SMIL (Synchronized Multimedia Integration Language) [37]. The language also allows the specification of traditional multimedia applications with no 360° content and/or sensory effects. In addition, MultiSEL aims at supporting the interoperability between mulsemmedia authoring tools.

As proof of concept, we have implemented MultiSEM in STEVE 2.0 (Spatio-temporal View Editor) [48, 83]. This authoring environment allows authors to specify 2D mulsemmedia applications using event-based temporal relations to synchronize traditional media and sensory effects. STEVE 2.0 also allows the definition of media presentation properties and sensory effect rendering properties. Also, it provides an integration with the Ginga-NCL middleware extended for sensory effects by exporting STEVE projects to NCL 4.0 documents [73]. Ginga-NCL [103, 2] is an NCL presentation engine to provide interoperability and harmonization among IPTV multimedia application frameworks. It is also a middleware for the Brazilian digital TV system. To integrate with the multisensory-extended Ginga-NCL, STEVE 2.0 exports mulsemmedia applications to NCL 4.0 documents, a new extension of NCL 3.0 for sensory effects [73].

Among existing mulsemmedia solutions, our approach is unique in that it explores the application level proposing a novel mulsemmedia conceptual model that aims to enhance the development of graphical tools for authoring mulsemmedia applications, a graphical authoring tool itself, and an XML language for 360° mulsemmedia applications for virtual reality.

## 1.4 Thesis Contributions

The contributions of the approach for authoring mulsemmedia applications based on events proposed in this thesis are the following:

1. An event-based conceptual model, named MultiSEM, for authoring mulsemmedia documents with sensory effects as first-class entities and enhancing the development of graphical authoring environments.
2. An XML-based language, called MultiSEL, for specifying 360° mulsemmedia documents based on MultiSEM.
3. A graphical authoring environment for 2D mulsemmedia applications, named STEVE 2.0, designed for users with no programming knowledge.
4. The study of several multimedia and mulsemmedia authoring environments and models encouraging researchers to explore new solutions for the mulsemmedia authoring phase [85].
5. The integration of STEVE 2.0 with an extended GINGA-NCL for sensory effects by exporting STEVE applications to NCL 4.0 documents.
6. The integration of MultiSEL with a mulsemmedia authoring tool in virtual reality (VR), named AMUSEVR.
7. The analysis of the STEVE's usability with authoring experiments highlighting challenges we have found for the mulsemmedia authoring phase.

## 1.5 Thesis Structure

We organize this thesis in the following chapters. Chapter 2 gives a multimedia background to support our study in the mulsemmedia field. We cite several non-recent references regarding traditional multimedia modeling and authoring since this field is a classic topic. We aim to bring back basic concepts of multimedia modeling and the multimedia community's contributions concerning the investigation of authoring tools over the years.

In Chapter 3, related work is addressed. It discusses several mulsemmedia solutions such as conceptual models, declarative languages, framework, and authoring tools. This

chapter also presents a set of features mulsemmedia authoring tools should provide to enhance the mulsemmedia production phase.

Chapter 4 presents the MultiSEM model. We discuss its entities to represent media and sensory effects nodes in mulsemmedia documents and the presentation properties MultiSEM defines for each type of media and sensory effects. In addition, the chapter discusses how MultiSEM synchronizes those items based on events. Moreover, a comparison between MultiSEM and the models on which it is based is given.

In Chapter 5, we discuss MultiSEL elements and their attributes divided into six functional areas. For each area, we explain the elements semantic, show code snippets, and summarize the elements and attributes in tables to give an overview of the language structure. Furthermore, we present a case study to show how MultiSEL represents an entire 360° mulsemmedia application.

Chapter 6 presents an implementation of MultiSEM in STEVE 2.0. We also describe a mulsemmedia application as a case study to show how MultiSEM represents it and how authors can use STEVE 2.0 to create it graphically. STEVE's architecture, data flow, and features are also depicted in this chapter.

Chapter 7 presents STEVE's experiments in order to evaluate its usability, specific features, and user experience. The chapter describes the methodologies we have used, the questionnaires for collecting users' feedback, and the analyses of the results.

Conclusion and future work are given in Chapter 8.

# Chapter 2

## Multimedia Authoring

This chapter presents a multimedia background to support our study in the mulsemedia field in the next chapter. We cite several non-recent references regarding traditional multimedia modeling and authoring since the multimedia field is a classic topic. We aim to bring back basic concepts of multimedia modeling and the multimedia community's contributions concerning the investigation of authoring tools over the years.

Therefore, we discuss studies regarding multimedia modeling and tools for authoring multimedia applications. Also, we present the temporal synchronization paradigms that underlie different multimedia conceptual models. Moreover, this chapter highlights the advantages and disadvantages of those paradigms and gives a comparison among them. Afterward, it discusses different authoring GUI (graphical user interface) approaches and authoring tools and presents some requirements for comparing them. Those requirements are also essential to support our discussion about mulsemedia authoring tools in the next chapter. We finalize this chapter by giving a comparison table of the characteristics of each multimedia authoring tool discussed.

### 2.1 Modeling: Temporal Synchronization Paradigms

Multimedia (or hypermedia) documents are composed of media items and spatio-temporal relationships among them. These document components are expressed through multimedia conceptual models, whose main entities are represented by nodes, links, and composite nodes. The temporal relationships among media items can be specified using different temporal synchronization paradigms. The main paradigms [116, 117, 30, 67, 68] discussed in the multimedia community are: script, timeline, graph, hierarchy/structure, constraint and event-based.

### 2.1.1 Script-based Paradigm

The script-based paradigm uses imperative programming languages to define the temporal and spatial behavior of multimedia documents. This paradigm is consequently more flexible and expressive than other paradigms. However, it requires programming knowledge from authors. The script paradigm also makes the spatio-temporal visualization harder as it does not provide a high-level abstraction to deal with the document structure. Video-book [94] and Harmony [59] are examples of tools for authoring hypermedia documents that use multimedia models based on the script paradigm.

In [17], a multimedia synchronization toolkit, called Nsync, is proposed. The toolkit does not provide an authoring tool but consists of a declarative synchronization definition language and a run-time presentation management system. Although this language is declarative, it uses scripts to specify the temporal layout of multimedia applications.

Adobe Animate [6] is a multimedia authoring and computer animation tool that also uses a script language, ActionScript 3.0 [5], to define the temporal and spatial behavior of multimedia applications. The tool provides a script window in which authors can add complex interactivity to applications using the ActionScript language.

In the web scenario, HTML5 [38] is also an example of a declarative authoring language using the script paradigm. Although it introduces new elements for improving multimedia content support, mainly video and audio, HTML5 uses ECMAScript code to provide more expressive multimedia documents for the web.

### 2.1.2 Timeline-based Paradigm

The timeline-based paradigm is effective when media object duration is known a priori or explicitly defined since this paradigm directly arranges media objects in a temporal axis. On the other hand, it is inappropriate to represent media object temporal order when multimedia presentations have asynchronous behavior as happens in hypermedia. Conditional synchronization cannot be directly defined among media items either. If any object duration is changed, the author needs to reorganize media items in time to keep the specified temporal dependencies. Using the timeline-based paradigm does not allow a formatter (multimedia player or presentation engine) to make runtime adjustments in case of a media network reception delay. In other words, multimedia presentations lose their temporal synchronization when that happens. Besides, content adaptation and timeless compositions cannot be expressed directly in a timeline representation. Despite those

limitations, this paradigm is widely used in commercial software and is very simple to be understood by non-expert authors. Section 2.3 presents some of these tools.

### 2.1.3 Graph-based Paradigm

The graph-based paradigm uses formalism for defining document synchronization. This paradigm thus takes advantage of several formal models available in the literature. We identify two dominant formal techniques: the flowchart model [129] and the directed graph model [29]. The latter type is mainly represented by Petri nets [100] and the statechart model [49]. The flowchart model is a chart that describes media items behavior step by step using graphical elements. These elements represent actions and decision points linked through arrows to specify the presentation control flow. This technique is similar to imperative programming to define the document's behavior but uses visual elements instead.

Timed Petri nets [100] are bipartite directed graphs that contain the following components: places, transitions, and arcs. Each place has tokens and a duration. Arcs are used to connect places and transitions. A transition is fired when all of its input places contain tokens. After a transition is triggered, those tokens are moved to their output places. Tokens then remain blocked until the place duration finishes. HTSPN [128], Trellis [60] and caT [92] are multimedia models and tools that use this synchronization paradigm. This approach has the advantage of providing formal verification of document temporal behavior.

Another formal definition based on directed graph models is the statechart-based paradigm, which uses statecharts to specify media items and temporal relations. In this chart, states represent media items and transitions define a state hierarchy. Using transitions, authors can define if media items are presented at the same time or not. In [49], the authors propose HMBS (Hypermedia Model Based on Statecharts). The model uses the structure and execution semantics of statecharts to define the browsing semantics and the structural organization of multimedia documents.

The graph-based paradigm requires knowledge of those formal notations to specify the behavior of multimedia documents. Moreover, the task can become complex when authors need to define temporal relations among media segments (anchors) because each anchor must be defined as a different vertex in the graph. In addition, it makes user interaction specification harder than other paradigms such as the event-based one, as will be discussed in Section 2.1.6.



### 2.1.4 Hierarchy-based or Structure-based Paradigm

As far as the structure-based paradigm [31, 123, 27, 51, 82, 61] is concerned, it uses compositions to define temporal relationships among media items. There are different composition types: sequential, parallel, and atemporal. All items that are part of a sequential composition are presented sequentially. On the other hand, those that are part of a parallel composition are presented simultaneously. The atemporal composition [67] is defined as a grouping of components with no associated temporal relations, which will be presented when the composition is activated at runtime by a temporal composition. The SMIL *excl* element [15] is an example of atemporal composition. Using these compositions, authors can define document synchronization as a tree of temporal containers.

In the structure-based paradigm, authors can also define a delay for media items and synchronization constraints. Such constraints can be defined in attributes of media assets or composition nodes, or external structures, as CMIF's (CWI Multimedia Interchange Format) sync arcs [123, 67]. The structure-based CMIF model consists of nested presentations, events, and channels. *Event* refers to a fragment of media data and a channel groups events of the same media type and their properties. The structure-based paradigm provides a simple synchronization model in which authors can create a storyboard using the document structure. Also, this paradigm allows authors to put the structure along a timeline as in CMIFed [123].

The concepts introduced by CMIF were applied to the SMIL language [37]. Thus, SMIL and MPEG-4 XMT authoring languages [58] are also based on the hierarchical paradigm. The hypermedia model presented in [109] and the ZYX model [26] also use compositions to define temporal relations as a tree model representation.

As with the graph-based paradigm, relations among anchors are also hard to represent in the structure-based approach because a whole media item (and not only one of its anchors) must be included in a composition. For multimedia documents with several nodes and temporal relations, the document specification can become complex with several nested compositions. The hierarchical structure is also limited concerning representing synchronization conditions composed of more than one type of event or state. Events can refer to the presentation of media items, which can be in different states, such as occurring (media is playing), sleeping (media is stopped) or paused; and to user interactions. To overcome this limitation, SMIL State [72] extends the hierarchy-based synchronization providing state variable manipulation.

### 2.1.5 Constraint-based Paradigm

The constraint-based paradigm specifies temporal constraints classified into two categories: reference point-based and interval-based synchronization. The latter defines the duration of an item as an interval. The interval synchronization is defined based on 13 relations between intervals discussed in [11]. Another approach can be based on 29 interval relations proposed in [125]. As in the hierarchical-based paradigm, the constraint-based synchronization is limited for defining user interaction and temporal relations among anchors. Madeus [76] and MPGS (Multimedia Presentation Generator System) [23] are authoring systems whose temporal synchronization model makes use of the interval-based paradigm.

Regarding point-based synchronization, it uses reference points to define the temporal synchronization among media items. These points can be the beginning and the end of an item or anchor. As a result, reference point-based synchronization is more flexible than interval-based synchronization. Firefly [29] is an example of an authoring system that uses a reference point-based model. The system has a hybrid synchronization model since it also uses the graph-based paradigm.

### 2.1.6 Event-based Paradigm

The event-based paradigm is based on event occurrences during multimedia application execution to define temporal relations among media items. Events can be classified into different types: presentation, selection, attribution, and preparation. While the first one refers to synchronous events (starting, stopping, and pausing a media item presentation), the selection event represents user interaction. Moreover, this paradigm defines the attribution event for changing variable values. In addition, it presents the preparation event [75] for buffering part of media content in the player to avoid delays when media presentation starts. The event-based paradigm is less intuitive than the timeline-based paradigm for authoring. As in the structure-based paradigm, multimedia documents with several media items and event-based relationships can complicate the document specification.

However, the event-based approach is very expressive for defining temporal relationships among media items [25]. This approach is advantageous when multimedia applications are designed to be delivered on the web since media items may suffer delays due to server and network load. This approach is also easily extended to new synchronization types [25], such as events with duration. The event-based paradigm also easily handles

asynchronous events, such as user interactions and variable tests, which cannot be represented using the timeline-based paradigm. Hypermedia models that use the event-based paradigm are NCM (Nested Context Model) [116, 117], on which NCL (Nested Context Language) [103] is based, Labyrinth [52], and SIMM (Simple Interactive Multimedia Model) [47]. NEXT [96] and Composer [66] are multimedia authoring tools that are based on the NCM model. The STEVE editor [47] is based on the SIMM model. SMIL 2.0 [15] also uses event-based synchronization to specify a media object *begin* and *end* attributes based on a different media object begin or end time.

## 2.2 Authoring GUI Approaches

The study presented in [30] provides an authoring paradigm taxonomy that is not based on temporal synchronization models, as we discussed in Section 2.1. The authors instead base their approach on user interfaces for authoring multimedia documents. That is, they use the term *authoring paradigm* to refer to the GUI (graphical user interface) approach used by authoring tools to display the multimedia document structure to users.

In contrast, in this work, we do not present multimedia authoring tools according to this classification due to the following reasons. First, this taxonomy does not take into account that graphical user interfaces (GUI) approaches (that is, authoring paradigms according to [30]) may not directly represent the underlying temporal synchronization paradigm of the conceptual multimedia model that supports tool implementations. The second reason is that an authoring tool GUI may offer more than one authoring paradigm regardless of the underlying temporal synchronization paradigm. Third, our goal is to concentrate on features presented in multimedia authoring tools [30, 87] to support our discussion regarding multimedia authoring tools.

Therefore, we analyze authoring tools highlighting their temporal synchronization paradigm and authoring GUI approach separately. For the authoring tools of Section 2.3, this thesis indicates the temporal synchronization paradigm according to Section 2.1. Regarding authoring GUI approaches, we classify them in textual, structural, layout, spatial, temporal view editing, wizard approach, and form-based interface.

### 2.2.1 Textual View Editing

The textual view editing corresponds to directly writing multimedia documents using a standard multimedia language or script. The tools that follow this approach [96, 66, 27,

51, 76, 94, 59] provide a text editor integrated with other views. That is, the text editor reflects its changes on the other views and vice versa. The textual view editing fits the model whereby authors with programming skills need to specify spatial and temporal behaviors that they cannot express using the graphical interface provided.

### 2.2.2 Structural View Editing

The structural approach refers to giving an editable graphical representation of media items and the relationships among them. This representation can be given using a tree, a set of compositions, or graphs. The multimedia authoring tools SMIL Builder [27] and NEXT [96] use a tree representation for giving a structural view, but only SMIL Builder allows editing the tree. NEXT also provides an editable graph view that allows users to specify nodes and links. GRiNs [31] and SMILAuthor2 [129] use compositions (sequential and parallel) according to the discussion of Section 2.1.4. The remaining tools in Table 2.1 that provide structural view use graph representation.

### 2.2.3 Layout View Editing

Regarding the layout view editing, it allows users to edit the regions where media items are presented. It is important to highlight that this view only presents the beginning positions and dimensions of media items. However, those media properties can be changed during a multimedia presentation, for example, using NCL links [103]. The Composer [66] and NEXT [96] authoring tools provide the layout view editing of NCL documents. MediaTouch [82] also gives this editing ability allowing users to define presentation characteristics of MHEG-5 Scenes.

### 2.2.4 Spatial View Editing

In contrast to layout view editing, the spatial approach allows authors to visualize and specify the presentation properties of media items for each time instant. To this end, authoring tools [47, 117, 123, 51, 76, 61] provide the spatial view integrated with the temporal view.

### 2.2.5 Temporal View Editing

Authoring tools [47, 117, 51, 76] provide temporal view editing to allow users to manipulate media items on the time axis. In other words, this view graphically presents the temporal order that media items are presented and their duration using a rectangle-based representation. In this view, authors can specify the beginning/end time instant of media items and their duration by manipulating these rectangles.

### 2.2.6 Wizard Approach

The wizard approach guides users through a sequence of steps that allows them to input information in a systematic way to produce a multimedia application. The NEXT editor [96] is the only one in Section 2.3 that offers this approach. It displays several windows in sequence, guiding users to choose a multimedia application template and select their media content to compose it.

### 2.2.7 Form-based Interface

Like the wizard approach, the form-based interface allows users to input all the information to specify the spatial and temporal behavior by filling empty fields. However, the form-based approach does not guide users through a step-by-step process. From the tools discussed in Section 2.3, only MPGS [23] uses this GUI approach.

## 2.3 Authoring Tools

This section gives an overview of multimedia authoring tools available in scientific papers to identify a set of features for enhancing multimedia authoring. We also cite some commercial tools to support our analysis. In Section 2.4, we summarize all these features. Then the next section presents a table associating the authoring tools discussed with these features.

### 2.3.1 Academic Tools

*STEVE* [47] is a spatio-temporal editor for authoring interactive multimedia documents. It is based on its event-based model called SIMM (Simple Interactive Multimedia Model). The model gives a causal interpretation to Allen's temporal relations [11]. STEVE thus

provides a temporal view that allows authors to synchronize media items using these temporal relations graphically. In addition, STEVE allows users to specify media presentation properties and verify how and where media items will be displayed during document execution through a spatial view. The editor also provides the definition and simulation of user interaction events. Moreover, it gives authors feedback regarding temporal consistency. The editor can not only export multimedia applications to NCL documents but also HTML5 files. In fact, STEVE translates NCL documents to HTML5 using NCL4WEB [112].

*NEXT (NCL Editor supporting XTemplate)* [96] is also a graphical editor for authoring NCL applications. It allows users with no knowledge of multimedia authoring languages to create NCL documents using templates. To this end, NEXT uses composite templates defined in XTemplate [53]. However, creating new templates is a hard task for non-programmers. As in Composer, users must also know NCM entities in order to use all features provided by NEXT.

*Composer* [66] is another NCL document authoring tool that provides different document views. However, the last version of Composer [16] does not offer a temporal view as previously supported [66]. Since the NCM model directly underlines the Composer's graphical environment, users must know some NCM entities, a non-trivial task. Authors should use the structural view to specify their documents by manipulating icons and rectangles representing NCM nodes and links.

*HyperProp* [117] is a hypermedia system that consists of an authoring tool and a formatter. The event-based NCM model [116] underlies the system. Therefore, the HyperProp authoring tool provides a structural view for specifying links among nodes based on events. Icons (content nodes) or rectangles (composite nodes) and lines represent NCM nodes and links, respectively. The tool also offers a temporal view integrated with the structural one. Moreover, it provides a spatial view allowing authors to define spatial relationships among nodes and their presentation properties. Authors can also receive feedback regarding temporal and spatial consistency.

*CMIFed* [123] provides three document views that give graphical representations of the CMIF's elements. The hierarchy/structural view represents the nested presentations and events using nested sets of boxes to express temporal compositions. The channel view provides a visualization of time that flows from top to bottom, separated by channels. However, the editor does not allow users to edit this temporal view. CMIFed also provides a player that allows the editing of layout aspects of presentations. Users can also preview

a presentation or part of it directly from the hierarchy and channel views. Moreover, the editor allows users to debug presentations by playing and checking when events are active in the channel view.

*GRI**Ns* [31] is an extension of CMIFed by implementing navigation facilities. It consists of an authoring and presentation system for SMIL [37] documents. The system uses a temporal synchronization model based on parallel and sequential compositions. Its structural view shows nested rectangles to represent SMIL compositions that give authors temporal synchronization. However, this representation may become difficult to be understood when applications have several temporal compositions nested on many levels. The tool also provides a timeline view but only for visualizing.

*SMIL Builder* [27] is also an editor for creating SMIL documents with incremental verification based on a hierarchical SMIL Petri Net model. The tool offers textual and structural views to edit SMIL documents. The structural view allows the specification of SMIL documents using a tree structure. The textual and temporal views display any modification in the structural one. Regarding the temporal view, it does not allow editing and is only for visualization. Moreover, it does not display the media items directly on the time axis. Instead, it uses graphical elements to represent the Petri-net-based model that underlies SMIL Builder.

*SMILAuthor 2.0* [129] is another tool for exporting SMIL documents. It provides a structural view using compositions as well and supports non-deterministic temporal behavior. Moreover, SMILAuthor 2.0 provides a layout view to edit spatial relationships among media items. The editor also offers a limited presentation preview, in which users can only preview part of the presentation by specifying the preview duration.

*LimSee2* [51] is a SMIL authoring tool that provides both spatial and temporal views. The spatial view allows users to edit SMIL regions by moving and resizing media items. In the temporal view, authors can synchronize media items by dragging and resizing them to define their start/end time and duration. However, the tool does not support user interaction in the document definition. Besides, *LimSee2* requires a basic knowledge of the hierarchical model to synchronize media items temporally.

*FireFly* [29] is an authoring tool that uses a constraint-based model and presents a structural view based on graphs. It also uses events to synchronize media items. However, this structural view may make temporal behavior verification difficult since media items are not directly placed in a time axis. This tool neither provides spatial view editing nor does it generate an application using a standardized format.

*Madeus* [76] is a multimedia authoring tool that allows spatial and temporal editing using a constraint-based specification. The tool provides a graphical temporal view in which users can check the temporal order of media items and the constraint relations among them. Authors can also edit this view by moving or resizing a media item along the horizontal axis. However, the editor requires users to create a multimedia document using the *Madeus* textual format before editing it through the graphical views. In the presentation view, authors can play a document, pause it and edit the spatial position of media items.

*MPGS* [23] is a multimedia presentation system that is underpinned by a constraint model. The system consists of two environments for specifying and generating the presentation. The specification environment allows users to define temporal and spatial constraints between two objects using a form-based approach. The system also performs spatial and temporal consistency checks during the authoring phase, and afterward, users complete the specification. Moreover, *MPGS* uses strategies to generate presentations when spatial and temporal constraints specified by users cannot be satisfied at run-time.

*MediaTouch* [82] is an authoring tool for creating MHEG-5 [54] objects. It provides structural and layout view editing. *MediaTouch* uses a structure-based paradigm for synchronizing media items and offers interactivity support through the Link Editor interface. The tool also provides a player.

*Gaggi and Celentano* [61] propose a tool that allows users to create multimedia presentations with parallel and sequential synchronization and hyperlinks for navigation. It offers structural view editing by using a tree representation and a graph view. It also provides a spatial layout view and a preview. However, neither a standard language is used to export applications, nor a player is provided.

*Videobook* [94] is a prototype hypermedia system that uses a script-based model to specify the presentation layout and display timing. The system is composed of a database, editors, and a player. The database stores the scripts associated with each model component, such as media, triggers (buttons), and scene (set of media and triggers). The system provides the scene Editor, which is the center of authoring. This editor provides a graphical representation of media alongside triggers over time and space according to their scripts. *Videobook* thus gives only a presentation visualization and does not allow users to specify the multimedia presentation graphically. To specify the scripts, *Videobook* provides a text editor. The system also offers a player that creates a schedule table using the scene scripts.



*Harmony* [59] is a multimedia presentation system that provides a database to store multimedia content and an authoring environment, the Harmony user interface. The interface allows users to create media nodes and links among them through a script-based interface. It also supports interactivity events to trigger links and displays the structure of multimedia documents in a tree graph, which illustrates the temporal synchronization among the media. However, this tree cannot be edited in order to specify the behavior of multimedia applications.

*I-HTSPN* [128] provides a graphical interface for creating multimedia documents based on Timed Petri nets. To build nets, users can graphically create places, transitions, and arcs. Authors can also specify attributes associated with these elements using dialog windows. In addition, a player is provided to run MHEG multimedia applications. The editor has also been implemented to produce Java multimedia applications.

*Trellis* [60] provides a graphical editing client and two other clients for displaying text and graphics images content in two independent windows. The editor represents multimedia applications using a Timed Petri-net-based view. With this view, users can edit the net structure through graphical elements that illustrate each component of the net, such as place, transition, and token.

*caT* [92] extends *Trellis* [60] in order to support context-aware documents that respond to environment changes, such as time, location and bandwidth/cost. To do that, *caT* provides user modeling, high-level Petri-net specification, and fuzzy knowledge. The fuzzy logic engine is invoked to infer good values from uncertain user contexts. The *caT* editor provides multiple subnets status in the simulation node to aid users in checking the document presentation. The *caT* system also features an analysis tool that helps authors verify the document's behavior and offers an interactive debugging tool.

### 2.3.2 Commercial Tools

There are also commercial software solutions that allow users to create video presentations, such as Final Cut Pro [14], Adobe Premiere [8], Davinci Resolve [24], Adobe Director [7], and Nero Video 2021 [9]. They use the timeline-based paradigm as discussed in Section 2.1 and are designed for video editing professionals. Thus, they offer many features and menus that make the creation process hard for ordinary users. These commercial tools do not produce hypermedia applications written in a standard multimedia authoring language either. Usually, they encode multimedia applications in video file formats, such as MP4 and MOV.

## 2.4 Desirable Features for Multimedia Authoring Tools

From the discussion regarding both academic and commercial authoring tools, we identified features that need to be provided by multimedia authoring tools to enhance the multimedia production phase. We do not include structural view editing in those features since this editing approach may turn the authoring complex when multimedia applications have several media items and relationships among them. The proposed features are as follows:

- *Temporal View Editing*: tools need to provide a temporal view for presenting the temporal behavior of media items. The items have to be directly placed on the time axis so that users can verify when media items are presented and their duration. Additionally, tools must allow users to edit the temporal view by changing the duration and beginning/end time instant of media items.
- *Spatial View Editing*: a spatial view must be provided by tools for presenting how and where media items are visually presented during execution. This view needs to be synchronized with the temporal view to allow users to check the spatial view for each instant of the multimedia presentation. Tools also need to provide a user interface to allow authors to define media properties such as size and position graphically.
- *Interactivity Support*: authors need to be able to define interactivity relations. Those relations refer to user interactions using input devices (e.g., keyboard, mouse, remote control, microphone, camera, touch screen, and eye tracker) with a media item. In other words, authors must be able to define interactive media with which users can interact. Also, actions must be triggered from that user interaction.
- *Presentation Preview*: authors need to visualize the temporal and spatial behavior of their multimedia applications during the authoring phase before publishing them. This visualization should give a graphical representation of the application layout over time according to the temporal synchronization. In the case of interactive multimedia applications, the preview may not provide user interaction. In other words, the preview may be a simulation and not the application execution itself.

- *Ordinary User Support*: tools should allow authors with no knowledge of multimedia language or model to create interactive multimedia applications. For providing that, tools should offer a graphical user interface approach that can guide authors in their production phase.
- *Template Support*: templates can be defined as generic structures that specify the spatio-temporal behavior of multimedia documents but without defining media content. Tools should provide templates so that users only need to complete a template definition with media content to produce whole multimedia applications.
- *Standard Publishing Format*: authors should be able to export their multimedia applications to documents using international standard multimedia languages, such as HTML5 and NCL (SMIL W3C Group closed its activities in 2012 [63]). This feature is more flexible than exporting an application to a video file, where users cannot interact with the content.
- *Content Delivery Analysis*: the analysis of performance regarding the delivery of multimedia content is essential in distributed multimedia systems. Even in the authoring phase, tools should provide authors with feedback on the delivery performance of media items added in their applications. Additionally, tools may also provide optimization methods for different types of media content delivery.
- *Error Analysis*: authors should have feedback about inconsistent spatial or temporal behaviors when they create or edit applications. This feature is handy in order to avoid execution errors [106] after applications are distributed to final users. Authoring tools based on formal models, such as the ones based on the graph-based paradigm discussed in Section 2.1.3, have advantages considering this feature.
- *Player*: tools need to provide an embedded multimedia player or to be integrated with an external player so that authors can fully interact with their multimedia applications.

## 2.5 Comparison of Multimedia Authoring Tools

Table 2.1 provides a summary for multimedia authoring tools discussed in Section 2.3.1 regarding the features presented in Section 2.4. In addition to those features, the table also indicates the temporal synchronization paradigm of each tool according to the paradigms discussed in Section 2.1. Concerning the temporal and spatial view editing feature, they are represented in the column *Authoring GUI Approach* using the words *temporal* and *spatial*. That column indicates the tools' graphical user interface approach for allowing authors to create multimedia applications graphically. The authoring approach types are in keeping with the discussion of Section 2.2.

These multimedia authoring tools that we discussed cater for traditional audiovisual-based multimedia. However, olfactory, wind, and thermal effects in mulsemedia applications are characterized by a new set of characteristics such as wafting and lingering, which affect intensity [89, 10, 62] in contrast to traditional media (intensity of audio volume or image brightness). Therefore, new requirements raised by mulsemedia applications are insufficiently catered by legacy multimedia editors. The following chapter discusses several studies that focus on the mulsemedia authoring phase to support us in identifying this new set of features to be provided by mulsemedia authoring tools.

Table 2.1: Features of Multimedia Authoring Tools

Feature / Tool	Temporal Synchronization Paradigm	Authoring GUI Approach	Interactivity Support	Presentation Preview	Ordinary User Support	Template Support	Publishing Formats	Content Delivery Analysis	Error Analysis	Player
<b>STEVE</b> [84]	event-based	temporal spatial	✓	✓	✓		NCL HTML5		✓	✓
<b>NEXT</b> [96]	event-based	layout structural textual wizard	✓		✓	✓	NCL			
<b>Composer</b> [66]	event-based	layout structural textual	✓				NCL			✓
<b>HyperProp</b> [117]	event-based	structural temporal spatial	✓				NCM HTML		✓	✓
<b>CMIFed</b> [123]	structure-based	spatial structural	✓	✓			CMIF			✓
<b>GRiNs</b> [31]	structure-based	structural	✓	✓			SMIL			✓
<b>SMIL Builder</b> [27]	petri-net-based	structural textual					SMIL		✓	
<b>SMILAuthor2</b> [129]	petri-net-based	structural layout	✓	+/-			SMIL			
<b>Limsee2</b> [51]	structure-based	temporal spatial textual		✓			SMIL			✓
<b>FireFly</b> [29]	event constraint -based	structural	✓				own format			✓
<b>Madeus</b> [76]	constraint-based	textual temporal spatial	✓	✓			Madeus SMIL		✓	✓
<b>MPGS</b> [23]	constraint-based	form-based					own format	✓	✓	✓
<b>MediaTouch</b> [82]	structure-based	structural layout	✓				MHEG- 5		✓	✓
<b>Gaggi</b> [61]	structure-based	structural spatial	✓	✓			own format		✓	✓
<b>Videobook</b> [94]	script-based	textual	✓	✓			own format			✓
<b>Harmony</b> [59]	script-based	textual	✓				own format			✓
<b>I-HTSPN</b> [128]	graph-based	structural	✓				MHEG Java		✓	✓
<b>Trellis</b> [60]	graph-based	structural	✓				own format		✓	✓
<b>caT</b> [92]	graph-based	structural	✓	✓			own fmt. HTML		✓	✓

✓ : fully supported feature; +/- : partially supported feature

# Chapter 3

## Related Work

This chapter addresses studies that present proposals for supporting mulsemmedia production. Additionally, it discusses several graphical environments for authoring mulsemmedia applications. The chapter also discusses simulators and players for mulsemmedia applications.

In addition to supporting the thesis's proposal, this chapter aims to encourage researchers to explore new solutions for the mulsemmedia authoring phase and stimulate the production of mulsemmedia applications [85].

### 3.1 Mulsemmedia Modeling

In this section, we highlight studies that support the representation of sensory effects and their characteristics. Those proposals involve conceptual mulsemmedia models, programming frameworks, description and programming languages to contribute to the mulsemmedia production phase.

#### 3.1.1 MPEG-V

The MPEG-V standard [78] defines a set of XML-based elements for specifying real-world objects, such as sensors, actuators, and virtual world objects. These elements aim at standardizing the data exchange between both worlds. MPEG-V defines *Control Information Description Language* (CIDL) to represent metadata related to actuator and sensor capability, and sensory effect and sensor adaptation preference of users. The standard also provides the Sensory Effect Description Language (SEDL) for describing sensory effects. An SEDL file, called Sensory Effect Metadata (SEM), annotates multimedia content with

sensory effects. To define the temporal synchronization between sensory effects and audiovisual contents, MPEG-V uses a timeline-based paradigm in which each effect has its beginning and ending synchronized with specific audiovisual content. That temporal model has several well-known limitations [25], mainly concerning the user interaction, as discussed in Section 2.1.2.

The standard also allows the specification of sensory effect intensity fade-in and fade-out in time, defining when effects reach their maximum intensity and null value, respectively. The intensity specification of sensory effects is essential for authoring mulsemmedia applications since it allows authors to define how strong sensory effects are rendered in the physical environment [74]. Additionally, the specification of fade-in and fade-out for sensory effects allows authors to define an intensity variation over time, increasing the quality of experience (QoE) of mulsemmedia applications.

Concerning the spatial model of MPEG-V, sensory effects have the *location* attribute. This attribute specifies the region where effects should be perceived by users that are immersed in the mulsemmedia content. The spatial model considers the user the central point of reference, and the effect location is defined using x, y, and z axes.

Furthermore, the SEDL language provides elements for grouping effects so that authors can create complex effects, for instance, an explosion that may include wind, light, and vibration effects. The language also allows authors to reuse sensory effects that are frequently used by declaring and referring them. Those sensory effects (SEM files) should be transformed into actuator commands in the real world to render them using a virtual world to real-world (VR) adapter, which is out of the scope of MPEG-V.

In addition, MPEG-V allows the description of virtual objects classified in general objects and avatars. For both types, the standard defines properties, such as identity, audio resource, scent, movement controls, and input events, to allow those objects to be controlled by the real-world inputs and used by different virtual worlds. Note that MPEG-V only defines virtual object properties and does not define their geometric shape, animation, and texture. To support these last specifications, MPEG-4 [97] may be integrated with MPEG-V.

With IIDL (*Interaction Interface Description Language*), MPEG-V defines two vocabularies: *Device Command Vocabulary* (DCV) and *Sensed Information Vocabulary* (SIV). DCV allows the description of actuator commands to render sensory effects in the real world. On the other hand, SIV is responsible for modeling information captured by sensors.

Although it provides a set of XML-based languages, MPEG-V does not support the definition of an entire mulsemedia application as can be done with our work, NCL 4.0 [73], and Guedes' framework [65]. In other words, MPEG-V cannot represent multiple media types, such as video, image, and text, and synchronize each of them with several sensory effects in time and space. Indeed, MPEG-V only specifies temporal annotations of sensory effects for a single video or audio.

### 3.1.2 NCL 4.0

NCL 4.0 [73] is a new extension of NCL 3.0 [103] for sensory effects. This extension follows the proposal presented in [74] by modeling sensory effects as first-class entities. To support sensory effects, the language adds a new module *Effect* to the *Components* Functional Area of NCL 3.0. Therefore, NCL 4.0 defines a new element named *effect* that can be treated as other NCL nodes, such as *media*, *context* and *switch* [103].

To specify the rendering characteristics of sensory effects, NCL 4.0 allows authors to define them as properties of the *effect* element or using the *descriptor* element defined in NCL 3.0. Those properties are based on the MPEG-V standard. In addition, the extended NCL adds new attributes to the original *region* element of NCL 3.0 to specify the sensory effect position using the spherical coordinate system. Using this system, the language aims to allow authors to define the sensory effect position regardless of the mulsemedia physical installation. Also, authors can define the position using the MPEG-V spatial model [78].

NCL 4.0 also extends NCL to support multimodal interaction and multiple users [20, 19].

### 3.1.3 Guedes et al.

Guedes et al. [65] propose a high-level programming framework in order to support multimodal user interfaces for multimedia applications. The authors propose the integration of concepts from the multimedia and Multimodal User Interfaces (MUIs) communities. The framework supports different types of input and output modalities.

Regarding input modalities, the framework provides user-generated input modalities, such as gestures and voice recognizers. For instance, it uses SRGS (*Speech Recognition Grammar Specification*) [69] files to define which speech should be recognized. In other words, these files define recognizer anchors to specify parts of the recognizer content. Con-



cerning output modalities, it offers traditional audiovisual content, speech synthesizers, and actuators. Those output modalities can stimulate different human senses (hearing, smell, touch, taste, or vision) using audiovisual devices and actuators to render different kinds of sensory effects.

To allow users to synchronize recognizers and synthesizers, the framework defines four actions. *Start*, which enables the recognizer or synthesizer; *stop* to deactivate them; *pause* to deactivate without releasing its resources; and *resume*, which reactivates a paused recognizer or synthesizer.

Additionally, the framework defines a user class base to identify user profiles and allows applications to adapt according to these profiles. In other words, the authors propose contextual elements to provide the modality selection based on the user's sensory capabilities (see, speak and hear). The framework also considers environment characteristics with the user description to complement this adaptation to make the modality selection. Furthermore, the authors present a case study using the NCL language to illustrate the proposed framework.

The main difference between the framework presented in [65] and our work is the abstraction level used for representing sensory effects in mulsemmedia applications. In [65], the authors provide sensory effects by representing sensors and actuators as nodes. On the other hand, we model sensory effects as document nodes in a higher-level abstraction.

### 3.1.4 ASAMPL

ASAMPL (Algebraic System of Aggregates and Mulsemmedia data Processing Language) [120, 99, 119] is a programming language for enabling the authoring of mulsemmedia applications with multimodal content. The language focuses on representing complex information composed of multiple data of different types. The main idea is to represent a real scene using audiovisual data and olfactory and taste data. ASAMPL also represents physical data of objects or environments, such as air, water, metal, and their properties, temperature, pressure, density to describe the real world.

The language defines the concept of multi-image to represent multimodal information by applying the Algebraic System of Aggregates' concepts. A multi-image consists of a set of muxels (multimodal elements). The muxel term is defined based on voxel graphics, a volume element representing 3D objects. A tuple of values describes a muxel representing different modalities of the object characteristic in this point (muxel) in a specific instant.

However, authors must use external sources to give the details of the multimodal content. Indeed, they must use the ASAMPL’s Source and Download statements to assign these details to a tuple or aggregate (set of tuples).

ASAMPL also provides built-in commands that allow developers to fulfill necessary actions on multimodal data. Using these actions, developers can download and upload data streams, temporally synchronize data of different modalities using a timeline-based paradigm and change the data duration.

### 3.1.5 Comparison of Mulsemmedia Models

Figure 3.1 illustrates four quadrants defined according to temporal synchronization paradigm (event-based and timeline-based) and abstraction level to give an overview of the mulsemmedia models discussed in this section. The horizontal axis ranks the models according to their abstraction level for representing sensory effects and traditional multimedia. Models that are closer to the right have a higher abstraction level than those closer to the left. Our proposal is called MultiSEM (Multimedia Sensory Effect Model) and is going to be presented in detail in Chapter 4. Figure 3.1 shows that MultiSEM and NCL 4.0 have the highest abstraction level since they are most to the right. Both proposals represent sensory effects as first-class entities. On the opposite side, the MPEG-V standard has the lowest abstraction level for representing sensory effects.

The vertical axis classifies those models into two groups. One of them defines the event-based models. This group contains the proposal of Guedes et al. [65], MultiSEM [48], and NCL 4.0 [73] since the models use the event-based paradigm to synchronize mulsemmedia contents temporally. The other group contains timeline-based models, MPEG-V and ASAMPL. For instance, ASAMPL is a timeline-based model classified as a low-level abstraction model. However, this model has an abstraction level higher than MPEG-V.

## 3.2 Mulsemmedia Authoring Tools

In addition to the previous studies that help us represent sensory effects in mulsemmedia applications proposing conceptual models or declarative languages, others focus on enhancing the mulsemmedia authoring phase by providing graphical tools. Those tools support the definition of sensory effect metadata and the specification of complete mulsemmedia applications using different authoring GUI approaches. Here we present different mulse-

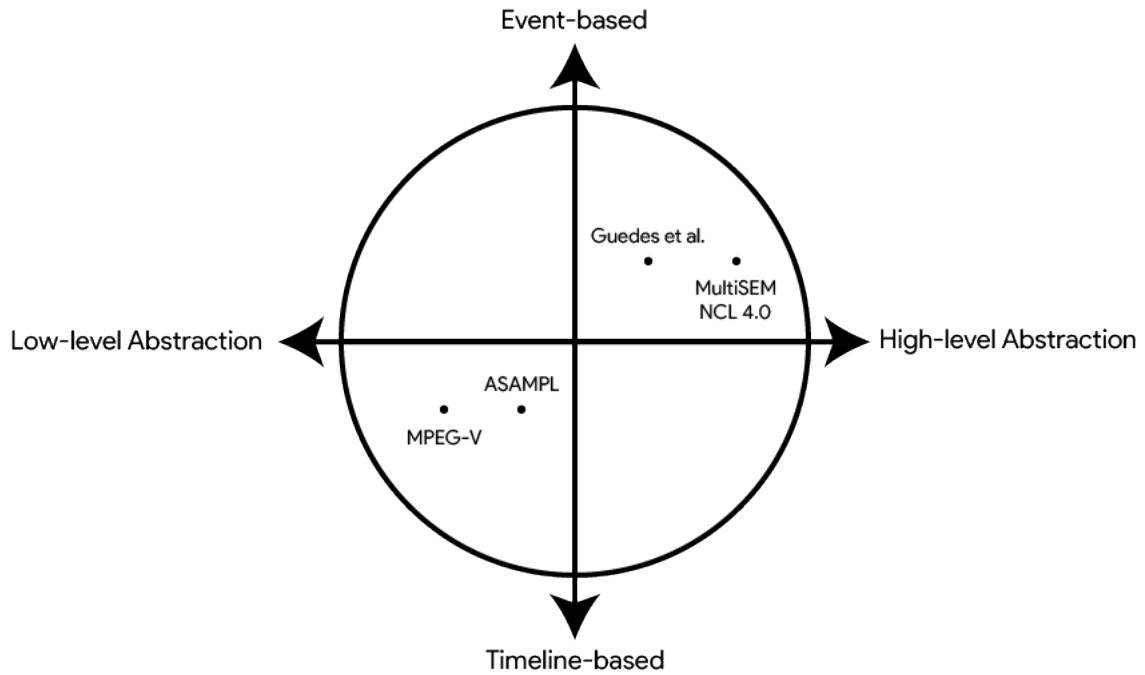


Figure 3.1: Mulsemedia Models Overview

media authoring tools, discussing their features and comparing them to raise requirements and challenges regarding graphical authoring tools.

### 3.2.1 CrowdMuse

Aiming at finding out whether a crowdsourcing approach can support mulsemmedia authoring, particularly sensory effect annotation associated with a specific video, authors in [44] present a web platform called CrowdMuse. This platform collects sensory effect annotations from the crowd allowing authors and users to work together in the mulsemmedia authoring phase. However, CrowdMuse does not focus on providing a complete authoring environment in which users can synchronize multiple media and sensory effects and edit rendering characteristics. Moreover, the tool does not provide a simulator or player. Instead, CrowdMuse focuses on collecting the contributions of users regarding video time intervals that they think to be appropriate to associate sensory effects. As the outcome of the crowdsourcing process, the tool exports the sensory effect annotation to the MPEG-V format so that other players compatible with this format can render the sensory effects synchronized with the video, for which the effects were annotated.

### 3.2.2 H-Studio

H-Studio [42] is an authoring tool that focuses on the creation of only two types of sensory effects: vibration and motion. The tool is composed of three main parts: a video preview to play the video to be annotated with sensory effects, a timeline where users can synchronize sensory effects with the video, and a menu to allow authors to define parameters of the current sensory effect selected in the timeline. For vibration parameters, H-Studio presents only two parameters: amplitude and frequency. On the other hand, the tool provides a more complex interface for defining the properties of motion effects according to the Six Degrees of Freedom (6DoF) of a rigid body in three-dimensional space. It means that an object can move forward/backward, up/down, left/right combined with rotation in three perpendicular axes. H-Studio offers three methods to support the specification of 6DoF movements. The first one allows authors to use a force-feedback device, the second defines the motion through a trajectory recording from a force-feedback device, and the last method allows users to import those parameters from the real world. The tool also provides a video player that can synchronize with a force-feedback device to preview the effects created.

### 3.2.3 Sang-Kyun Kim et al.

In [80], authors propose a method to extract temperature sensory effects from audiovisual content. To this end, the method consists of extracting the color temperature of scenes and mapping their properties to the temperature effect attributes. They also introduce an authoring tool to apply the proposed method. The tool contains a video control component, a temperature effect component, and an effect sensory timeline component. The video control component allows authors to go forward or backward frame by frame in a specific video. With the temperature effect module, authors can select frame intervals to extract the color temperature. Then from color temperature properties, the tool creates temperature effect metadata specified in MPEG-V. Moreover, the authoring tool provides a timeline for showing the calculated color temperature categories and their correlated temperature effects along the time axis. Authors can also define frame intervals using this timeline.

### 3.2.4 SEVino

SEVino (Sensory Effect Video Annotation) [126] is a graphical tool that provides a timeline divided into channels. Each channel represents a type of sensory effect, such as wind, vibration, and light. In the timeline, users can create rectangles whose sizes define sensory effect duration. Also, the tool provides video frames along the time axis to allow authors to synchronize effects with single video content. In addition, SEVino allows users to define some properties of the corresponding sensory effect selected on the timeline. Moreover, SEVino can export sensory effect annotation to MPEG-V and import MPEG-V SEM description files so that users can modify or extend them using SEVino's GUI. In addition to these features, SEVino is integrated with a player, Sensory Effect Media Player (SEMP), and a simulator, Sensory Effect Simulator (SESim). Both of them are also proposed in [126] and will be discussed in more detail in Section 3.5.

### 3.2.5 RoSE

RoSE (Representation of Sensory Effects) Studio [34] is an authoring tool that also uses a timeline to synchronize sensory effects with audiovisual contents. As SEVino, RoSe exports sensory effects to SEM files using MPEG-V. In addition, RoSe multiplexes the effects using the MPEG-2 TS standard. RoSe's GUI comprises a video player, a region for editing sensory effect properties, and a timeline. The tool also offers a specific graphical interface to define the sensory effect position in the 3D space based on the MPEG-V spatial model. Although the tool provides several sensory effects, it does not allow users to synchronize them with several multimedia items. Instead, RoSe only supporting the sensory effect annotation with a single video.

### 3.2.6 SMURF

Another tool for creating MPEG-V SEM files using a timeline is SMURF (Sensible Media Authoring Factory) [77]. It is also based on the MPEG-V standard, and its GUI is composed of a video player, a region for defining the sensory effect rendering characteristics, and a timeline for synchronizing the sensory effect with a single video. Additionally, SMURF provides an interface to create groups of effects. In this interface, users can define several types of effects to compound a complex effect, such as an explosion effect involving temperature, light, flash, and wind effects. As a result, only one effect represents the explosion effect, rendering different types of primary effects. Furthermore, SMURF

allows authors to reuse a declared frequently-used effect. Last but not least, the tool can import and export MPEG-V SEM files.

### 3.2.7 Real 4D Studio

In [111], authors propose an authoring environment composed of three tools: Real4DAExtractor, Real4DEmaker, and Real4DASudio. The first is responsible for extracting rigid body motion, light, and flash effects from visual contents through frame analysis. Real4DEmaker allows users to generate SEM files for each sensory effect at a time. Users can also import a SEM file and simulate the effect in a 3D virtual world. In addition, Real4DEmaker provides an interface for editing effect position based on the MPEG-V spatial model.

Real4DStudio allows authors to synchronize sensory effects created in Real4DEmaker with a single video using a timeline approach. A 3D simulation for checking the synchronization of several effects is also available in Real4DStudio. Additionally, this tool presents two types of interfaces according to the purpose of usage: media-based and event-based interfaces. Both of them use a timeline approach. However, the media-based interface provides authoring targeting specific media. The event-based interface provides an effect preview instead of a canvas view to focus on authoring a particular sensory effect.

### 3.2.8 MulSeMaker

MulSeMaker [50] aims at enhancing the authoring of mulsemmedia applications for authors with no programming knowledge by providing a template-based interface approach. It uses web components to define custom XML tags for defining and integrating sensory effects with HTML media objects. This integration uses a timeline-based paradigm for synchronizing continuous media with sensory effects. For discrete media, it uses the event-based paradigm. In this case, users can define interactivity events, such as a user's click to start or stop sensory effects. Also, users can start sensory effects when a specific discrete media begins its presentation. MulSeMaker's graphical interface is based on wizards and templates. Authors should select a template, choose media objects to be part of the document and define application sensory effects. For each effect, authors associate it with a media object and define its beginning and ending time by using a table. The use of wizard and templates support authors with no knowledge of programming. However, this approach limits the expressiveness of the authoring tool by restricting authors to the pre-

defined template set. Moreover, the authors are not able to modify the behavior of these applications. Therefore, Mulsemaker does not allow authors to define new mulsemmedia applications by defining document nodes and their spatio-temporal relationships.

### 3.2.9 FeelReal

FeelReal [56] is a commercial tool for synchronizing sensory effects with a single video. The editor focuses on scent effects since its vendor provides a scent generator mask to integrate with several brands of virtual reality headsets. This mask can also render water mist, wind, heat, and vibration effects. FeelReal provides an interface based on timeline and panel with different scents that the mask can render. The editor also allows users to define the effect position but is limited to the left and right sides, which corresponds to the sides of the mask device. It does not provide a simulator or export the sensory effect description to a standard format, such as MPEG-V, due to the description being directly sent to the mask device.

## 3.3 Desirable Features for Mulsemmedia Authoring Tools

Taking the discussion of mulsemmedia authoring tools into account and based on multimedia authoring features presented in Section 2.4, we also identify a set of features to be provided by mulsemmedia authoring tools for enhancing the mulsemmedia production phase. They are as follows:

- *Temporal View Editing*: tools need to provide a temporal view for presenting the temporal behavior of traditional multimedia items and sensory effects. Media items and effects have to be directly placed on the time axis so that users can verify when they are presented and their duration. Additionally, tools must allow users to edit the temporal view by changing the duration and beginning/end time instant of media items and sensory effects.
- *Synchronization of Multiple Media and SE (Sensory Effects)*: tools should allow users to temporally synchronize not only a single video but also several types of media, such as video, image, text, and audio, with various sensory effects at the same time through a temporal view.

- *SE Group*: an interface for defining groups of effects so that authors can create complex effects using primary sensory effects should be provided. For instance, an explosion effect can be composed of the primary effects of wind, light, and vibration.
- *SE Reuse*: tools need to provide the reuse of sensory effects that are frequently used. Authors should be able to declare and refer them.
- *Spatial View Editing*: a spatial view must be provided by tools for presenting and editing how (presentation and rendering properties), and where (media and sensory effect position) media items and sensory effects are presented or rendered during execution. It is noteworthy that editing rendering properties include changing the intensity value of sensory effects. As in the multimedia context, this view has to be synchronized with the temporal view so that authors can verify the spatial view for each time instant of the mulsemmedia presentation.
- *Interactivity Support*: as in multimedia applications, authors also need to be able to define interactivity relations in mulsemmedia applications to allow users to interact through an input device (e.g., remote control) with (generally visual) media items. In order to specify that feature, authors must define an interactive media with which users can interact to trigger an action that can modify the temporal or spatial behavior of other media items or sensory effects.
- *Statement Assessment Support*: this requirement refers to allowing authors to define statement assessment using information captured from sensor devices to trigger actions on media items or sensory effect(s) based on the particular sensed information.
- *Simulator*: tools should allow authors to visualize the temporal and spatial behavior in an integrated simulation virtual environment so that authors can verify when, how and where media items and sensory effects would be presented in the real world before publishing the mulsemmedia application. In the case of interactive applications, the simulation may not provide user interaction.



- *Ordinary User Support*: tools should allow authors with no knowledge of mulsemmedia language or model to create mulsemmedia applications offering a graphical user interface approach that can guide authors in their production process.
- *Template Support*: templates are generic structures that specify the spatio-temporal behavior of mulsemmedia applications but with no definition of media contents and sensory effect annotation. In other words, tools should provide templates so that users have to define media contents and sensory effect annotation for each media item that requires it in the template specification to create mulsemmedia applications.
- *Automatic extraction*: tools should allow authors to automatically extract sensory effects from audiovisual contents to enhance sensory effect annotation.
- *Standard Publishing Format*: authors should be able to export their mulsemmedia applications from the tool in which they were created to documents written in standard mulsemmedia languages.
- *SE Preparation Support*: for distributed mulsemmedia systems, the performance analysis concerning the sensory effect rendering in the real world is essential since delays can occur during the preparation of actuator devices [75], which is responsible for rendering sensory effects in the real world. Therefore, techniques [75] can be applied to configure the preparation of rendering devices in the authoring phase to avoid delays in effect rendering.
- *Error Analysis*: authors should have feedback about inconsistent spatial or temporal behaviors when they create or edit mulsemmedia applications. This feature is handy in order to avoid execution errors [106] after applications are distributed to final users.
- *Player*: tools need to provide an embedded mulsemmedia player or to be integrated with an external player so that authors can fully be immersed in mulsemmedia applications and feel sensory effects in real life as end users will.

## 3.4 Comparison of Mulsemmedia Authoring Tools

Table 3.1 presents a comparison among the mulsemmedia authoring tools discussed in Section 3.2. We compare them based on the functional requirements for mulsemmedia authoring tools presented in the previous section. Additionally, the comparison table presents the temporal synchronization paradigm for each mulsemmedia tool. The tools cited in this survey use the event-based or timeline-based paradigms, which follow the definition presented in Section 2.1. Furthermore, the table describes the authoring GUI approach, which also follows the same classification presented in Section 2.2 although the mulsemmedia tools discussed in this study only use the temporal, spatial, or wizard approach.

Our proposal for multimedia authoring tool is called STEVE 2.0 [48]. It is an extension of STEVE [47, 46], discussed in Section 2.3.1, for integrating sensory effects with traditional multimedia content. STEVE 2.0 will be presented in Chapter 6 in detail.

In order to present a more detailed comparison regarding the spatial view editing, we divided this feature into four subtopics: *SE Position Editing*, *Rendering Editing*, *SE Spatial View* and *Multimedia Spatial View Editing*. The *SE Position Editing* feature allows authors to edit the 3D position from where sensory effects are rendered. Some of the tools discussed in Section 3.2 provide this feature based on the MPEG-V spatial model. H-Studio partially supports this feature since the tool only allows authors to edit motion effect position. FeelReal also partially provides position editing since authors can only define the left and right sides of the headset to render effects. The *Rendering Editing* feature refers to an interface to allow authors to edit the several rendering properties of sensory effects. Those properties can also be based on the characteristics defined in MPEG-V. MulSeMaker and FeelReal partially provide this feature since both tools only allow the effect intensity editing. The tool presented in [80] also supports the rendering properties partially since it only allows authors to edit properties of temperature sensory effects. The *SE Spatial View* subtopic corresponds to the tool's ability to allow users to verify the spatial behavior of sensory effects together with traditional multimedia content for each time instant synchronized with the temporal view. Finally, *Multimedia Spatial View Editing* indicates whether tools specifically provide a spatial view for traditional multimedia items, which is the case of STEVE 2.0.

Moreover, Table 3.1 indicates the *Asynchronous Event Support* feature, which includes the requirements related to the interactivity and statement assessment support described in Section 3.3. Those features correspond to events that may occur during the application

Table 3.1: Features of Mulsemmedia Authoring Tools

		STEVE 2.0 [48]	CrowdMuse [44]	H-Studio [42]	Kim et al. [80]	SEVino [126]	RoSE [34]	SMURF [77]	Real 4D Studio [111]	MulSeMaker [50]	FeelReal [56]
<b>Temporal Sync. Paradigm</b>		E	T	T	T	T	T	T	T	E & T	T
<b>Authoring GUI Approach</b>		T & S	W	T	T	T	T	T	T	W	T
<b>Sync. of Multiple Media and Sensory Effects</b>		✓								✓	
<b>Sensory Effect Group</b>								✓			
<b>Sensory Effect Reuse</b>								✓			
<b>Spatial View Editing</b>	<b>SE Position Editing</b>	✓		+/-			✓	✓	✓		+/-
	<b>Rendering Editing</b>	✓		✓	+/-	✓	✓	✓	✓	+/-	+/-
	<b>Sensory Effect Spatial View</b>										
	<b>Multimedia Spatial View Editing</b>	✓									
<b>Asynchronous Events</b>	<b>Interactivity Support</b>	✓								+/-	
	<b>Statement Assessment Support</b>										
<b>Simulator</b>						✓			✓		
<b>Ordinary User Support</b>		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
<b>Template Support</b>										✓	
<b>Automatic Extraction</b>		✓			+/-				✓		
<b>Publishing Formats</b>		NCL 4.0	MV	none	MV	MV	MV	MV	MV	HTML5	none
<b>SE Preparation Support</b>											
<b>Error Analysis</b>		✓									
<b>Player</b>						✓					✓
<b>Feature Score</b>		8	1	2.5	2	4	3	5	5	4	3

+/- (partially supported feature); ✓ (fully supported feature);

SE (Sensory Effect)

Temporal Synchronization Paradigm: E (event-based); T (timeline-based);

Authoring GUI Approach: T (temporal); S (spatial); W (wizard)

Publishing Formats: MV (MPEG-V)

execution, and we cannot predict when they will occur during authoring time. MulSeMaker partially supports the interactivity feature. Although the predefined templates available in MulSeMaker may contain interactivity events already specified, users cannot create new interactivity relations or edit the existing ones.

The tool introduced in [80] partially provides the *Automatic Extraction* feature since it extracts only temperature sensory effects from visual contents. On the other hand, STEVE 2.0 and Real 4D Studio extract different types of sensory effects. Concerning

the *Player* column, it indicates which tool provides a mulsemmedia player. Each player is discussed in Section 3.5. The last line of the table, *Feature Score*, provides a rank among the tools we discussed by checking how many features they support. For each feature the tools provide, they earn 1 point. For features that are partially supported, the tools earn 0.5 points. For example, H-Studio totally provides *Ordinary User Support* (1 point) and *Rendering Editing* (1 point). However, this tool partially supports *SE Position Editing*, which results in 0.5 points. H-Studio thus has 2.5 as *Feature Score*.

Most tools we discussed in this chapter are based on the MPEG-V temporal synchronization paradigm. Therefore, they inherit the limitations of the timeline paradigm. One limitation is the lack of support for defining asynchronous events, such as user interaction and state variable assessments. Another limitation is that those tools only support the authoring of sensory effect metadata (SEM files). They do not specify sensory effects in the context of mulsemmedia applications. Consequently, these MPEG-V-based tools do not define temporal relationships among nodes (traditional media or sensory effects). In contrast, only STEVE 2.0 and MulSeMaker define nodes and temporal relationships to specify an application behavior and support the event-based temporal synchronization paradigm.

Furthermore, STEVE 2.0 is the only tool that offers a spatial GUI approach that provides spatial view editing for traditional multimedia and sensory effects. CrowdMuse and MulSeMaker are tools that use the wizard approach. However, CrowdMuse only provides a single interface with a video player and buttons to select the beginning and end time instants and intervals to annotate sensory effects with a single video, while MulSeMaker gives a step-by-step window to guide authors through the authoring process. The remaining tools only provide a timeline view as an authoring GUI approach.

Figure 3.2 illustrates four quadrants defined according to temporal synchronization paradigm (event-based and timeline-based) and features to give an overview of the mulsemmedia tools discussed. We placed the tools in the horizontal axis according to the *Feature Score* line in Table 3.1. This axis is divided in half in value 4.5. Thus, STEVE 2.0, which has the highest feature score (8), is most to the right, while CrowdMuse is leftmost with the lowest score.

The vertical axis classifies those tools into two groups. One of them defines the tools that are based on events to synchronize mulsemmedia documents temporally. This first group contains STEVE 2.0 and MulSeMaker. Although MulSeMaker uses the timeline paradigm for continuous media, it also uses the event-based paradigm, as discussed in

Section 3.2.8. The other group refers to tools that use the timeline-based synchronization paradigm. For example, SEVino has a score of 4, and its temporal synchronization is based on the timeline paradigm. We can remark that the majority of tools use the timeline-based paradigm and do not provide several desirable features for mulsemmedia authoring tools listed in Section 3.3.

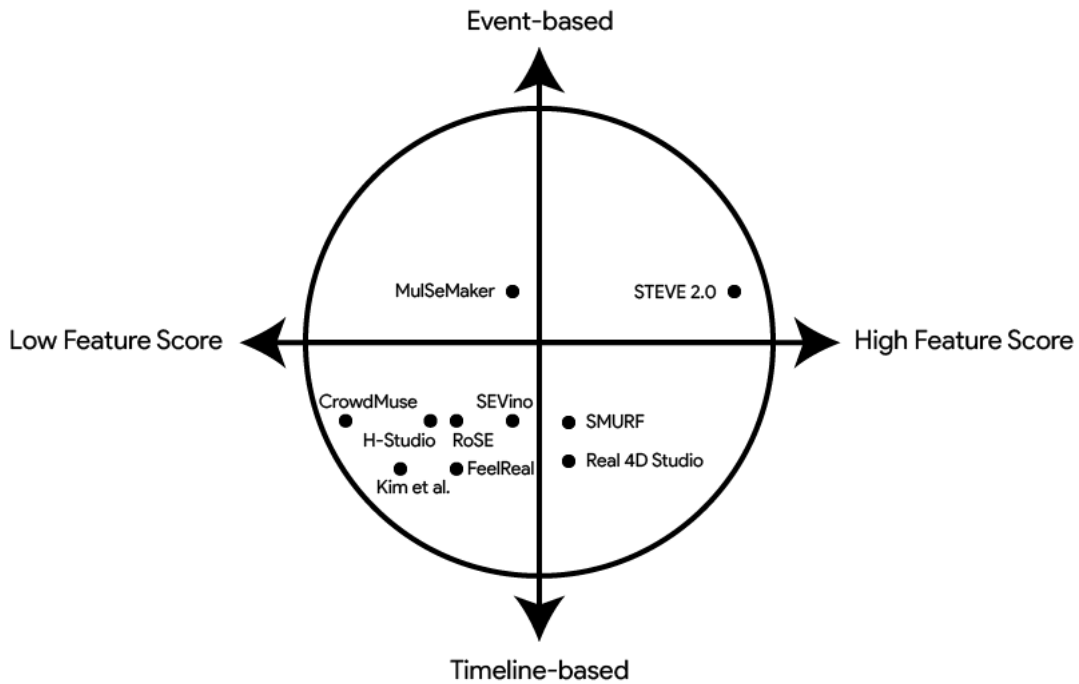


Figure 3.2: Mulsemmedia Tools Overview

### 3.5 Mulsemmedia Simulators and Players

In order to run mulsemmedia applications, several players have been proposed in the literature. Providing a player embedded into authoring tools is essential to enhance the whole process of deploying mulsemmedia applications in the real world. Such players can present traditional multimedia content and render sensory effects. To do this, these players communicate with multimedia displays and with actuator devices, sending them commands so that those devices can render sensory effects.

In addition to players, simulators have also been proposed. They use virtual actuators to support the author's effort of checking a mulsemmedia application execution. Sensory effect simulators represent those virtual actuators using graphical symbols. Therefore, when an actuator should be activated, its graphical representation is highlighted to indicate that the actuator is rendering a sensory effect. In the following, some tools for playing and simulating mulsemmedia applications are discussed.

### 3.5.1 SESim - Sensory Effect Simulator

Waltl et al. [126] proposed a simulator to aid the development of applications that make use of the MPEG-V standard to specify sensory effects. The simulator is called SESim, Sensory Effect Simulator. SESim receives a SEM file, which can be created using SEVino, and the audio/video (A/V) files as inputs. Then, the SESim XML parser extracts the sensory effects from that SEM file. Afterward, those effects are forwarded to its simulator module. That module sends the A/V files to the player module and the extracted effects to the timer module. The timer module also receives the current playback time to activate/deactivate the corresponding virtual actuator.

### 3.5.2 SEMP - Sensory Effect Media Player

SEMP (Sensory Effect Media Player) is a mulsemmedia application player that is also proposed in [126]. The player supports the following actuators for rendering sensory effects: amBX, Cyborg Gaming Lights, and *Vortex Activ* systems [4]. The first one provides a wind effect through two fans with around 5000 rounds per minute (RPM). It also produces light effects using LEDs. Cyborg Gaming Lights also provide light effects. However, that system produces more powerful light effects than amBX. The Vortex Activ system provides scent effects. To do that, it consists of a set of four fans to spread scents in the environment. As SESim, SEMP should receive the audiovisual content and SEM files as inputs to render sensory effects. However, in SEMP, commands are forwarded to real devices for rendering sensory effects.

### 3.5.3 Sensible Media Simulator

Another simulator that is also compatible with MPEG-V is the Sensible Media Simulator [79]. This simulator was designed to take advantage of car devices to produce sensory effects. The in-car entertainment system provides a graphical interface that shows the available car devices used as actuators. Users can also check the location of each device. As SESim, a SEM file and the audiovisual content are received as input. In addition, the capabilities of sensory devices in a car are described using CIDL (Control Information Description Language) of the MPEG-V standard. User sensory preferences are also specified with CIDL. The control information contained therein allows an application to be adapted to users and device capabilities. Additionally, the Sensible Media Simulator uses IIDL (Interaction Information Description Language) to describe sensed information and

device commands.

### 3.5.4 Sensorama

Sensorama [33] is a player that is also able to receive SEM files as input. Additionally, it allows users to define an event list that represents a composition of sensory effects. For instance, an explosion can be an event composed of motion, temperature, and wind effects. Sensorama's architecture is based on a 4D effect device control engine. This engine is responsible for loading and parsing SEM files, managing the synchronization between media and sensory effects, mapping devices and data, and sending control signals to actuators. To achieve the goal of providing a 4D movie theater experience, the player uses a cave automatic virtual environment (CAVE). The CAVE system supports wind, light, fog, flashlight, and motion effects.

### 3.5.5 Multimedia Multisensorial 4D Platform

In [21], authors present a multimedia multisensorial 4D platform that focuses on delivering audiovisual content synchronized with only olfactory and thermal effects. The platform architecture is based on a central MCU (Microcontroller Unit) that represents the multi-sensory module. It is responsible for receiving sensory information from a server, performing parsing necessary to exchange data between the virtual and real worlds, and sending commands to actuator devices, which render the olfactory and thermal sensory effects. To describe those commands and sensed information, the platform uses the MPEG-V standard. However, the high coupling of the MCU module implementation does not allow its extension to integrate new types of sensory effects and devices, and neither does it allow the reuse of the MCU with other mulsemmedia applications.

### 3.5.6 Real 4D Studio Simulator

Real 4D Studio [111] is an authoring tool that provides an embedded simulator to allow authors to verify in a 3D virtual environment when sensory effects are activated during the mulsemmedia application execution. The 3D simulation is performed by parsing SEM files written in MPEG-V and using the media control, part of the Real4DEMaker module. The simulation presents 3D graphics representing sensory effect actuators and offers a media player together to show the synchronization between the audiovisual content (single video) and sensory effects. Since Real 4D Studio focuses on providing a 4D movie experience

with armchair motion effects, the simulation also presents visual elements to represent each armchair movement.

### 3.5.7 PlaySEM SER 2

A mulsemedia framework, called PlaySEM SER 2, for dealing with heterogeneous applications and devices is proposed in [104]. The framework allows mulsemedia applications, whether timeline-based or event-based, to be rendered in the real world, handling hardware configurations, communications issues, and heterogeneous devices. Therefore, in the mulsemedia authoring phase, authors can focus only on the details of the mulsemedia application specification, such as the mulsemedia content and the spatio-temporal synchronization. This framework supports several communication protocols, metadata standards, connectivity interfaces, and rendering devices. Previously in [105], PlaySEM SER did not provide a flexible architecture in which new protocols and devices could be added with no changes in its components. The first version of PlaySEM SER also only allowed MPEG-V to describe the sensory effect metadata.

PlaySEM SER 2's architecture is divided in *Connectivity Interface*, *Sensory Effects Processing* and *Communication Broker*. The first is responsible for establishing connections with multiple and heterogeneous devices using a set of protocols, such as *Wi-Fi*, *Bluetooth* and *USB/Serial*. This module also provides a set of communication protocols, such as *CoAP*, *MQTT* and others to exchange messages between the devices and PlaySEM SER 2. The Sensory Effects Processing module reads the metadata of sensory effects and converts them into messages to control the devices regardless of the specific type of device. This module allows mulsemedia systems to work with MPEG-V and other metadata types, giving them flexibility. Concerning the Communication Broker module, it is a bridge between PlaySEM SER 2 and mulsemedia applications. In other words, this part conveys SEM descriptions to PlaySEM SER 2 and provides essential services for temporally synchronizing multimedia content with sensory effects such as starting, pausing, and stopping. This communication is also made using different protocols such as UPnP, CoAP, MQTT, and WebSocket.

Although this proposal handles several issues concerning communication and connectivity with heterogeneous devices, it does not involve the specification of mulsemedia applications. For each new event or behavior of the application, a software component must be implemented in a procedural language to recognize such an event or behavior.



### 3.5.8 Josué et al.

Authors in [74] propose a sensory effect simulator to enable the preview of mulsemmedia applications in a 3D room by receiving SEM files written in MPEG-V. This 3D environment consists of a center point and spread objects representing actuators. Using a WYSIWYG (What You See is What You Get) approach, users can organize physical objects, such as chairs, TV, and actuators. They can also add and remove actuators. The direction of the actuators is always the center point of the room. The simulator uses colors to represent different sensory effect types and to indicate when actuators are active. The color intensity represents the effect intensity associated with the actuator. In this simulator, the authors propose extending the MPEG-V's spatial model to place actuators in the 3D room. This extended model uses a spherical coordinate system to overcome the MPEG-V's spatial model limitation of representing slight variations in the position of sensory effects. Moreover, the MPEG-V's spatial model is unsuitable for handling animations, in which effect position changes over time. The simulator, however, supports both spatial models.

## 3.6 Gaps of Authoring Tools

Based on our comparison of mulsemmedia authoring tools in Section 3.4, we have identified gaps regarding the proposals of those authoring tools. All those gaps should be considered in future proposals and investigated to advance research on mulsemmedia authoring.

- *Temporal Synchronization Paradigm*: Most tools are based on the timeline temporal synchronization paradigm, which, however, presents limitations, as discussed in Section 2.1. The event-based paradigm, which is more expressive, is rarely explored by the tools. In this review, only STEVE 2.0 [48] and MulSeMaker [50] use the event-based paradigm to temporally synchronize traditional multimedia and sensory effects.
- *Synchronization of Multiple Media Items*: STEVE 2.0 and MulSeMaker are the only tools that support synchronization with multiple media items. In other words, those tools allow users to synchronize multiple sensory effects with several traditional multimedia items (audio, video, image and text) in time and space. On the other hand, the majority of tools presented in this study support the synchronization of several sensory effects with only one video. Exploring this gap in authoring tools

can help to understand how users deal with several items at same time. This will also encourage the production of complex mulsemmedia applications.

- *Sensory Effect Group and Reuse*: Although MPEG-V defines groups of sensory effects, SMURF [77] is the unique proposal that allows users to define groups of effects and reuse frequently-used sensory effects/groups. Users can take advantage of this feature to define more complex combinations of sensory effects.
- *Sensory Effect Spatial View*: None of the tools presented in this study provides an effect spatial view where authors can verify the spatial behavior of sensory effects and traditional multimedia items at the same time integrated with the temporal view.
- *Asynchronous Events*: Asynchronous events is an important feature for creating interactive and dynamic mulsemmedia applications. However, this feature is rarely provided by the tools. This feature includes the definition of interactivity relations and statement assessment tests.
- *Template Support*: The template support is not also often explored by mulsemmedia tools. Templates can accelerate the authoring phase by giving users predefined structures of mulsemmedia applications. To provide those templates, the scientific community needs to investigate several mulsemmedia scenarios in different application fields (learning, health, entertainment etc).
- *Sensory Effect Preparation*: The preparation of sensory effects is not supported by any tool discussed in this study. However, this feature is fundamental for dealing with delays that actuators may have [75, 73] while rendering sensory effects. For example, actuators to render scent effects need time to be effective. In other words, users may experience the scent effect rendered after a delay.

This chapter discussed studies regarding mulsemmedia modeling, and graphical authoring tools, highlighting their differences among them and identified challenges in the authoring phase. The following chapter presents our mulsemmedia conceptual model, MultiSEM.

## Chapter 4

# MultiSEM - Multimedia Sensory Effect Model

This chapter presents the Multimedia Sensory Effect Model (MultiSEM) [48, 83] for integrating and synchronizing multiple sensory effects with traditional multimedia content in hypermedia applications. MultiSEM is a specific domain model that aims to enhance the development of mulsemedia applications and support the implementation of mulsemedia graphical authoring environments based on temporal views for authors with no programming skills. With that goal in mind, MultiSEM is based on the NCM (Nested Context Model) [116, 117] model and the MPEG-V standard proposing a simplification of NCM and integrating MPEG-V's sensory effect vocabulary into our mulsemedia model. We will discuss the differences between MultiSEM and those solutions later in this chapter.

MultiSEM allows the synchronization of sensory effects with traditional media in time and space. To this end, the model represents sensory effects as nodes, following the approach discussed in [74]. That is, they are modeled as first-class entities using an approach with high-level abstraction. Therefore, the spatio-temporal behavior of mulsemedia applications can be defined regardless of the devices used for implementing mulsemedia applications in the real world. MultiSEM also uses the event-based temporal synchronization paradigm. This paradigm is applied to several proposals of authoring tools [47, 66, 96] and multimedia models [116, 117, 52] since it is more expressive compared to the other paradigms [25], as we discussed in Section 2.1. Additionally, the proposed model uses the concept of hypermedia connector [91] to represent spatio-temporal relations among nodes, whether traditional multimedia content or sensory effects. Therefore, the model allows sensory effects to be temporally synchronized with other nodes (media or effect) according to events in the mulsemedia application.

To define the sensory effect entity and their subclasses (e.g. wind effect, heat effect, etc.), MultiSEM is based on *Part 3: Sensory Information* of MPEG-V [122]. Sections 4.1 and 4.2 present respectively the different types of sensory effects and their rendering properties, such as intensity value and range (e.g., *lux* unit is used for light effects), position, and specific rendering properties of each effect type. The MultiSEM spatial model for sensory effects is also based on the MPEG-V standard. Therefore, effects can be located in physical environments according to the X, Y, and Z axes [78]. Sections 4.3 and 4.4 present how MultiSEM handles events and relations among nodes following NCM concepts. Differences among MultiSEM, NCM, and MPEG-V will be discussed in detail in Section 4.5.

## 4.1 Nodes

In multimedia modeling [117], content parts are represented as nodes or components. A content node comprises a set of information units that depends on the type of media that the node represents. Information units can be audio samples, video frames, text characters, or image pixels. That node concept is extended in [74] to represent sensory effects in mulsemmedia applications. According to this definition, a node represents sensory stimuli to users and may represent multimedia content or sensory effects, stimulating user senses. MultiSEM uses this concept by modeling sensory effects as mulsemmedia application nodes to integrate effects with traditional multimedia content, as the diagram of Figure 4.1 shows.

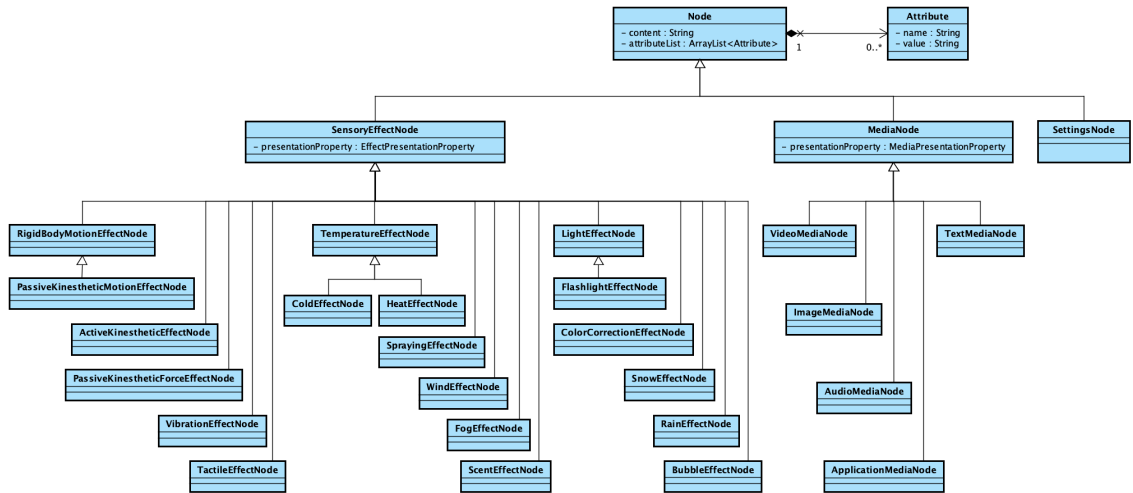


Figure 4.1: MultiSEM Node Representation

In MultiSEM, the *Node* entity is responsible for creating a generalization of *Sensory-*

*EffectNode*, *MediaNode* and *SettingsNode* entities. It may also contain a list of attributes defined by the author. The *SensoryEffectNode* class contains the *presentationProperty* attribute to represent the presentation/rendering characteristics of effects. These effect properties are defined according to the type of effect. In Section 4.2, we discuss more details about those properties.

The *SensoryEffectNode* class has also subclasses to represent different effect types, which are based on *Part 3: Sensory Information* of MPEG-V [122]. Following the order of effects presented in Figure 4.1, the *RigidBodyMotionEffectNode* describes the movement of a rigid object such as a motion chair commonly offered in cinemas. *PassiveKinestheticMotionEffectNode* is an extension of *RigidBodyMotionEffectNode* since the passive kinesthetic motion effect is a variation of a rigid body motion. The passive motion provides users with trajectory motions guided by a kinesthetic device. That is, users wear this device and it guides the user's body (hands, legs, or arms) according to the defined trajectory. In [93], kinesthetic devices are used to provide a paired system for combining the kinesthetic experiences between two persons. In this case, the trajectory is defined by one of the devices and reproduced on the other. This kind of device can enhance interactions such as clinical gait rehabilitation and ease sharing of physical experiences with Parkinson's patients. On the other hand, *ActiveKinestheticEffectNode* supports the active kinesthetic torque/force effect generated by a real or virtual object when users touch it. Different from *ActiveKinestheticEffectNode*, *PassiveKinestheticForceEffectNode* produces torque/force effect delivered by a kinesthetic device according to a defined torque/force. Regarding *VibrationEffectNode*, it creates vibration effects that can be rendered in the environment or a specific object such as the motion chair as the rigid body motion effect. *TactileEffectNode* supports a kind of tactile display that simulates a feeling of touching object surfaces.

In contrast to MPEG-V, MultiSEM defines *ColdEffectNode* and *HeatEffectNode* as an extension of *TemperatureEffectNode* because a cold or a heat effect is a special variation of temperature effect. The cold and heat effects have a default temperature value to reproduce the sensation of cold and heat respectively. *SprayingEffectNode* renders spraying effects that can be composed by any liquid or even a powder type. *WindEffectNode* and *FogEffectNode* are responsible for rendering wind and fog in the immersive environment respectively. *ScentEffectNode* is an effect that reproduces different scent types.

*LightEffectNode* describes any light effects and can also produce colored light in the environment. The *FlashlightEffectNode* type is modeled as a subclass of *LightEffectNode*

since a flash effect is a special variation of light. *ColorCorrectionEffectNode* is used to effectively reproduce the color of visual contents or their specific regions/objects based on the content provider intention. This color correction process also considers the display environment as the ambient light and characteristics of the display device used by the consumer. *SnowEffectNode* is an effect to simulate snow in the immersive environment. As the snow effect, *RainEffectNode* and *BubbleEffectNode* are not defined by MPEG-V. They produce rain and bubbles, respectively.

With regard to *MediaNode*, it represents multimedia content and has subclasses to model different media types. As extensions of this entity, MultiSEM thus define the following subclasses. *VideoMediaNode*, *ImageMediaNode*, *AudioMediaNode* and *TextMediaNode*, which represent video, image, audio and text media respectively. In *MediaNode*, the *content* attribute inherited from *Node* is a reference, such as URI (*Uniform Resource Identifier*), to the content itself or a byte sequence. For instance, the content of *VideoMediaNode* refers to a video file such as an MP4 file. An image media may refer to a PNG file. An MP3 file can be referenced by an audio media. As *SensoryEffectNode*, *MediaNode* contains the *presentationProperty* attribute to represent the presentation characteristics of media objects. These characteristics are also defined according to the type of media, which are discussed with further details in Section 4.2.

To support imperative objects in a declarative mulsemmedia language, MultiSEM define *ApplicationMediaNode* as an extension of *MediaNode*. It is defined using the same abstraction of *Node*. It refers to a code written in a scripting language, such as Lua [71] and JavaScript, aiming at implementing functionality not supported by the domain-specific mulsemmedia model.

The last node presented in MultiSEM is the *SettingsNode* entity. It is a special node that represents a list of global variables. These variables can be defined by the author or can represent variables directly controlled by the formatter. In the last case, the variables can represent the properties of display devices on the consumer side or define properties of the immersive environment.

## 4.2 Presentation Properties

The *PresentationProperty* entity is specialized in *EffectPresentationProperty* and *MediaPresentationProperty* as shown in Figure 4.2. These entities represent the rendering characteristics of effects and media object presentation properties respectively. The fol-

lowing subsections give more details about these properties defined in MultiSEM.

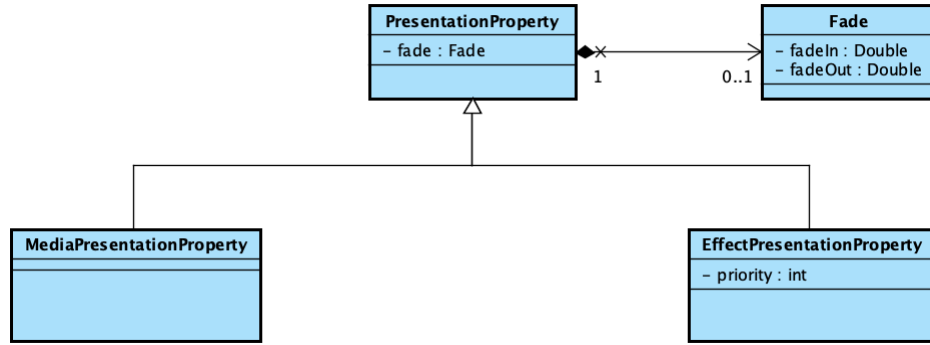


Figure 4.2: Presentation Property Entity

Additionally, *PresentationProperty* has an attribute called *fade*, that specifies fade-in and fade-out times. For sensory effects, they shall reach the defined intensity within the fade-in time. For visual media nodes, the transparency level shall reach the defined transparency value 100% when totally transparent. For the audio media node, the audio node shall reach the defined volume from value 0. The fade temporal behavior for both media and sensory effect nodes is illustrated in Figure 4.3.

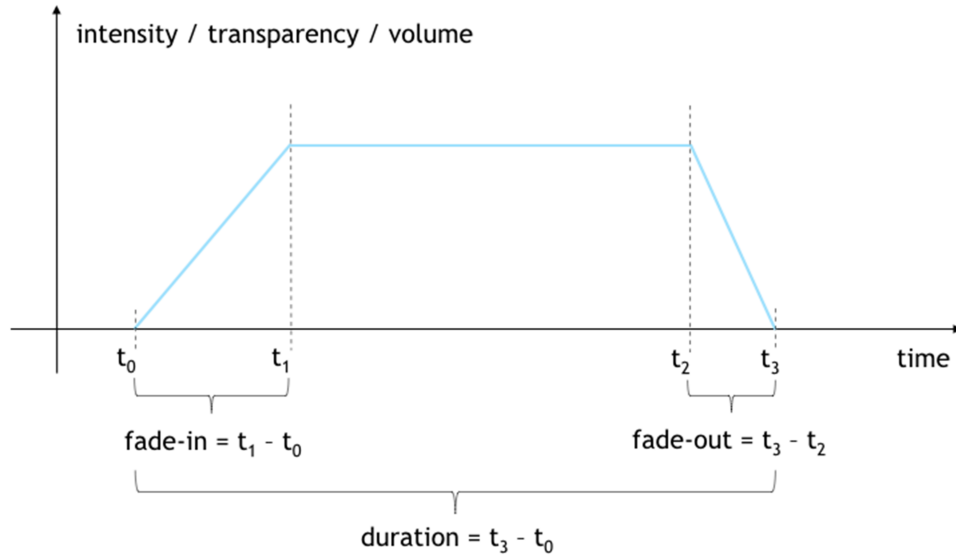


Figure 4.3: Fade-in and Fade-out Times in MultiSEM

The y-axis can represent either effect intensity, transparency level of visual media, or audio volume depending on the node type. The x-axis represents time. For example, at the  $t_0$  instant, the node starts its presentation. From  $t_0$  to  $t_1$ , the node shall reach the defined intensity/transparency/volume value according to its type. That is, the interval  $t_1-t_0$  is the fade-in time. When the node presentation finishes at  $t_2$ , the value of intensity/transparency/volume shall decrease to null value at  $t_3$ . Therefore the interval  $t_3-t_2$

represents the fade-out time.

### 4.2.1 Sensory Effect Properties

Figure 4.4 presents the diagram that defines the entities responsible for modeling the presentation properties of sensory effects. The *EffectPresentationProperty* entity contains two common attributes among the different types of effect: *priority* and *effectPositionProperty*. The *priority* attribute indicates the processing order of effects when a set of effects should be rendered at the same time. A value of 1 indicates the highest priority, and greater than one represents lower priorities.

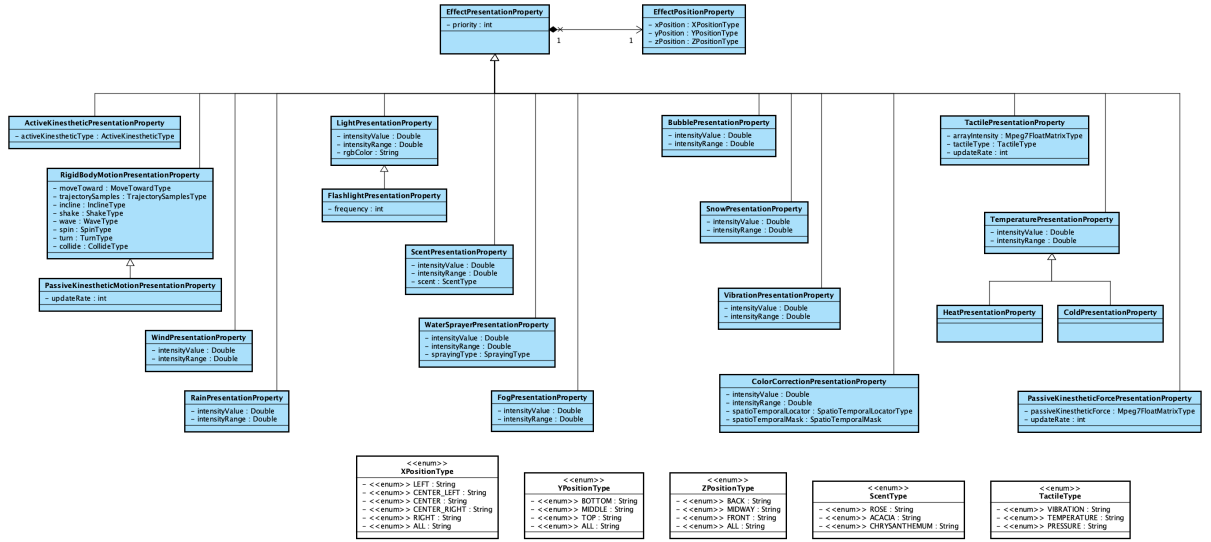


Figure 4.4: Sensory Effect Presentation Property

The second attribute refers to the *EffectPositionProperty* entity. It defines the spatial model provided by MultiSEM, which is based on the MPEG-V spatial model [78]. Therefore MultiSEM also uses the X, Y, and Z axes to place effects in immersive environments. An effect position describes the region where this effect is expected to be received from the user's perspective. Figure 4.5 illustrates the different regions we can specify to place an effect.

All types of regions are modeled in the *XPositionType*, *YPositionType* and *ZPositionType* enumerations as shown in Figure 4.4. They also define a value of *ALL* indicating any location on the axis. For instance, we can place an effect defining *LEFT* for X-axis, *ALL* for Y-axis and *BACK* for Z-axis. That is, it describes that the effect is received from the left-back side of the user and any location on the Y-axis.

The *EffectPresentationProperty* class has subclasses to represent the rendering charac-



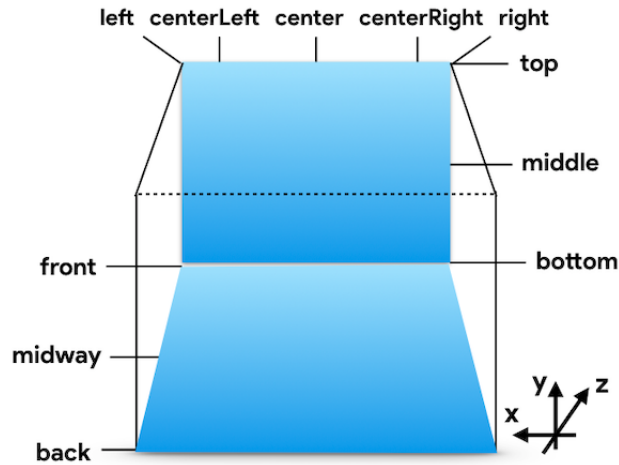


Figure 4.5: MPEG-V Spatial Model

teristics of each effect type. All these properties are based on *Part 3: Sensory Information* of MPEG-V [122]. Following the order of presentation property types depicted in Figure 4.4, the *ActiveKinestheticPresentationProperty* entity has an attribute, *activeKinestheticType*, to describe three forces ( $f_x$ ,  $f_y$ ,  $f_z$  in newton) and three torques ( $t_x$ ,  $t_y$ ,  $t_z$  in N-mm) as a reaction from a user interaction. *RigidBodyMotionPresentationProperty* has attributes to define the movement of a rigid object such as *moveToward* defining the horizontal and vertical directions of the motion in terms of angle, its trajectory, rotation (pitching (X-axis), yawing (Y-axis) and rolling (Z-axis)) represented by the *incline* attribute, *shake* describing continuously repeated movement, *wave* referring to a continuous up and down motion, *spin* describing the spinning motion, *turn* defining the motion of moving objects toward left or right, and *collide* describing the motion of a moving object colliding against another object. In addition to the inherited attributes from *RigidBodyMotionPresentationProperty*, *PassiveKinestheticMotionEffectProperty* contains the *updateRate* attribute to indicate the number of data updates per second.

The *intensityValue* and *intensityRange* attributes are defined for effects related to wind, rain, light, flashlight, scent, water sprayer, fog, bubble, snow, vibration, color correction, and temperature. The first attribute describes the absolute intensity within the intensity range in units according to the effect type. The default unit for wind effect is the Beaufort scale; for rain, scent, water sprayer, fog, bubble, and snow effects is ml/h; for light and flashlight is lux; for vibration effect is Hertz; for color correction effect is *on* and *off* with respect to 1 (on) and 0 (off); and for temperature effects is Celsius. Using the Hertz scale, we can describe a vibration effect with an intensity of 20 Hertz in a range between 0 and 50.

With regard to *LightPresentationProperty*, it has the *rgbColor* attribute to describe the color of the light effect as RGB. Its subclass, *FlashlightPresentationProperty*, adds the *frequency* attribute to describe flickering in times per second. *ScentPresentationProperty* has the *scent attribute*, which is an enumeration of scent types such as Rose, Acacia, Chrysanthemum etc as defined in Annex A.2.8 of MPEG-V, Part 6 [130]. In addition to the intensity attributes, *WaterSprayerPresentationProperty* has another attribute, *sprayingType*, to identify the sprayed material such as purified water. The entity that describes the color correction effect has two attributes in addition to the intensity ones. The *spatioTemporalLocator* or *spatioTemporalMask* attribute defines the region within a video segment to which the color correction effect is applied. Both regions are defined according to the MPEG-7 standard [1]. If no region is defined, the color correction effect is applied to the entire scene. *TactilePresentationProperty* has the *arrayIntensity* attribute that provides the intensity value of individual actuator of a tactile device. The intensity unit is defined according to the type of tactile effect. For example, the intensity unit of a vibrotactile effect is mm (amplitude). For the thermal-tactile effect, the value is specified in Celsius. For the pressure-tactile effect, the intensity is defined in  $N/mm^2$ . The *tactile-Type* attribute is an enumeration of tactile types such as vibrotactile, thermal-tactile and pressure-tactile. As in *PassiveKinestheticMotionEffectProperty*, the *updateRate* attribute defines the number of data updates per second.

*TemperaturePresentationProperty* has two subclasses to represent the heat and cold effect. For both effects, the intensity value and range are defined. *PassiveKinestheticForcePresentationProperty* has two attributes: *passiveKinestheticForce* and *updateRate*. The first one is an  $m \times 6$  matrix, in which  $m$  rows are the number of samples and the six columns contain three forces in newton and three types of torques in N.mm. The *updateRate* attribute defines the number of data updates per second.

### 4.2.2 Media Properties

The diagram that contains entities to represent the presentation properties of media nodes is depicted in Figure 4.6. *MediaPresentationProperty* has four subclasses: *AudiovisualPresentationProperty*, *AudioPresentationProperty*, *VisualPresentationProperty* and *TextPresentationProperty*. Each one of them defines properties related to the media type such as video, audio, image and text.

*VisualPresentationProperty* contains attributes to characterize visual media such as video, image, and text. The *transparency* attribute indicates the degree of transparency of

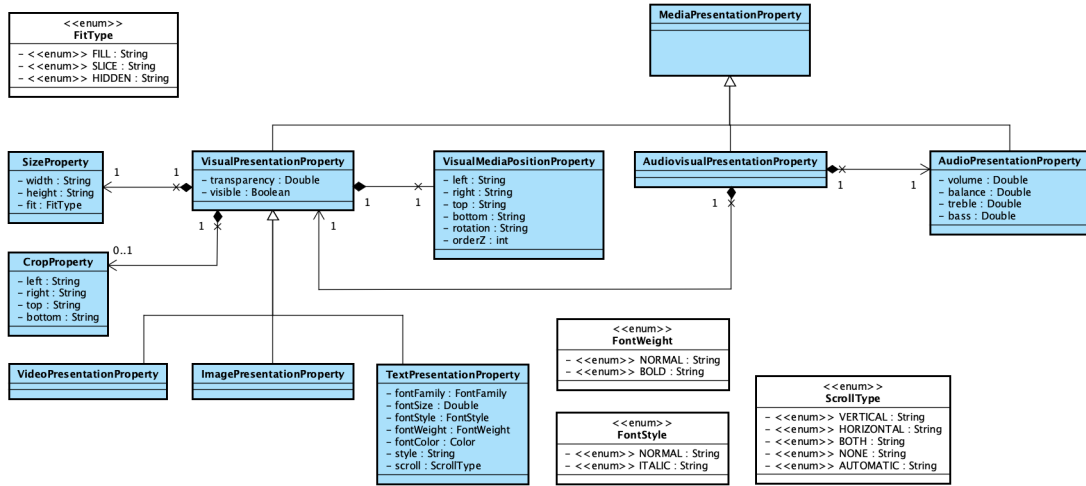


Figure 4.6: Media Node Presentation Property

visual media nodes. Its value is within a range between 0 (totally opaque) and 1 (totally transparent). The *visible* attribute is a boolean value to indicate if the presentation of media nodes should be shown or hidden on the display device. *VisualPresentationProperty* also contains an attribute of the *SizeProperty* type. It defines the width and height of the media node. Additionally the *fit* attribute is defined in *SizeProperty*. This attribute defines how visual media nodes are presented on the display screen. It contains three values: *Fill*, *Slice* and *Hidden*. The first one resizes the media node to fill the entire screen without keeping the media's aspect ratio. Differently from *Fill*, the *Slice* type keeps the media aspect ratio filling the entire screen. *Hidden* does not change the media size. That is, it keeps the original dimension of the media node. The *CropProperty* entity defines a crop of the media dimension specifying the following attributes: *left*, *right*, *top* and *bottom*. *VisualMediaPositionProperty* specifies the left, right, top, bottom to place the visual media nodes related to the screen. Additionally this entity contains the rotation and order-z attributes. The first one defines the rotation of the media node related to its center. *Order-z* defines the position of media nodes in the z-axis. That is, a media node, *m1*, with an order-z value bigger than the order-z value of another media *m2*, is shown over *m2* on the display screen. Figure 4.7 illustrates the attributes of the crop, position and size properties.

Regarding *TextPresentationProperty*, it adds specific attributes for textual media such as the font type, size, style, weight, and color. It also defines the *style* attribute to allow the association of a style sheet file with the media node to format its textual content. The scroll attribute indicates which scroll type (vertical, horizontal, both, or automatic) is used to visualize the textual content. *AudioPresentationProperty* specifically defines

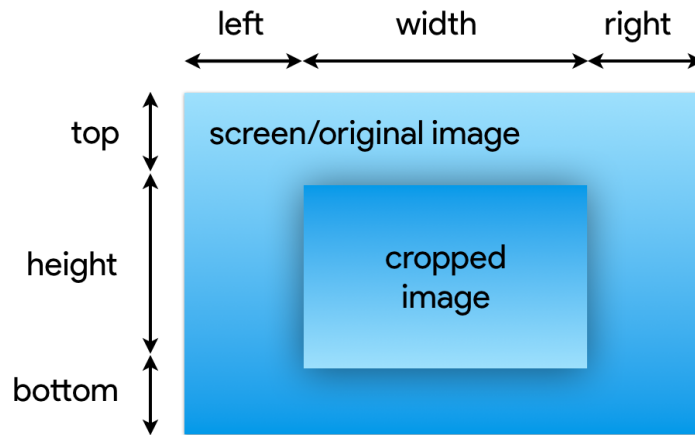


Figure 4.7: Attributes of Crop, Position and Size Entities

properties for audio media nodes such as volume, balance, treble, and bass. The *Audio-visualPresentationProperty* entity is composed of audio and visual properties to describe media nodes that contain both audio and video content.

## 4.3 Events

So far, we discussed how nodes are described in MultiSEM. As in multimedia models, MultiSEM also specifies temporal relationships among nodes in order to synchronize them. Temporal synchronization can be represented using different techniques. One of them is the event-based paradigm, which offers greater expressiveness [25] and is applied to several proposals of authoring tools and multimedia models [47, 66, 96, 116, 117, 52]. MultiSEM also uses the event-based paradigm based on the NCM events [116, 117]. The model thus allows effects to be temporarily synchronized with other nodes, media, or sensory effect nodes, according to the occurrence of events in mulsemmedia applications.

In the temporal scenario, an event can be defined as an occurrence in time that can be instantaneous or can last some time interval [98]. Moreover, it can be synchronous, when its occurrence instant is known a priori, or asynchronous, when the event occurrence instants are not previously known. User interactions with multimedia applications represent asynchronous events. Therefore, we cannot know if the interaction event will happen or when if it will. Furthermore, the state variable assessments also represent asynchronous events. In the event-based paradigm, the temporal scenario is specified through temporal relationships among events. For instance, a video node is started when an image node finalizes its presentation. To define those relationships, MultiSEM entities are based on events and will be discussed in detail in Section 4.4.

MultiSEM defines the following types of events: presentation, attribution, and selection events, which are represented in the class diagram of Figure 4.8. The presentation event refers to the presentation of a node. It can indicate, for example, if a video node is playing or paused or if a sensory effect is being rendered in the physical environment. The selection event represents the user interaction with nodes using an input device such as a keyboard, mouse, or remote control. The interaction moment occurs when the node that should be selected is being presented. Last but not least, the attribution event is responsible for representing the value changes of node properties such as the transparency of a video, volume of audio, or the intensity of a sensory effect. Those properties are defined in the *attributeName* of the *AttributionEvent* class.

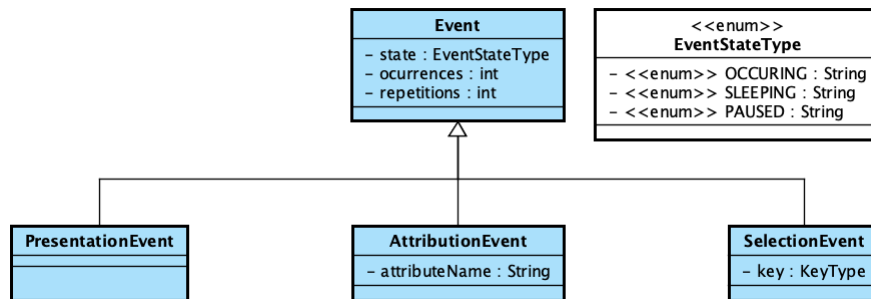


Figure 4.8: MultiSEM Events

MultiSEM assigns a state machine to each one of those events defined to manage the several changes that might occur with an event during document presentation. This machine state is defined in the NCM model [116, 117] and it presents the following states: *occurring*, *paused* and *sleeping*. *Occurring* means that the node is being presented, *paused*, its presentation is paused and *sleeping* defines that the node presentation is stopped. Additionally, each event has attributes. The first one is *occurrences*, which counts how many times the event has occurred. The *repetitions* attribute defines the number of times the event should occur from the last occurrence. This attribute is only applied to presentation and attribution events. Figure 4.9 shows the state machine to manage event states in MultiSEM and its transitions.

In this machine, the initial state is represented as a small black circle and the other states are represented by a node (rectangle). Edges (arrows) show transitions between states. Arrows are labeled with action names that trigger that transition. Those actions are triggered when a condition is fulfilled. In Section 4.4, the relation between actions and conditions will be discussed in detail. In Figure 4.9, an action named *start* triggers state transition from *sleeping* to *occurring*. From the *occurring* state, there are three outgoing edges. Two of them reach the *sleeping* state through the *abort* and *stop* or *natural end*

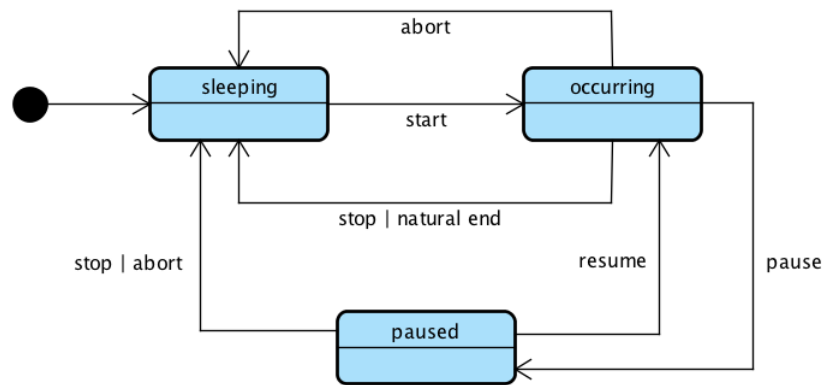


Figure 4.9: Event State Machine

actions. The other one reaches the *paused* state whose input is the *pause* action. The *paused* state has two outgoing edges. One directs to *sleeping* state with stop or abort action and the other to *occurring* state with the *resume* action. For instance, suppose a temporal relation defines that a video node is started when an image node is ended. The behavior of this machine is the following. There are two event state machines: one for the presentation event of the video node and a second machine for the presentation event of the image node. When the transition from the *occurring* state to *sleeping* state occurs (the *stops* transition) for the presentation event of the image node, the presentation of the video node begins. To start the video node, in its state machine, the *starts* transition occurs changing the state from *sleeping* to *occurring*. Table 4.1 summarizes those transitions and actions.

Table 4.1: State Machine Transitions [116, 117]

Transition	Name
sleeping → occurring	starts
occurring → sleeping	stops
occurring → sleeping	aborts
occurring → paused	pauses
paused → occurring	resumes
paused → sleeping	stops
paused → sleeping	aborts

Notice that the *Event* entity does not make part of the authoring model definition. However, the roles in relationships might be based on these transitions. This entity and its state machine are only instantiated in the execution phase of mulsemmedia documents. In other words, players/renderers are responsible for implementing those event state machines.

## 4.4 Relations

In previous sections, we defined document nodes, their presentation and rendering characteristics, and events. This section discusses how MultiSEM represents the temporal relations among those nodes using state machine transitions. Figure 4.10 shows the class diagram for modeling relations in MultiSEM based on the concept of hypermedia connectors [91].

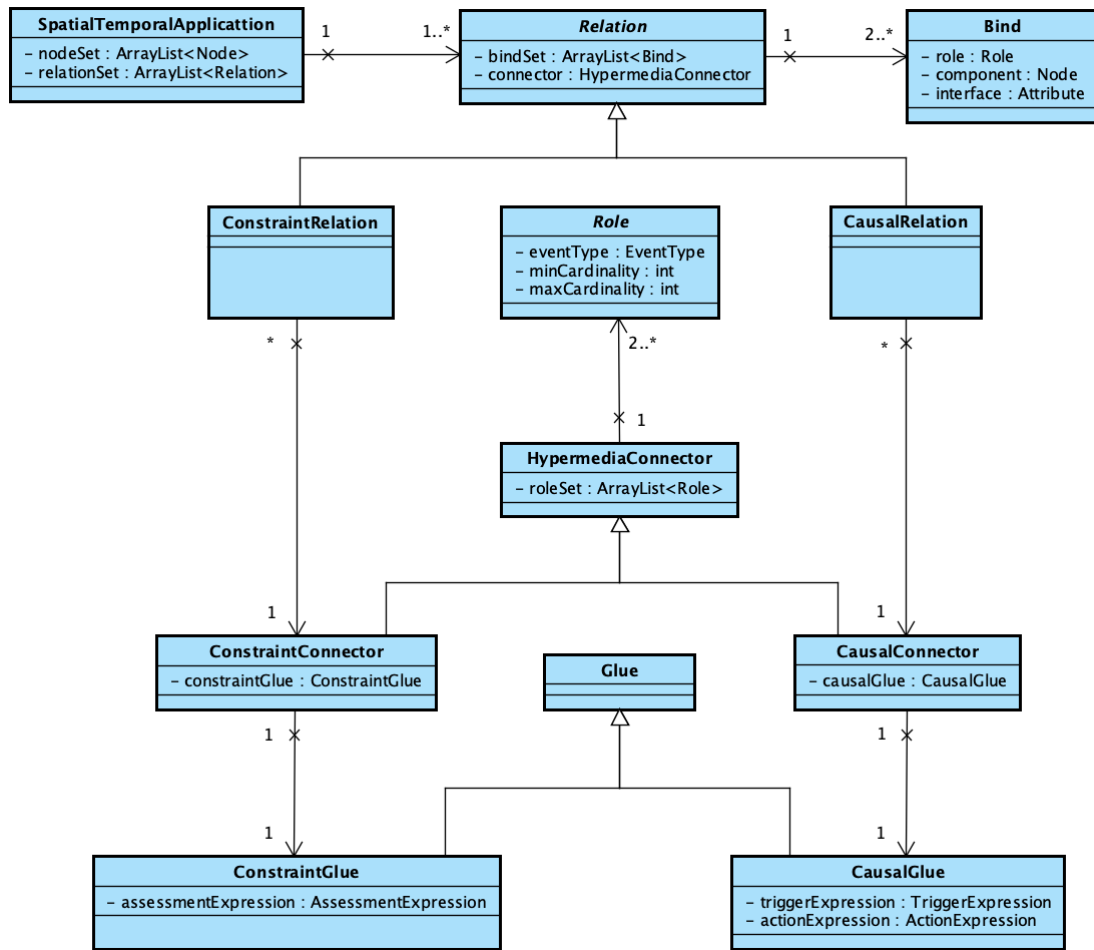


Figure 4.10: Relation and Connector Class Diagram

The *SpatialTemporalApplication* entity represents the mulsemmedia document. It is basically composed of nodes (*nodeSet*) and relations (*relationSet*) among them. A relation can be a constraint or causal relation, as we can see in Figure 4.10. With the first one, the model allows the definition of temporal or spatial constraints based on events without any causality involved in relationships. For instance, a constraint can define that a node presentation must be finished with the start of another node. Notice that the occurrence of one node's presentation without the other's occurrence also satisfies the constraint.

On the other hand, causal relations allow the expression of causality in relationships. Those causal relations are also based on events and define a condition that must be satisfied to trigger an action. For instance, a causal relation can start a node presentation when another node presentation finishes.

To define both types of relation, MultiSEM makes use of hypermedia connectors (*connector*) [91] and a set of binds (*bindSet*). A hypermedia connector specifies a multipoint relation regardless of which nodes will interact using the relation. To this end, the *HypermediaConnector* class specifies a set of interface points (*roleSet*), called roles, which are represented by the *Role* class in Figure 4.10.

Regarding the *Bind* class, it contains the *role* attribute that refers to a role defined in the connector. It also contains a *component* attribute that specifies a node in order to associate it with the bind role. Each role defines an event type, specified by the *eventType* attribute. Therefore a role can refer to the presentation, selection or attribution events. The *Role* class also has the *minCardinality* and *maxCardinality* attributes to respectively indicate the minimal and maximal number of participants that can be associated with this role by using the *Bind* element. If a role event type is a selection, the *Event* entity may define the selection key used for interacting. In case of an attribution event type, *Event* must specify the associated attribute name. The *Role* entity is specialized in different types that will be discussed in Section 4.4.1.

According to the relation type, *ConstraintConnector* or *CausalConnector* is used to define the relation. In addition to the set of roles, both connectors refer to another element, called *Glue*. It describes how roles interact by combining events according to constraint or causal semantics. Therefore connectors give semantics to relations. Both relation types refer to a connector and associate different nodes with connector roles by the *bindSet* attribute. This way, relations can reuse the same connector expressing the same causal or constraint semantics for different nodes.

Analogous to *HypermediaConnector*, *Glue* is also specialized in *ConstraintGlue* and *CausalGlue*. The first one contains an assessment expression of event and node attributes. *CausalGlue* contains a trigger and action expressions in order to relate the connector roles. Both glue types will be discussed in details in Section 4.4.2.



### 4.4.1 Role

As discussed in the previous section, a *Role* must be associated with an event type. According to the relation type, different roles may be used to define a hypermedia connector. The *Role* entity has three specific subclasses as Figure 4.11 shows: *ConditionRole*, *PropertyRole* and *ActionRole*. All role types can be used in a causal connector, but only *PropertyRole* can be used in constraint connectors. A causal connector must have at least one condition role and one action role. Concerning a constraint connector, it must contain at least two property roles to define a constraint relation.

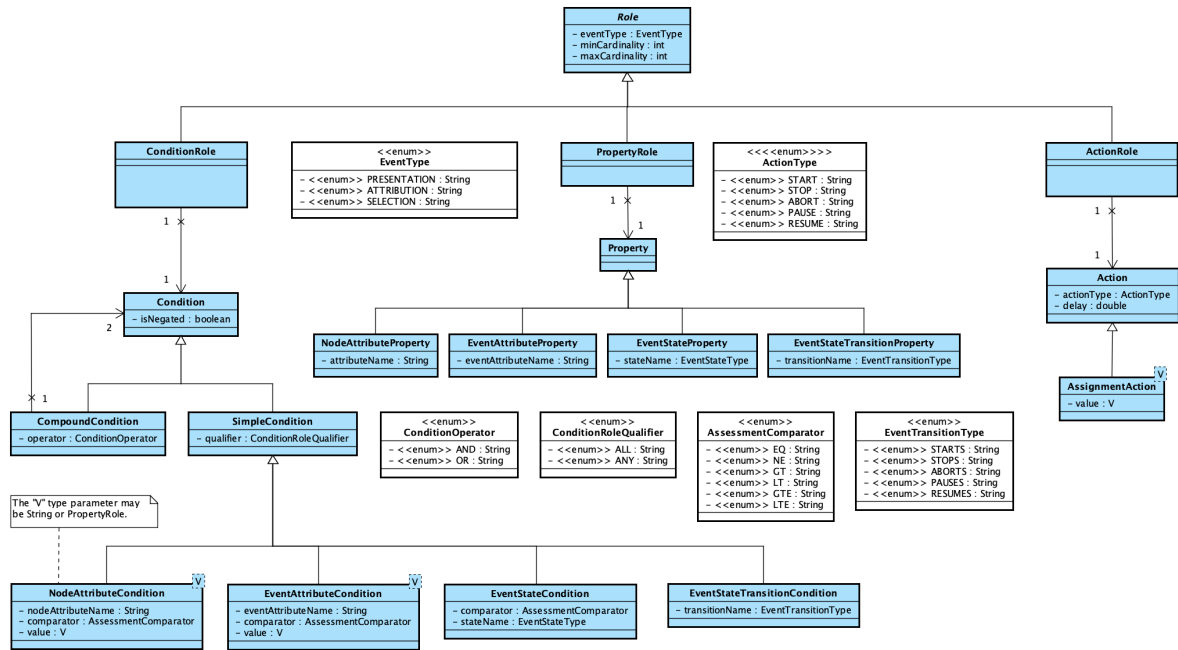


Figure 4.11: Role Class Diagram

The *ConditionRole* entity represents a condition role that must be satisfied to trigger an action defined in a causal relation. A condition can be simple (*SimpleCondition*) or compound (*CompoundCondition*). Both types contain the *isNegated* attribute to negate any condition. Regarding the *CompoundCondition* entity, it also contains two conditional statements (simple or compound) over the same event, and the operator between the statements (*AND* or *OR*).

A simple condition has four subclasses according to the type of properties to be evaluated. The *NodeAttributeCondition* entity represents the assessment of attributes of the condition role node (*nodeAttributeName*) with a specific value or with other node attributes associated with a property role. The *PropertyRole* entity is discussed later. To refer to different values, the *value* attribute type is parameterized. Therefore it can

indicate a specific value or refer to a property defined in a *PropertyRole*. The last attribute of *NodeAttributeCondition* is *comparator* to indicate a comparator to assess the values.

Analogously *EventAttributeCondition* uses a comparator and a parameterized value attribute to assess a property. However, in this case the property corresponds to an event attribute such as the *occurrences* attribute. *EventStateCondition* assesses the state of the event (*sleeping*, *occurring* and *paused*) associated with the condition role. Additionally *EventStateTransitionCondition* evaluates when the role event executes the transition specified by the *transitionName* attribute. This last condition type is used to temporally synchronize nodes based on transitions that occur in their event state machines.

For all types of simple conditions, the *qualifier* attribute is defined. It should be specified when several nodes may be associated with the same role. In this case, the *ALL* value indicates that all conditions should be satisfied. On the other hand, *ANY* indicates that at least one condition should be satisfied to trigger the action associated with the causal connector.

Concerning the *PropertyRole* entity, it represents different properties that can be referred to in the mulsemmedia document. As shown in Figure 4.11, this role can represent different types of property. A property may refer to a node attribute, an event attribute, an event state, or an event state transition similar to the condition types of *ConditionRole*.

In addition to the condition role, it is needed to specify an action role (*ActionRole*) in order to define a causal connector. An action role contains the *actionType* attribute to indicate the action type (*START*, *STOP*, *ABORT*, *PAUSE* and *RESUME*) to be triggered when the corresponding condition role is satisfied. *ActionRole* also defines a delay to trigger actions.

Additionally a specific action role is also defined as *AssignmentAction*. This action type represents an action to set a value to the node attribute associated with the action role by a bind specification. This action type is related to the attribution event. Analogously to *NodeAttributeCondition* and *EventAttributeCondition*, the *AssignmentAction* entity contains the parameterized attribute *value* to refer to not only an absolute value but also a property. Note that all property types can be used in condition and action roles to assess a value and set a node attribute, respectively.

### 4.4.2 Glue

Hypermedia connectors are composed of roles and a specific glue element according to the relation type they represent, constraint or causal. In the exact connector specification, we can define several roles for different event types. Each role is related to the others by the *Glue* entity to give the connector the constraint or causal semantics. As discussed in the previous section, there are two types of relation and glue: constraint and causal. As shown in Figure 4.12, the constraint glue only defines an assessment expression between property roles with no causality involved. That is, no condition role triggers an action role.

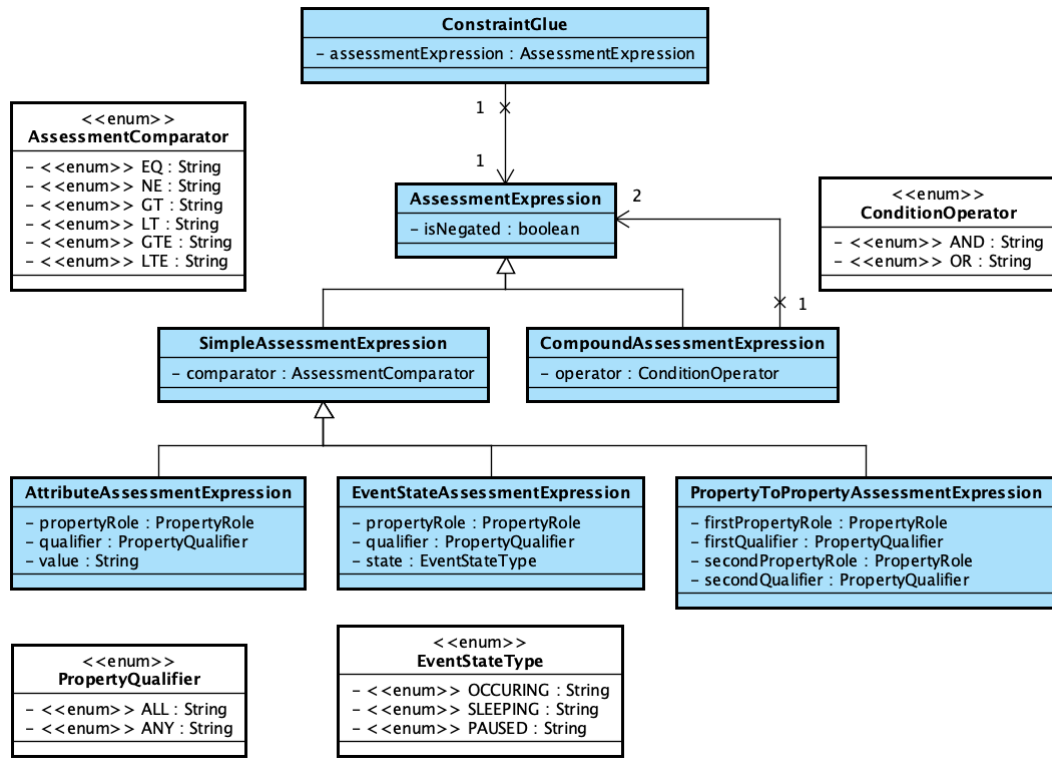


Figure 4.12: Constraint Glue Class Diagram

Similar to the compound condition for *ConditionRole*, *AssessmentExpression* has an attribute (*isNegated*) to negate the expression. The expression can also be simple or compound. The compound assessment expression has an operator (*AND* or *OR*) to relate the two expressions, which can be simple or compound. The simple assessment expression has three types of subclasses as shown in Figure 4.12. *AttributeAssessmentExpression* is responsible for comparing a property role related to node or event attributes with a value. *EventStateAssessmentExpression* verifies if the value of a property role that refers to an event state is equal to the value of the *state* attribute. And *PropertyToPropertyAssessmentExpression* allows the comparison between two property roles. Qualifiers may be

defined for all types of simple assessment expressions when more than one component is associated with the property role. The *ALL* qualifier indicates that all properties must be considered. *ANY* indicates at least one must be considered. Notice that there is a difference between that expression composition and the condition composition in the condition role representation. In the glue element, roles of different event types are combined, while only one type of event is involved in the composition of the condition role.

The constraint glue is very helpful to define spatial synchronization among media nodes. For example, it can specify that two media nodes must be aligned to the top by using a simple statement for testing if the node *top* property values are equal. In this case, the role event type is an attribution.

Concerning a *CausalGlue*, it contains one trigger expression and one action expression, as shown in Figure 4.13. The trigger expression can be negated and may define a delay that is used in its evaluation. Suppose that at the  $t$  instant, a trigger expression  $C$  is true. In the interval  $[t, t + \text{delay}]$ , the trigger remains true. A trigger expression can be simple or compound. *SimpleTriggerExpression* refers to a condition role and defines a qualifier (*ALL* or *ANY*) in case of multiple components be associated with that condition role. *CompoundTriggerExpression* is composed of an operator and two expressions: two trigger expressions or one assessment (*AssessmentExpression*) and one trigger expression. A compound trigger can be composed of other compound trigger expressions. Consequently, it can contain any number of condition and property roles. However, there is a limitation to keep the causal relation consistency. At least one condition role must refer to an *EventStateTransitionCondition*.

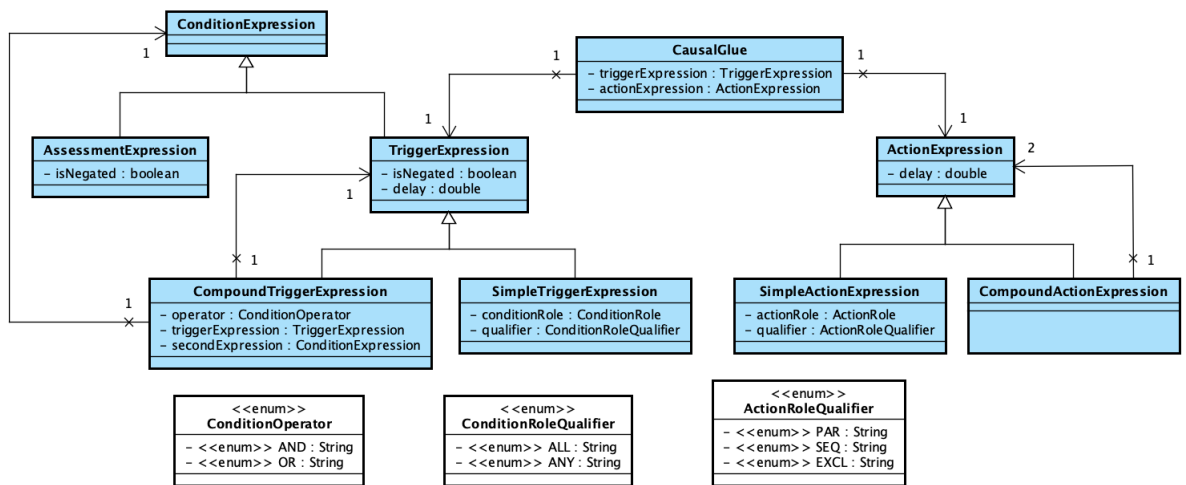


Figure 4.13: Causal Glue Class Diagram

Concerning an *ActionExpression*, it can also be simple or compound. For both cases,

the action expression defines a delay to trigger the actions. The simple action refers to an action role (*actionRole*) defined in the corresponding causal connector and a qualifier of the *ActionRoleQualifier* type. The *PAR* and *SEQ* qualifiers define that the action defined in *actionRole* must be triggered in parallel and sequence for all nodes associated with that action role, respectively. The *excl* qualifier specifies that only one of the action roles should be triggered (the mulsemmedia formatter or user should decide which one is triggered). In addition, the *CompoundActionExpression* entity contains the *operator* attribute, which also has the *PAR*, *SEQ* and *EXCL* values. However, in this case, they operate on the different actions that compose the compound action expression. For instance, an action composed of *START* and *STOP* actions with the *PAR* operator defines that those actions need to be triggered in parallel. Appendix B shows the full UML diagram of MultiSEM.

### 4.4.3 Predefined Connector Base

This section presents the predefined hypermedia connector base MultiSEM provides to support mulsemmedia graphical tools with temporal view editing as the primary GUI approach. We focus on that GUI approach since most mulsemmedia authoring tools use it, as discussed in Section 3.4. In Chapter 6, we present how those predefined connectors can aid STEVE 2.0 in providing temporal relations. Moreover, they also support our proposal MultiSEL, an XML-based language for specifying 360° mulsemmedia applications. This language will be discussed in Chapter 5.










We divide the MultiSEM's predefined connector base into four parts: connectors for Allen's temporal relations, connectors for interactivity relations, connectors for synchronous relations with attribution action, and connectors for relations with statement assessment.

#### 4.4.3.1 Part 1: Allen's Temporal Relations

This part defines several synchronous temporal relations giving a causal interpretation to the Allen's temporal relations between time intervals [11]. They are as follows: *Starts*, *Starts\_Delay*, *Finishes*, *Finishes\_Delay*, *Meet*, *Meet\_Delay*, *Met\_By*, *Met\_By\_Delay* and *Before*. Table 4.2 shows the graphical representations and descriptions for those relations.

Those synchronous relations represent the presentation event, as discussed in Section 4.3, and comprise one primary node (green rectangle) and one or more secondary nodes

Table 4.2: Temporal Relations of the Part 1 of MultiSEM's Predefined Connector Base

Relation	Symbol	Description
<i>Starts</i>		Nodes begin when the primary node starts
<i>Starts _Delay</i>		Nodes begin with delay when the primary node starts
<i>Finishes</i>		Nodes end when the primary node finishes
<i>Finishes _Delay</i>		Nodes end with delay when the primary node finishes
<i>Meet</i>		Nodes begin when the primary node finishes
<i>Meets _Delay</i>		Nodes begin with delay when the primary node finishes
<i>Met_By</i>		Nodes end when the primary node starts
<i>Met_By _Delay</i>		Nodes end with delay when the primary node starts
<i>Before</i>		Present nodes sequentially with a delay between them when the primary node finishes

(gray rectangle). The primary node plays the connector's condition role (*starts* or *stops*) and the secondary ones play the connector's action role (*start* or *stop*). Those types of condition and action are defined according to our discussion in Section 4.4.1.

For instance, Listing 4.1 shows the causal hypermedia connector that represents the *Starts* relation in the XConnector language [91]. The connector has a simple condition role whose event type is *presentation* and its event state transition is *starts*. The connector's action role defines the action type *start* for the *presentation* event. Therefore, the primary node of the *Starts* relation is bound to the connector's condition role (conditionRole id="x") and the secondary ones to the action role (actionRole id="y"). We specify the causal connectors for the other relation types using the XConnector language [91] in Appendix A.1.

```

1
2 <!-- Relation "Starts" -->
3 <xconnector id="starts" xsi:type="CausalHypermediaConnector" >
4   <conditionRole id="x" eventType="presentation">
5     <condition xsi:type="EventTransitionCondition" transition="starts"/>
6   </conditionRole>
7   <actionRole id="y" eventType="presentation" actionType="start"/>
8   <glue>
9     <triggerExpression xsi:type="SimpleTriggerExpression" conditionRole="x" />
10    <actionExpression xsi:type="SimpleActionExpression" actionRole="y"/>
11  </glue>
12 </xconnector>

```

Listing 4.1: Causal Connector for *Starts* Relation

For *Starts \_Delay*, *Finishes \_Delay*, *Meet \_Delay*, and *Met\_By \_Delay*, the connector's action role also has a *delay* attribute. In this case, the action is triggered after that delay

when the condition is satisfied.

#### 4.4.3.2 Part 2: Interactivity Relations

As in synchronous relations, we also have the primary and secondary nodes for representing interactivity. However, the primary node represents the media item users need to interact with using the interactivity key to trigger the associated action. For example, the relation type *onSelectionStart* specifies that when the user interacts with the relation primary node, the *secondary* one will be started. Listing 4.2 shows that interactivity relation specified in XConnector.

```

1
2 <!-- Relation "OnSelectionStart" -->
3 <xconnector id="onSelectionStart" xsi:type="CausalHypermediaConnector">
4   <conditionRole id="x" eventType="selection">
5     <condition xsi:type="EventTransitionCondition" transition="ends"/>
6   </conditionRole>
7   <actionRole id="y" eventType="presentation" actionType="start"/>
8   <glue>
9     <triggerExpression xsi:type="SimpleTriggerExpression" conditionRole="x" />
10    <actionExpression xsi:type="SimpleActionExpression" actionRole="y"/>
11  </glue>
12 </xconnector>

```

Listing 4.2: Causal Connector for *OnSelectionStart* Relation

In Appendix A.2, we present other types of interactivity relations such as *OnSelectionStop*, *OnSelectionSet*, *OnSelectionStartStop* and *OnSelectionStartStopDelay*.

#### 4.4.3.3 Part 3: Synchronous Relations with Attribution Action

This part specifies the same synchronous relations defined in Part 1, adding an attribution action (value assignment) to them. For example, we define the *StartsSet* connector. It represents the same causal temporal semantic of the *Starts* relation, additionally, sets a value to a document or node property. Listing 4.3 shows the XConnector specification for the *StartsSet* relation. The specifications for the other synchronous relations with an attribution action are given in Appendix A.3.

```

1
2 <!-- Relation "StartsSet" -->
3 <xconnector id="startsSet" xsi:type="CausalHypermediaConnector">
4   <param name="setValue"/>
5   <conditionRole id="x" eventType="presentation">
6     <condition xsi:type="EventTransitionCondition" transition="starts"/>
7   </conditionRole>

```

```

8   <actionRole id="y" eventType="presentation" actionType="start"/>
9   <actionRole id="z" eventType="attribution" actionType="start" finalValue="$setValue"
   />
10  <glue>
11    <triggerExpression xsi:type="SimpleTriggerExpression" conditionRole="x" />
12    <actionExpression xsi:type="CompoundActionExpression" operator="par">
13      <firstAction actionRole="y" qualifier="par"/>
14      <SecondAction actionRole="z" qualifier="par"/>
15    </actionExpression>
16  </glue>
17 </xconnector>

```

Listing 4.3: Causal Connector for *StartsSet* Relation

#### 4.4.3.4 Part 4: Relations with Statement Assessment

Part 4 provides connectors that apply statement assessments to interactivity and synchronous relations. Those assessments are defined according to MultiSEM's entities presented in Section 4.4.1. In those relations, actions are triggered only if the statement is satisfied (returns *true*) and the presentation or selection event happens with the relation's primary node. For instance, the *conditionalStarts* relation defines that the secondary nodes will start only if the primary node starts and the statement assessment is also satisfied. Listing 4.4 specifies that relation in XConnector. The other relations with statement assessments MultiSEM provides are specified in Appendix A.4.

```

1  <!-- Relation "ConditionalStarts" -->
2  <xconnector id="conditionalStarts" xsi:type="CausalHypermediaConnector">
3    <param name="pAttributeName">
4    <param name="pValue">
5    <param name="pComparator">
6    <propertyRole id="p" eventType="attribution">
7      <property xsi:type="NodeAttributionProperty" attributeName="pAttributeName"/>
8    </propertyRole>
9    <conditionRole id="x" eventType="presentation">
10     <condition xsi:type="EventTransitionCondition" transition="starts"/>
11   </conditionRole>
12   <actionRole id="y" eventType="presentation" actionType="start"/>
13   <glue>
14     <triggerExpression xsi:type="CompoundTriggerExpression" operator="AND">
15       <trigger xsi:type="SimpleTriggerExpression" conditionRole="x"/>
16       <property xsi:type="AttributeToValueExpression" propertyRole="p" value="
pValue" comparator="pComparator"/>
17     </triggerExpression>
18     <actionExpression xsi:type="SimpleActionExpression" actionRole="y"/>
19   </glue>
20 </xconnector>

```

Listing 4.4: Causal Connector for *ConditionalStarts* Relation



## 4.5 Comparison

Since MultiSEM is based on NCM 3.0 [116, 117], this section discusses in detail which entities have been added or removed related to NCM for enhancing the development of mulsemmedia authoring environments. Also, Section 4.5.2 compares MultiSEM with the MPEG-V's sensory effect representation.

### 4.5.1 MultiSEM vs NCM

#### 4.5.1.1 Nodes

Regarding *SensoryEffectNode*, presented in Section 4.1, it is added as first-class entity in the modeling of NCM node according to the definition presented in [74]. In other words, nodes may be not only traditional media contents or settings nodes but also sensory effects. Additionally *SensoryEffectNode* has several subclasses to represent different types of effects, which are based on the MPEG-V's Sensory Effect Vocabulary (SEV) [122]. Figure 4.14 shows the NCM node modeling according to NCM 3.0 [115].

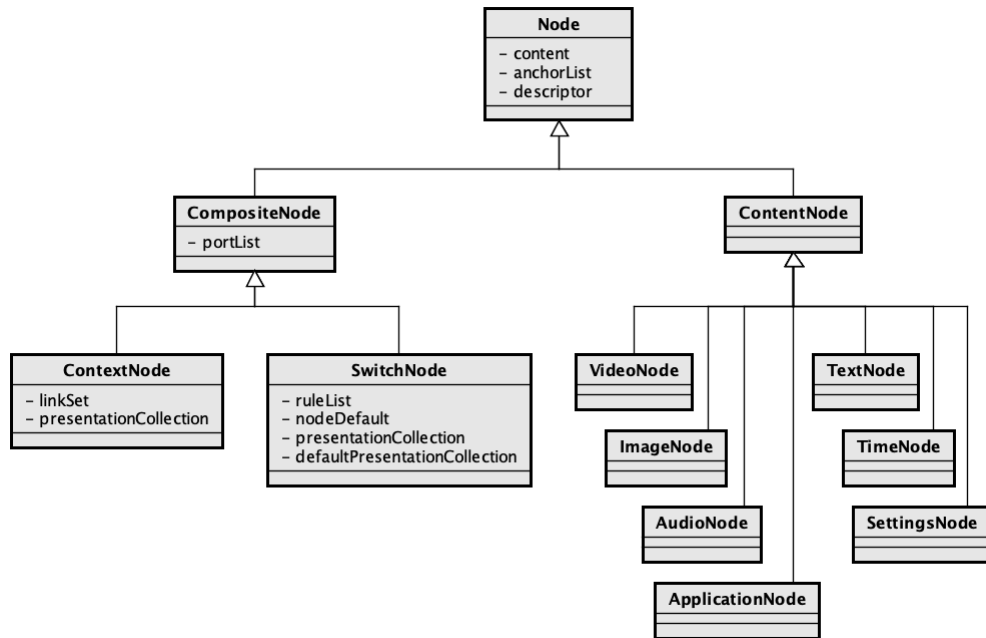


Figure 4.14: NCM 3.0 Node Diagram

In addition to sensory effect nodes, MultiSEM also defines traditional media nodes, as discussed in Section 4.1. In NCM, we have another node type called time node [115]. It aims to represent absolute time or relative time based on events. However, MultiSEM does not define that node since its temporal relations can represent time intervals between

events specifying delays in trigger and action expressions. Removing that node type, MultiSEM aims to simplify the authoring environments avoiding the graphical representation of one more node type in the authoring tools.

#### 4.5.1.2 Composition: *Context*

Also, in contrast to NCM, MultiSEM does not define composition nodes, such as context and switch nodes, which are the core of NCM. The NCM *Context* element is a type of composition node that may contain media nodes and other composition nodes recursively. In addition to these contents, they can also contain relationships among the contained nodes.

That simplification aims to avoid the representation complexity of compositions in graphical authoring environments. With that simplification, on the other hand, MultiSEM cannot represent logical structures (group of nodes and relations) that other document elements can reuse. Therefore, MultiSEM has less expressiveness regarding the element composition to gain simplicity for developing mulsemmedia authoring environments.

#### 4.5.1.3 Composition: *Switch*

Concerning the *switch* element, which is also a composite node, it improves the support of content adaptation. *SwitchNode*, as shown in Figure 4.14, represents a group of nodes in which only one is selected to be presented according to predefined rules. Those alternative nodes can also be other switches, context or media nodes.

For instance, a switch can select an audio media according to the system language. If the system language is Portuguese, then the Portuguese audio media is selected. Otherwise, the English audio is played instead. Although MultiSEM does not define the switch element, content adaptation can be specified using causal relations.

#### 4.5.1.4 Content Anchors

Another change in MultiSEM compared to NCM is that the MultiSEM node does not contain content anchors, which allow temporal synchronization with internal segments of media nodes.

Not modeling the content anchor, MultiSEM avoids the difficulties of representing them graphically in authoring environments based on temporal view. An alternative to

these anchors in the graphical authoring tools is to define media segments graphically, transforming those segments into independent media nodes.

Furthermore, the temporal synchronization between internal parts of nodes can be defined by using temporal relations with a delay to trigger their actions. For instance, we can use a causal relation to specify the following scenario. An image node should be presented when a specific content anchor of a video node begins. This content anchor corresponds to the [5s, 10s] segment of the video node. We can thus define a causal relation that specifies when the video node begins, the image node is started after a 5s delay.

#### 4.5.1.5 Presentation Properties

With regard to node presentation properties discussed in Section 4.2, MultiSEM does not define the *Descriptor* and *Region* entity as in the NCM model. Those NCM entities are responsible for modeling the presentation properties of media nodes.

NCM defines a data object representing an NCM node with all operations for manipulating this node but without presentation properties. The association of a data object with an NCM descriptor is called a presentation object. Therefore, a node can be associated with different descriptors generating several presentation objects for that same data object. In other words, this representation allows the reuse of descriptors.

MultiSEM cannot support this reuse due to the incorporation of all presentation properties into the *Node* entity defining them in the *MediaPresentationProperty* and *EffectPresentationProperty* entities, as presented in Section 4.2. The MultiSEM presentation property modeling aims to enhance the development of an authoring environment by separating the presentation properties of media and sensory effect nodes according to their types. In addition, it simplifies the specification of presentation objects by defining that a specific set of presentation properties is associated with only one node.

Table 4.3 summarizes those modeling differences between MultiSEM and NCM in order to support the development of graphical authoring tools based on temporal view for mulsemmedia applications.

### 4.5.2 MultiSEM vs MPEG-V

Regarding the differences between MultiSEM and MPEG-V, the MPEG-V standard defines the *Sensory Effect Description Language* (SEL) [122] to provide essential building

Table 4.3: Modeling Difference between MultiSEM and NCM

	MultiSEM	NCM 3.0 [116, 117]
<b>Nodes</b>	Sensory Effects and Traditional Media	Only Traditional Media
<b>Composition</b>	X	✓
<b>Content Anchors</b>	X	✓
<b>Presentation Properties</b>	<i>MediaPresentationProperty</i> and <i>EffectPresentationProperty</i>	<i>Descriptor</i> and <i>Region</i>

X (does not support); ✓ (supports).

blocks for authoring sensory effect metadata. The data format of each type of sensory effect is defined as a sensory effect vocabulary (SEV) [122]. That is, MPEG-V allows only the specification of sensory effect metadata. Unlike MultiSEM, MPEG-V cannot specify an entire mulsemmedia application by defining media items and effects' spatial and temporal behavior. Moreover, MPEG-V uses the timeline-based paradigm, which has several inherent limitations [25].

Compared to MPEG-V sensory effects, MultiSEM added other sensory effects to provide them directly in the graphical authoring environment. The following effects are added: cold and heat effects as subtypes of MPEG-V temperature effect, snow, bubble, and rain effect.

Concerning the basic attributes of sensory effects defined in MPEG-V, MultiSEM removes the ones that are not appropriate for an approach for authoring mulsemmedia documents based on events. They are as follows: the *activate*, *duration*, and *alt* attributes. The *activate* and *duration* attributes respectively describe whether the effect shall be activated and its duration. The *alt* attribute describes alternative effects for a specific effect. This structure is similar to the NCM switch element, and it is not represented in MultiSEM either owing to difficulties in representing it in graphical tools, as discussed in Section 4.5.1.2.

The next chapter presents our XML language proposal, MultiSEL (Multimedia Sensory Effect Language), for developing mulsemmedia applications with 360° visual content for virtual reality environments. That language is based on the MultiSEM entities and its predefined connector base discussed in Section 4.4.3.

## Chapter 5

# MultiSEL - An XML-based Language for 360° Mulsemmedia Applications

MultiSEL (Multimedia Sensory Effect Language) is an XML-based language for specifying mulsemmedia applications with 360° visual content for virtual reality environments. Therefore, the language allows authors to define 360° visual content with multiple sensory effects. In the VR context, those sensory effects can be rendered by multi-sensory devices that can be integrated with HMDs (Head-Mounted Displays) [13]. Moreover, authors can write MultiSEL documents with no 360° content or sensory effects.

The language is based on MultiSEM's entities, presented in Chapter 4, and we use the declarative approach to propose it. This approach is widely used by other multimedia languages, such as HTML, NCL, and SMIL, as well as the MPEG-V's sensory effect representation. Our proposal of a mulsemmedia language aims at aiding the interoperability between mulsemmedia authoring tools. Indeed, we discuss the integration between our proposal STEVE 2.0 and AMUSEVR [45], a VR mulsemmedia authoring tool, in Chapter 6.

We divide MultiSEL's functionality into six functional areas based on the modular approach widely used in the specification of XML-based languages [103, 15]. In this approach, each one of these areas also contains modules. Those modules are sets of XML elements, attributes and values that describe specific language functionalities. Table 5.1 summarizes the functional areas and their respective modules in MultiSEL as an example.

In the following sections, we describe each functional area defined for MultiSEL and their modules. Also, this chapter presents the language structure and an entire mulsemmedia document written in MultiSEL.

Table 5.1: MultiSEL's Functional Areas and Modules

Functional Areas	Modules
Structure	Structure
Components	Scene Media SensoryEffect
Interfaces	SceneInterface
Presentation and Rendering Specification	Property
Relations	TemporalSynchronization
Metainformation	Metainformation

## 5.1 *Structure* Functional Area

The *Structure* functional area contains one module named *Structure*. This module defines the basic structure of a MultiSEL document. It specifies the following elements: the document root, called *multisel*, the *head* and *body* elements. The *multisel* element has the following attributes: *id*, which is required and can contain any string to identify the MultiSEL document, *title* for giving extra information about the document, and *xmlns*, that defines the profile language. Both *head* and *body* are children elements of the root. The *head* element contains *metainformation*, which we will discuss later. And the *body* element contains *scene* and *relation* elements, which give the document behaviour itself. Those elements will be discussed in the next sections.

## 5.2 *Metainformation* Functional Area

The *Metainformation* functional area contains the *Metainformation* module. This module has two elements: *meta* and *metadata*. The *meta* element allows the definition of simple metadata about the mulsemmedia document such as its author and creation date. It has two attributes to do that. The *name* and *content* attributes to define the metadata name and its value respectively. The *metadata* element defines complex metadata by using an RDF (Resource Description Framework) tree. The *metadata* element itself is the root element of this RDF tree. The metadata can be specified in the *head* or *body* element.

## 5.3 *Components* Functional Area

The *Components* functional area contains three modules: *Scene*, *Media* and *SensoryEffect*.

The *Scene* module has the *scene* element to represent the spatio-temporal behaviour of the mulsemmedia document. Therefore, this element contains the document nodes (media and sensory effects) and their relations. In addition, it has the *port* and *property* child elements. The *port* element provides external access to the scene's inner elements. It will be discussed in Section 5.5. The *property* element defines scene variables to be used in the relations, which will be discussed in Section 5.6. Regarding the Scene's attributes, it has the *id* and *primaryComponent* attributes. The latter defines the first media node to be presented when the mulsemmedia document starts.

The *Media* module contains the *media* element, which defines traditional media items, such as audio, video, text and image. This element contains three attributes to describe it. The *src* attribute for specifying a URI that refers to the item content, the *type* attribute for defining the type of media (video, audio, etc) and an *id* attribute to uniquely identify the media item.

The *SensoryEffect* module contains the *effect* element to represent sensory effects as a first-class entity [74], following the same approach of MultiSEM. The effect element also has an *id* attribute and a *type* for specifying the type of sensory effect, such as wind, light and scent effects, according to the MultiSEM specification.

## 5.4 *Presentation and Rendering Specification* Functional Area

The *Presentation and Rendering Specification* functional area contains the *Property* module. This module defines the presentation characteristics of media nodes and the rendering characteristics of sensory effects. To this end, the module has the *property* element to specify the property name and its value. This element contains the *name* attribute to define the media property (e.g. volume, width, transparency, etc.) or sensory effect property (e.g. intensity, scent, frequency, etc.) and the *value* attribute to define the property value itself. MultiSEL specifies the position property for both media and effect nodes using XYZ coordinates to place them in the virtual reality environment. All the MultiSEL's presentation and rendering properties are based on the MultiSEM's specification. In addition, MultiSEL defines the *pip* property to indicate whether a media node is always positioned in front of the user view in the VR environment or not. The position does not need to be defined when *pip* is enabled (*true* as value). Also, the language defines the *background* property for visual media items. It defines that the media item is the scene

background when its value is set up to *true*. Another attribute to support 360° is the *loop* property. It defines that the media item is restarted when its duration ends. Moreover, we can define the *viewAngle* property to specify if the video or image item should be presented in 180° or 360° angle.

Moreover, MultiSEL has a special property for sensory effects called *intensitySpatialAttenuation*. This attribute allows the specification of a spatial attenuation of the sensory effect intensity. In other words, the intensity of sensory effects can be spatially synchronized with visual contents as shown in Figure 5.1. MultiSEL defines this attenuation by applying the inverse-square law, as the computer graphics area uses for light attenuation of 3D objects [12]. The MultiSEL’s attenuation function defines the distance between the sensory effect source (i.e. its position in the VR environment) and each tile of the visual content. We will discuss the tile concept further. This attenuation thus generates a matrix of effect intensity to represent a intensity gradient. These intensity matrices and the visual content tiles must be synchronized in both temporal and spatial domains. In Figure 5.1, the user experiences a 360° video in a multi-sensory HMD that provides scent effects. For this 360° video, the figure illustrates the intensity matrix of the scent sensory effect. The sensory effect position is defined on the bottom right corner of the 360° video, where there are some vivid purple flowers. The further away from that corner, the scent intensity decreases. The sensory effect position must be set according to the video so that the attenuation function can update all matrix’s intensities to synchronize this matrix with the video in time. This approach is based on the 360° mulsemmedia concept presented in [40].

To define this attenuation, the language provides the *intensitySpatialAttenuation* attribute for sensory effect elements. There are two ways authors can define this attenuation. In the simpler one, authors just need to define the *value* attribute for the *intensitySpatialAttenuation* property with *strong*, *moderate* or *weak*. The other way is more specific, in which authors need to define the *tileSize* and *coefficient* attributes. For the *tileSize* attribute, authors need to specify the tile width and height in px. The larger the tile size, the less grainy the attenuation is. Big tiles generate less different intensity values spatially synchronized with the visual media item. The *coefficient* attribute is used to define parameters for the attenuation function. For the inverse square law function, we have three values of coefficient to make fine adjustment to the attenuation. In addition, MultiSEL defines the *functionType* attribute to specify other attenuation functions to be implemented by mulsemmedia players.



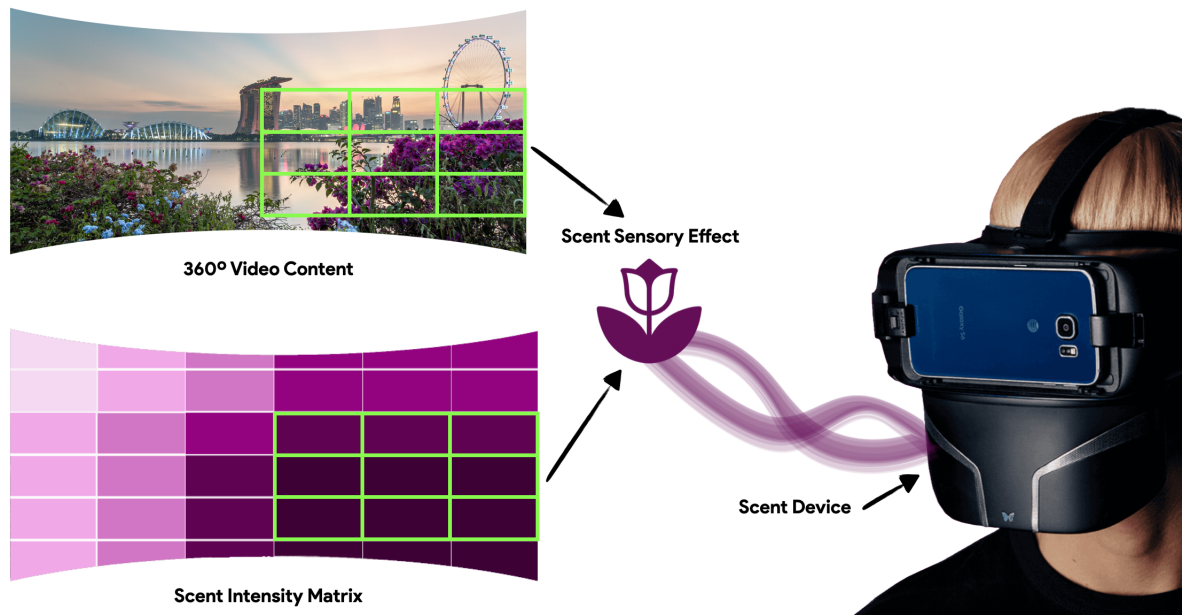


Figure 5.1: Intensity Spatial Attenuation Concept (scent device [56])

## 5.5 *Interfaces* Functional Area

The *Interfaces* functional area contains the *SceneInterface* module, which has the *port* element. This element defines interface points that node relations use to refer to inner nodes of scenes. We discuss those relations among document nodes in Section 5.6. The *port* element contains the *id* attribute for identifying the interface point and the *component* attribute for referring to an inner node of the scene which the interface point belongs. That is, the *port* element provides external access to inner items of the scene. This *component* attribute can also reference scene properties. In addition, the *port* element may contain the *interface* attribute to reference media and sensory effect properties.

## 5.6 *Relations* Functional Area

The *Relations* functional Area contains the *TemporalSynchronization* module. This module defines the *relation* element to specify temporal relations among Component's elements, *Scene*, *Media* and *SensoryEffect*. We define two basic attributes for the *relation* element: *type* and *delay*. The *type* attribute allows authors to give the temporal semantic among the relation elements. To do that, MultiSEL is based on the predefined temporal relations defined in MultiSEM's temporal synchronization model. Those predefined relations were discussed in Section 4.4.3 and specified in Appendix A. Therefore, the *type* attribute can assume the following values to define synchronous temporal relation: *starts*, *startsDe-*

*lay*, *finishes*, *finishesDelay*, *meet*, *meetDelay*, *metBy*, *metByDelay* and *before*. The *delay* attribute may be necessary according to *type*. For example, if the *type* attribute is *startsDelay* then the *delay* attribute should be defined. In this case, the relation starts the secondary nodes after the time defined by this delay. The delay can also be specified for the relation's secondary participants separately.

The *relation* element has two basic children: the *primary* and *secondary* elements. Those elements represent the primary and secondary nodes that participate in a temporal relation, as presented in Section 4.4.3. Both elements have the *component* attribute to define the scene, media or sensory effect as the primary or secondary node of the temporal relation. Also, those elements may have the *interface* attribute in order to reference inner media or sensory effect items of scenes. In that case, the *component* attribute must specify a scene identifier, and the *interface* attribute must specify a port identifier to reference an inner item of the identified scene.

MultiSEL only allows the reference of inner nodes of scenes from outside the scenes, that is, from the *body* element, and by using the port and interface elements. This brings to the MultiSEL's structure the compositionality concept [57], which NCL [103] also makes use of it. In MultiSEL, we apply this concept by interpreting the *scene* element as a composition that exposes its media and effect items to the *body* element. Compositionality supports document validation mechanisms, such as the temporal consistency, and the document maintenance. Therefore, we cannot create temporal relations inside a scene that reference an item of other scenes. We need to create a relation in the *body* element and use the *port* and *interface* elements to relate items of different scenes.

In addition, MultiSEL allows the specification of temporal relations between scenes without defining an inner node. That is, only the *component* attribute of the primary and secondary elements is defined with the scene identifiers. In this case, the relation's condition is associated with the scene's primary component if the condition is the beginning of the scene or associated with all scene's constituents if the condition is the scene end. Regarding the relation's action, it is triggered in the primary component of the scene (secondary node of the relation) when this action is a *starts* or is triggered in all scene's constituent when the action is a *stops*.

Temporal relations can also be asynchronous in MultiSEL. The language allows authors to define user interactions based on the connectors for interactivity relations, as discussed in Section 4.4.3 and specified in Appendix A.2. MultiSEL provides the *onS-electionStart* and *onSelectionStop* values for the *type* attribute of interactivity relations.

Those types defines that when the user interacts with the relation primary component, the secondary ones will start and stop respectively. The *relation* element also has the *keyCode* attribute in this case. It defines the interactivity key that users need to press to trigger the action associated with the interactivity relation. As in synchronous relations, we can also define a delay to trigger the action after user interactions.

The *set* action (value assignment) discussed in Section 4.4.1 can also be defined for both synchronous and asynchronous relations in MultiSEL. Therefore, for instance, *startsSet*, *finishesSet* or *onSelectionSet* can be defined as value for the relation's *type* attribute. The semantic of those relations were discussed in Section 4.4.3. In this case, the *relation* element has another child element, called *set*. The *set* element has three attributes: *component*, *interface* and *value*. The *component* attribute defines the media, sensory effect or scene, the *interface* attribute defines the component's property to be set, and the *value* attribute specifies the value itself.

Moreover, we can define a statement assessment, based on Section 4.4.1, to trigger the actions of synchronous and asynchronous relations in MultiSEL. In this case, the *relation* element has another child element to specify this conditional statement, named *assessment*. The *assessment* element has the *expression* attribute to define the boolean conditional expression to be assessed. To define this expression, MultiSEL provides comparators (*equal*, *not equal*, *greater than*, *less than*, *greater than or equal to* and *less than or equal to*), as presented in Section 4.4.2. This expression can also be compound by using the *AND* or *OR* operators, following the compound assessment expression we presented in Section 4.4.2. Note that the relation action is triggered only if this expression returns *true* and the presentation or selection event happens with the primary node according to the relation type. For instance, we can define the *conditionalStarts* and *conditionalFinishes* types in MultiSEL, both specified in Appendix A.4. The *conditionalStarts* relation defines that the secondary nodes will start only if the primary node starts and the expression is satisfied.

## 5.7 Language Structure

This section presents tables to summarize all the elements, their attributes and contents (child elements). We also provide a code snippet for each table to illustrates the usage of the elements. Each table represents a MultiSEL's functional area discussed in the previous sections. The elements' content specification presents the following symbols: (?)

Table 5.2: *Structure* Functional Area

Elements	Attributes	Content
multisel	<u>id</u> , title, <u>xmlns</u>	(head?, body?)
head		(meta*, metadata*)
body	<u>id</u>	(scene*, relation*, meta*, metadata*)

Table 5.3: *Metainformation* Functional Area

Elements	Attributes	Content
meta	<u>name</u> , <u>value</u>	empty
metadata		RDF tree

optional - the element may not exist or there is only one occurrence of it, (|) or, (\*) zero or more occurrences, and (+) one or more occurrences. The elements' attributes that are underlined indicate they are required.

Listing 5.1 gives a code snippet to illustrate how the MultiSEL's structure is organized using the elements of Table 5.2 and their contents.

```

1 <multisel id="sensoryEffectApp" title="My Mulsemmedia App" xmlns="MultiSEProfile">
2   <head>
3     <meta/> <!-- simple metadata -->
4     <metadata>
5       <!-- structured metadata (RDF tree) -->
6     </metadata>
7   </head>
8   <body>
9     <!-- meta and metadata -->
10    <!-- scenes -->
11  </body>
12 </multisel>

```

Listing 5.1: MultiSEL's Structure

Listing 5.2 shows a code snippet that represent how the MultiSEL's metainformation is structured using the elements of Table 5.3 and their contents. This code contains simple metadata to give document information about the author, data and language. Also, it contains an RDF tree to describe more complex metadata.

```

1 ...
2 <meta name="author" content="Douglas Mattos"/>
3 <meta name="data" content="2021-09"/>
4 <meta name="language" content="en"/>
5 <metadata>
6   <rdf:RDF
7     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
8     xmlns:dc="https://purl.org/dc/elements/1.1/">

```

Table 5.4: *Components* Functional Area

Elements	Attributes	Content
scene	id, primaryComponent	(port property media effect relation)*
media	id, src, type	(property*)
effect	id, type	(property*)

```

9
10 <rdf:Description rdf:about="https://www.midiacom.uff.br/multiselEx">
11   <dc:author>Douglas</si:author>
12   <dc:title>My first MultiSEL document</si:title>
13   <dc:description>
14     A MultiSEL document to illustrate how it works.
15   </description>
16   ...
17 </rdf:Description>
18 </rdf:RDF>
19 </metadata>
20 ...

```

Listing 5.2: MultiSEL's Structure

Listing 5.3 illustrates how the MultiSEL's components are organized inside the body element using the elements of Table 5.4 and their contents.

```

1 ...
2 <body>
3   ...
4   <scene id="scene1" primaryComponent="introVideo">
5     <!-- ports -->
6     <!-- properties -->
7     <media id="introVideo" src="path_to_the_video_file" type="video">
8       <!-- properties -->
9     </media>
10    <effect id="" type="scentEffect">
11      <!-- properties -->
12    </effect>
13    <!-- relations -->
14  </scene>
15  <scene id="scene2" primaryComponent="video2">
16    ...
17  </scene>
18  ...
19 </body>
20 ...

```

Listing 5.3: MultiSEL's Components

We present in Listing 5.4 how the MultiSEL's *property* element defines presentation characteristics of media items, rendering properties of sensory effects, and scene properties.

Table 5.5: *Presentation and Rendering Specification* Functional Area

Elements	Attributes	Content
property	<u>name</u> , <u>value</u>	empty

Table 5.6: *Interfaces* Functional Area

Elements	Attributes	Content
port	<u>id</u> , <u>component</u> , <u>interface</u>	empty

This code snippet uses elements and attributes of Table 5.5 and 5.4.

```

1 ...
2 <scene id="scene1" primaryComponent="introVideo">
3   <property name="var1" value="10"/>
4   <property name="var2" value="true"/>
5   <media id="introVideo" src="path_to_the_video_file" type="video">
6     <property name="background" value="true"/>
7     <property name="transparency" value="40%"/>
8     <property name="size" value="100px,200px"/>
9     <property name="pip" value="true"/>
10  </media>
11  <effect id="scent1" type="scentEffect">
12    <property name="intensityValue" value="50%"/>
13    <property name="position" value="0,6,12"/>
14    <property name="intensitySpatialAttenuation" tileSize="5,7" coefficient="2,4,3"
15    functionType="inverse-square-law"/>
16  </effect>
17  ...
18 </scene>
19 ...

```

Listing 5.4: MultiSEL's Presentation and Rendering Properties

Listing 5.5 shows how the MultiSEL's port element references scene's media and sensory effect items through the *id* and *component* attributes. We also demonstrate how the port's interface attribute gives the body's relations external access to the properties of the scene itself and to the properties of media and sensory effect items. This code snippet uses the elements of Tables 5.6 , 5.4 and 5.5.

```

1 ...
2 <scene id="scene1" primaryComponent="introVideo"/>
3   <!-- interface points to media and effect items of scene1 -->
4   <port id="portIntroVideo" component="introVideo"/>
5   <port id="portScentEffect" component="scent1"/>
6   <!-- interface point to the var1 scene property -->
7   <port id="portVar1" component="var1"/>
8   <!-- interface point to the intensityValue property of scent1 -->
9   <port id="portIntensityValue" component="scent1" interface="intensityValue"/>
10

```

Table 5.7: *Relation* Functional Area

Elements	Attributes	Content
relation	id, <u>type</u> , delay, keyCode	(primary, secondary+, set*, assessment*)
primary	<u>component</u> , interface	empty
secondary	<u>component</u> , interface, delay	empty
set	<u>component</u> , <u>interface</u> , <u>value</u>	empty
assessment	<u>expression</u>	empty

```

11 <property name="var1" value="10"/>
12 ...
13 <media id="introVideo" src="path_to_the_video_file" type="video">
14 ...
15 </media>
16 <effect id="scent1" type="scentEffect">
17   <property name="intensityValue" value="50%"/>
18   ...
19 </effect>
20 ...
21 </scene>
22 ...

```

Listing 5.5: MultiSEL's Interface

Listing 5.6 presents three relation specifications in MultiSEL in order to illustrate the use of the elements and attributes shown in Table 5.7. The *r1* relation represents a temporal relation that starts the secondary components when the primary one begins after a 2s delay. The *r2* relation shows how we can add a conditional assessment to the relation's condition, which is associated with its primary component. This relation compares a property of *scene1* with the 5 value. To do that, the relation accesses a *scene1*'s property by referencing its respective port (*portIntensityValue*). In addition, this relation contains the *set* element to set a *scene2*'s property with the 50%. The *r3* relation illustrates an interactivity relation. It defines that users can interact with the *scene1*'s component *media1* through the *ENTER* key to stop the *scene2*'s component *media3*, which is associated with the *secondary* element.

```

1 ...
2 <scene id="scene1">
3   ...
4   <relation id="r1" type="startsDelay" delay="2s">
5     <primary component="media1"/>
6     <secondary component="media2"/>
7     <secondary component="effect1"/>
8   </relation>
9   ...

```

```

10 </scene>
11 ...
12 <relation id="r2" type="conditionalFinishesSet">
13   <primary component="scene1"/>
14   <assessment expression="scene1.portIntensityValue > 5"/>
15   <secondary component="scene2" interface="portEffect2"/>
16   <set component="scene2" interface="portVideoTransparency" value="50%"/>
17 </relation>
18
19 <relation id="r3" type="onSelectionStop" keyCode="ENTER">
20   <primary component="scene1" interface="portMedia1"/>
21   <secondary component="scene2" interface="portMedia3"/>
22 </relation>
23 ...

```

Listing 5.6: MultiSEL's Relations

MultiSEL provides two profiles: *MultiSEProfile* and *MultiMediaProfile*. The first one is the complete profile. It includes all the MultiSEL's modules for creating interactive 360° mulsemmedia documents. *MultiMediaProfile* focuses on traditional mutimedia contents allowing authors to create documents without sensory effects. Therefore, this profile does not include the *SensoryEffect* module. Appendix C presents *MultiSEProfile* specified in XML Schema.

## 5.8 Case Study

This section describes an example of a MultiSEL document using *MultiSEProfile*. This document specified a 360° mulsemmedia application designed for VR environments using a HMD with 3DoF. We divide this document into three parts to make it easier to understand. The first part contains the Scene 1 specification, the second one has the relations defined in the body document, and the third part contains the Scene 2 specification.

This application starts with a 360 video (*360gardenVideo*) as the VR environment background representing a garden, as shown in Listing 5.7. In the first scene, there is an interactive image (*rosePortalImage*) to illustrate a portal that takes users to the second scene. To interact with the portal, users need to press the HMD controller's *X* key. This interactivity relation is defined by the *r5* relation in the body document, shown in Listing 5.8. Scene 1 also has a scent sensory effect (*roseScent*) to render the rose scent that comes from the portal. In addition, Scene 1 has another interactive image that works as a toggle button for users to allow or not increasing the scent sensory effect intensity when they go to the second scene. Users need to use the A/B buttons of the HMD motion



controllers [13] to interact with this button image to enable/disable the effect intensity increasing. Those interactions are defined by the scene1's  $r3$  and  $r4$  relations. Both images stop being displayed and the scent sensory effect stop being rendered when the *360gardenVideo* finishes. This temporal synchronization is defined by the scene1's  $r1$  and  $r2$  relations. Figure 5.2 shows the structure view of this first scene.

```

1 <multisel id="360MulsemmediaApp" title="MultiSEL Case Study" xmlns="MultiSEProfile">
2   <head>
3     <meta name="author" value="Douglas Mattos"/>
4     <meta name="year" value="2021"/>
5   </head>
6   <body>
7     <scene id="scene1" primaryComponent="360gardenVideo">
8       <!-- ports -->
9       <port id="portIncreaseIntensity" component="allowIncreaseIntensity"/>
10      <port id="portRosePortalImage" component="rosePortalImage"/>
11      <port id="portRoseScent" component="roseScent"/>
12      <port id="portRoseInt" component="roseScent" interface="intensityValue"/>
13      <port id="portGardenVideo" component="360gardenVideo"/>
14      <port id="portEnableIncreaseImage" component="enableIncreaseImage"/>
15      <!-- scene property -->
16      <property name="allowIncreaseIntensity" value="false"/>
17      <!-- media and sensory effect items -->
18      <media id="360gardenVideo" src="garden.mp4" type="360video">
19        <property name="background" value="true"/>
20      </media>
21      <media id="rosePortalImage" src="portal.jpeg" type="image">
22        <property name="position" value="0,2,6"/>
23      </media>
24      <media id="enableIncreaseImage" src="enableIncrease.png" type="image">
25        <property name="pip" value="true"/>
26      </media>
27      <effect id="roseScent" type="scentEffect">
28        <property name="scent" value="rose"/>
29        <property name="intensityValue" value="50%"/>
30        <property name="position" value="0,2,6">
31      </effect>
32      <!-- relations -->
33      <relation id="r1" type="starts">
34        <primary component="360gardenVideo"/>
35        <secondary component="rosePortalImage"/>
36        <secondary component="roseScent"/>
37        <secondary component="enableIncreaseImage"/>
38      </relation>
39      <relation id="r2" type="finishes">
40        <primary component="360gardenVideo"/>
41        <secondary component="rosePortalImage"/>
42        <secondary component="roseScent"/>
43        <secondary component="enableIncreaseImage"/>
44      </relation>
45      <relation id="r3" type="onSelectionSet" keyCode="A">
46        <primary component="enableIncreaseImage"/>

```

```

47     <set component="allowIncreaseIntensity" value="true"/>
48   </relation>
49   <relation id="r4" type="onSelectionSet" keyCode="B">
50     <primary component="enableIncreaseImage"/>
51     <set component="allowIncreaseIntensity" value="false"/>
52   </relation>
53 </scene>
54 ...
55 </body>
56 </multisel>

```

Listing 5.7: MultiSEL Document Part 1

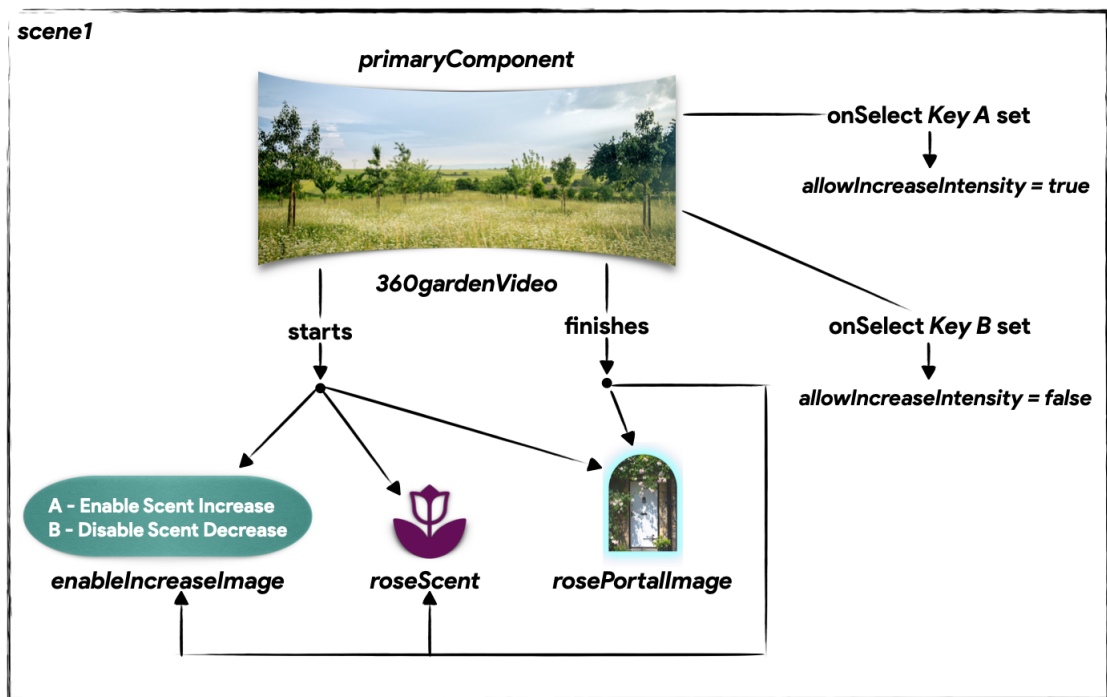


Figure 5.2: Scene 1 Structural View

The document body, shown in Listing 5.8, has relations between items from Scene 1 and 2. The *r5* relation defines an interactivity in order to start *scene2* when users interact with *rosePortallImage* by pressing the *X* key of the HMD controller. This relation uses a port element to access *rosePortallImage* of Scene 1 according to compositionality concept we discussed in Section 5.6. The *r6* relation defines that the garden video and the images of Scene 1 stop when Scene 2 begins. It represents the user entrance to the second garden (*roseGarden*). The *r7* relation increases the scent effect intensity and shows a warning image for that increase when Scene 2 begins and if the user enables the intensity increase in Scene 1. The last relation just finishes the scent effect when Scene 2 ends. Figure 5.3 shows the structure view of the document body's relations.

```

1 <multisel id="360MulsemmediaApp" title="MultiSEL Case Study" xmlns="MultiSEProfile">

```

```

2    ...
3    <body>
4        ...
5        <!-- relations between scenes -->
6        <relation id="r5" type="onSelectionStarts" keyCode="X">
7            <primary component="scene1" interface="portRosePortalImage"/>
8            <secondary component="scene2"/>
9        </relation>
10       <relation id="r6" type="metBy">
11           <primary component="scene2"/>
12           <secondary component="scene1" interface="portGardenVideo"/>
13           <secondary component="scene1" interface="portRosePortalImage"/>
14           <secondary component="scene1" interface="portEnableIncreaseImage"/>
15       </relation>
16       <relation id="r7" type="conditionalStartsSet">
17           <primary component="scene2"/>
18           <assessment expression="scene1.portIncreaseIntensity == true">
19               <secondary component="scene2" interface="portWarning"/>
20               <set component="scene1" interface="portRoseInt" value="100%"/>
21           </assessment>
22       </relation>
23       <relation id="r8" type="finishes">
24           <primary component="scene2"/>
25           <secondary component="scene1" interface="portRoseScent"/>
26       </relation>
27       ...
28   </body>
29 </multisel>

```

Listing 5.8: MultiSEL Document Part 2

Listing 5.9 shows the Scene 2 specification. It has a second video (*roseGarden*) to represent the other garden that users enter after interacting with the portal of Scene 1. In addition, Scene 2 has an image to warn about the scent effect intensity increase. This image is used by the *r7* relation defined in Listing 5.3. Moreover, Scene 2 starts a wind sensory effect (*r9* relation) and finishes the warning image and the wind effect when the *roseGarden* video ends. Figure 5.4 shows the Scene 2 structure view. Appendix D presents all the document parts together.

```

1 <multisel id="360MulsemmediaApp" title="MultiSEL Case Study" xmlns="MultiSEProfile">
2     ...
3     <body>
4         ...
5         <scene id="scene2" primaryComponent="roseGarden">
6             <!-- port -->
7             <port id="portWarning" component="strongScentWarningImage"/>
8             <!-- media and sensory effect items -->
9             <media id="roseGarden" src="roseGarden.png" type="360Image">
10                 <property name="background" value="true"/>
11                 <property name="explicitDur" value="30s"/>
12             </media>
13             <media id="strongScentWarningImage" src="warning.png" type="image"/>

```

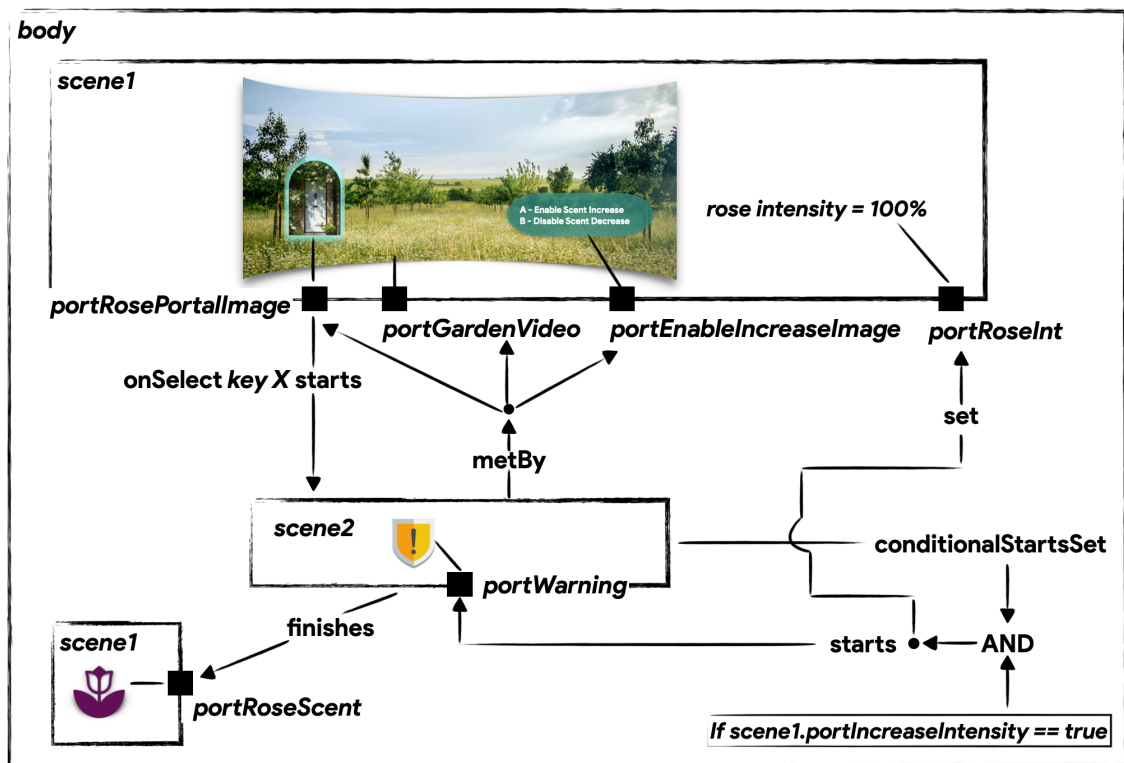


Figure 5.3: Body Structural View

```

14     <effect id="wind" type="windEffect">
15         <property name="intensityValue" value="50%"/>
16         <property name="position" value="0,6,12"/>
17     </effect>
18     <!-- relations -->
19     <relation id="r9" type="starts" delay="10s">
20         <primary component="roseGarden"/>
21         <secondary component="wind"/>
22     </relation>
23     <relation id="r10" type="finishes">
24         <primary component="roseGarden"/>
25         <secondary component="strongScentWarningImage"/>
26         <secondary component="windEffect"/>
27     </relation>
28 </scene>
29 </body>
30 </multisel>

```

Listing 5.9: MultiSEL Document Part 3

This chapter discussed our proposal for MultiSEL, an XML language based on MultiSEM to specify 360° mulsemmedia applications. The following chapter presents STEVE 2.0, a graphical authoring environment, also based on MultiSEM, to create interactive mulsemmedia applications.

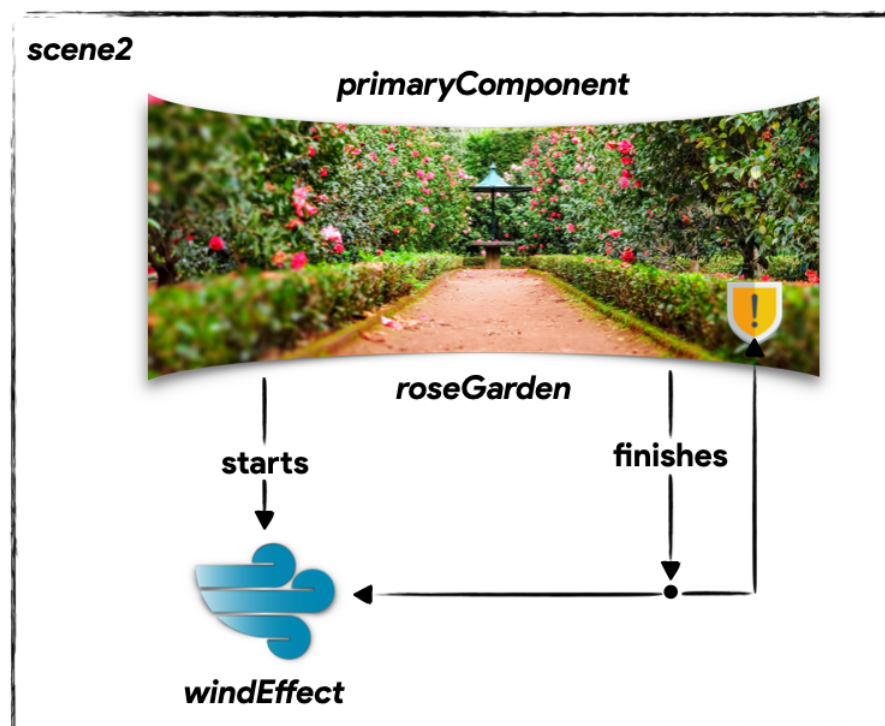


Figure 5.4: Scene 2 Structural View

## Chapter 6

# STEVE 2.0 - An Authoring Environment for Mulsemmedia Applications

In this chapter, we present a case study in order to demonstrate how MultiSEM represents the mulsemmedia application described in Section 6.1. In Section 6.2, we present our mulsemmedia authoring tool proposal, STEVE 2.0 [48], implemented using the MultiSEM model. In that section, we also present STEVE’s architecture, data flow, and graphical user interface. Section 6.3 discusses STEVE’s integrations with other mulsemmedia technologies providing an end-to-end tool chain.

### 6.1 Case Study

This section describes an example of mulsemmedia application<sup>1</sup>, which is based on the application presented in [74]. The application describes an immersive environment with multiple sensory effects for cognitive activities with children with autism. It aims to demonstrate how a graphical authoring tool based on MultiSEM can support the development of mulsemmedia applications and how MultiSEM represents the application. In this interactive application, users are sat on an armchair in the center of a room. There are actuators and sensors spread in the immersive environment. There is also a screen to display the audiovisual content that is synchronized with sensory effects. At the beginning of the application, the screen displays a living room represented by an image media node (*background\_image*) as illustrated in Figure 6.1.

After 5s of the application beginning, the video and audio that illustrate the burning fireplace starts. At the same time, a light effect also starts to represent the luminosity

---

<sup>1</sup><https://bit.ly/3DOu6hW> - Demo video for the mulsemmedia application

from the flames. After 5s of the light effect beginning, a temperature effect starts to heat the environment due to the flames. Figure 6.1 shows the structural view of the application's first scenario.

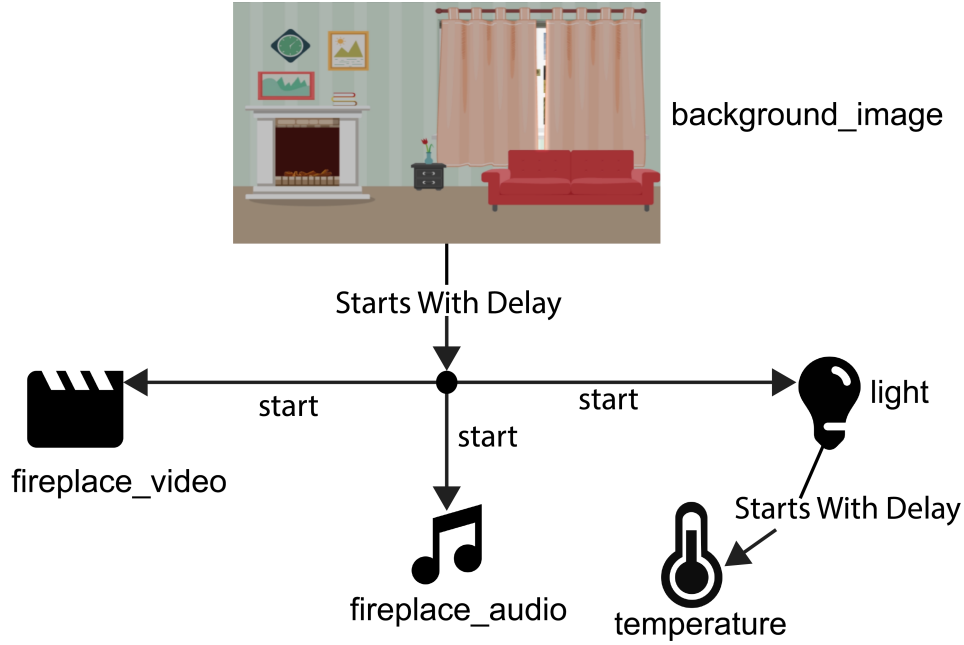


Figure 6.1: Structural View 1 [74]

To represent this scenario in MultiSEM, the background image is modeled as a media node using the *Media* class. The light and temperature effects are modeled as *Sensory-Effect* nodes. Also, the *effectPresentationProperty.intensity* attribute of the light effect is set to 50 lux. Additionally, the intensity attribute of the temperature effect is set to 32 Celsius degrees. To temporally synchronize these nodes according to the *Starts With Delay* relation presented in Figure 6.1, we first need to specify a *CausalConnector*. The causal connector has a simple condition role whose event type is *presentation* and the event state transition is *starts*. Its *ActionRole* has the *eventType* attribute defined as *presentation* and *actionType* as *start*. Additionally, a delay attribute is defined as a parameter to fire the action. The connector's glue has a simple trigger expression that is associated with the condition role *onBegin* and a simple action associated with the action role *startAction*. Listing 6.1 presents the specification of that connector (*starts\_delay*) using the XConnector language [91].

```

1 <!-- XConnector "Starts with Delay" -->
2 <xconnector id="starts_delay" xsi:type="CausalHypermediaConnector">
3   <param name="pDelay"/>
4   <conditionRole id="onBegin" eventType="presentation">
5     <condition xsi:type="EventTransitionCondition" transition="starts"/>
6   </conditionRole>

```

```

7   <actionRole id="startAction" eventType="presentation" actionType="start" delay="$
    pDelay"/>
8   <glue>
9       <triggerExpression xsi:type="SimpleTriggerExpression" conditionRole="onBegin" />
10      <actionExpression xsi:type="SimpleActionExpression" actionRole="startAction"/>
11  </glue>
12 </xconnector>
13
14 <!-- MultiSEL Specification for Relation "Starts with Delay" -->
15 <relation id="fireplace_relation" type="startsDelay" delay="5s">
16     <primary component="background_image" />
17     <secondary component="fireplace_video" />
18     <secondary component="fireplace_audio" />
19     <secondary component="light" />
20 </relation>

```

Listing 6.1: Connector and Causal Relation Example

That XML snippet also shows the MultiSEL specification of the relation *fireplace\_relation*, which uses the connector *starts\_delay* to give the relation causal temporal semantics. The relation associates the background image with the connector condition role using a *Bind* element, as shown in Listing 6.1. Also, the relation associates the *fireplace\_video*, *fireplace\_audio* and *light* nodes with the connector action role. For each node, the delay parameter is defined as 5s. Another relation is created using the same connector to associate the light effect node with the condition role and the temperature effect with the action role.

Figure 6.3 illustrates the second scenario of our mulsemmedia application example to show how MultiSEM represents interactivity events and statement assessments. In that scenario, the interactivity relation defines that when the user interacts with the application, pressing the green key of the remote control, two media items and three sensory effects start only if a statement assessment is also satisfied. The *window\_video* and *window\_audio* media items represent the window opening and the outside view, resulting in the scene illustrated in Figure 6.2. The triggered sensory effects are *light*, *wind* and *temperature* to represent the luminosity from the outside and the cold wind from the window respectively. The statement assessment is defined to evaluate if the window is open (*windowsOpen=1*) or close (*windowsOpen=0*). If it is close, then the interactivity relation sets the *windowsOpen* value to 1 and starts the media items and sensory effects. The *windowsOpen* variable is modeled as a *SettingsNode*, as discussed in Section 4.1.

Listing 6.2 shows the XConnector specification for that interactivity relation and its MultiSEL specification. The connector's glue has a compound trigger expression with an *and* operator to define the user interaction and the statement assessment as condition





Figure 6.2: After User Interaction [74]

to trigger the relation actions (*start* and *set*). Therefore, the connector has a condition role whose event type is the selection event and its transition is set to *ends* (when the user click finishes). The causal relation links that condition role to the *background\_image* item and defines the interactivity key (*green\_key*). The other condition has the event type set to *attribution* to represent the statement assessment. The compound trigger expression in the *glue* element has the parameters *assessmentValue* and *pComparator* to define the boolean expression to be assessed. The causal relation binds the *windowsOpen* node with the connector property role, *statementAssessment*, and defines the values for the *assessmentValue* and *pComparator* parameters as *0* and *equal* respectively.

Regarding the action, the relation defines two types of action, *start* and *set*, represented by the *startAction* and *setAction* action roles respectively. The first one is associated to the presentation event and the second one with the attribution event. The causal relation links the start action with *window\_video* and *window\_audio* items and the sensory effects. Additionally, the set action role has the *pValue* parameter. It represents the value to be set to the *windowsOpen* node and is defined in the relation *bindParam* element. Furthermore, the connector triggers both the actions simultaneously, defined by the operator *par* in the glue action expression.

In addition to the structural view of our mulsemmedia application, we provide its temporal view using STEVE 2.0 in Section 6.2.4.

```

1
2 <!-- XConnector "OnSelectionStartSetAssessment" -->
3 <xconnector id="onSelectionStartSetAssessment" xsi:type="CausalHypermediaConnector">
4   <param name="pValue">
5   <param name="assessmentValue">
6   <param name="pComparator">
7   <param name="pKeyCode">
8   <conditionRole id="onSelection" key="$pKeyCode" eventType="selection">

```

```

9      <condition xsi:type="EventTransitionCondition" transition="ends"/>
10    </conditionRole>
11    <propertyRole id="statementAssessment" eventType="attribution">
12      <property xsi:type="NodeAttributionProperty"/>
13    </propertyRole>
14    <actionRole id="startAction" eventType="presentation" actionType="start"/>
15    <actionRole id="setAction" eventType="attribution" actionType="set"
16      finalValue="$pValue"/>
17    <glue>
18      <triggerExpression xsi:type="CompoundTriggerExpression" operator="AND">
19        <trigger xsi:type="SimpleTriggerExpression" conditionRole="onSelection"/>
20        <property xsi:type="AttributeToValueExpression" propertyRole="
21          statementAssessment" value="assessmentValue" comparator="pComparator"/>
22      </triggerExpression>
23      <actionExpression xsi:type="CompoundActionExpression" operator="par">
24        <firstAction actionRole="startAction" qualifier="par"/>
25        <SecondAction actionRole="setAction" qualifier="par"/>
26      </actionExpression>
27    </glue>
28  </xconnector>
29  <!-- MultiSEL Specification for Relation "conditionalOnSelectionStartSet" -->
30  <relation id="user_interaction_relation" type="conditionalOnSelectionStartSet" keyCode="
31    green_key">
32    <primary component="background_image" />
33    <assessment expression="windowsOpen == 0" />
34    <secondary component="window_video" />
35    <secondary component="window_audio" />
36    <secondary component="light" />
37    <secondary component="wind" />
38    <secondary component="temperature" />
39    <set component="windowsOpen" value="1" />
40  </relation>

```

Listing 6.2: Connector and Causal Relation Example

## 6.2 STEVE 2.0

In order to demonstrate how MultiSEM supports the development of mulsemmedia authoring environments based on temporal view, we implemented the model in STEVE [47, 84, 46]. This implementation extends STEVE to support the integration and synchronization of sensory effects with traditional multimedia content. We call that new version STEVE 2.0 [48, 83].

The first version of STEVE was based on SIMM (Simple Multimedia Model) [47, 46]. That version only supports the authoring of traditional multimedia documents with no sensory effects. Moreover, its underlying model is less expressive than MultiSEM since

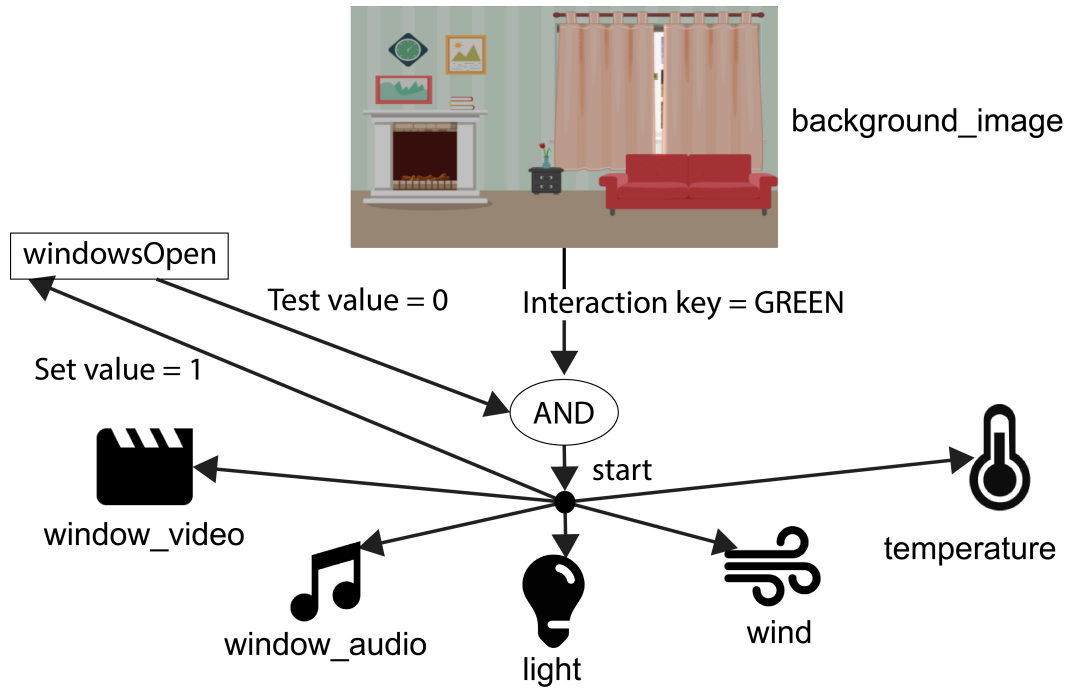


Figure 6.3: Structural View 2 [74]

SIMM represents temporal relations in a higher-level abstraction and it does not support variable tests [46].

Since STEVE 2.0 is based on MultiSEM, it uses an event-based temporal synchronization paradigm to define the temporal behavior of mulsemmedia applications. Its authoring GUI approach is based on temporal view, as discussed in 2.2, to allow users with no programming skills to define mulsemmedia applications. The tool graphically provides causal temporal relations based on MultiSEM's relations and gives authors feedback about temporal synchronization inconsistencies. Additionally, users can create interactivity relations to activate, for example, sensory effects due to user interactions with the mulsemmedia application. The editor also provides spatial view editing to allow users to edit rendering characteristics (e.g., intensity, scent type, light frequency) and physical positions of sensory effects. Moreover, we can edit the presentation properties of traditional multimedia content. All presentation and rendering properties STEVE 2.0 provides are based on the MultiSEM's specification proposed in Section 4.2.

Moreover, STEVE 2.0 allows authors to check the temporal and spatial behavior of mulsemmedia applications by providing a graphical temporal and spatial view in a synchronized way. After the production phase, STEVE mulsemmedia applications can be distributed by exporting them to documents written in NCL 4.0 [73], which can run in the multisensory-extended Ginga-NCL middleware presented in [73]. That integration

will be discussed in Section 6.3.

### 6.2.1 STEVE 2.0 Architecture

Figure 6.4 shows STEVE 2.0 modular architecture (source code<sup>2</sup>) highlighting the new components added (green boxes) to STEVE 1.0 [47, 84, 46] and which components were updated (yellow boxes) to support mulsemmedia applications. The architecture uses the Model-View-Controller (MVC) design pattern, which separates the system into three modules that communicates with each other: *View*, *Model* and *Controller*. The first one is responsible for implementing the graphical user interface of STEVE, getting model data from *Model*, and notifying *Model* about user interactions with *View* using the *Controller* model. *Model* describes the STEVE business logic part, which implements data retrieval, and MultiSEM’s event-based synchronization paradigm and its entities. Also, it notifies *View* about model changes in order to update STEVE’s GUI.

In *Model*, we have the STEVE extension core to support mulsemmedia applications, the class *SensoryEffectNode*, which implements sensory effects as first-class entity following MultiSEM representation. In addition, we have added the new package *Sensory Effect Properties* into *Spatial View* to represent the rendering characteristics of sensory effects according to MPEG-V’s specification, on which MultiSEM is based. Moreover, the *Temporal View* package was modified to use MultiSEM’s temporal synchronization entities, as discussed in Section 4.4. Additionally, STEVE 2.0 uses MultiSEM’s predefined connector base (*multisemConnectorBase.xml* artifact) to represent the temporal causal relations it provides. We have also extended ANA API [107] in order to export STEVE 2.0 projects into NCL 4.0 documents. ANA API is a metamodel specifically created to represent NCL documents in model-oriented environments enhancing the NCL code manipulation [107].

The *Repository* and *HTML Support* packages, also presented in *Model*, were not modified. The first one implements the data and business logic for STEVE’s repository, in which users manipulate multimedia files. *HTML Support* is responsible for exporting STEVE projects with no sensory effect content, i.e. only interactive multimedia applications, into HTML5 documents. The package uses the NCL4Web tool (*ncl4web.xsl* artifact), which uses XSLT and JavaScript libraries, to transform NCL documents into HTML5 applications.

Regarding the *View* module, STEVE 2.0 has added the *Sensory Effect Pane* and

---

<sup>2</sup><https://github.com/dougpmattos/steve>

*SE Properties Pane* modules. The first one implements a graphical control element that provides the list of sensory effects defined in MultiSEM. The *SE Properties Pane* implements a pane to allow users to edit sensory effect rendering properties according to the effect type. In addition, we have modified the *Preview Pane* package to help authors to visualize the temporal synchronization between sensory effects and audiovisual content. Furthermore, *Temporal View Pane* was updated to create the graphical representation of sensory effects in STEVE's temporal view. We will discuss those graphical elements for sensory effects in Section 6.2.3.

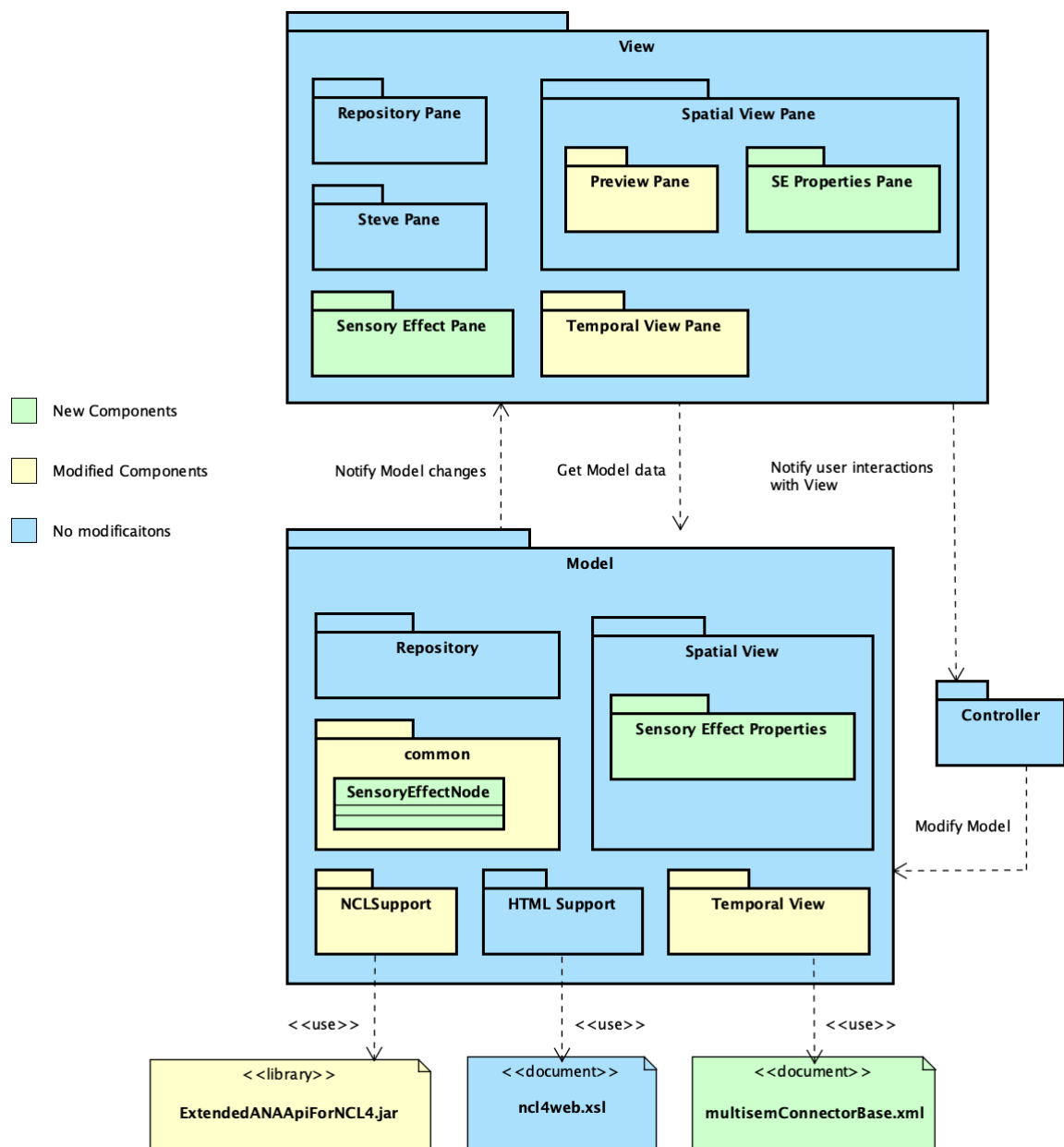


Figure 6.4: STEVE 2.0 architecture highlighting the modifications against STEVE 1.0

### 6.2.2 STEVE 2.0 Data Flow

STEVE's data flow diagram, presented in Figure 6.5, shows three input types the editor can receive: new projects, pre-existing STEVE 2.0 projects and NCL 3.0 [103] documents. The STEVE project serialized file is interpreted in order to recover MultiSEM entities. When NCL 3.0 documents are imported into STEVE, they are mapped into MultiSEM entities through processes 1, 2, and 3 indicated in Figure 6.5.

*Process 1* transforms the NCL document into a graph model called Hypermedia Temporal Graph (HTG) [39]. HTG represents the multimedia application temporal behavior in a digraph structure. It contains all predictable and unpredictable events, which can modify NCL media item presentation event states (*sleeping*, *paused* and *occurring*). In *Process 2*, STEVE generates another data structure called *Presentation Plan* [39] to indicate event execution times. Finally, in *Process 3*, the NCL document is represented using MultiSEM entities. Notice that the original NCL links from the imported document are not directly mapped into MultiSEM relations, except the user interaction relations. Instead, STEVE creates *Starts with Delay* relations to temporally synchronize the document nodes according to their execution times defined in the presentation plan.

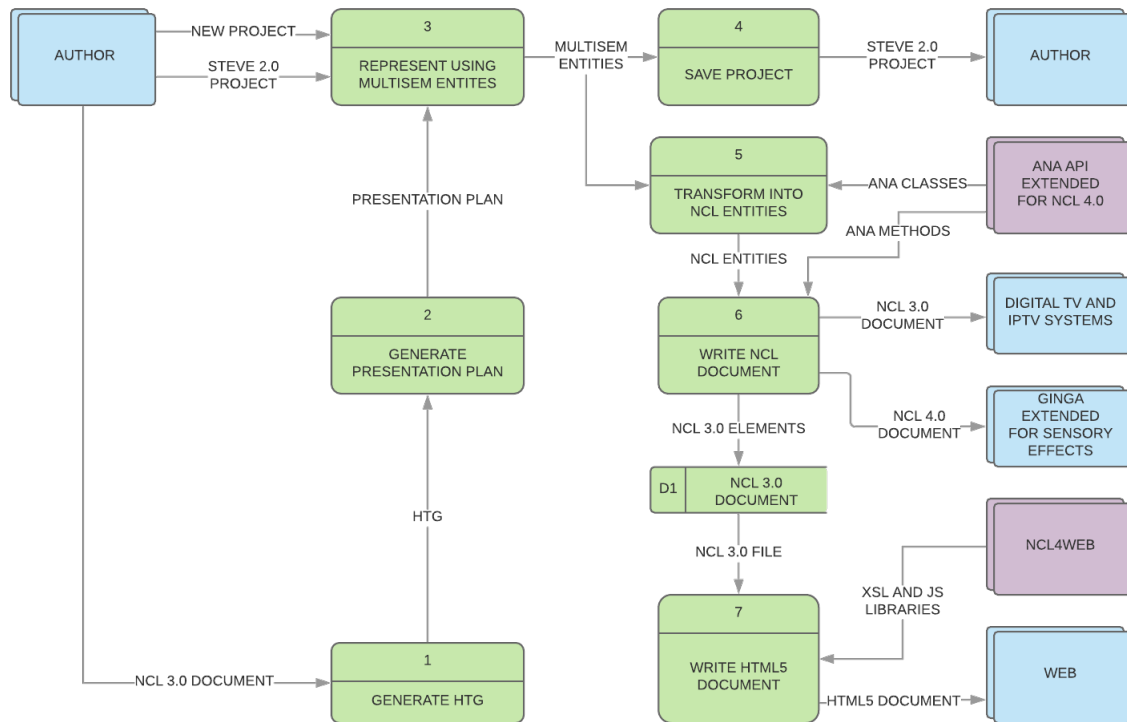


Figure 6.5: STEVE 2.0 Data Flow

In order to export STEVE applications to NCL documents, the tool transforms Mul-

tiSEM entities into NCL entities, shown in *Process 5* in Figure 6.5, using the ANA API [107], which represents NCL elements as Java classes. The editor also uses ANA methods to write NCL documents in *Process 6*. We have extended ANA API to support the sensory effect node present in NCL 4.0. Therefore, STEVE 2.0 exports its applications to NCL 3.0, ready to run in Digital TV and IPTV systems [103], and NCL 4.0 [73], being able to run in an extended Ginga-NCL for sensory effects [73].

STEVE also allows exporting applications to HTML5 documents. In this case, first, the editor generates the NCL 3.0 document and stores it (*Data Store D1*). Then STEVE uses NCL4Web [113] to write the HTML5 document (*Process 7*) representing the stored NCL document. Moreover, STEVE applications that are opened in the editor can be saved (*Process 4*) in STEVE file format to be available for future editing.

### 6.2.3 Graphical User Interface

Figure 6.6 shows the interface of STEVE 2.0 to support multiple sensory effects. The interface of STEVE consists of a multimedia content repository in the upper left corner, a panel in the middle for users to edit media presentation properties and sensory effect rendering characteristics, a preview pane to play audiovisual content in the upper right corner, and a temporal view in the bottom region. The pane for editing the rendering of sensory effects is based on the MultiSEM effect properties specification discussed in Section 4.2.1. In addition, STEVE 2.0 provides a graphical control element with a list of sensory effects above the temporal view, as shown in Figure 6.6. That sensory effect list is based on the types of effect defined in MultiSEM in Section 4.1.

Users can select one of those sensory effects and drag them to the timeline to temporally synchronize them with other nodes. To do that temporal synchronization, users can use the causal temporal relations STEVE graphically provides below the timeline, as shown in Figure 6.6. Notice that some sensory effects available in the STEVE interface are not defined in MultiSEM since they are implemented by grouping different effects defined in MultiSEM. For example, the *Rainstorm* effect is mapped into the *Flashlight*, *Fog*, *Wind* and *Rain* effects.

Moreover, the preview pane also displays icons that represent sensory effects temporally synchronized with audiovisual content. For instance, in Figure 6.6, a hot effect is shown in preview synchronized with the first video of the application.

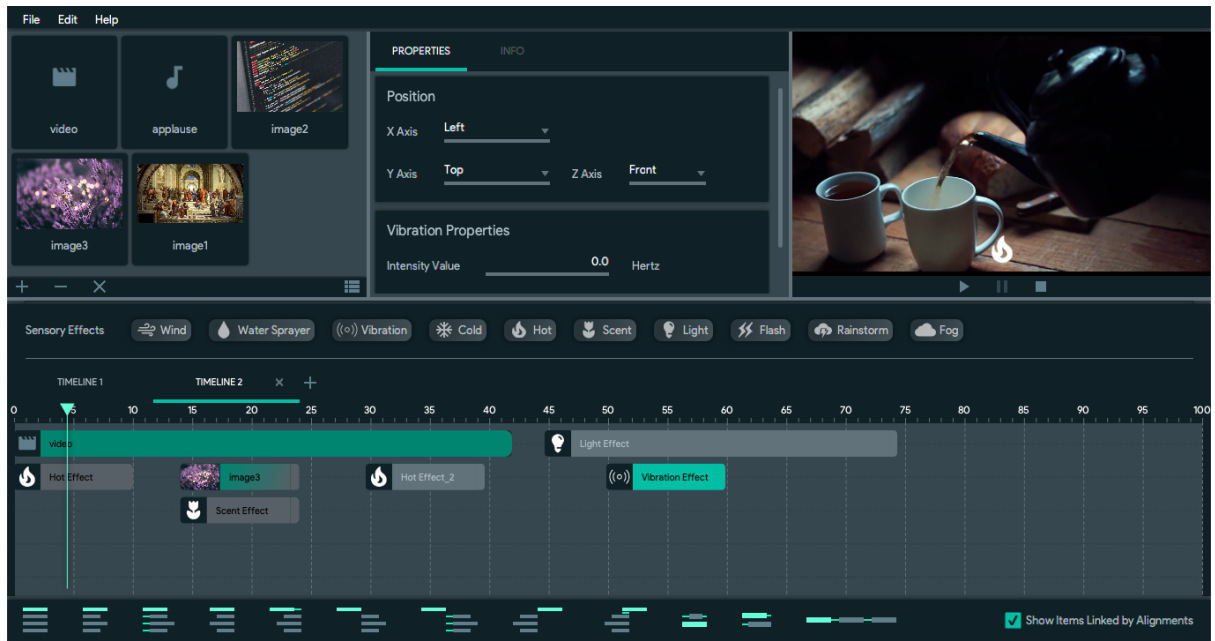


Figure 6.6: Graphical Interface of STEVE 2.0

### 6.2.4 Temporal View

To support authors to synchronize their mulsemmedia applications, STEVE provides causal temporal relations graphically in the button bar below the timeline, as shown in Figure 6.6. STEVE 2.0 uses the part 1 of MultiSEM’s predefined connector base (*Allen’s Temporal Relations*), as proposed in Section 4.4.3.1, to define those relations.

Figure 6.7a shows the temporal view in STEVE 2.0 that corresponds to the structural view of Figure 6.1. To create this temporal view in STEVE 2.0, authors should first drag the media and sensory effect nodes from the media repository and the sensory effect bar, respectively, to the timeline. After that, they should synchronize them using the temporal relation buttons.

In addition to synchronous temporal relations, STEVE 2.0 provides the definition of asynchronous relations to define user interactions. To create the user interaction illustrated in Figure 6.3, authors need to specify the interactivity key and which application’s media items or sensory effects stop and start after user interaction. In our example, authors need to define the *Green* key as the interactivity key and select the *window\_video*, *window\_audio*, *light*, *wind* and *hot* (temperature) nodes to start after users interact pressing the *Green* key, as demonstrated in Figure 6.8. After defining that interactivity relation, STEVE automatically creates a new timeline, *Timeline 2* as shown in Figure 6.7b, to contain the new nodes that start. Therefore, for each interactivity relation defined, STEVE





Figure 6.7: STEVE Event-based Timelines for Our Mulsemedia Application

2.0<sup>3</sup> creates a new event-based timeline.

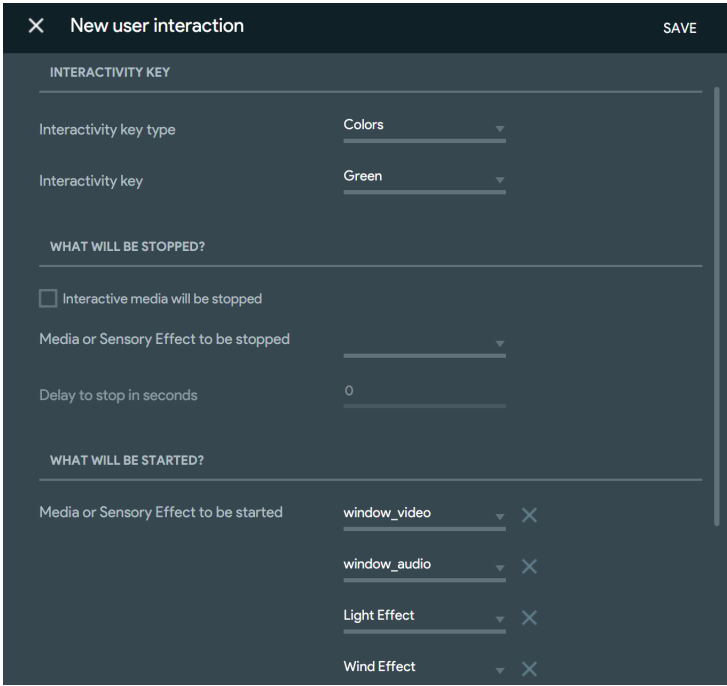


Figure 6.8: STEVE's *New User Interaction* Window

<sup>3</sup><https://dougpmattos.github.io/#13> - Demo Videos for the Main Features of STEVE 2.0

## 6.3 STEVE's Integrations: on the Need for Mulsemmedia Platforms

This section discusses how STEVE 2.0 can be integrated with other mulsemmedia technologies to provide an end-to-end tool chain as a complete mulsemmedia platform accomplishing the three phases [41] of the development of mulsemmedia applications: authoring, distribution, and rendering in the physical environment.

Regarding the authoring phase, STEVE 2.0 can be integrated with other authoring environments using our proposal MultiSEL, which can provide interoperability between different 2D/3D mulsemmedia authoring tools. Indeed, a virtual reality mulsemmedia authoring tool, called AMUSEVR (*Authoring 360° Multimedia and Sensory Effects in VR*) [45], applies MultiSEL into its modeling of 360° mulsemmedia applications. The VR tool also imports MultiSEL documents and exporting AMUSEVR projects to MultiSEL documents.

STEVE 2.0 was integrated with a machine learning solution to enhance visual content's sensory effect annotation. That content-driven component, named STEVEML<sup>4</sup> (STEVE Machine Learning) [3], gives STEVE 2.0 the ability to automatically extract sensory effects from video content. With that integration, authors can select video or image items in STEVE's temporal view and request the sensory effect annotation for the selected media according to the effect types chosen. After the extraction, authors can also make manual adjustments in the temporal synchronization of the extracted effects.

Concerning the distribution and rendering phases, STEVE 2.0 was integrated<sup>5</sup> with the multisensory Ginga-NCL proposed in [73] by exporting STEVE projects to documents written in NCL 4.0 [73]. That mulsemmedia formatter proposes an effect player for each effect type and implements three APIs *DeviceScent*, *DeviceWind* e *DeviceLight* to render scent, wind, and light effects respectively.

The component diagram of Figure 6.9 illustrates how STEVE 2.0 is integrated with each mulsemmedia technology discussed in this section. MultiSEL supports the integration of STEVE with AMUSEVR [45], STEVE's modular architecture aids the integration with the machine learning component STEVEML [3], and STEVE 2.0 integrates with the multisensory Ginga-NCL [73] by exporting mulsemmedia applications to NCL 4.0 documents.

<sup>4</sup><https://dougpmattos.github.io/#28> - Demo Video for STEVEML

<sup>5</sup><https://bit.ly/3DOu6hW> - STEVE application running on the multisensory-extended Ginga-NCL

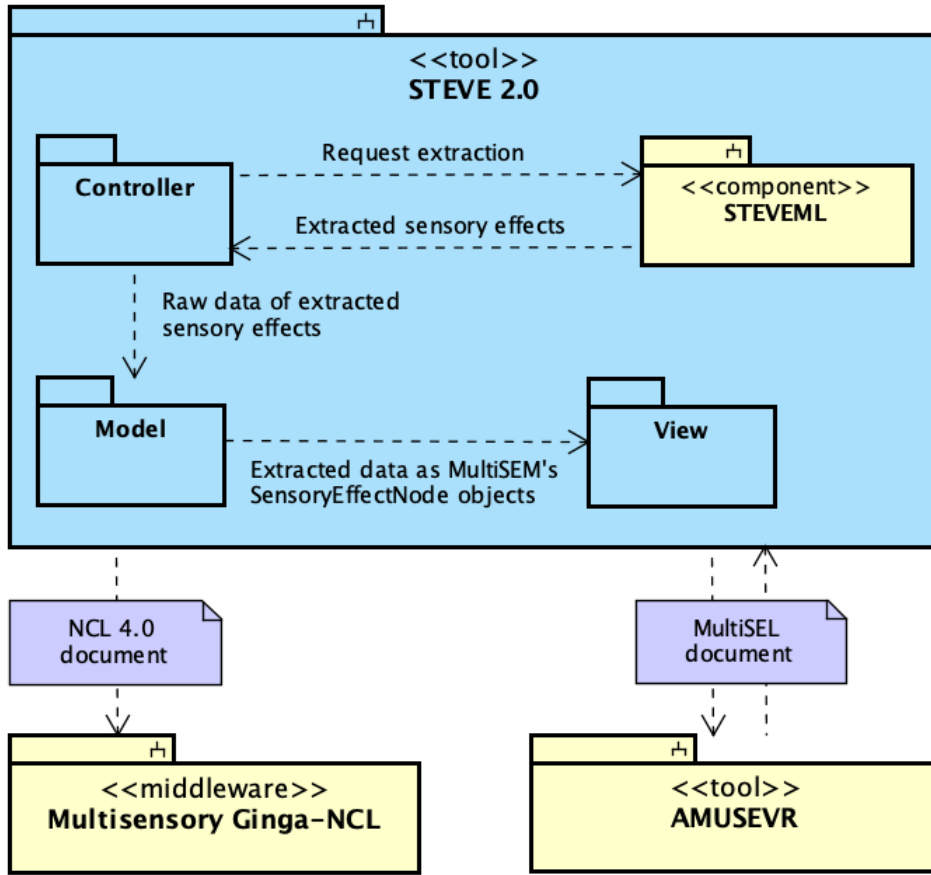


Figure 6.9: STEVE's Integrations

## 6.4 Final Remarks

Concerning the gaps of authoring tools we have identified in Section 3.6, in this section, we pointed out which of them STEVE 2.0 satisfies and which we leave as future improvements in STEVE 2.0.

- *Temporal Synchronization Paradigm*: STEVE 2.0 is the only tool in our literature review that uses the event-based paradigm to synchronize document nodes temporally, avoiding the timeline paradigm limitations, as discussed in Section 2.1.2.
- *Synchronization of Multiple Media Items*: STEVE provides the temporal synchronization of multiple media items, including traditional media and sensory effects, using MultiSEM's causal temporal relations.
- *Sensory Effect Group and Reuse*: We leave as future improvements in STEVE 2.0.
- *Sensory Effect Spatial View*: We leave as future improvements in STEVE 2.0.

- *Asynchronous Events*: STEVE allows users to create interactivity relations but does not allow the definition of relations with assessment statements.
- *Template Support*: We leave as future improvements in STEVE 2.0.
- *Sensory Effect Prefetching*: We leave as future improvements in STEVE 2.0.

The next chapter presents the STEVE 2.0 evaluation applying different methodologies for assessing its usability, specific features, and user experience.

# Chapter 7

## Evaluation

This chapter presents the analysis of STEVE's usability with authoring experiments. Section 7.1 describes the methodology used for carrying out the STEVE's usability test. In Section 7.2, we present the questionnaires used for collecting the users' feedback. In Section 7.3, we analyze that feedback to measure STEVE's usability, user experience and to evaluate STEVE's specific features. Section 7.4 highlights challenges we have found for the mulsemmedia authoring phase.

In the STEVE's authoring experiments, 43 users participated, of which 35 are computer science students and nine are from other areas, such as cinema, medicine, mathematics, physics, history, and law. We have applied three different methodologies for evaluating STEVE: the Goal Question Metric (GQM) [22], the System Usability Scale (SUS) [28], and the User Experience Questionnaire (UEQ) [81, 110]. In the following sections, we present how they were applied to STEVE's experiments.

### 7.1 Methodology

We applied the Goal Question Metric (GQM) [22] approach to structure our evaluation. The GQM mechanism can be viewed as a directed graph in which the flow is from the goal nodes to the questions nodes to the metric nodes. Also, we based on a GQM application for usability test presented in [70]. Additionally, the GQM guidelines propose to define the purpose and the perspective of the goals. The purpose defines the object of study and why we are analyzing it. The perspective defines a particular angle or aspect for evaluation and from whom that evaluation is given.

Therefore, Table 7.1 presents the goals used for evaluating STEVE. All those goals

Table 7.1: STEVE’s Experiment Goals

Goals	Description
<b>G1</b>	Analyze STEVE’s manual synchronization between sensory effect and traditional media for the purpose of evaluation with respect to perspicuity and effectiveness from the point of view of the users.
<b>G2</b>	Analyze STEVE’s temporal relations for the purpose of evaluation with respect to perspicuity and effectiveness from the point of view of the users.
<b>G3</b>	Analyze STEVE’s interactivity relation feature for the purpose of evaluation with respect to perspicuity and effectiveness from the point of view of the users.
<b>G4</b>	Analyze STEVE for the purpose of evaluation with respect to its overall usability from the point of view of the users.
<b>G5</b>	Analyze STEVE for the purpose of evaluation with respect to users’ experience.

focus on the STEVE’s temporal synchronization feature (its temporal view). For each goal, the description column gives the object of study (e.g., G2 - STEVE’s temporal relations), the purpose of the analysis (e.g., G2 - evaluation), the aspect for evaluation (e.g., G2 - perspicuity and effectiveness), and the point of view (e.g., G2 - users). Then in Section 7.2, we present the set of questions for each goal. For G4, we used the SUS questionnaire [28] and, for G5, we used the User Experience Questionnaire (UEQ) [81, 110]. For the other goals, we defined a specific questionnaire.

In our evaluation, we use the term *effectiveness* in G1 and G2 according to the definition in [28]. It means the ability of users to complete tasks using the system and the quality of the output. The term *perspicuity* that we also use in those goals follows the definition presented in [110]. It is related to the ease of understanding of the evaluated product.

To make the STEVE’s usability test available for users, we have created a web presentation <sup>1</sup> that guides users through the test step-by-step. Thus, the users have performed the experiments remotely with no extra help. First, we introduce the multisensory application concept and motivate users to participate in the experiments. Then, we present the STEVE’s main features demonstrating them using short videos for users to get familiar with it. Those features are as follows:

- Import media files to STEVE.

---

<sup>1</sup><https://dougpmattos.github.io/>

- Drag and drop media and sensory effects into the temporal view.
- Define the duration of sensory effects and media items.
- Preview STEVE's mulsemmedia applications.
- Create temporal alignments to synchronize items.
- Create interactivity events.

Afterward, we give the users the directions to download and install STEVE, available for macOS<sup>2</sup> and Windows<sup>3</sup>. Before starting the tasks, we also provide the assets necessary to complete the tasks, such as image and video files. Furthermore, after completing each task, we ask the users to upload the STEVE project created and give their feedback by filling out the questionnaires for the respective task. We will present those questionnaires in Section 7.2. Indeed, we emphasize that their feedback is essential regardless if they have completed the task or not. After performing the last task, we ask the users to fill out the SUS and UEQ questionnaires, also presented in Section 7.2. All questionnaires were made available to users through Google Form.

In the next section, we present the tasks users should perform. With our goals in mind, we have defined three tasks focused on the temporal synchronization of a mulsemmedia application.

### 7.1.1 Tasks

**Task 1** proposes a simple mulsemmedia application with two sensory effects, hot and cold, and a video media object. Then, the task goal is to synchronize both sensory effects with the video scenes. Also, users were not allowed to use the automatic extraction of sensory effects [43] and the temporal alignments STEVE provides graphically. That is, users must synchronize the items manually by dragging the effect and media items into the STEVE's temporal view, as presented in Section 6.2.4.

**Task 2** adds a new image media into the mulsemmedia application introduced in Task 1. In this task, users must show that image, which represents fire, together with the hot sensory effects created in Task 1 using the graphical temporal alignments STEVE provides. Figure 7.1 shows the temporal view of a mulsemmedia application created to perform Task 1 and the result after completing Task 2.

---

<sup>2</sup><https://bit.ly/3BwjvGO>

<sup>3</sup><https://bit.ly/3gRICM7>

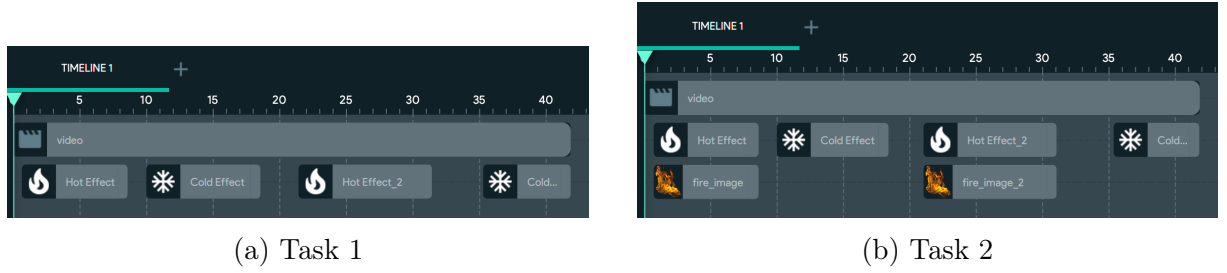


Figure 7.1: STEVE's Temporal View for Task 1 and 2

**Task 3** proposes another application in which users must create an interactivity relation using three items: an image representing an interactive button and two video files that represent a movie trailer and the movie itself. First, to perform the task, users must show the button image together with the trailer video. In other words, the image needs to be presented when the video starts playing and stops its presentation when the video ends. Then, users must make the button image interactive by defining that when the ENTER key is pressed, the application presents the movie video and stops the trailer. Figure 7.2 shows the temporal view of an application built to perform Task 3. That temporal view contains two timelines, and the second one is derived from the interactivity defined in Timeline 1.

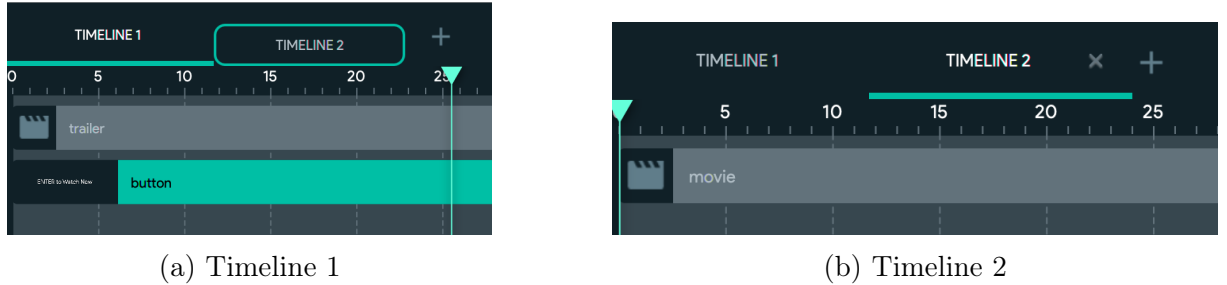


Figure 7.2: STEVE's Temporal View for Task 3

## 7.2 Questionnaires

In this section, we present the questions for each goal defined in Table 7.1. For goals  $G1$ ,  $G2$ , and  $G3$ , we have defined a set of questions for each one. And, for  $G4$  and  $G5$ , we have applied the questionnaires SUS [28] and UEQ respectively [81, 110].

### 7.2.1 G1 Questions

To achieve our first goal **G1**, *Analyze STEVE's manual synchronization between sensory effect and traditional media for the purpose of evaluation with respect to perspicuity and*



Table 7.2: Questions for G1 Goal

Question	Description
<b>Q1</b>	I found STEVE’s sensory effect icons easy to understand.
<b>Q2</b>	I found changing sensory effect duration hard in STEVE.
<b>Q3</b>	I found synchronizing sensory effects with video scenes easy in STEVE.
<b>Q4</b>	Did you successfully complete this task?

Table 7.3: Metrics for G1 Questions

Metric	Description	Question
<b>PP</b>	<i>Perspicuity</i> [110], the ease of understanding the evaluated product, measured using a five-point Likert scale.	Q1, Q2 and Q3
<b>EF</b>	<i>Effectiveness</i> [28], the ability of users to complete tasks using the system, and the quality of the output. Respectively, measured in the number of positive answers to Q4 and the quality of the users’ STEVE project for Task 1.	Q4

*effectiveness from the point of view of the users*, we have defined the following questions presented in Table 7.2. Questions *Q1*, *Q2*, and *Q3* uses a Likert scale [28] that divides the responses into a five-point scale indicating the degree of disagreement and agreement (from 1 - Strongly disagree to 5 - Strongly agree).

Table 7.3 presents the metrics used to answer *G1* questions. We use the same metric *PP* (Perspicuity) for answering the questions *Q1*, *Q2*, and *Q3*. For *Q4*, we use the *EF* (Effectiveness) metric, which is based on the number of positive answers (the user completed the task successfully) and the quality of the users’ STEVE project for Task 1. We classify the quality of STEVE’s projects in *successful* when the user’s project matches the expected output for the task and *unsuccessful* otherwise.

## 7.2.2 G2 Questions

To achieve Goal **G2**, *Analyze the STEVE’s temporal relations for the purpose of evaluation with respect to perspicuity and effectiveness from the point of view of the users*, we have defined the following questions presented in Table 7.4. We use a five-point Likert scale to indicate the degree of disagreement and agreement for *Q1’*, *Q2’*, and *Q3’*. For *Q4’*, we provide the STEVE’s temporal alignments in a list for users to select one or more of them they have used while performing Task 2. Users also can select an option to indicate that they could not use any temporal alignments. This question supports us in identifying if users managed to understand those alignments and use them properly.

Table 7.5 presents the metrics used to answer *G2* questions. We use the same metric

Table 7.4: Questions for G2 Goal

Question	Description
<b>Q1'</b>	I found the temporal synchronization between media and sensory effects easy to use.
<b>Q2'</b>	I found the temporal alignments hard to use.
<b>Q3'</b>	I found the temporal alignment icons easy to understand.
<b>Q4'</b>	Which temporal alignments did you use?
<b>Q4</b>	Did you successfully complete this task?

Table 7.5: Metrics for G2 Questions

Metric	Description	Question
<b>PP</b>	<i>Perspicuity</i> [110], the ease of understand of the evaluated product, measured using a five-point Likert scale.	Q1', Q2' and Q3'
<b>EF</b>	<i>Effectiveness</i> [28], the ability of users to complete tasks using the system, and the quality of the output. Respectively, measured in the number of positive answers to Q4 and the quality of the users' STEVE project for Task 2 supported by the Q4' answers.	Q4' and Q4

*PP*, introduced in Table 7.3, for answering questions *Q1'*, *Q2'*, and *Q3'*. For *Q4'* and *Q4*, we use the *EF* metric, also presented in Table 7.3, which is based on the number of users that completed the task successfully and the quality of the users' STEVE project for Task 2. In addition, the quality of the STEVE projects is also supported by the temporal alignments user have used to synchronize sensory effects and media items.

### 7.2.3 G3 Questions

To achieve Goal **G3**, *Analyze STEVE's interactivity relation feature for the purpose of evaluation with respect to perspicuity and effectiveness from the point of view of the users*, we have defined the following questions presented in Table 7.6. *Q1''* uses a five-point Likert scale to indicate the degree of disagreement and agreement as we discussed for the previous goals.

Table 7.7 presents the metrics used to answer the *G3* questions. We use the metric *PP* from Table 7.3 to answer question *Q1''*. For *Q4* question, we use the *EF* metric based

Table 7.6: Questions for G3 Goal

Question	Description
<b>Q1''</b>	I found the interactivity functionality easy to use.
<b>Q4</b>	Did you successfully complete this task?

Table 7.7: Metrics for G3 Questions

Metric	Description	Question
<b>PP</b>	<i>Perspicuity</i> [110], the ease of understand of the evaluated product, measured using a five-point Likert scale.	Q1”
<b>EF</b>	<i>Effectiveness</i> [28], the ability of users to complete tasks using the system, and the quality of the output. Respectively, measured in the number of positive answers to Q4 and the quality of the users’ STEVE project for Task 3.	Q4

Table 7.8: SUS Questionnaire for STEVE

Question	Description
<b>1</b>	I think that I would like to use STEVE frequently.
<b>2</b>	I found STEVE unnecessarily complex.
<b>3</b>	I thought STEVE was easy to use.
<b>4</b>	I think that I would need the support of a technical person to be able to use STEVE.
<b>5</b>	I found the various functions in STEVE were well integrated.
<b>6</b>	I thought there was too much inconsistency in STEVE.
<b>7</b>	I would imagine that most people would learn to use STEVE very quickly.
<b>8</b>	I found STEVE very cumbersome (awkward) to use.
<b>9</b>	I felt very confident using STEVE.
<b>10</b>	I needed to learn a lot of things before I could get going with STEVE.

on the number of positive answers and the quality of the users’ STEVE project for Task 3.

#### 7.2.4 SUS Questionnaire for G4

To achieve goal **G4**, *Analyze STEVE for the purpose of evaluation with respect to its overall usability from the point of view of the users*, we have applied the SUS questionnaire [28]. The System Usability Scale (SUS) provides a reliable tool for measuring usability. It consists of a ten-item questionnaire that uses a five-point Likert scale. Users need to select one of the five options for each question from 1 (strongly disagree) to 5 (strongly agree). Table 7.8 shows the SUS questionnaire for STEVE.

SUS yields a single number (SUS Score) to represent a composite measure of the overall usability [28]. Moreover, the score for each SUS question is not meaningful on its own. Therefore, we do not provide a table of metrics for each question of G4. The SUS final score is responsible for answering our goal G4 regarding STEVE’s usability.

### 7.2.5 UEQ Questionnaire for G5

To achieve goal **G5**, *Analyze STEVE for the purpose of evaluation with respect to users' experience*, we have applied the UEQ questionnaire [81, 110]. The User Experience Questionnaire (UEQ) is a fast and reliable questionnaire to measure the user experience of interactive products. It assesses STEVE giving us the users' impressions about 26 pairs of terms with opposite meanings. For each pair, users can rate it on a 7-point Likert scale. Thus, the answers to a pair of terms range from -3 (fully agree with the negative term) to +3 (fully agree with the positive term). In the questionnaire, half of the terms start with the positive attribute and the others with the negative. In addition, those pairs are listed in random order.

Table 7.9 shows the UEQ questionnaire, which is divided into six scales (*Attractiveness*, *Perspiciuity*, *Efficiency*, *Dependability*, *Stimulation*, and *Novelty*). We can interpret those scales as our metrics to support us to answer goal *G5* regarding the user experience. Additionally, those scales are measured calculating their mean value from the users' responses. In Table 7.9, we present a description and the opposite meanings for each scale.

In addition, we can group the scales into *Attractiveness*, *Pragmatic* and *Hedonic Quality* [110]. The first one is a pure valence dimension, i.e., emotional reaction on a pure acceptance/rejection dimension. *Pragmatic Quality* describes interaction qualities regarding the user tasks and contains the scales *Perspiciuity*, *Efficiency*, and *Dependability*. *Hedonic Quality* has the *Stimulation* and *Novelty* scales and describes aspects concerning pleasure or fun while using the product. Thus, Table 7.9 shows the UEQ scales grouped by those three classifications.

To summarize the structure of our goals, questions and metrics, we provide a directed graph [110], shown in Figure 7.3, whose flow is from the goal nodes to the question nodes to the metric nodes. Note that *G1*, *G2*, and *G3* share the *Q4* question. And the *PP* and *EF* metrics are also shared among the questions.

## 7.3 Results

For each goal, this section provides the analysis of the data collected from the responses of the questionnaires to evaluate STEVE regarding its usability, user experience, and the effectiveness and perspicuity of the STEVE specific features discussed in Section 7.2.

Table 7.9: UEQ's Questionnaire

		Description	Opposite Meanings
Pragmatic	Attractiveness	Overall impression of the product. Do users like or dislike it? Is it attractive, enjoyable or pleasing?	annoying / enjoyable bad / good unlikable / pleasing unpleasant / pleasant unattractive / attractive unfriendly / friendly
	Perspicuity	Is it easy to get familiar with the product? Is it easy to learn? Is the product easy to understand and clear?	not understandable / understandable easy to learn / difficult to learn complicated / easy clear / confusing
	Efficiency	Can users solve their tasks without unnecessary effort? Is the interaction efficient and fast? Does the product react fast to user input?	fast / slow inefficient / efficient impractical / practical organized / cluttered
	Dependability	Does the user feel in control of the interaction? Can he or she predict the system behavior? Does the user feel safe when working with the product?	unpredictable / predictable obstructive / supportive secure / not secure meets expectations / does not meet expectations.
Hedonic	Stimulation	Is it exciting and motivating to use the product? Is it fun to use?	valuable / inferior boring / exciting not interesting / interesting motivating / demotivating
	Novelty	Is the product innovative and creative? Does it capture users' attention?	creative / dull inventive / conventional usual / leading- edge conservative / innovative

For the  $G1$ ,  $G2$ , and  $G3$  questionnaires, we calculate the mean value for each question to achieve the goals. We use the SUS and UEQ methodologies to achieve  $G4$  and  $G5$  respectively. For SUS, we calculate the SUS final score and, for UEQ, we use the UEQ's data analysis tool<sup>4</sup>, which calculates the mean value for each UEQ scale.

<sup>4</sup>Link to the UEQ's Data Analysis Tool

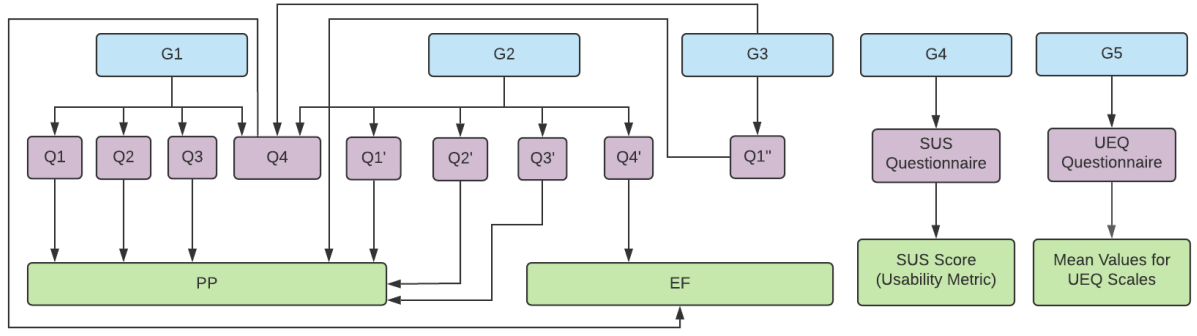


Figure 7.3: Directed Graph for our Goals, Questions and Metrics

### 7.3.1 G1 Analysis

According to Section 7.2.1, we define two metrics to achieve our *G1* goal (*Analyze the STEVE's manual synchronization between sensory effect and traditional media for the purpose of evaluation with respect to perspicuity and effectiveness from the point of view of the users*): *PP* (Perspicuity) and *EF* (Effectiveness). The first one is measured by calculating the mean value for the *Q1*, *Q2*, and *Q3* *G1* question responses using a five-point Likert scale. Therefore, a mean value greater than 2.5 for one of those questions defines that users agree, on average, with its statement.

Figure 7.4 shows the mean value (vertical axis) for *G1*, *G2* and *G3* questions (horizontal axis). Analyzing the mean value for *G1* questions *Q1*, *Q2*, and *Q3*, described in Table 7.2, we can evaluate the following. *Q1* has obtained the highest mean value (4.67) indicating the graphical representation for sensory effect types STEVE provides is easy to understand by the participants. For question *Q2*, the mean value of 2.28 indicates that, on average, users did not agree with *I found changing sensory effect duration hard in STEVE*. *Q3* indicates that users, on average (3.91), found synchronizing sensory effects with video scenes easy.

Regarding the *Q4* question, the *EF* metric is measured using the number of positive answers supported by the quality of the users' STEVE project for Task 1. In the experiment, all 43 users completed Task 1 and had their STEVE projects classified as successful. We can thus evaluate that the STEVE's manual synchronization between sensory effect and traditional media is effective from the users' point of view.

Therefore, we can evaluate that STEVE's manual synchronization between sensory effect and traditional media is perspicuous and effective from the point of view of the users, achieving our goal *G1*.

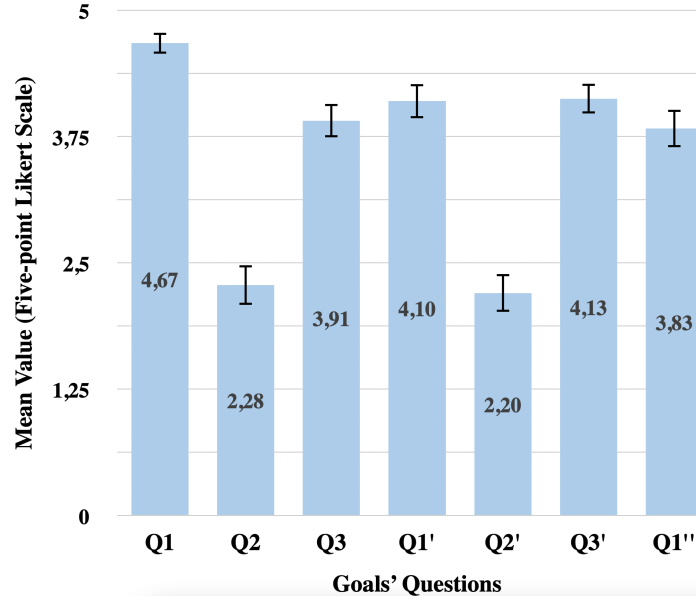


Figure 7.4: Mean Value for  $G1$ ,  $G2$ , and  $G3$  Questions

### 7.3.2 G2 Analysis

Regarding the  $G2$  goal (*Analyze the STEVE's temporal relations for the purpose of evaluation with respect to perspicuity and effectiveness from the point of view of the users*), it also uses the same  $G1$  metrics,  $PP$  (Perspicuity) and  $EF$  (Effectiveness). According to Figure 7.4,  $Q1'$  has obtained a high mean value of 4.10, suggesting that users found the temporal synchronization between media and sensory effects easy to use. For  $Q2'$ , we have obtained a mean value of 2.20, indicating users did not find the temporal alignments hard to use on average.  $Q3'$ , with a high mean value of 4.13, points out that users found the temporal alignments icons easy to understand. We can thus evaluate that STEVE's temporal relations are perspicuous from the users' point of view.

Since  $Q4$  also makes part of  $G2$ , we questioned users whether they managed to complete *Task 2* successfully. Among 44 users, only one could not complete the task, and those who completed provided succeeded STEVE projects. In addition, with  $Q4'$ , we collected which temporal alignments users made use. Most of them used the *Equal* temporal relations to synchronize the Task 1 image media with the sensory effects (users had to show an image together with the sensory effects). It also indicates that users understood how to use the temporal alignments. Therefore, we can evaluate that the STEVE's temporal relations are effective from the point of view of the users, achieving our goal  $G2$ .

### 7.3.3 G3 Analysis

Figure 7.4 shows the *Q1* question for *G3* (*Analyze the STEVE's interactivity relation feature for the purpose of evaluation with respect to perspicuity and effectiveness from the point of view of the users*). It obtained a mean value of 3.83, pointing out that users found the interactivity functionality easy to use. However, that mean value was the lowest among the others and nine users (three from non-technological area) could not complete Task 3 successfully, suggesting that STEVE's interactivity feature can be improved. Although we have concluded that the feature can be improved, we can evaluate that STEVE's interactivity relation feature is perspicuous and effective from the point of view of the users, achieving our goal *G3*.

### 7.3.4 SUS Score for G4

To achieve the *G4* goal (*Analyze STEVE for the purpose of evaluation with respect to its overall usability from the point of view of the users*), we have applied the SUS questionnaire, as discussed in Section 7.2.4. That methodology defines SUS Score for each response to its questionnaire to represent a measure of the overall usability [28]. Therefore, we calculated the SUS Score for each user and the SUS Score average among users.

We have obtained 76.62 as SUS Score average for the STEVE's experiments with a standard deviation of 12.48. That score is above average (68) [108], therefore, STEVE's overall usability can be classified as *acceptable* [18] in a range from 0 to 100, in which under 50 is *unacceptable*, between 50-70 is *marginally acceptable*, and above 70 is *acceptable*. Thus, we have achieved the *G4* goal evaluating the STEVE's overall usability.

In addition, we can classify the STEVE's usability as *Good* in an adjective scale [18] that is divided into six categories. Figure 7.5 shows those scales associated with raw SUS Score, in which the letter "E" means *Excellent* (between 80 and 85 in the SUS Score scale). Thus, we can evaluate that STEVE's temporal relations are effective from the point of view of the users, achieving our goal *G2*.

### 7.3.5 UEQ Analysis Tool for G5

For the *G5* goal (*Analyze STEVE for the purpose of evaluation with respect to users' experience*), we have analyzed the UEQ questionnaire filled out by users after completing all tasks in the experiment. As we discussed in Section 7.2.5, that methodology has six



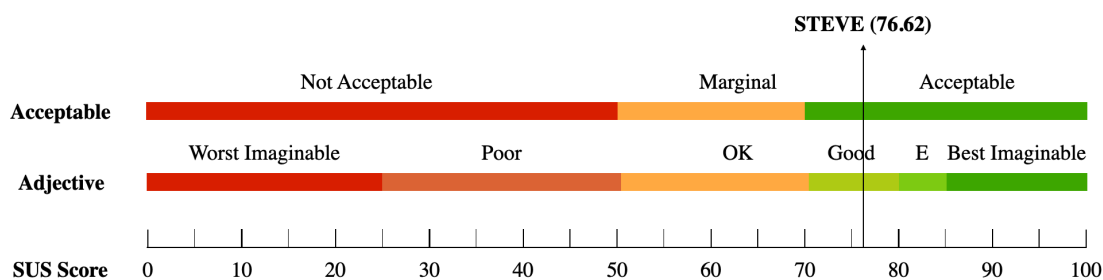


Figure 7.5: Acceptability and Adjective Scales associated with raw SUS Score

scales and, for each one, it gives us a mean value calculated from the user responses for the 26 pairs of opposite terms.

UEQ does not produce an overall score for the user experience. Instead, we have a mean value for each scale so that we can interpret it correctly. Before our interpretation, we have removed four suspicious responses (random or not serious answers) according to the simple heuristic proposed in [110]<sup>5</sup>. To detect those problematic responses, that heuristic checks how much the best and worst evaluation of an item in a scale differs. Based on the number of scales that present suspicious answers for each user, the heuristic removes all the user's responses.

Figure 7.6 shows the mean value for each UEQ scale with the respective variances we obtained in the STEVE's experiments. The range of the scales is between -3 (horribly bad) and +3 (extremely good). For all scales, except *Dependability*, STEVE was classified as *Excellent* compared with the benchmark presented in [110]. That benchmark has a data set that contains data from 468 studies regarding different products. *Excellent* means that the obtained result is in the range of the 10% best results among those studies.

For the *Dependability* scale, STEVE was classified as *Above Average*, 25% of results better, 50% of results worse. This classification is still a positive result, although it indicates that STEVE may have unexpected behaviors from the point of view of the users, according to the *Dependability* definition presented in Table 7.9.

The *Perspicuity* and *Efficiency*, which we consider the most important UX aspects for STEVE, obtained a high value of 2.012 and 2.073 respectively. In addition, *Attractiveness* and *Simulation* also obtained high values of 2.13 and 2.00 respectively. Following the scales definition of Table 7.9, we can thus conclude that STEVE is easy to understand and allows users to solve their tasks without unnecessary effort from the point of view of the users' experience. Appendix E presents the mean value per each scale item and the

<sup>5</sup>[https://www.ueq-online.org/Material/Data\\_Analysis\\_Tools.zip](https://www.ueq-online.org/Material/Data_Analysis_Tools.zip)

distribution of answers per item.

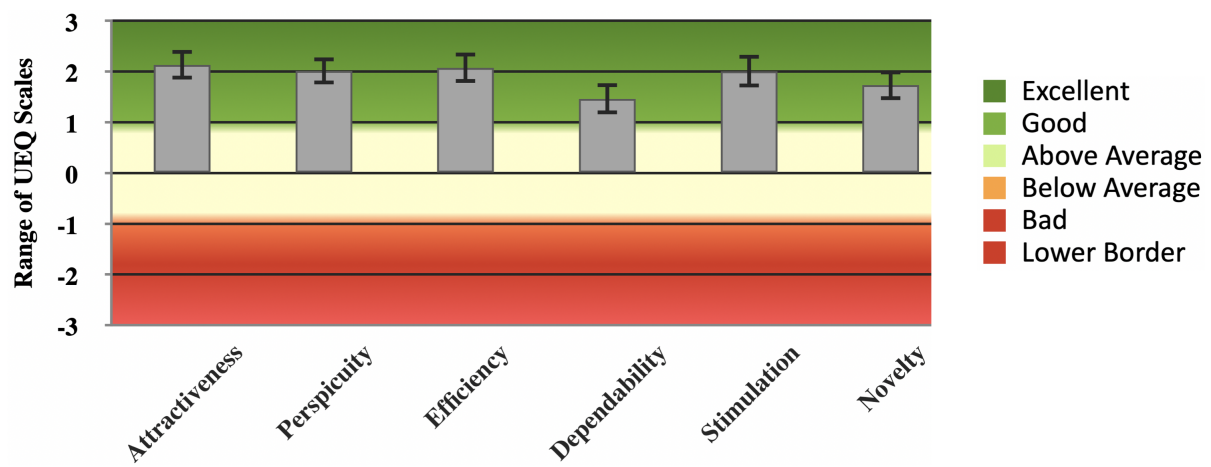


Figure 7.6: Mean Value per UEQ Scales

As discussed in Section 7.2.5, we can group the UEQ scales into *Attractiveness*, and *Pragmatic* and *Hedonic* Quality [110]. Figure 7.7 shows the mean values per each group. The *Attractiveness* has the highest value indicating that STEVE was considered by the users excellent in its overall impression, attractive, enjoyable, or pleasing. For the other groups, it was also classified as *Excellent* [110] by the users. Therefore, we can evaluate the users’ experience in the STEVE experiments as excellent regarding pragmatic and hedonic aspects, achieving our *G5* goal.

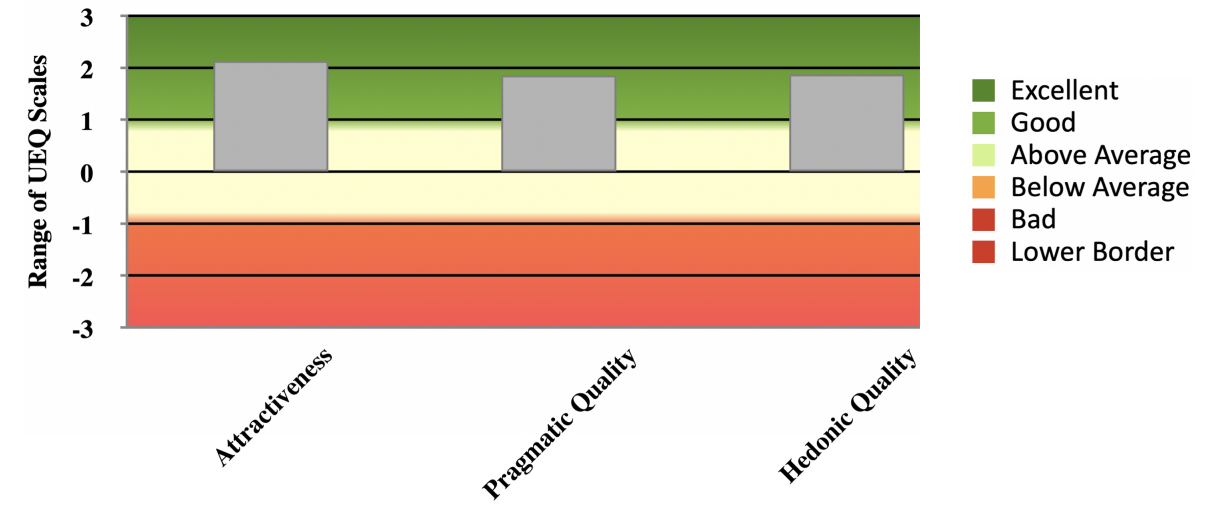


Figure 7.7: Mean Value per UEQ Group

## 7.4 Final Remarks

Following our GQM-based evaluation structure, we defined five goals for our STEVE experiments. We defined a set of questions for each goal, and for each question, we defined metrics to achieve the goals. We analyzed the effectiveness and perspicuity of STEVE's manual synchronization and temporal and interactivity relations. In addition, we used the SUS and UEQ methodologies to evaluate the STEVE's overall usability and user experience.

To overview the experiment's results, most users successfully used the manual and temporal relation synchronizations and the interactivity feature. However, some users reported they could not understand the interactivity definition, which is according to our G3 analysis. From SUS analysis, we evaluated STEVE's overall usability as *Acceptable/Good* [108, 18]. Using the UEQ methodology, we achieved our *G5* goal, classifying STEVE user experience as *Excellent* [110] for all UEQ scales, except for *Dependability*, which was evaluated as *Above Average* [110]. That scale indicates that we can improve STEVE to be more consistent in avoiding unexpected behaviors.

As a consequence of our STEVE experiment, we also demonstrated that STEVE allowed users with no programming skills (participants with no computer science background) to create their mulsemmedia applications. Moreover, we can emphasize that specifying interactivity relations in graphical authoring tools based on temporal view is a non-trivial task and requires more investigation to obtain better results in a future STEVE's usability test.

The next chapter concludes our study highlighting the thesis contributions and giving future directions in the authoring phase of mulsemmedia applications.

# Chapter 8

## Conclusions

Multimedia applications have been presented in many areas and are available on different devices. Not only massive volume of media content is consumed, but it is also produced for ordinary users. However, multimedia applications rarely provide content that explores other human senses beyond sight and hearing. In that context, the demand for producing mulsemmedia applications has encouraged several studies in the authoring phase, in particular regarding mulsemmedia authoring tools.

Therefore, this work focused on various proposals of mulsemmedia authoring tools to investigate how we can enhance the use of other human senses, such as olfaction and tactile, in mulsemmedia applications. We also presented studies concerning mulsemmedia modeling since they support the development of authoring tools and are essential in representing the structure of mulsemmedia documents. Additionally, our literature review included a multimedia background that discusses multimedia models and authoring tools to support our study in a mulsemmedia context. Indeed, we outlined desirable features for multimedia authoring tools that worked as a basis for identifying essential features for mulsemmedia ones. That set of mulsemmedia features is also supported by our analysis of several mulsemmedia authoring tools. In addition to those contributions to the mulsemmedia community, we identified gaps and future directions in researching and developing mulsemmedia authoring tools.

### 8.1 Contributions

This thesis proposed an approach for authoring mulsemmedia applications based on events that includes a mulsemmedia conceptual model (MultiSEM), an XML-based language (MultiSEL), and a graphical authoring environment for mulsemmedia applications (STEVE 2.0).

A mulsemmedia conceptual model is essential to represent the spatio-temporal behavior of mulsemmedia documents in a structured fashion. Furthermore, conceptual models aid the specification of document nodes (media and sensory effects), providing entities to represent their content and presentation characteristics. In addition, those models are crucial to define a mulsemmedia language since any language should be based on a conceptual model. The definition of a language allows authors to write mulsemmedia documents specifying their spatio-temporal behavior based on the conceptual model entities. Furthermore, providing an authoring tool enhances the development of mulsemmedia applications giving authors a graphical interface to define the mulsemmedia document elements. MultiSEM integrates sensory effects into traditional multimedia applications using the event-based temporal synchronization paradigm and modeling sensory effects as first-class entities. It aims to enhance the development of mulsemmedia applications and support mulsemmedia graphical tools with temporal view editing as the primary GUI approach. MultiSEM defines a predefined connector base that specifies different classes of temporal relations that can represent complex mulsemmedia applications to achieve that goal. Also, that connector base supports the MultiSEL language. In addition, the model defines entities to represent the presentation properties of media nodes and the rendering characteristics of sensory effects. We based on NCM (Nested Context Model) [116, 117] to define MultiSEM by simplifying NCM's structure in order to enhance the development of mulsemmedia authoring environments. The model is also based on MPEG-V's sensory effect vocabulary to define the types of sensory effects and their rendering properties. Using the concept of hypermedia connectors makes MultiSEM extensible for new relation and event types.

Additionally, we proposed the MultiSEL language, based on MultiSEM, for specifying 360° mulsemmedia applications for virtual reality environments. It aims at aiding the interoperability between mulsemmedia authoring tools. In our discussion regarding STEVE's integrations, we highlighted the integration between STEVE 2.0 and the VR authoring tool AMUSEVR [45] using MultiSEL. The proposed language also allows the specification of traditional multimedia applications.

Moreover, we proposed STEVE 2.0, an authoring environment for mulsemmedia applications. It demonstrated how MultiSEM supports the development of mulsemmedia environments based on temporal view. With STEVE's experiments, we analyzed its usability, specific features, and user experience, demonstrating that our approach for authoring mulsemmedia applications can enhance the authoring phase of mulsemmedia applications.

As another contribution, we also integrated STEVE with NCL 4.0 and Ginga-NCL,

on the need for mulsemmedia platforms. Those platforms can be seen as end-to-end tools that act in the production, distributing, and rendering phases of mulsemmedia applications.

## 8.2 Answering the Research Questions

- *How we can enhance the authoring phase of interactive multimedia applications with multiple sensory effects?* For answering our main research question, we have proposed an approach for authoring mulsemmedia applications based on events that includes a conceptual mulsemmedia model, a declarative language for mulsemmedia applications, and a graphical authoring tool based on temporal view supporting our statements described in Chapter 1. To support that central question, we answer each secondary question below.
- *What are the necessary entities a mulsemmedia conceptual model should define in order to support the development of graphical authoring environments based on temporal view for mulsemmedia applications?* We have proposed a novel mulsemmedia conceptual model in Chapter 4 describing its entities and event-based paradigm for temporal synchronization. That model aims to support mulsemmedia authoring tools that have the temporal view editing as the primary GUI approach. As proof of concept, we have implemented MultiSEM in STEVE 2.0, discussed in Chapter 6.
- *What are the necessary elements a declarative mulsemmedia language should have to support the writing of mulsemmedia documents with 360° visual content with that can be exchanged between different mulsemmedia systems?* Based on our mulsemmedia conceptual model, we have proposed a new XML-based language in Chapter 5 for 360° mulsemmedia applications describing all its elements.
- *What are the desirable features for mulsemmedia authoring tools in order to enhance the development of mulsemmedia applications even by users with no knowledge of multimedia or mulsemmedia modeling and standard languages?* In Chapter 3, based on our literature review of mulsemmedia authoring tools, we have proposed a set of desirable features for mulsemmedia authoring tools.
- *What are the gaps of the mulsemmedia authoring tools presented in our literature review?* We have identified those gaps in Chapter 3. In addition, we outlined which one of them STEVE 2.0 satisfies in the end of Chapter 6.

- *Can users with no knowledge of multimedia or mulsemmedia modeling and standard languages create their mulsemmedia applications using a graphical authoring tool?* Through STEVE's experiments reported in Chapter 7, we have demonstrated that STEVE allowed users with no programming skills (participants with no computer science background) to create their mulsemmedia applications.

## 8.3 Limitations

This section addresses limitations that need to be mentioned as they impact our results. The limitations we outlined below can be beneficial for future researchers.

- We did not manage to carry on STEVE's usability tests with different authors' profiles to analyze how those profiles can give us distinct perceptions regarding STEVE 2.0 features. Indeed, we had only two identified user groups: computer science students and users from other areas (non-technological). Moreover, we had only eight users from the non-technological group. The pandemic made STEVE's experiment a challenge since finding users available to participate in our study was a non-trivial task.
- Also, in STEVE evaluation, we had to conduct the experiment remotely in an asynchronous fashion due to the pandemic instead of a classical lab test in which users can be supervised. Thus, a portion of the preparation was passed to the users since they needed to have the appropriate equipment to run STEVE and install the software [86]. Moreover, we could not certificate the equipment they have used, if external factors negatively impact the execution of the experiment's tasks, and if others influenced them when performing the tasks and filling the experiment's questionnaires. Furthermore, both SUS and UEQ questionnaires recommend that users fill in them right after completing the tasks, which we could not guarantee.
- We were unable to carry on analyses in MultiSEL to evaluate it from the authors' point of view. This is an important future work.
- Due to time and the pandemic, we could not carry on usability tests with STEVE integrated with the mulsemmedia technologies discussed in Section 6.9. Those experiments would be beneficial in evaluating how STEVE 2.0 can enhance the development of mulsemmedia applications when other mulsemmedia technologies are integrated into its environment, providing an end-to-end tool chain.

## 8.4 Publications

This section presents the scientific papers I have published in conferences and journals.

- de Mattos, Douglas Paulo, Débora C. Muchaluat-Saade, and Gheorghita Ghinea. "Beyond Multimedia Authoring: On the Need for Mulsemmedia Authoring Tools." *ACM Computing Surveys (CSUR)* 54.7 (2021): 1-31.
- de Farias, Flávio Miranda, Douglas Paulo de Mattos, and Débora C. Muchaluat-Saade. "AMUSEVR: A Virtual Reality Authoring Environment for Immersive Mulsemmedia Applications." *Proceedings of the 1st Workshop on Multisensory Experiences - SensoryX'21*. SBC, 2021.
- Soares, Arthur Albuquerque Zopellaro, Leonardo F. Soares, Douglas Paulo de Mattos, et al. "Enabling Emulation and Evaluation of IEC 61850 Networks With TITAN." *IEEE Access*, 2021.
- de Abreu, Raphael Silva, Douglas Paulo de Mattos, Joel dos Santos et al. "Toward Content-Driven Intelligent Authoring of Mulsemmedia Applications." *IEEE MultiMedia* 28.1 (2020): 7-16.
- de Mattos, Douglas Paulo, Débora C. Muchaluat-Saade, and Gheorghita Ghinea. "An approach for authoring mulsemmedia documents based on events." *International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 2020.
- de Abreu, Raphael, Douglas Paulo de Mattos, et al. "Semi-automatic synchronization of sensory effects in mulsemmedia authoring tools." *Proceedings of the 25th Brazilian Symposium on Multimedia and the Web*. 2019.
- de Mattos, Douglas Paulo, et al. "IEC 61850 packet generator for testing substation communication." *2019 IEEE PES Asia-Pacific Power and Energy Engineering Conference (APPEEC)*. IEEE, 2019.
- de Mattos, Douglas Paulo, and Débora C. Muchaluat-Saade. "MultiSEM: A mulsemmedia model for supporting the development of authoring tools." *Proceedings of the 24th Brazilian Symposium on Multimedia and the Web*. 2018.
- de Mattos, Douglas Paulo, and Débora C. Muchaluat-Saade. "Steve: a hypermedia authoring tool based on the simple interactive multimedia model." *Proceedings of the ACM Symposium on Document Engineering*. 2018.



- Josué, Marina, Raphael Abreu, FÃ¡bio Barreto, Douglas Paulo de Mattos, et al. "Modeling sensory effects as first-class entities in multimedia applications." Proceedings of the 9th ACM Multimedia Systems Conference. 2018.
- de Mattos, Douglas Paulo, and Dorina Popovici. "VR4STEM-A 3D virtual world for assisting young people to gain entrepreneurship skill in the STEM and ICT domains." INTED2018-12th International Technology, Education and Development Conference. 2018.
- de Mattos, Douglas Paulo, et al. "Desafios da modelagem de aplicações multimídia com múltiplos efeitos sensoriais." Anais Estendidos do XXII Simpósio Brasileiro de Sistemas Multimídia e Web. SBC, 2016.
- de Mattos, Douglas Paulo, and Débora Christina Muchaluat-Saade. "STEVE: Spatial-Temporal View Editor for Authoring Hypermedia Documents." Proceedings of the 22nd Brazilian Symposium on Multimedia and the Web. 2016.

## 8.5 Future Directions

As future directions, we highlight graphical authoring tools for 360° mulsemedia applications [36, 40, 104]. In that context, users are immersed in 360° videos that are also synchronized with sensory effects. Users have the freedom to control the view from a full spherical panorama. This new class of mulsemedia applications comes with new challenges and requirements for the authoring phase. Therefore, it is essential to advance in studies that explore the integration of 360° videos in mulsemedia authoring environments. Indeed, virtual reality (VR) technologies have been explored for enhancing 360° mulsemedia authoring tools [35, 45].

Another future direction that could be addressed in the context of authoring tools is that of crossmodal correspondences [88], which may affect our perceptual experiences. This concept refers to a compatibility effect between attributes or dimensions of a stimulus in different sensory modalities. In [41], a list of crossmodal correspondences is presented among smell, taste, touch, hearing, and sight. For instance, authors describe the bouba/kiki effect experiment [118], which demonstrates that we can associate a shape with a specific sound. Accordingly, it is not inconceivable that mulsemedia authoring tools can leverage crossmodal concepts to create effective mulsemedia applications. Last but not least, mulsemedia authoring tools of

the future would be greatly aided by the emergence of modeling languages targeting mulsemmedia, as well as by mulsemmedia simulators and players - all are research efforts needing to be addressed by the community.

In addition, we can extend STEVE 2.0 to provide:

- the grouping and reuse of sensory effects to allow authors to define more complex combinations of sensory effects;
- a spatial view for sensory effects in which users can verify the spatial behavior of sensory effects and traditional multimedia items at the same time integrated with the temporal view;
- the definition of relations with statement assessments, templates for mulsemmedia applications so that users only need to define their content;
- the specification of sensory effect preparation [75, 73];
- the export of STEVE projects to MultiSEL and HTML5 documents.

Moreover, as future studies, we can carry on a new usability test in STEVE 2.0, avoiding the limitations inherent in asynchronous remote experiments, as discussed in Section 8.3, find other user profiles to collect different perceptions, and evaluate STEVE when integrated with the mulsemmedia technologies we presented in Section 6.3.

Additionally, we can evaluate the MultiSEL language carrying on usability tests, extend MultiSEM and MultiSEL to support multi-device presentations, multimodal interactions with multiple users [19, 20], and sensors' data in order to allow authors to define relations with statement assessments using those data.

# References

- [1] Iso/iec 15938 - mpeg7. <http://mpeg7.org/mpeg-7-standard/>. February, 2019 (last access).
- [2] ABNT. Digital terrestrial television - data coding and transmission specification for digital broadcasting - part 2: Ginga-ncl for fixed and mobile receivers - xml application language for application coding, 2011. ABNT NBR 15606-2:2011 standard.
- [3] ABREU, R., MATTOS, D., SANTOS, J. A. D., GHINEA, G., MUCHALUAT-SAADE, D. C. Towards content-driven intelligent authoring of mulsemmedia applications. *IEEE MultiMedia* 28, 1 (2020), 7–16.
- [4] ADEMOYE, O. A., GHINEA, G. Information recall task impact in olfaction-enhanced multimedia. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 9, 3 (2013), 1–16.
- [5] ADOBE. Actionscript references and documentation. [https://help.adobe.com/en\\_US/FlashPlatform/reference/actionscript/3/index.html](https://help.adobe.com/en_US/FlashPlatform/reference/actionscript/3/index.html).
- [6] ADOBE. Adobe animate. <https://www.adobe.com/pt/products/animate.html>, 2021.
- [7] ADOBE. Adobe Director. [adobe.com/br/products/director.html](https://www.adobe.com/br/products/director.html), 2021.
- [8] ADOBE. Adobe Premiere. [http://www.adobe.com/products/premiere.html](https://www.adobe.com/products/premiere.html), 2021.
- [9] AG, N. Nero Video 2021. [nero.com/ptb/products/nero-video/](https://www.nero.com/ptb/products/nero-video/), 2021.
- [10] ALKASASBEH, A. A., GHINEA, G. Using olfactory media cues in e-learning - perspectives from an empirical investigation. *Multimedia Tools and Applications* 79, 27 (2020), 19265–19287.
- [11] ALLEN, J. F. Maintaining knowledge about temporal intervals. *Communications of the ACM* 26, 11 (1983), 832–843.
- [12] ANGLE, E., SHREINER, D. Interactive computer graphics: A topdown approach with shader-based opengl, 2011.
- [13] ANTHES, C., GARCÍA-HERNÁNDEZ, R. J., WIEDEMANN, M., KRANZLMÜLLER, D. State of the art of virtual reality technology. In *2016 IEEE Aerospace Conference* (2016), IEEE, p. 1–19.
- [14] APPLE, INC. Final Cut Pro X. [apple.com/br/final-cut-pro/](https://www.apple.com/br/final-cut-pro/), 2021.

- [15] AYERS, Y., COHEN, A., BULTERMAN, D., OTHERS. Synchronized multimedia integration language (smil) 2.0. *W3C Recommendations* (2001).
- [16] AZEVEDO, R. G. A., ARAÚJO, E. C., LIMA, B., SOARES, L. F. G., MORENO, M. F. Composer: meeting non-functional aspects of hypermedia authoring environment. *Multimedia tools and applications* 70, 2 (2014), 1199–1228.
- [17] BAILEY, B., KONSTAN, J. A., COOLEY, R., DEJONG, M. Nsync-a toolkit for building interactive multimedia presentations. In *Proceedings of the sixth ACM international conference on Multimedia* (1998), p. 257–266.
- [18] BANGOR, A., KORTUM, P. T., MILLER, J. T. An empirical evaluation of the system usability scale. *Intl. Journal of Human–Computer Interaction* 24, 6 (2008), 574–594.
- [19] BARRETO, F. *Uma Proposta de Extensão do Middleware Ginga-NCL para Interação Multimodal e Suporte Multiusuário em Ambientes Hipermedia*. Tese de Doutorado, Universidade Federal Fluminense, 2021.
- [20] BARRETO, F., DE ABREU, R. S., MONTEVECCHI, E. B. B., JOSUÉ, M. I., VALENTIM, P. A., MUCHALUAT-SAADE, D. C. Extending ginga-ncl to specify multimodal interactions with multiple users. In *Proceedings of the Brazilian Symposium on Multimedia and the Web* (2020), p. 281–288.
- [21] BARTOCCI, S., BETTI, S., MARCONE, G., TABACCHIERA, M., ZANUCCOLI, F., CHIARI, A. A novel multimedia-multisensorial 4d platform. In *2015 AEIT International Annual Conference (AEIT)* (2015), IEEE, p. 1–6.
- [22] BASILI, V. R. Software modeling and measurement: the goal/question/metric paradigm. Relatório Técnico, 1992.
- [23] BERTINO, E., FERRARI, E., STOLF, M. Mpgs: An interactive tool for the specification and generation of multimedia presentations. *IEEE Transactions on Knowledge and Data Engineering* 12, 1 (2000), 102–125.
- [24] BLACKMAGIC. Davinci Resolve 17. [blackmagicdesign.com/products/davinciresolve](https://blackmagicdesign.com/products/davinciresolve), 2021.
- [25] BLAKOWSKI, G., STEINMETZ, R. A media synchronization survey: Reference model, specification, and case studies. *IEEE journal on selected areas in communications* 14, 1 (1996), 5–35.
- [26] BOLL, S., KLAS, W. Z/sub y/xa multimedia document model for reuse and adaptation of multimedia content. *IEEE transactions on knowledge and data engineering* 13, 3 (2001), 361–382.
- [27] BOUYAKOUB, S., BELKHIR, A. Smil builder: An incremental authoring tool for smil documents. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 7, 1 (2011), 1–30.
- [28] BROOKE, J. Sus-a quick and dirty usability scale. *Usability evaluation in industry* 189, 194 (1996), 4–7.

- [29] BUCHANAN, M. C., ZELLWEGER, P. T. Specifying temporal behavior in hypermedia documents. In *Proceedings of the ACM conference on Hypertext* (1992), p. 262–271.
- [30] BULTERMAN, D. C., HARDMAN, L. Structured multimedia authoring. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 1, 1 (2005), 89–109.
- [31] BULTERMAN, D. C., HARDMAN, L., JANSEN, J., MULLENDER, K. S., RUTLEDGE, L. Grins: A graphical interface for creating and playing smil documents. *Computer Networks and ISDN systems* 30, 1-7 (1998), 519–529.
- [32] CHAN, S. W.-C., THOMPSON, D. R., CHAU, J. P., TAM, W. W., CHIU, I. W., LO, S. H. The effects of multisensory therapy on behaviour of adult clients with developmental disabilities - a systematic review. *International Journal of Nursing Studies* 47, 1 (2010), 108–122.
- [33] CHO, H. Y. Event-based control of 4d effects using mpeg rose. In *Master's thesis - School of Mechanical, Aerospace and S. Engineering. Korea Adv. Inst. of Science and Technology* (2010).
- [34] CHOI, B., LEE, E.-S., YOON, K. Streaming media with sensory effect. In *2011 International Conference on Information Science and Applications* (2011), IEEE, p. 1–6.
- [35] COELHO, H., MELO, M., MARTINS, J., BESSA, M. Collaborative immersive authoring tool for real-time creation of multisensory vr experiences. *Multimedia Tools and Applications* 78, 14 (2019), 19473–19493.
- [36] COMŞA, I.-S., SALEME, E. B., COVACI, A., ASSRES, G. M., TRESTIAN, R., SANTOS, C. A., GHINEA, G. Do i smell coffee? the tale of a 360° mulsemedia experience. *IEEE MultiMedia* 27, 1 (2019), 27–36.
- [37] CONSORTIUM, W. W. W. W. Synchronized Multimedia Integration Language. [w3.org/TR/SMIL/](http://w3.org/TR/SMIL/), 2008.
- [38] CONSORTIUM, W. W. W. W. HTML5. [w3.org/TR/html5/](http://w3.org/TR/html5/), 2014.
- [39] COSTA, R. M. D. R., MORENO, M. F., GOMES SOARES, L. F. Intermedia synchronization management in dtv systems. In *ACM Symposium on Document Engineering* (2008), p. 289–297.
- [40] COVACI, A., TRESTIAN, R., SALEME, E. B., COMSA, I.-S., ASSRES, G., SANTOS, C. A., GHINEA, G. 360° mulsemedia: a way to improve subjective qoe in 360° videos. In *Proceedings of the 27th ACM International Conference on Multimedia* (2019), p. 2378–2386.
- [41] COVACI, A., ZOU, L., TAL, I., MUNTEAN, G.-M., GHINEA, G. Is multimedia multisensorial ? - a review of mulsemedia systems. *ACM Computing Surveys (CSUR)* 51, 5 (2018), 91.

- [42] DANIEAU, F., BERNON, J., FLEUREAU, J., GUILLOT, P., MOLLET, N., CHRISTIE, M., LÉCUYER, A. H-studio: an authoring tool for adding haptic and motion effects to audiovisual content. In *Proceedings of the adjunct publication of the 26th annual ACM symposium on User interface software and technology* (2013), p. 83–84.
- [43] DE ABREU, R. S., MATTOS, D., DOS SANTOS, J., GHINEA, G., MUCHALUAT-SAADE, D. C. Toward content-driven intelligent authoring of mulsemmedia applications. *IEEE MultiMedia* 28, 1 (2020), 7–16.
- [44] DE AMORIM, M. N., SALEME, E. B., DE ASSIS NETO, F. R., SANTOS, C. A., GHINEA, G. Crowdsourcing authoring of sensory effects on videos. *Multimedia Tools and Applications* 78, 14 (2019), 19201–19227.
- [45] DE FARIAS, F. M., DE MATTOS, D. P., MUCHALUAT-SAADE, D. C. Amusevr: A virtual reality authoring environment for immersive mulsemmedia applications. In *Proceedings of the 1st Workshop on Multisensory Experiences-SensoryX’21* (2021), SBC.
- [46] DE MATTOS, D. P., MUCHALUAT-SAADE, D. C. Steve - editor gráfico da visão espaço-temporal para autoria de documentos hipermídia. master’s thesis. Dissertação de Mestrado, Universidade Federal Fluminense, 2016.
- [47] DE MATTOS, D. P., MUCHALUAT-SAADE, D. C. Steve: a hypermedia authoring tool based on the simple interactive multimedia model. In *Proceedings of the ACM Symposium on Document Engineering 2018* (2018), ACM, p. 1–10.
- [48] DE MATTOS, D. P., MUCHALUAT-SAADE, D. C., GHINEA, G. An approach for authoring mulsemmedia documents based on events. In *2020 International Conference on Computing, Networking and Communications (ICNC)* (2020), IEEE, p. 273–277.
- [49] DE OLIVEIRA, M. C. F., TURINE, M. A. S., MASIERO, P. C. A statechart-based model for hypermedia applications. *ACM Transactions on Information Systems (TOIS)* 19, 1 (2001), 28–52.
- [50] DE SOUSA, M. F., FERRAZ, C. A. G., KULESZA, R., AYRES, I., LIMA, M. Mulsem-maker: An mdd tool for mulsemmedia web application development. In *Proceedings of the 23rd Brazilian Symposium on Multimedia and the Web* (2017), ACM, p. 317–324.
- [51] DELTOUR, R., LAYAIDA, N., WECK, D. Limsee2: A cross-platform smil2.0 authoring tool. *The European Research Consortium for Informatics and Mathematics-ERCIM News*, 62 (2005).
- [52] DÍAZ, P., AEDO, I., PANETSOS, F. Modeling the dynamic behavior of hypermedia applications. *IEEE Transactions on Software Engineering* 27, 6 (2001), 550–572.
- [53] DOS SANTOS, J. A. F., MUCHALUAT-SAADE, D. C. Xtemplate 3.0: spatio-temporal semantics and structure reuse for hypermedia compositions. *Multimedia Tools and Applications* 61, 3 (2012), 645–673.

- [54] ECHIFFRE, M., MARCHISIO, C., MARCHISIO, P., PANICCIARI, P., DEL ROSSI, S. Mhec-5-aims, concepts, and implementation issues. *IEEE MultiMedia* 5, 1 (1998), 84–91.
- [55] ESCOBEDO, L., TENTORI, M., QUINTANA, E., FAVELA, J., GARCIA-ROSAS, D. Using augmented reality to help children with autism stay focused. *IEEE Pervasive Computing* 13, 1 (2014), 38–46.
- [56] FEELREAL, INC. Feelreal. <https://feelreal.com>, 2021.
- [57] FELIX, M. F. *Formal Analysis of Software Models Oriented by Architectural Abstractions*. Tese de Doutorado, Pontifícia Universidade Católica do Rio de Janeiro, 2004. in Portuguese.
- [58] FOR STANDARDIZATION, I. O. Information technology – coding of audio-visual objects – part 11: Scene description and application engine, 2015. ISO/IEC 14496-11:2015.
- [59] FUJIKAWA, K., SHIMOJO, S., MATSUURA, T., NISHIO, S., MIYAHARA, H. Multimedia presentation system "harmony" with temporal and active mediay 0.
- [60] FURUTA, R., STOTTS, P. D. Trellis: A formally defined hypertextual basis for integrating task and information. *Coordination theory and collaboration technology* (2001), 341.
- [61] GAGGI, O., CELENTANO, A. A visual authoring environment for prototyping multimedia presentations. In *Fourth International Symposium on Multimedia Software Engineering, 2002. Proceedings.* (2002), IEEE, p. 206–213.
- [62] GHINEA, TIMMERER, LIN, GULLIVER. Mulsemedia: State of the art, perspectives, and challenges. *ACM Transactions on Multimedia Computing, Communications, and Applications* 11, 1s (2014), 17.
- [63] GROUP, W. S. W. <https://www.w3.org/AudioVideo/>, 2012.
- [64] GSÖLLPOINTNER, K., SCHNELL, R., SCHULER, R. K. *Digital Synesthesia: A Model for the Aesthetics of Digital Art*. Walter de Gruyter GmbH & Co KG, 2016.
- [65] GUEDES, A. L. V., DE ALBUQUERQUE AZEVEDO, R. G., BARBOSA, S. D. J. Extending multimedia languages to support multimodal user interactions. *Multimedia Tools and Applications* 76 (2017), 5691–5720.
- [66] GUIMARÃES, R. L., DE RESENDE COSTA, R. M., SOARES, L. F. G. Composer: Authoring tool for itv programs. In *European Conference on Interactive Television* (2008), Springer, p. 61–71.
- [67] HARDMAN, H. L., OTHERS. *Modelling and Authoring Hypermedia Documents*. Ph.d. thesis, Univ. Amsterdam, 1998.
- [68] HARDMAN, L., BULTERMAN, D. C. Authoring support for durable interactive multimedia presentations. *STAR Report in Eurographics* 95 (1995).

- [69] HUNT, A., MCGLASHAN, S. E. Speech recognition grammar specification. W3C, 2004.
- [70] HUSSAIN, A., KUTAR, M. Usability metric framework for mobile phone application. *PGNet, ISBN 2099* (2009), 978–1.
- [71] IERUSALIMSKY, R. *Programming in lua*, 2nd ed. Roberto Ierusalimsky, March 2006.
- [72] JANSEN, J., BULTERMAN, D. C. Smil state: an architecture and implementation for adaptive time-based web applications. *Multimedia Tools and Applications* 43, 3 (2009), 203–224.
- [73] JOSUÉ, M. *Preparação de Objetos de Mídia e Efeitos Sensoriais para Formatação de Documentos Mulsemídia*. Ph.d. thesis, Universidade Federal Fluminense, 2021.
- [74] JOSUÉ, M., ABREU, R., BARRETO, F., MATTOS, D., AMORIM, G., DOS SANTOS, J., MUCHALUAT-SAADE, D. Modeling sensory effects as first-class entities in multimedia applications. In *Proceedings of the 9th ACM Multimedia Systems Conference* (2018), ACM, p. 225–236.
- [75] JOSUÉ, M., MORENO, M., MUCHALUAT-SAADE, D. Mulsemmedia preparation: a new event type for preparing media object presentation and sensory effect rendering. In *Proceedings of the 10th ACM Multimedia Systems Conference* (2019), p. 110–120.
- [76] JOURDAN, M., LAYAÏDA, N., ROISIN, C., SABRY-ISMAÏL, L., TARDIF, L. Madeus, and authoring environment for interactive multimedia documents. In *Proceedings of the sixth ACM international conference on Multimedia* (1998), p. 267–272.
- [77] KIM, S. Authoring multisensorial content. *Signal Processing: Image Communication* 28, 2 (2013).
- [78] KIM, S., HAN, J. Text of white paper on mpeg-v. In *MPEG Group Meeting, ISO/IEC JTC* (2014), vol. 1.
- [79] KIM, S.-K., JOO, Y.-S., LEE, Y. Sensible media simulation in an automobile application and human responses to sensory effects. *ETRI Journal* 35, 6 (2013), 1001–1010.
- [80] KIM, S.-K., YANG, S.-J., AHN, C. H., JOO, Y. S. Sensorial information extraction and mapping to generate temperature sensory effects. *ETRI Journal* 36, 2 (2014), 224–231.
- [81] LAUGWITZ, B., HELD, T., SCHREPP, M. Construction and Evaluation of a User Experience Questionnaire. *HCI and Usability for Education and Work, 4th Symposium of the Workgroup Human-Computer Interaction and Usability Engineering of the Austrian Computer Society* 5298 (2008), 63–76.
- [82] MARCHISIO, C., MARCHISIO, P. Mediatouch: A native authoring tool for mheg-5 applications. *Multimedia Tools and Applications* 14, 1 (2001), 5–22.



- [83] MATTOS, D. P., MUCHALUAT-SAADE, D. C. Multisem: A mulsemedia model for supporting the development of authoring tools. In *Proceedings of the 24th Brazilian Symposium on Multimedia and the Web* (2018), ACM, p. 109–116.
- [84] MATTOS, D. P. D., MUCHALUAT SAADE, D. C. Steve: Spatial-temporal view editor for authoring hypermedia documents. In *Proceedings of the 22nd Brazilian Symposium on Multimedia and the Web* (2016), p. 63–70.
- [85] MATTOS, D. P. D., MUCHALUAT-SAADE, D. C., GHINEA, G. Beyond multimedia authoring: On the need for mulsemedia authoring tools. *ACM Computing Surveys (CSUR)* 54, 7 (2021), 1–31.
- [86] MCFADDEN, E., HAGER, D. R., ELIE, C. J., BLACKWELL, J. M. Remote usability evaluation: Overview and case studies. *International journal of human-computer interaction* 14, 3-4 (2002), 489–502.
- [87] MEIXNER, B. Hypervideos and interactive multimedia presentations. *ACM Computing Surveys (CSUR)* 50, 1 (2017), 1–34.
- [88] MESFIN, G., HUSSAIN, N., KANI-ZABIHI, E., COVACI, A., SALEME, E. B., GHINEA, G. Qoe of cross-modally mapped mulsemedia: an assessment using eye gaze and heart rate. *Multimedia Tools and Applications* 79, 11 (2020), 7987–8009.
- [89] MESFIN, G., SALEME, E. B., ADEMOYE, O., KANI-ZABIHI, E., SANTOS, C., GHINEA, G. Less is (just as good as) more-an investigation of olfactory intensity and hedonic valence in mulsemedia qoe using heart rate and eye tracking. *IEEE Transactions on Multimedia* (2020).
- [90] MONKS, J., OLARU, A., TAL, I., MUNTEAN, G. Quality of experience assessment of 3d video synchronised with multisensorial media components. In *Broadband Multimedia Systems and Broadcasting (BMSB), 2017 IEEE International Symposium on* (2017), IEEE, p. 1–6.
- [91] MUCHALUAT-SAADE, D. C., SOARES, L. F. G. Xconnertor and xtemplate: improving the expressiveness and reuse in web authoring languages. *New review of hypermedia and multimedia* 8, 1 (2002), 139–169.
- [92] NA, J.-C., FURUTA, R. Dynamic documents: authoring, browsing, and analysis using a high-level petri net-based hypermedia system. In *Proceedings of the 2001 ACM Symposium on Document engineering* (2001), ACM, p. 38–47.
- [93] NISHIDA, J., SUZUKI, K. Biosync: A paired wearable device for blending kinesthetic experience. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (2017), ACM, p. 3316–3327.
- [94] OGAWA, R., HARADA, H., KANEKO, A. Scenario-based hypermedia: A model and a system. In *ECHT* (1990), vol. 90, p. 38–51.
- [95] PARÉS, N., CARRERAS, A., DURANY, J., FERRER, J., FREIXA, P., GÓMEZ, D., KRUGLANSKI, O., PARÉS, R., RIBAS, J. I., SOLER, M., OTHERS. Promotion of creative activity in children with severe autism through visuals in an interactive multisensory environment. In *Proceedings of the 2005 conference on Interaction design and children* (2005), ACM, p. 110–116.

- [96] PAULO DE MATTOS, D., VARANDA DA SILVA, J., MUCHALUAT-SAADE, D. C. Next: graphical editor for authoring ncl documents supporting composite templates. In *Proceedings of the 11th european conference on Interactive TV and video* (2013), p. 89–98.
- [97] PEREIRA, F., EBRAHIMI, T. *The MPEG-4 book*. Prentice-Hall, 2002.
- [98] PÉREZ-LUQUE, M. J., LITTLE, T. D. C. A temporal reference framework for multimedia synchronization. *Journal on Selected Areas in Communications* 14, 1 (January 1996), 36–51.
- [99] PESCHANSKII, V. Y. Timewise data processing with programming language asampl. 132–137.
- [100] PETERSON, J. L. Petri net theory and the modeling of systems.
- [101] PIERETTI, R. A., KAUL, S. D., ZARCHY, R. M., O’HANLON, L. M. Using a multimodal approach to facilitate articulation, phonemic awareness, and literacy in young children. *Communication Disorders Quarterly* 36, 3 (2015), 131–141.
- [102] RAINER, B., WALTL, M., CHENG, E., SHUJAU, M., TIMMERER, C., DAVIS, S., BURNETT, I., RITZ, C., HELLWAGNER, H. Investigating the impact of sensory effects on the quality of experience and emotional response in web videos. In *Quality of Multimedia Experience, Fourth International Workshop on* (2012), IEEE, p. 278–283.
- [103] REC, I. H. 761, nested context language (ncl) and ginga-ncl for iptv services, geneva, apr. 2009, 2012.
- [104] SALEME, E. B., COVACI, A., MESFIN, G., SANTOS, C. A., GHINEA, G. Mulse-media diy: a survey of devices and a tutorial for building your own mulsemedia environment. *ACM Computing Surveys (CSUR)* 52, 3 (2019), 1–29.
- [105] SANTOS, C. A. S., NETO, A. N. R., SALEME, E. B. An event driven approach for integrating multi-sensory effects to interactive environments. In *2015 IEEE International Conference on Systems, Man, and Cybernetics* (2015), IEEE, p. 981–986.
- [106] SANTOS, J. A. D., MUCHALUAT-SAADE, D. C., ROISIN, C., LAYAÏDA, N. A hybrid approach for spatio-temporal validation of declarative multimedia documents. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 14, 4 (2018), 1–24.
- [107] SANTOS, J. A. F., SILVA, J. V., VASCONCELOS, R., SCHAU, W., WERNER, C., MUCHALUAT-SAADE, D. C. aNa: API for NCL Authoring. *WebMedia - Workshop on Tools and Applications* (2012).
- [108] SAURO, J., LEWIS, J. R. *Quantifying the user experience: Practical statistics for user research*. Morgan Kaufmann, 2016.
- [109] SCHERP, A., BOLL, S. Paving the last mile for multi-channel multimedia presentation generation. In *Multimedia Modelling Conference, 2005. MMM 2005. Proceedings of the 11th International* (2005), IEEE, p. 190–197.

- [110] SCHREPP, M., HINDERKS, A., THOMASCHEWSKI, J. Construction of a Benchmark for the User Experience Questionnaire (UEQ). *International Journal of Interactive Multimedia and Artificial Intelligence* 4, 4 (2017), 40.
- [111] SHIN, S.-H., HA, K.-S., YUN, H.-O., NAM, Y.-S. Realistic media authoring tool based on mpeg-v international standard. In *2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)* (2016), IEEE, p. 730–732.
- [112] SILVA, E. C. O., DOS SANTOS, J. A., MUCHALUAT-SAADE, D. C. Ncl4web: translating ncl applications to html5 web pages. In *Proceedings of the 2013 ACM symposium on Document engineering* (2013), p. 253–262.
- [113] SILVA, E. C. O., DOS SANTOS, J. A., MUCHALUAT-SAADE, D. C. NCL4WEB: translating NCL applications to HTML5 web pages. *ACM Symposium on Document Engineering*, p. 253–262.
- [114] SOARES, L. F. G., ET AL. Modeling, authoring and formatting hypermedia documents in the hyperprop system. *Multimedia Systems* (2000).
- [115] SOARES, L. F. G., RODRIGUES, R. F. Nested context model 3.0 part 1 - ncm core. Relatório Técnico, Informatics Department, PUC-Rio, Rio de Janeiro, May 2005.
- [116] SOARES, L. F. G., RODRIGUES, R. F. Nested context model 3.0: Part 1-ncm core. *Technical Report No. 18. Telemidia Lab, PUC-Rio, Rio de Janeiro* (2005).
- [117] SOARES, L. F. G., RODRIGUES, R. F., SAADE, D. C. M. Modeling, authoring and formatting hypermedia documents in the hyperprop system. *Multimedia systems* 8, 2 (2000), 118–134.
- [118] SPENCE, C. Crossmodal correspondences: A tutorial review. *Attention, Perception, & Psychophysics* 73, 4 (2011), 971–995.
- [119] SULEMA, Y. Asampl: Programming language for mulsemedia data processing based on algebraic system of aggregates. In *Interactive Mobile Communication, Technologies and Learning* (2017), Springer, p. 431–442.
- [120] SULEMA, Y., GLINSKII, V. Semantics and pragmatics of programming language asampl. *Problems in Programming*, 1 (2020), 74–83.
- [121] THIEME, H., MORKISCH, N., BORGETTO, B., DOHLE, C. Movement representation techniques for treating limb pain—a systematic review and metaanalysis. *Physiotherapy* 101 (2015), e1507–e1508.
- [122] TIMMERER, C. Iso/iec cd 23005-3 3rd edition sensory information. <https://mpeg.chiariglione.org/standards/mpeg-v/sensory-information>, 2009.
- [123] VAN ROSSUM, G., JANSEN, J., MULLENDER, K. S., BULTERMAN, D. C. Cmifed: a presentation environment for portable hypermedia documents. In *Proceedings of the first ACM international conference on Multimedia* (1993), p. 183–188.
- [124] VOGT, T., HERPERS, R., ASKEW, C. D., SCHERFGEN, D., STRÜDER, H. K., SCHNEIDER, S. Effects of exercise in immersive virtual environments on cortical neural oscillations and mental state. *Neural plasticity* 2015 (2015).

- [125] WAHL, T., ROTHERMEL, K. Representing time in multimedia systems. In *ICMCS* (1994), p. 538–543.
- [126] WALTL, M., RAINER, B., TIMMERER, C., HELLWAGNER, H. An end-to-end tool chain for sensory experience based on mpeg-v. *Signal Processing: Image Communication* 28, 2 (2013), 136–150.
- [127] WALTL, M., TIMMERER, C., HELLWAGNER, H. Improving the quality of multimedia experience through sensory effects. In *Quality of Multimedia Experience, Second International Workshop on* (2010), IEEE, p. 124–129.
- [128] WILLRICH, R., DE SAQUI-SANNES, P., SÉNAC, P., ENSICA, F., DIAZ, M. Ht-spn: an experience in formal modeling of multimedia applications coded in mhev or java. *Design and Management of Multimedia Information Systems: Opportunities and Challenges: Opportunities and Challenges* (2000), 380.
- [129] YANG, C.-C., CHU, C.-K., WANG, Y.-C. Extension of timeline-based editing for non-deterministic temporal behavior in smil2.0 authoring. *Journal of Information Science & Engineering* 24, 5 (2008).
- [130] YOON, K. Iso/iec cd 23005-6 3rd edition common types and tools. <https://mpeg.chiariglione.org/standards/mpeg-v/common-types-and-tools>.
- [131] YUAN, Z., GHINEA, G., MUNTEAN, G. Beyond multimedia adaptation: Quality of experience-aware multi-sensorial media delivery. *IEEE Transactions on Multimedia* 17, 1 (2015), 104–117.
- [132] ZINK, M., SITARAMAN, R., NAHRSTEDT, K. Scalable 360° video stream delivery: Challenges, solutions, and opportunities. *Proceedings of the IEEE* 107, 4 (2019), 639–650.

## APPENDIX A – MultiSEM’s Connector Base

### A.1 Connectors for Allen’s Temporal Relations

```

1
2 <?xml version="1.0"?>
3 <connectorBase name="Allen's Temporal Relations"
4 xmlns="http://www.telemidia.puc-rio.br/specs/xml/XConnector"
5 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
6 xsi:schemaLocation="http://www.telemidia.puc-rio.br/specs/xml/XConnector
7 http://www.telemidia.puc-rio.br/specs/xml/XConnector.xsd">
8
9   <!-- Relation "Starts" -->
10  <xconnector id="starts" xsi:type="CausalHypermediaConnector">
11    <conditionRole id="x" eventType="presentation">
12      <condition xsi:type="EventTransitionCondition" transition="starts"/>
13    </conditionRole>
14    <actionRole id="y" eventType="presentation" actionType="start"/>
15    <glue>
16      <triggerExpression xsi:type="SimpleTriggerExpression" conditionRole="x" />
17      <actionExpression xsi:type="SimpleActionExpression" actionRole="y"/>
18    </glue>
19  </xconnector>
20
21  <!-- Relation "Starts with Delay" -->
22  <xconnector id="starts_delay" xsi:type="CausalHypermediaConnector">
23    <param name="delay"/>
24    <conditionRole id="x" eventType="presentation">
25      <condition xsi:type="EventTransitionCondition" transition="starts"/>
26    </conditionRole>
27    <actionRole id="y" eventType="presentation" actionType="start" delay="$delay"/>
28    <glue>
29      <triggerExpression xsi:type="SimpleTriggerExpression" conditionRole="x" />
30      <actionExpression xsi:type="SimpleActionExpression" actionRole="y"/>
31    </glue>
32  </xconnector>
33
34  <!-- Relation "Finishes" -->
35  <xconnector id="finishes" xsi:type="CausalHypermediaConnector">
36    <conditionRole id="x" eventType="presentation">
37      <condition xsi:type="EventTransitionCondition" transition="stops"/>
38    </conditionRole>
39    <actionRole id="y" eventType="presentation" actionType="stop"/>

```

```

40     <glue>
41         <triggerExpression xsi:type="SimpleTriggerExpression" conditionRole="x" />
42         <actionExpression xsi:type="SimpleActionExpression" actionRole="y"/>
43     </glue>
44 </xconnector>
45
46 <!-- Relation "Finishes with Delay" -->
47 <xconnector id="finishes_delay" xsi:type="CausalHypermediaConnector">
48     <param name="delay"/>
49     <conditionRole id="x" eventType="presentation">
50         <condition xsi:type="EventTransitionCondition" transition="stops"/>
51     </conditionRole>
52     <actionRole id="y" eventType="presentation" actionType="stop" delay="$delay"/>
53 <glue>
54     <triggerExpression xsi:type="SimpleTriggerExpression" conditionRole="x" />
55     <actionExpression xsi:type="SimpleActionExpression" actionRole="y"/>
56 </glue>
57 </xconnector>
58
59 <!-- Relation "Meets" -->
60 <xconnector id="meets" xsi:type="CausalHypermediaConnector">
61     <conditionRole id="x" eventType="presentation">
62         <condition xsi:type="EventTransitionCondition" transition="stops"/>
63     </conditionRole>
64     <actionRole id="y" eventType="presentation" actionType="start"/>
65 <glue>
66     <triggerExpression xsi:type="SimpleTriggerExpression" conditionRole="x"/>
67     <actionExpression xsi:type="SimpleActionExpression" actionRole="y"/>
68 </glue>
69 </xconnector>
70
71 <!-- Relation "Meets with Delay" -->
72 <xconnector id="meets_delay" xsi:type="CausalHypermediaConnector">.
73     <param name="delay"/>
74     <conditionRole id="x" eventType="presentation">
75         <condition xsi:type="EventTransitionCondition" transition="stops"/>
76     </conditionRole>
77     <actionRole id="y" eventType="presentation" actionType="start" delay="$delay"/>
78 <glue>
79     <triggerExpression xsi:type="SimpleTriggerExpression" conditionRole="x"/>
80     <actionExpression xsi:type="SimpleActionExpression" actionRole="y"/>
81 </glue>
82 </xconnector>
83
84 <!-- Relation "Met_By" -->
85 <xconnector id="met_by" xsi:type="CausalHypermediaConnector">
86     <conditionRole id="x" eventType="presentation">
87         <condition xsi:type="EventTransitionCondition" transition="starts"/>
88     </conditionRole>
89     <actionRole id="y" eventType="presentation" actionType="stop"/>
90 <glue>
91     <triggerExpression xsi:type="SimpleTriggerExpression" conditionRole="x" />
92     <actionExpression xsi:type="SimpleActionExpression" actionRole="y"/>
93 </glue>

```

```

94 </xconnector>
95
96 <!-- Relation "Met_By with Delay" -->
97 <xconnector id="met_by_delay" xsi:type="CausalHypermediaConnector">
98   <param name="$delay"/>
99   <conditionRole id="x" eventType="presentation">
100     <condition xsi:type="EventTransitionCondition" transition="starts"/>
101   </conditionRole>
102   <actionRole id="y" eventType="presentation" actionType="stop" delay="$delay"/>
103   <glue>
104     <triggerExpression xsi:type="SimpleTriggerExpression" conditionRole="x" />
105     <actionExpression xsi:type="SimpleActionExpression" actionRole="y"/>
106   </glue>
107 </xconnector>
108
109 </connectorBase>

```

## A.2 Connectors for Interactivity Relations

```

1
2 <?xml version="1.0"?>
3 <connectorBase name="Asynchronous Relations"
4 xmlns="http://www.telemedia.puc-rio.br/specs/xml/XConnector"
5 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
6 xsi:schemaLocation="http://www.telemedia.puc-rio.br/specs/xml/XConnector
7 http://www.telemedia.puc-rio.br/specs/xml/XConnector.xsd">
8
9   <!-- Relation "OnSelectionStart" -->
10   <xconnector id="onSelectionStart" xsi:type="CausalHypermediaConnector">
11     <conditionRole id="x" eventType="selection">
12       <condition xsi:type="EventTransitionCondition" transition="ends"/>
13     </conditionRole>
14     <actionRole id="y" eventType="presentation" actionType="start"/>
15     <glue>
16       <triggerExpression xsi:type="SimpleTriggerExpression" conditionRole="x" />
17       <actionExpression xsi:type="SimpleActionExpression" actionRole="y"/>
18     </glue>
19   </xconnector>
20
21   <!-- Relation "OnSelectionStop" -->
22   <xconnector id="onSelectionStop" xsi:type="CausalHypermediaConnector">
23     <conditionRole id="x" eventType="selection">
24       <condition xsi:type="EventTransitionCondition" transition="ends"/>
25     </conditionRole>
26     <actionRole id="y" eventType="presentation" actionType="stop"/>
27     <glue>
28       <triggerExpression xsi:type="SimpleTriggerExpression" conditionRole="x" />
29       <actionExpression xsi:type="SimpleActionExpression" actionRole="y"/>
30     </glue>
31   </xconnector>
32

```

```

33 <!-- Relation "OnSelectionSet" -->
34 <xconnector id="onSelectionSet" xsi:type="CausalHypermediaConnector">
35   <param name="setValue"/>
36   <conditionRole id="x" eventType="selection">
37     <condition xsi:type="EventTransitionCondition" transition="ends"/>
38   </conditionRole>
39   <actionRole id="y" eventType="attribution" actionType="start" finalValue="$
setValue"/>
40   <glue>
41     <triggerExpression xsi:type="SimpleTriggerExpression" conditionRole="x" />
42     <actionExpression xsi:type="SimpleActionExpression" actionRole="y"/>
43   </glue>
44 </xconnector>
45
46 <!-- Relation "OnSelectionStartStop" -->
47 <xconnector id="onSelectionStartStop" xsi:type="CausalHypermediaConnector">
48   <conditionRole id="x" eventType="selection">
49     <condition xsi:type="EventTransitionCondition" transition="ends"/>
50   </conditionRole>
51   <actionRole id="y" eventType="presentation" actionType="start"/>
52   <actionRole id="z" eventType="presentation" actionType="stop"/>
53   <glue>
54     <triggerExpression xsi:type="SimpleTriggerExpression" conditionRole="x" />
55     <actionExpression xsi:type="CompoundActionExpression" operator="par">
56       <firstAction actionRole="y" qualifier="par"/>
57       <SecondAction actionRole="z" qualifier="par"/>
58     </actionExpression>
59   </glue>
60 </xconnector>
61
62 <!-- Relation "OnSelectionStartStopDelay" -->
63 <xconnector id="onSelectionStartStopDelay" xsi:type="CausalHypermediaConnector">
64   <param name="delayStart"/>
65   <param name="delayStop"/>
66   <conditionRole id="x" eventType="selection">
67     <condition xsi:type="EventTransitionCondition" transition="ends"/>
68   </conditionRole>
69   <actionRole id="y" eventType="presentation" actionType="start" delay="$delayStart
"/>
70   <actionRole id="z" eventType="presentation" actionType="stop" delay="$delayStop"
/>
71   <glue>
72     <triggerExpression xsi:type="SimpleTriggerExpression" conditionRole="x" />
73     <actionExpression xsi:type="CompoundActionExpression" operator="par">
74       <firstAction actionRole="y" qualifier="par"/>
75       <SecondAction actionRole="z" qualifier="par"/>
76     </actionExpression>
77   </glue>
78 </xconnector>
79
80 </connectorBase>

```



## A.3 Connectors for Synchronous Relations with Attribution Action

```

1
2 <?xml version="1.0"?>
3 <connectorBase name="Synchronous Relations with Attribution Action"
4 xmlns="http://www.telemedia.puc-rio.br/specs/xml/XConnector"
5 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
6 xsi:schemaLocation="http://www.telemedia.puc-rio.br/specs/xml/XConnector
7 http://www.telemedia.puc-rio.br/specs/xml/XConnector.xsd">
8
9   <!-- Relation "StartsSet" -->
10  <xconnector id="startsSet" xsi:type="CausalHypermediaConnector">
11    <param name="setValue"/>
12    <conditionRole id="x" eventType="presentation">
13      <condition xsi:type="EventTransitionCondition" transition="starts"/>
14    </conditionRole>
15    <actionRole id="y" eventType="presentation" actionType="start"/>
16    <actionRole id="z" eventType="attribution" actionType="start" finalValue="$
17    setValue"/>
18    <glue>
19      <triggerExpression xsi:type="SimpleTriggerExpression" conditionRole="x" />
20      <actionExpression xsi:type="CompoundActionExpression" operator="par">
21        <firstAction actionRole="y" qualifier="par"/>
22        <SecondAction actionRole="z" qualifier="par"/>
23      </actionExpression>
24    </glue>
25  </xconnector>
26
27  <!-- Relation "FinishesSet" -->
28  <xconnector id="finishesSet" xsi:type="CausalHypermediaConnector">
29    <param name="setValue"/>
30    <conditionRole id="x" eventType="presentation">
31      <condition xsi:type="EventTransitionCondition" transition="stops"/>
32    </conditionRole>
33    <actionRole id="y" eventType="presentation" actionType="stop"/>
34    <actionRole id="z" eventType="attribution" actionType="start" finalValue="$
35    setValue"/>
36    <glue>
37      <triggerExpression xsi:type="SimpleTriggerExpression" conditionRole="x" />
38      <actionExpression xsi:type="CompoundActionExpression" operator="par">
39        <firstAction actionRole="y" qualifier="par"/>
40        <SecondAction actionRole="z" qualifier="par"/>
41      </actionExpression>
42    </glue>
43  </xconnector>
44 </connectorBase>

```

## A.4 Connectors for Relations with Statement Assessment

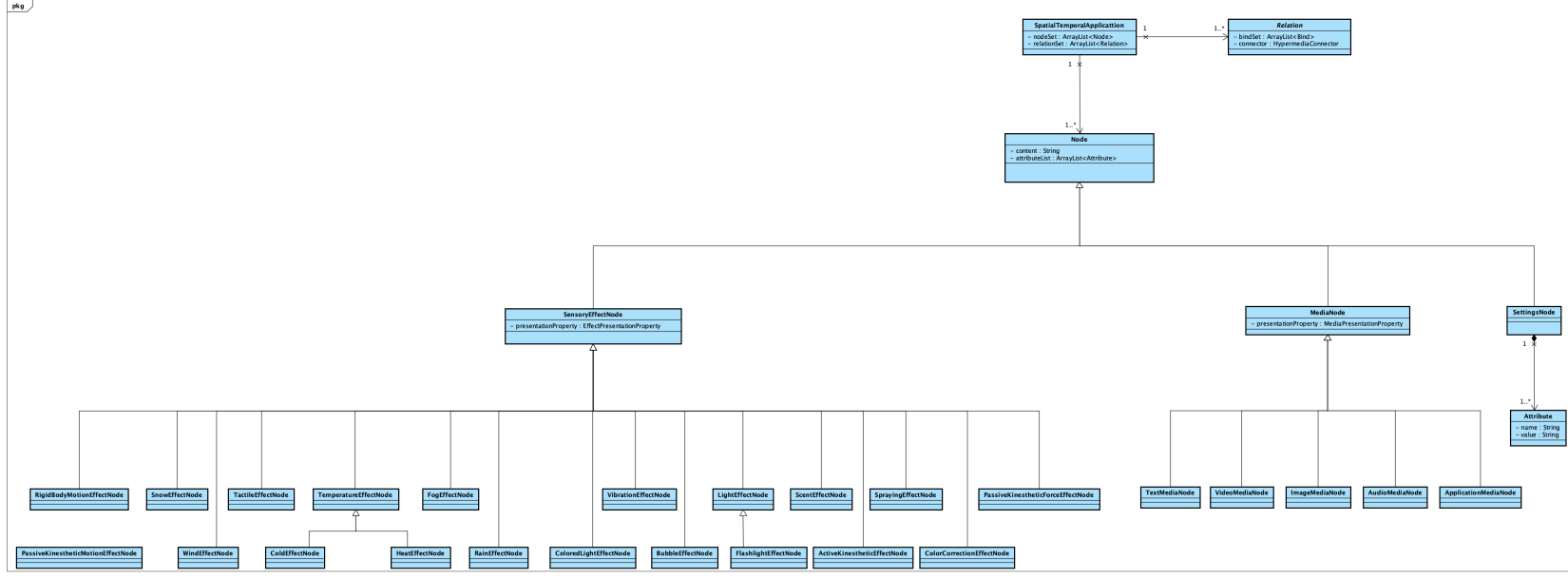
```

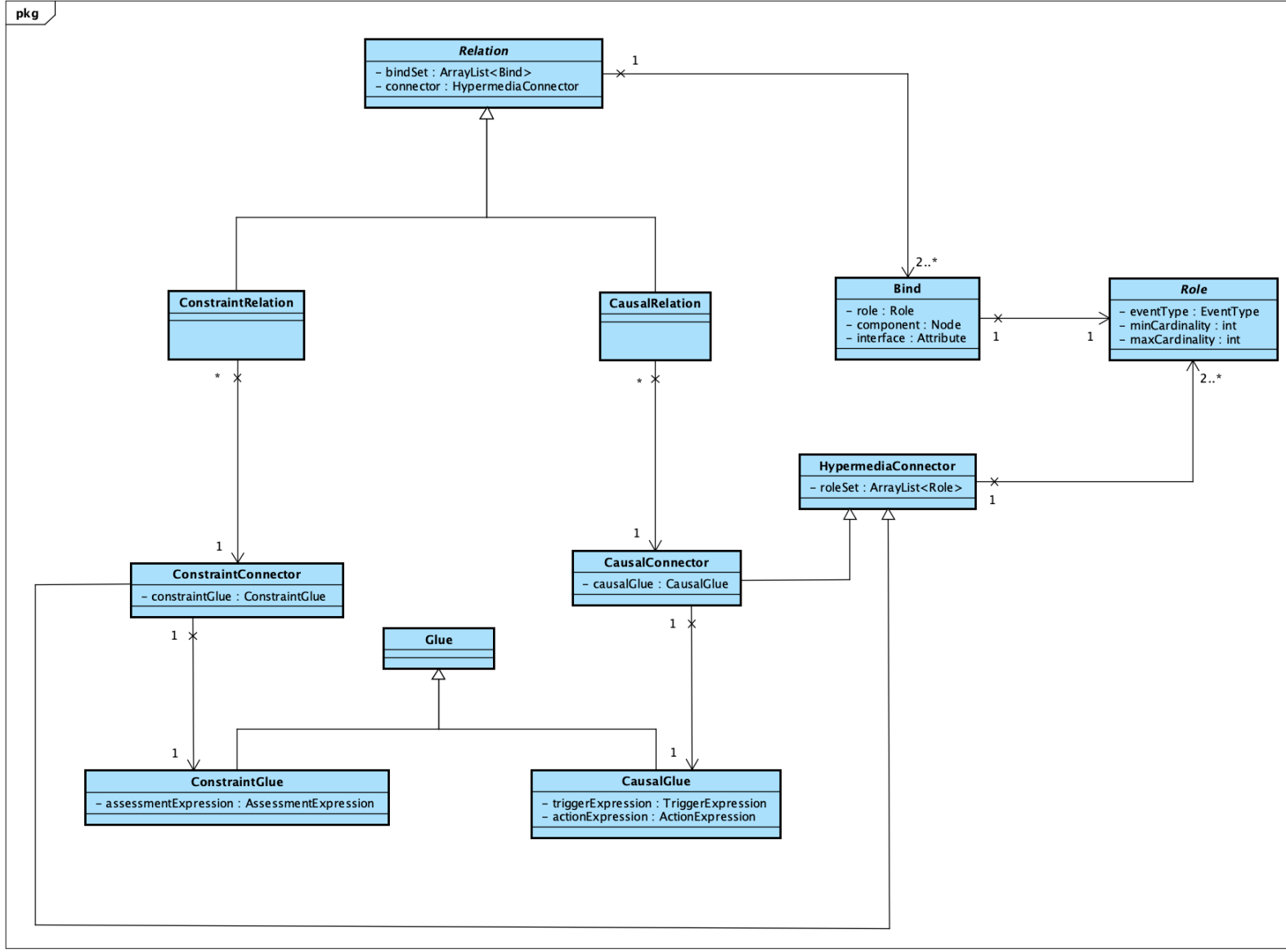
1
2 <?xml version="1.0"?>
3 <connectorBase name="Conditional Relations"
4 xmlns="http://www.telemidia.puc-rio.br/specs/xml/XConnector"
5 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
6 xsi:schemaLocation="http://www.telemidia.puc-rio.br/specs/xml/XConnector
7 http://www.telemidia.puc-rio.br/specs/xml/XConnector.xsd">
8
9   <!-- Relation "ConditionalStarts" -->
10  <xconnector id="conditionalStarts" xsi:type="CausalHypermediaConnector">
11    <param name="pValue">
12    <param name="pComparator">
13    <conditionRole id="x" eventType="presentation">
14      <condition xsi:type="EventTransitionCondition" transition="starts"/>
15    </conditionRole>
16    <propertyRole id="p" eventType="attribution">
17      <property xsi:type="NodeAttributionProperty"/>
18    </propertyRole>
19    <actionRole id="y" eventType="presentation" actionType="start"/>
20    <glue>
21      <triggerExpression xsi:type="CompoundTriggerExpression" operator="AND">
22        <trigger xsi:type="SimpleTriggerExpression" conditionRole="x"/>
23        <property xsi:type="AttributeToValueExpression" propertyRole="p" value="$
pValue" comparator="$pComparator"/>
24      </triggerExpression>
25      <actionExpression xsi:type="SimpleActionExpression" actionRole="y"/>
26    </glue>
27  </xconnector>
28
29  <!-- XConnector "OnSelectionStartSetAssessment" -->
30  <xconnector id="onSelectionStartSetAssessment" xsi:type="CausalHypermediaConnector">
31    <param name="pValue">
32    <param name="assessmentValue">
33    <param name="pComparator">
34    <param name="pKeyCode">
35    <conditionRole id="onSelection" key="$pKeyCode" eventType="selection">
36      <condition xsi:type="EventTransitionCondition" transition="ends"/>
37    </conditionRole>
38    <propertyRole id="statementAssessment" eventType="attribution">
39      <property xsi:type="NodeAttributionProperty"/>
40    </propertyRole>
41    <actionRole id="startAction" eventType="presentation" actionType="start"/>
42    <actionRole id="setAction" eventType="attribution" actionType="set" finalValue="$
pValue"/>
43    <glue>
44      <triggerExpression xsi:type="CompoundTriggerExpression" operator="AND">
45        <trigger xsi:type="SimpleTriggerExpression" conditionRole="onSelection"/>
46        <property xsi:type="AttributeToValueExpression" propertyRole="
statementAssessment" value="assessmentValue" comparator="pComparator"/>
47      </triggerExpression>

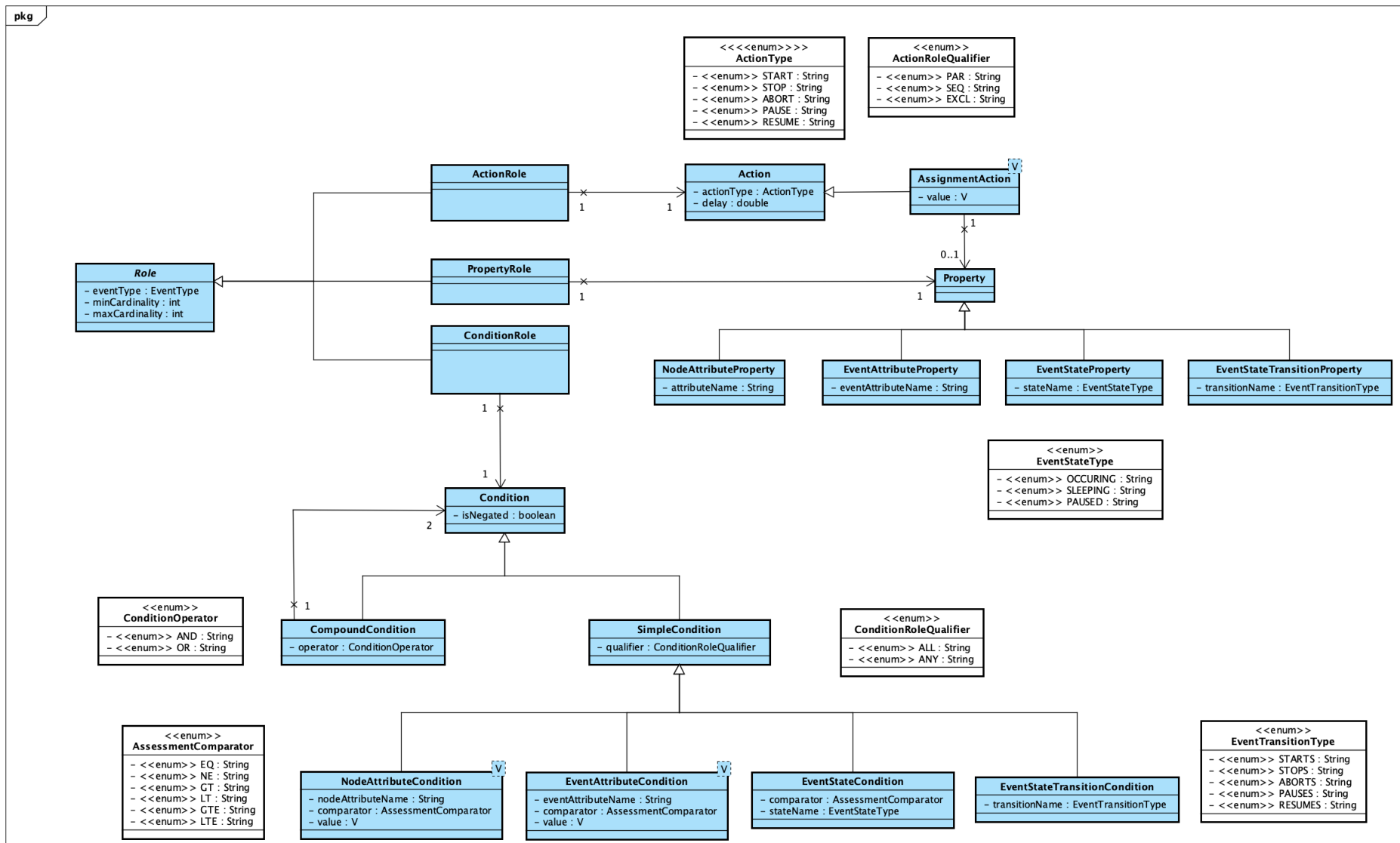
```

```
48     <actionExpression xsi:type="CompoundActionExpression" operator="par">
49         <firstAction actionRole="startAction" qualifier="par"/>
50         <SecondAction actionRole="setAction" qualifier="par"/>
51     </actionExpression>
52 </glue>
53 </xconnector>
54
55 </connectorBase>
```

## APPENDIX B - Complete UML Diagram of MultiSEM







## APPENDIX C - MultiSEL's XML Schema

### C.1 Functional Areas

```

1 <!--
2 XML Schema for the MultiSEL Language
3 This is MultiSEL
4 Copyright: 2021 LABORATORIO MIDIACOM, All Rights Reserved.
5 See https://www.midiacom.uff.br
6 Public URI: https://www.midiacom.uff.br/specs/xml/multisel/MultiSEL.xsd
7 Author: Douglas Paulo de Mattos
8 Revision: 2021/08/20
9 -->
10 <schema xmlns="http://www.w3.org/2001/XMLSchema"
11 targetNamespace="https://www.midiacom.uff.br/specs/xml/multisel"
12 elementFormDefault="qualified" attributeFormDefault="unqualified"
13 >
14   <!-- include the schema files for the building block types -->
15   <include schemaLocation="MultiSEL-structure.xsd"/>
16   <include schemaLocation="MultiSEL-metainfo.xsd"/>
17   <include schemaLocation="MultiSEL-components.xsd"/>
18   <include schemaLocation="MultiSEL-presentation.xsd"/>
19   <include schemaLocation="MultiSEL-interface.xsd"/>
20   <include schemaLocation="MultiSEL-relations.xsd"/>
21 </schema>

```

### C.2 Structure Functional Area

```

1 <!--
2 XML Schema for the MultiSEL Language
3 This is MultiSEL
4 Copyright: 2021 LABORATORIO MIDIACOM, All Rights Reserved.
5 See https://www.midiacom.uff.br
6 Public URI: https://www.midiacom.uff.br/specs/xml/multisel/MultiSEL-structure.xsd
7 Author: Douglas Paulo de Mattos
8 Revision: 2021/08/20
9 -->
10 <schema xmlns="http://www.w3.org/2001/XMLSchema"
11 targetNamespace="https://www.midiacom.uff.br/specs/xml/multisel"
12 elementFormDefault="qualified" attributeFormDefault="unqualified"

```



```

13 >
14   <element name="multisel">
15     <complexType>
16       <attribute name="xmlns" type="string" use="required"/>
17       <attribute name="id" type="string" use="required"/>
18       <attribute name="title" type="string"/>
19       <sequence>
20         <element name="head" type="headType" minOccurs="0" maxOccurs="1"/>
21         <element name="body" type="bodyType" minOccurs="0" maxOccurs="1"/>
22       </sequence>
23     </complexType>
24   </element>
25
26   <!-- define complex types -->
27   <complexType name="headType">
28     <all>
29       <element name="meta" minOccurs="0" maxOccurs="unbounded" type="metaType"/>
30       <element name="metadata" minOccurs="0" maxOccurs="unbounded" type="
31 metadataType"/>
32     </all>
33   </complexType>
34
35   <complexType name="bodyType">
36     <attribute name="id" type="string"/>
37     <all>
38       <element name="meta" minOccurs="0" maxOccurs="unbounded" type="metaType"/>
39       <element name="metadata" minOccurs="0" maxOccurs="unbounded" type="
40 metadataType"/>
41       <element name="scene" minOccurs="0" maxOccurs="unbounded" type="sceneType"/>
42       <element name="relation" minOccurs="0" maxOccurs="unbounded" type="
43 relationType"/>
44     </all>
45   </complexType>
46 </schema>

```

## C.3 Metainformation Functional Area

```

1 <!--
2 XML Schema for the MultiSEL Language
3 This is MultiSEL
4 Copyright: 2021 LABORATORIO MIDIA COM, All Rights Reserved.
5 See https://www.midiacom.uff.br
6 Public URI: https://www.midiacom.uff.br/specs/xml/multisel/MultiSEL-metainfo.xsd
7 Author: Douglas Paulo de Mattos
8 Revision: 2021/08/20
9 -->
10 <schema xmlns="http://www.w3.org/2001/XMLSchema"
11 xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema"
12 targetNamespace="https://www.midiacom.uff.br/specs/xml/multisel"
13 elementFormDefault="qualified" attributeFormDefault="unqualified"
14 >

```

```

15 <complexType name="metaType">
16   <attribute name="name" type="string"/>
17   <attribute name="value" type="string"/>
18 </complexType>
19 <complexType name="metadataType">
20   <sequence>
21     <element name="rdfTree" type="rdfs:rdf-schema"/>
22   </sequence>
23 </complexType>
24 </schema>

```

## C.4 *Components* Functional Area

```

1 <!--
2 XML Schema for the MultiSEL Language
3 This is MultiSEL
4 Copyright: 2021 LABORATORIO MIDIA COM, All Rights Reserved.
5 See https://www.midiacom.uff.br
6 Public URI: https://www.midiacom.uff.br/specs/xml/multisel/MultiSEL-components.xsd
7 Author: Douglas Paulo de Mattos
8 Revision: 2021/08/20
9 -->
10 <schema xmlns="http://www.w3.org/2001/XMLSchema"
11   targetNamespace="https://www.midiacom.uff.br/specs/xml/multisel"
12   elementFormDefault="qualified" attributeFormDefault="unqualified"
13 >
14   <complexType name="sceneType">
15     <attribute name="id" type="string"/>
16     <attribute name="primaryComponent" type="string" use="required"/>
17     <all>
18       <element name="port" minOccurs="0" maxOccurs="unbounded" type="portType"/>
19       <element name="property" minOccurs="0" maxOccurs="unbounded" type="
propertyType"/>
20       <element name="media" minOccurs="0" maxOccurs="unbounded" type="mediaType"/>
21       <element name="effect" minOccurs="0" maxOccurs="unbounded" type="effectType"
/>
22       <element name="relation" minOccurs="0" maxOccurs="unbounded" type="
relationType"/>
23     </all>
24   </complexType>
25
26   <complexType name="mediaType">
27     <attribute name="id" type="xs:string" use="required"/>
28     <attribute name="src" type="xs:string"/>
29     <attribute name="type" type="xs:string"/>
30     <sequence>
31       <element name="property" minOccurs="0" maxOccurs="unbounded" type="
propertyType"/>
32     </sequence>
33   </complexType>
34   <complexType name="effectType">

```

```

35     <attribute name="id" type="xs:string" use="required"/>
36     <attribute name="type" type="xs:string" use="required"/>
37     <sequence>
38         <element name="property" minOccurs="0" maxOccurs="unbounded" type="
propertyType"/>
39     </sequence>
40 </complexType>
41 </schema>

```

## C.5 *Presentation and Rendering Specification* Functional Area

```

1 <!--
2 XML Schema for the MultiSEL Language
3 This is MultiSEL
4 Copyright: 2021 LABORATORIO MIDIA COM, All Rights Reserved.
5 See https://www.midiacom.uff.br
6 Public URI: https://www.midiacom.uff.br/specs/xml/multisel/MultiSEL-presentation.xsd
7 Author: Douglas Paulo de Mattos
8 Revision: 2021/08/20
9 -->
10 <schema xmlns="http://www.w3.org/2001/XMLSchema"
11 targetNamespace="https://www.midiacom.uff.br/specs/xml/multisel"
12 elementFormDefault="qualified" attributeFormDefault="unqualified"
13 >
14     <complexType name="propertyType">
15         <attribute name="name" type="string" use="required"/>
16         <attribute name="value" type="string"/>
17         <attribute name="tileSize" type="string"/>
18         <attribute name="coefficient" type="string"/>
19         <attribute name="functionType" type="string"/>
20     </complexType>
21 </schema>

```

## C.6 *Interfaces* Functional Area

```

1 <!--
2 XML Schema for the MultiSEL Language
3 This is MultiSEL
4 Copyright: 2021 LABORATORIO MIDIA COM, All Rights Reserved.
5 See https://www.midiacom.uff.br
6 Public URI: https://www.midiacom.uff.br/specs/xml/multisel/MultiSEL-interface.xsd
7 Author: Douglas Paulo de Mattos
8 Revision: 2021/08/20
9 -->
10 <schema xmlns="http://www.w3.org/2001/XMLSchema"
11 targetNamespace="https://www.midiacom.uff.br/specs/xml/multisel"
12 elementFormDefault="qualified" attributeFormDefault="unqualified"

```

```

13 >
14 <complexType name="portType">
15   <attribute name="id" type="string" use="required"/>
16   <attribute name="component" type="string" use="required"/>
17   <attribute name="interface" type="string"/>
18 </complexType>
19 </schema>

```

## C.7 *Relations* Functional Area

```

1 <!--
2 XML Schema for the MultiSEL Language
3 This is MultiSEL
4 Copyright: 2021 LABORATORIO MIDIACOM, All Rights Reserved.
5 See https://www.midiacom.uff.br
6 Public URI: https://www.midiacom.uff.br/specs/xml/multisel/MultiSEL-relations.xsd
7 Author: Douglas Paulo de Mattos
8 Revision: 2021/08/20
9 -->
10 <schema xmlns="http://www.w3.org/2001/XMLSchema"
11 targetNamespace="https://www.midiacom.uff.br/specs/xml/multisel"
12 elementFormDefault="qualified" attributeFormDefault="unqualified"
13 >
14   <complexType name="relationType">
15     <attribute name="id" type="string"/>
16     <attribute name="component" type="string" use="required"/>
17     <attribute name="delay" type="string"/>
18     <attribute name="keyCode" type="string"/>
19     <all>
20       <element name="primary" type="primaryType"/>
21       <element name="secondary" type="secondaryType"/>
22       <element name="set" type="setType"/>
23       <element name="assessment" type="assessmentType"/>
24     </all>
25   <complexType>
26     <complexType name="primaryType">
27       <attribute name="component" type="string" use="required"/>
28       <attribute name="interface" type="string"/>
29     </complexType>
30     <complexType name="secondaryType">
31       <attribute name="component" type="string" use="required"/>
32       <attribute name="interface" type="string"/>
33       <attribute name="delay" type="string"/>
34     </complexType>
35     <complexType name="setType">
36       <attribute name="component" type="string" use="required"/>
37       <attribute name="interface" type="string" use="required"/>
38       <attribute name="value" type="string" use="required"/>
39     </complexType>
40     <complexType name="assessmentType">
41       <attribute name="expression" type="string" use="required"/>

```

---

```
42     </complexType>  
43 </schema>
```

---

## APPENDIX D - MultiSEL Document Example

```

1 <multisel id="360MulsemmediaApp" title="MultiSEL Case Study" xmlns="MultiSEProfile">
2   <head>
3     <meta name="author" value="Douglas Mattos"/>
4     <meta name="year" value="2021"/>
5   </head>
6   <body>
7     <scene id="scenel" primaryComponent="360gardenVideo">
8       <!-- ports -->
9       <port id="portIncreaseIntensity" component="allowIncreaseIntensity"/>
10      <port id="portRosePortalImage" component="rosePortalImage"/>
11      <port id="portRoseInt" component="roseScent" interface="intensityValue"/>
12      <port id="portGardenVideo" component="360gardenVideo"/>
13      <port id="portEnableIncreaseImage" component="enableIncreaseImage"/>
14      <!-- scene property -->
15      <property name="allowIncreaseIntensity" value="false"/>
16      <!-- media and sensory effect items -->
17      <media id="360gardenVideo" src="garden.mp4" type="360video">
18        <property name="background" value="true"/>
19      </media>
20      <media id="rosePortalImage" src="portal.jpeg" type="image">
21        <property name="position" value="0,2,6"/>
22      </media>
23      <media id="enableIncreaseImage" src="enableIncrease.png" type="image">
24        <property name="pip" value="true"/>
25      </media>
26      <effect id="roseScent" type="scentEffect">
27        <property name="scent" value="rose"/>
28        <property name="intensityValue" value="50%"/>
29        <property name="position" value="0,2,6"/>
30      </effect>
31      <!-- relations -->
32      <relation id="r1" type="starts">
33        <primary component="360gardenVideo"/>
34        <secondary component="rosePortalImage"/>
35        <secondary component="roseScent"/>
36        <secondary component="enableIncreaseImage"/>
37      </relation>
38      <relation id="r2" type="finishes">
39        <primary component="360gardenVideo"/>
40        <secondary component="rosePortalImage"/>
41        <secondary component="roseScent"/>
42        <secondary component="enableIncreaseImage"/>

```

```

43     </relation>
44     <relation id="r3" type="onSelectionSet" keyCode="A">
45         <primary component="enableIncreaseImage"/>
46         <set component="allowIncreaseIntensity" value="true"/>
47     </relation>
48     <relation id="r4" type="onSelectionSet" keyCode="B">
49         <primary component="enableIncreaseImage"/>
50         <set component="allowIncreaseIntensity" value="false"/>
51     </relation>
52 </scene>
53 <!-- relations between scenes -->
54 <relation id="r5" type="onSelectionStart" keyCode="X">
55     <primary component="scene1" interface="portRosePortalImage"/>
56     <secondary component="scene2"/>
57 </relation>
58 <relation id="r6" type="metBy">
59     <primary component="scene2"/>
60     <secondary component="scene1" interface="portGardenVideo"/>
61     <secondary component="scene1" interface="portRosePortalImage"/>
62     <secondary component="scene1" interface="portEnableIncreaseImage"/>
63 </relation>
64 <relation id="r7" type="conditionalStartsSet">
65     <primary component="scene2"/>
66     <assessment expression="scene1.portIncreaseIntensity == true"/>
67     <secondary component="scene2" interface="portWarning"/>
68     <set component="scene1" interface="portRoseInt" value="100%"/>
69 </relation>
70 <relation id="r8" type="finishes">
71     <primary component="scene2"/>
72     <secondary component="scene1" interface="roseScent"/>
73 </relation>
74 <!-- scene 2 -->
75 <scene id="scene2" primaryComponent="roseGarden">
76     <!-- port -->
77     <port id="portWarning" component="strongScentWarningImage"/>
78     <!-- media and sensory effect items -->
79     <media id="roseGarden" src="roseGarden.png" type="360Image">
80         <property name="background" value="true"/>
81         <property name="explicitDur" value="30s"/>
82     </media>
83     <media id="strongScentWarningImage" src="warning.png" type="image">
84         <effect id="wind" type="windEffect">
85             <property name="intensityValue" value="50%"/>
86             <property name="position" value="0,6,12"/>
87         </effect>
88     <!-- relations -->
89     <relation id="r9" type="starts" delay="10s">
90         <primary component="roseGarden"/>
91         <secondary component="wind"/>
92     </relation>
93     <relation id="r10" type="finishes">
94         <primary component="roseGarden"/>
95         <secondary component="strongScentWarningImage"/>
96         <secondary component="windEffect"/>

```

```
97         </relation>  
98     </scene>  
99 </body>  
100 </multisel>
```

Listing D.1: MultiSEL Document Example



## APPENDIX E - UEQ Results

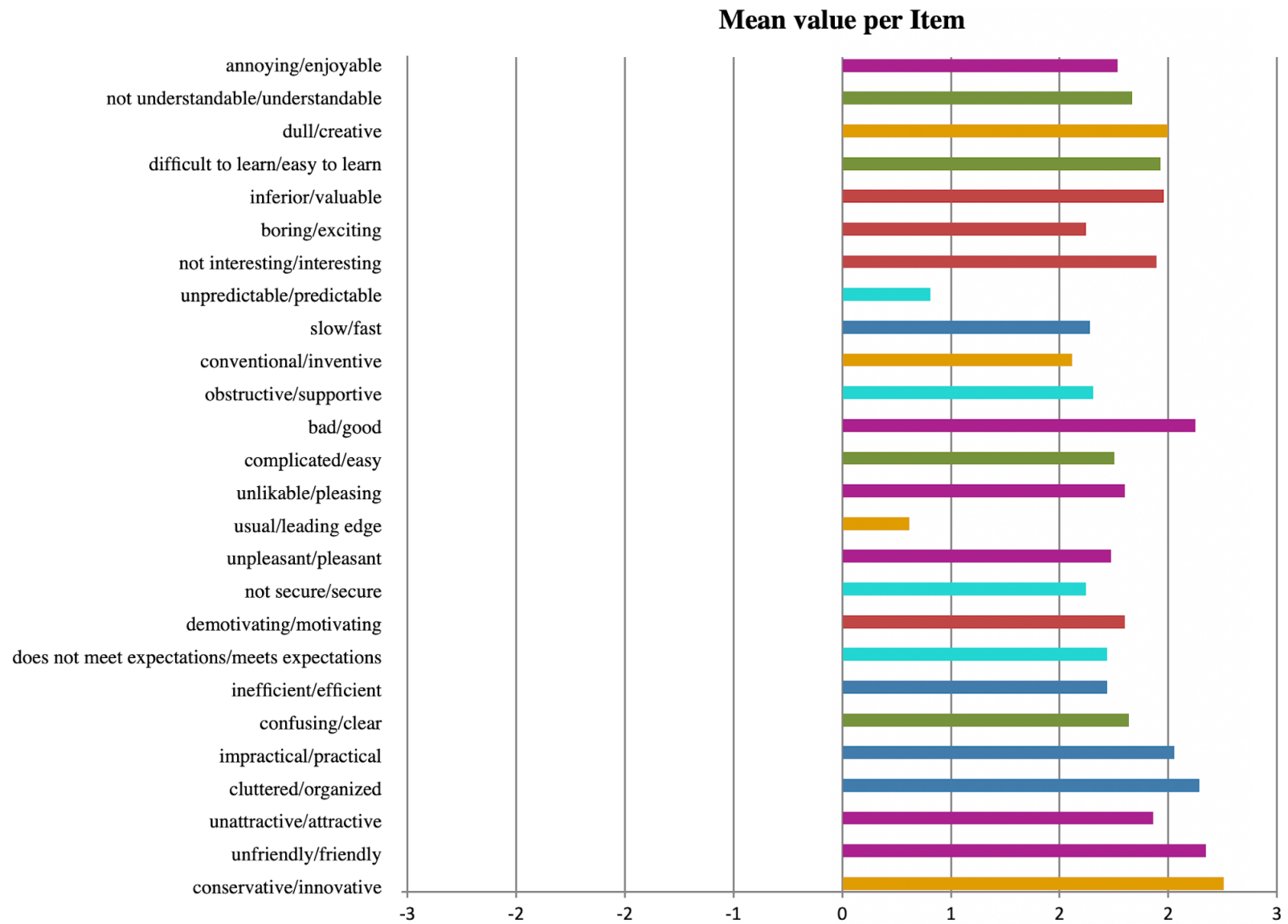


Figure E.1: Mean Value per UEQ Item

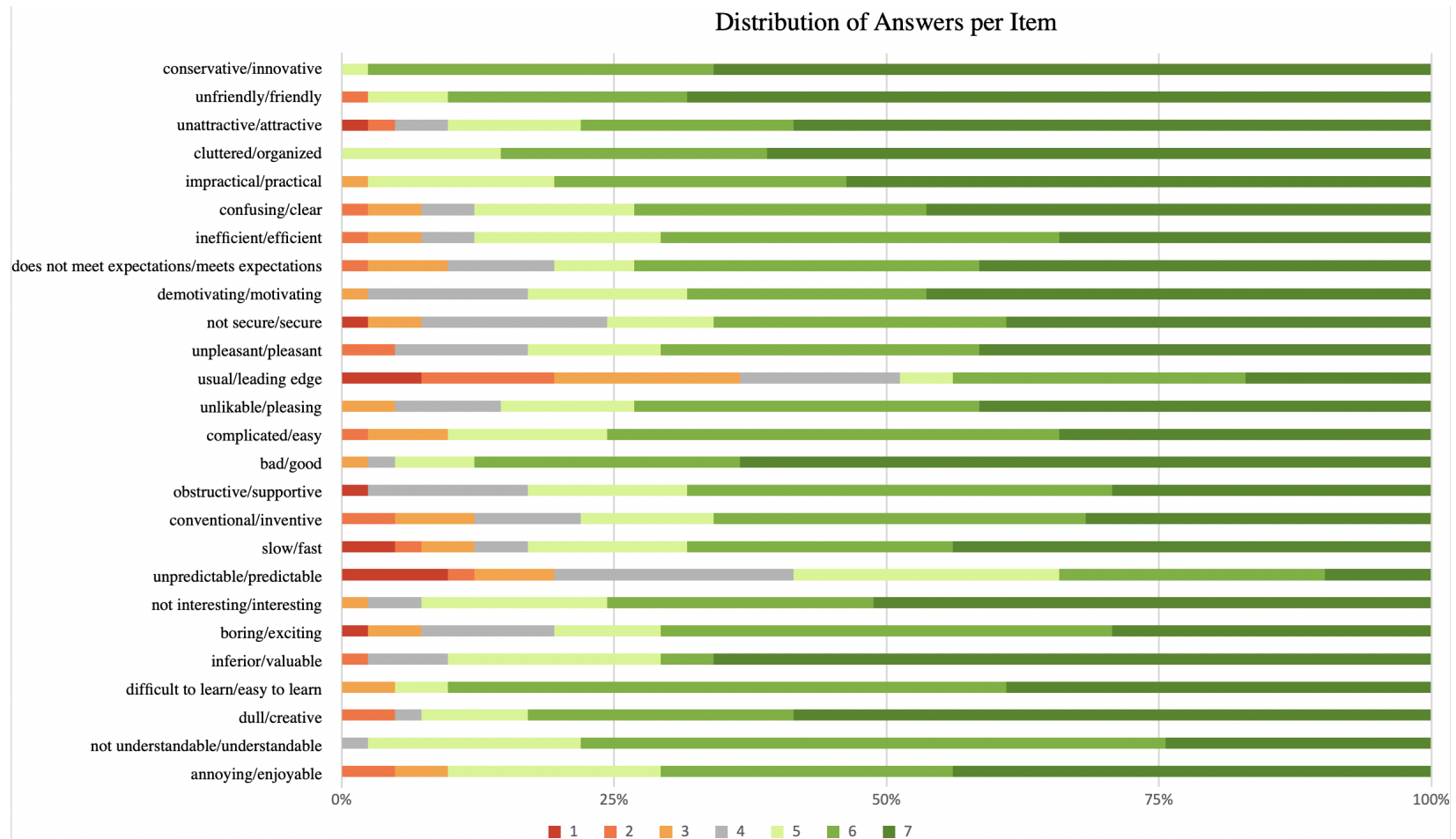


Figure E.2: Distribution of Answers per UEQ Item