Experience Requirements

A Thesis Submitted to the
College of Graduate Studies and Research
in Partial Fulfillment of the Requirements
for the degree of Doctor of Philosophy
in the Department of Computer Science
University of Saskatchewan
Saskatoon

By
David J. Callele

Permission to Use

Abstract

Video game development is a high-risk effort with low probability of success. The interactive nature of the resulting artifact increases production complexity, often doing so in ways that are unexpected. New methodologies are needed to address issues in this domain.

Video game development has two major phases: preproduction and production. During *preproduction*, the game designer and other members of the creative team create and capture a vision of the intended player experience in the game design document. The game design document tells the story and describes the game – it does not usually explicitly elaborate all of the details of the intended player experience, particularly with respect to how the player is intended to feel as the game progresses. Details of the intended experience tend to be communicated verbally, on an as-needed basis during iterations of the production effort.

During *production*, the software and media development teams attempt to realize the preproduction vision in a game artifact. However, the game design document is not traditionally intended to capture production-ready requirements, particularly for software development. As a result, there is a communications chasm between preproduction and production efforts that can lead to production issues such as excessive reliance on direct communication with the game designer, difficulty scoping project elements, and difficulty in determining reasonably accurate effort estimates.

We posit that defining and capturing the intended player experience in a manner that is influenced and informed by established requirements engineering principles and techniques will help cross the communications chasm between preproduction and production. The proposed experience requirements methodology is a novel contribution composed of:

1. a model for the elements that compose experience requirements,
2. a framework that provides guidance for expressing experience requirements, and
3. an exemplary process for the elicitation, capture, and negotiation of experience requirements.

Experience requirements capture the designer' s intent for the user experience; they represent user experience goals for the artifact and constraints upon the implementation and are not expected to be formal in the mathematical sense. Experience requirements are evolutionary in intent – they incrementally enhance and extend existing practices in a relatively lightweight manner using language and representations that are intended to be mutually acceptable to preproduction and to production.

Acknowledgements

I would like to take this opportunity to acknowledge the many people who have helped me to complete this work.

First and foremost I would like to thank Dr. Eric Neufeld for his strength and patience over the many years it took to complete this degree. I have learned far more than I ever expected from you and you will always have a special place in my heart. You dealt with three different thesis topics and never once did you complain. You always gave me good advice – in retrospect I wish I had taken more of it to heart earlier than I did.

I would like to thank Dr. Kevin Schneider for agreeing to be my co-supervisor once I had embraced topic number three. You helped me to understand the perspectives taken by software engineering academia, no matter how unusual they seemed to me on first introduction. Your guidance has been greatly appreciated.

My thanks to the remaining members of my final committee: Dr. Mark Keil, Dr. Chanchal Roy, Dr. Ron Bolton (Cognate), Dr. Samira Sadaoui (External), and Dr. Mark Eramian (Chair).

To the other members of my committees over the years, official and unofficial, my gratitude is yours.

Dedication

Completing this thesis was very challenging. I worked (at least) full time during the entirety of the thesis, often supplementing my daytime income with contract work in the evenings. I was a single parent during much of this time and all of the thesis work was done in what I could facetiously describe as my "spare time". I had three different thesis topics – I was beaten to publishing on the first topic, and I was not "at the right University" to get published on the second topic. However, "third time is the charm" and I have achieved success with this work.

I dedicate this thesis to my family and friends who stood by me through all of these challenges.

Most particularly, I dedicate this thesis to my first wife Michele, who passed away from cancer about five years before I started my studies, and to our daughter Yasmin, who has grown up watching her Dad working on homework every night, just like her.

I was blessed to fall in love a second time, and even though our marriage ended, I want to thank Mary for her support in facing many of life's challenges before starting the thesis and for her support during the early stages of the thesis.

Finally, I dedicate this thesis to those family members who passed away during the course of my studies and won't be able to be there to celebrate with me as I become the first in our family to receive their doctorate. I miss you still and I will remember you always...

Michele Moore	Feb. 20, 1961 – Feb. 12, 1995	Wife
Rose Lewis	Aug. 20, 1926 – Mar. 2, 1999	Aunt
Sue Andre	Oct. 4, 1917 – Dec. 4, 2003	Aunt
Josephine Hassen	Feb. 6, 1935 – June 25, 2004	Aunt
Mary Harris	May 31, 1919 – July 19, 2005	Aunt
Len Lewis	Oct. 11, 1916 – June 3, 2005	Uncle
Michael Wandzura	Nov. 3, 1930 – June 13, 2006	Cousin
Victor Held	June 14, 1935 – Feb. 7, 2008	Uncle
Raymond (Bob) Schwab	Sept. 25, 1951 – Feb. 29, 2008	Cousin
Lynn Callele	Sept. 30, 1960 – Apr. 18, 2008	Sister
John Callele	July 17, 1931 – Sept. 16, 2008	Father
Catherine Callele	Oct. 12, 1931 – Jan. 5, 2009	Aunt
Francis Niemeier	Oct. 14, 1947 – Jan. 15, 2009	Cousin
Charles Held	July 10, 1914 – Mar. 18, 2009	Cousin
Tony Andre	Feb. 2, 1925 – Mar. 4, 2010	Uncle

And to the many other family members who passed away before I started this work – you are always in our hearts, thoughts, and prayers.

With love to you all, and with tears in my eyes, thank you for all your love and support.

David Callele

Contents

Permission to Use **i**

Abstract **ii**

Acknowledgements **iv**

Contents **vi**

List of Tables **xi**

List of Figures **xii**

1 Introduction **1**
- 1.1 Problem Identification, Statement and Investigation: Chapter 2 2
- 1.2 Introducing Emotional Requirements: Chapters 3, 4 5
- 1.3 Security Requirements and Emotional Requirements: Chapters 5, 6 6
- 1.4 Extending the Experience Requirements Framework: Chapters 7, 8 8
- 1.5 Experience Requirements and Cognitive Gameplay Requirements: Chapters 9, 10 12
- 1.6 Implementation: Chapter 11 16
- 1.7 Guide to the Document 16

2 Requirements Engineering and the Creative Process in the Video Game Industry **17**
- 2.1 Introduction 18
- 2.2 Background 19
  - 2.2.1 Emotional Factors 20
  - 2.2.2 Language and Ontology 21
  - 2.2.3 Elicitation, Feedback and Emergence 21
- 2.3 Video Game Development 22
  - 2.3.1 Development Process 22
  - 2.3.2 The Game Design Document 23
- 2.4 The Transition from Preproduction to Production 24
- 2.5 Review of Postmortem Columns 25
- 2.6 Examples From Real Games 29
  - 2.6.1 Documentation Transformation 29
  - 2.6.2 Implication 30
  - 2.6.3 Evaluation 36
- 2.7 Summary and Conclusions 36
- 2.8 Challenges for Requirements Engineering 38
  - 2.8.1 Media and Technology 38
  - 2.8.2 The Importance of NFRs 39

2.8.3 Gameplay . . . 39
2.9 Future Work . . . 40
2.10 Acknowledgements . . . 40

3 Emotional Requirements in Video Games 41
3.1 Introduction . . . 42
3.2 Related Work . . . 43
3.3 Emotional Requirements . . . 43
3.3.1 Designer Intent . . . 44
3.3.2 Artistic Context . . . 45
3.4 Representing Emotional Requirements . . . 47
3.5 Summary and Future Work . . . 49

4 Emotional Requirements 50
4.1 Introduction . . . 50
4.2 Requirements Challenges . . . 51
4.3 Induced Emotional Requirements . . . 52
4.3.1 Representation . . . 52
4.3.2 Cultural Conditioning . . . 53
4.4 Contextual Information Requirements . . . 53
4.4.1 Positional . . . 53
4.4.2 Temporal . . . 54
4.4.3 Relational . . . 55
4.5 Conclusions . . . 55

5 Balancing Security Requirements and Emotional Requirements in Video Games 56
5.1 Introduction . . . 57
5.2 Related Work . . . 58
5.3 Evaluating Threats . . . 58
5.4 Resolving Requirement Conflicts . . . 59
5.5 Summary and Future Work . . . 60

6 Requirements in Conflict: Player *vs.* Designer *vs.* Cheater 61
6.1 Introduction . . . 62
6.2 Stakeholders . . . 63
6.2.1 Producers . . . 64
6.2.2 Consumers . . . 65
6.2.3 An Example . . . 65
6.2.4 The Exception to the Rule . . . 66
6.3 Related Work . . . 67
6.3.1 Security Requirements . . . 67
6.3.2 Misuse, Abuse, and Anti-Requirements . . . 67
6.3.3 Threats and Attacks . . . 68
6.3.4 Player Types . . . 68

6.3.5 Cheating . . . 68
6.3.6 Grief Play and Griefers . . . 69
6.3.7 Emotions in Requirements . . . 70
6.3.8 Negotiating Requirements . . . 70
6.4 Security and Video Games . . . 70
6.5 The Player's Perspective . . . 73
6.5.1 Alternative Play as Threat . . . 74
6.6 The Developer's Perspective . . . 75
6.7 A Process . . . 77
6.8 Resolving Requirement Conflicts . . . 77
6.9 Summary and Future Work . . . 79

7 Augmenting Emotional Requirements with Emotion Markers and Emotion Prototypes 81
7.1 Introduction . . . 82
7.2 Elicitation and Capture . . . 84
7.3 Specifying Emotional Requirements . . . 85
7.4 Conclusions . . . 86

8 Visualizing Emotional Requirements 87
8.1 Introduction . . . 88
8.2 Related Work . . . 90
8.2.1 Storyboards . . . 90
8.2.2 Film Studies and Cognitive Psychology . . . 90
8.3 Emotion Prototypes and Markers . . . 91
8.4 Scenario Concept . . . 92
8.5 Designing the Player Experience . . . 94
8.5.1 Emotions and Emoticons . . . 95
8.6 Generating the Emotional Intensity Map . . . 97
8.7 From Requirements to Design and Implementation . . . 100
8.7.1 Difficulty and Emotional State . . . 101
8.7.2 Constraints on Emotional Requirements . . . 102
8.8 The Final Product . . . 103
8.9 Elicitation and Capture . . . 104
8.10 Specifying Emotional Requirements . . . 106
8.11 Conclusions . . . 107
8.12 Future Work . . . 108

9 An Introduction To Experience Requirements 110
9.1 Introduction . . . 111
9.2 Experience Requirements . . . 111
9.3 A Model . . . 112
9.4 Potential Benefits . . . 113
9.5 Experience Requirements in Video Games . . . 114
9.6 Conclusions and Future Work . . . 115

10 A Proposal for Cognitive Gameplay Requirements 116
10.1 Introduction 117
10.2 Gameplay Requirements 119
10.3 Related Work 120
10.4 Cognitive Engagement in Games 122
10.5 Field Observations of Gameplay Design 123
10.5.1 General Design 124
10.5.2 Observations on the Review Meeting 126
10.5.3 Gameplay 1: Platform Maze 127
10.5.4 Gameplay 2: Sliding Walls 130
10.5.5 Gameplay 3: Shifting Sands 131
10.5.6 Summary 132
10.6 The Elements of Cognitive Gameplay Requirements 133
10.6.1 Preconditions 135
10.6.2 Cognitive Challenge 137
10.6.3 Post-Conditions 138
10.7 Conclusions 140
10.8 Future Work 140

11 Physualization: Going Beyond Paper Prototyping 142
11.1 Introduction 143
11.2 Physualization 143
11.3 Session Goals 144
11.4 Session Results 145
11.5 Resources and Tools 145
11.6 Questions for Consideration 146
11.7 Leading a Session 146
11.8 Some Examples 147
11.9 Other Work of Interest 150
11.9.1 Printed Materials 150
11.9.2 Physical Visualizations 151
11.9.3 YouTube Videos 151

12 Applying Experience Requirements and Conclusions 154
12.1 Windblown Adventure 154
12.2 First Person Shooter (FPS) 161
12.3 Negotiating Requirements at Runtime 162
12.3.1 An Exemplary Dispute Resolution Process 164
12.3.2 Player *vs.* Griefer 165
12.3.3 Player *vs.* Publisher 167
12.3.4 Summary 169
12.4 Conclusions 170

References 173

A Emotional Requirements Definition **182**

List of Tables

2.1 Documentation Transformation 29
6.1 Emotional Requirements 71
11.1 Requirements Tasks 144

List of Figures

2.1 Video game development 23
2.2 Observational Report Analysis 27
2.3 Correlation Within a Project 28
2.4 Akeladoor Puzzle Description, *used with permission* 32
2.5 Pyramid Puzzle Description, *used with permission* 34
2.6 Pyramid puzzle prototype, *used with permission* 35

3.1 First-Person Shooter (a) and Racing Game (b) designs, *used with permission* 44
3.2 Emotional intensity timeline 47
3.3 Representing emotional requirements 48

4.1 Far Vista Studios (www.farvistastudios.com) Run the Gauntlet in-game promotional scenes: The players environment might promote feelings of fear, relief, or elation. (used with permission) 52
4.2 An emotional timeline for Far Vista Studios Run the Gauntlet. (used with permission) 54

6.1 Generic Video Game Architecture 72

7.1 Sniper scenario emotional intensity map. Black fill denotes safety, white fill denotes danger, pattern fill indicates possible locations for emotion markers. Emoticons represent player emotion. 83

8.1 Sniper scenario pre-visualizations. The upper image is a plan view of the scene, the lower image is a minimum cost 3D rendering. 93
8.2 Top: A selection of typical emoticons from the EmotRG font. Bottom: Sketch artist cartoon face samples. Authors unknown. 96
8.3 Sniper scenario emotional intensity map. Black denotes safety, white denotes danger. 98
8.4 Color in emotional intensity maps 99
8.5 Sniper scenario emotional intensity map. Pattern fill indicates possible locations for emotion markers. 101
8.6 Gameplay difficulty in the emotional intensity map. White denotes danger, black denotes safety. From the top, an easy path, a difficult path, and a complex path. 102
8.7 Late prototype, plan view. Black arrow indicates Runner danger zone, white arrows indicate possible Sniper positions. 103
8.8 Three perspectives: (a)Isometric View, (b) Sniper View, (c) Runner View . . 104

10.1 Desert Puzzle concept sketches 125
10.2 Platform maze concept sketch 128
10.3 Platform maze design sketches 129

11.1 Starting to physualize 148
11.2 Simple gameplay scenarios 152
11.3 Other game requirements 153

12.1 Windblown Game Level Design 155
12.2 Windblown Game Design Review 155
12.3 WindBlown Adventure Emotional Journey 156
12.4 Windblown Design Review Details 157
12.5 Capturing the gameplay experience 158
12.6 Experience requirement production guidance 159
12.7 First-person shooter experience requirements 160
12.8 Emotional requirements 163

Chapter 1
Introduction

This manuscript thesis collects a set of publications that together investigate unique software production problems in the videogame industry then proposes experience requirements as a solution model, framework and methodology in response.

The addressed problem can be succinctly summarized. Modern videogames are a complex software artifact subject to significant technical restrictions. However, the appeal, and subsequent economic success, of the finished product is also dependent on the experience that it provides to its users. In practice, videogame development has two major but distinct phases: preproduction and production. Preproduction predominantly involves (so-called) creative personnel who brainstorm to capture a vision of the intended player experience in a game design document.

Production receives this document and media and software teams then attempt to realize the creative vision in a technological artifact - a piece of software. However, given the different perspectives of the members of the two teams and how such documents are presently written, many of the experiential features of the game are challenging for those charged with implementation. Generally, vagaries in the game design document are resolved verbally as needed. However, communications between preproduction and production breaks down, leading to production issues such as excessive reliance on direct communication with the game designer, difficulty scoping project elements, and difficulty in determining reasonably accurate effort estimates.

Collocating both teams and intertwining the phases may be a solution, but it may not always be practical, possible, or even desirable (for many reasons that are outside of the domain of this work). The present work argues that Experience Requirements (ER), a general term for a variety of methodologies introduced herein to help designers to better capture the intended player experience in the game design document, can greatly mitigate these problems. After an initial investigation of the problem, the early papers sketch a methodology that provides a model for the elements that compose experience requirements, creates a framework that provides guidance for expressing experience requirements, and demonstrates an exemplary process for the elicitation, capture, and negotiation of experience requirements. The subsequent papers refine and extend this methodology, demonstrate its viability, and point the direction to future deployment.

1.1 Problem Identification, Statement and Investigation: Chapter 2

Video games are a special type of multimedia entertainment application. Unlike movies, for example, they require active participation by the user, and this ability of the game to interact creates value for the player. It is difficult to formally draw boundaries between entertainment categories, or even play categories, but we assume a common understanding of the nature of this genre.

Games are developed by multi-disciplinary teams, the software engineering process in video game development is not clearly understood, and the development of reliable practices and processes for this domain is hindered by these complexities. The importance of nonfunctional requirements, such as entertaining the user, create unusual demands for the Requirements Engineering (RE) process, particularly if the team is relying upon RE methods from the productivity domain. Requirements like "fun" and "absorbing" are not well understood concepts in requirements engineering, compounding communication issues between game designers and software engineers. More practically, RE relies upon domain experts to resolve particular communications issues. Complicating matters is the breadth of knowledge required: game designers may not understand, for example, the limitations of artificial intelligence when designing non-player characters, while software engineers may not understand the creative vision or they may be too willing to compromise vision to ship the product.

Requirements engineering – the systematic analysis of requirements – within a community of common interest is difficult. Quantifying, communicating and capturing stakeholder wants and needs is not an exact science. Zave [113] classifies the problems addressed by requirements engineering, defining the domain, in part, as "...translation from informal observations of the real world to mathematical specification languages." Traditional requirements engineering techniques [51, 92] assume these communications issues can be overcome in a few iterations.

In game development, this is true in only a limited way. Many of the tasks (such as artwork or animation) do not have reasonable translations to a mathematical specification language and the problem may be further exacerbated by the diverse backgrounds of different participants in the videogame enterprise.

To quantify our understanding of this problem, we began with the *Postmortem* columns in Game Developer magazine [8], some of which are extracted in *POSTMORTEMS from Game Developer* [39], as a source of observational reports. The purpose of the Postmortem column is to help developers understand what succeeded and what failed in a particular venture. We analyzed 50 such reports, published between May 1999 and June 2004 to identify factors that led to success or failure. The analysis suggested that project management issues are the greatest contributors to success or failure, and that in the case of failure, many of these issues can be traced back to inadequate requirements engineering during the transition from preproduction to production – an observation that motivated this work. Our study [16] also identified the need to solve three problems:

1. Documentation must be transformed from its preproduction form to a form that can be used as a basis for production.

2. It must be possible to identify implicit information in preproduction documents and make it sufficiently explicit that production decisions can be made.

3. It is necessary to be able to apply domain knowledge to preproduction without hindering the creative process, particularly knowledge of constraints.

Three examples from real video games provided further evidence of the importance of properly managing the transition from preproduction to production. They illustrated the challenges associated with transforming preproduction documents to production documents, the importance of detecting implied information as early as possible, and the effects of applying a priori knowledge from the production domain to the transition from preproduction to production.

For example, the Pyramid Puzzle investigated in this work showed that when early versions of preproduction documentation are fed forward to the production team then the production team may be able to provide important feedback to the preproduction team. This communication cycle enabled earlier identification of emergent requirements and production constraints and may improve the reliability of the transition from preproduction to production. However, the introduction of production personnel into the preproduction process may have a negative effect on the creativity of the preproduction team, as the two groups may have different goals.

A game design document typically contains significant quantities of implicit information. In the study, we argued that requirements engineers can identify at least three kinds of implication: those implications that would be derived by nearly all readers, the implications that would be made by those with experience with the genre, and the implications that would be made by those with deep knowledge of implementation details of the target architecture.

A common language [52, 105], ontology [60, 105, 14], or vision [60] is often mentioned as the solution to communications issues between disparate stakeholders. We believe that such a formal process could provide a solution to manage the transition from preproduction to production. However, rigorous formalism may introduce problems. For instance, imagine axiomatizing in a formal language the deep concept of a common substance like water that can be gathered from the ocean into pails, and then poured back. Then imagine teaching the design and production teams to work within this structure. Our goal is also pragmatic – while we seek a formal shared communication system, much of it must be lightweight and easy to learn by stakeholders.

This work also introduced the technique of datamining *post mortem* reports to investigate systemic issues with industrial practices.

1.2 Introducing Emotional Requirements: Chapters 3, 4

The requirements process introduced at this stage of the research [17] was called *emotional requirements*. Requirements engineering for video games has to address a range of functional and nonfunctional requirements, including the player experience: the means by which the player's consciousness is cognitively engaged while simultaneously inducing emotional responses (which became the *cue* and *action* in later work (Section 1.4)) [21, 22]. In a later paper [24], we generalized this idea to experience requirements (Section 1.5) since not all experiences in a game are necessarily associated with emotions. We will refer to both emotional requirements and experience requirements with the acronym ER, since the meaning is clear by context.

A preliminary investigation began to identify the components of this kind of requirement. Emotional requirements were initially defined as a tuple: the emotion that the designer intended to induce in the player, and the means by which that emotion was induced. Support for spatial and temporal annotations were included in the notation.

This work used a first-person shooter game as a testbed. A relatively simple scenario within the game provided insight to the complexity of the general problem: Emotions are subjective, which makes identifying, specifying and representing them inherently difficult.

Within the context of the first-person shooter example, we investigated the representational power of emotional terrain maps, emotional intensity maps, and emotion timelines as visual mechanisms for capturing and expressing emotional requirements. For example, the designer may wish a player to feel apprehension on entry to a room and emotional terrain maps and emotional intensity maps lay out experiences and their properties on the two-dimensional play terrain. Emoticons can also be placed on the terrain to locate a particular emotional experience, and other annotations may describe associated sound and lighting artifacts that induce this feeling. The intensity of the experience is visualized by a gradient on the terrain. Some details may be left unspecified, providing creative opportunities for the production team. Finally, an emotional intensity timeline is a separate visualization that plots the intensity of a player's experience against time.

These contributions generated considerable interest within the requirements engineering community. The initial paper analyzing requirements engineering and the creative process in the video game industry resulted in an invited article in IEEE Software that introduced emotional requirements to a wider audience [19].

1.3 Security Requirements and Emotional Requirements: Chapters 5, 6

In the typical requirements engineering demarcation between Functional Requirements (FRs) and Non Functional Requirements (NFRs), emotional requirements are considered NFRs. Identifying and managing interactions between FRs and NFRs, and prioritizing within and between categories, are common tasks for the domain practitioner.

Stakeholders often have differing opinions on the relative importance of a requirement. For productivity applications, the set of stakeholders is typically dominated by the users of the application and their immediate management. The stakeholder domain for video games must be more diverse. The entertainment aspect of the product means that emotions are involved and that there may be interactions that are not necessarily logical. In productivity software, FRs tend to dominate NFRs. However, videogame NFRs such as *fun* and *entertainment* tend to dominate all other requirements in this application domain. In this work [18, 20], we investigate the effects of introducing a new (and potentially dominant) class of NFRs into the requirements process. In particular, we look at interactions between emotional requirements and security requirements in this domain.

Emotional requirements, as originally envisioned, were a constructively motivated, creative affordance, used to help deliver the intended emotional experience to a willing audience. All stakeholders were assumed to be similarly constructively motivated, desiring the best entertainment experience that they could achieve. However, further investigation and analysis suggested emotional requirements contain other challenges.

Not all players are the ‘good guys’. Not everyone plays fair. In fact, a significant number cheat and, even worse, some of them actively attempt to disrupt or destroy the game experience for other players. Certain kinds of gamers (cheaters) appear to enjoy a destructive experience – subverting the game experience of other players. If players are deriving emotional experiences by subverting the game, and ignoring the experiences intended by the game, the game may also be a commercial failure. We can model these destructive stakeholders as security threats of a very particular type: they are willing, but apparently hostile users. This model led to an exploration of the application of emotional requirements to this unusual stakeholder class.

Sometimes the negative play is a consequence of a bad playing experience. When the playing experience goes poorly then player emotions, attitudes, motivations, and actions change dramatically. The player finds that their emotional requirement for fun is not being met. For example, they may perceive that their efforts to play are being thwarted, they may feel betrayed by the game, or they may even feel threatened by other players. The player now views some element(s) of the game playing experience in an adversarial manner. The exceptions to this model are the *griefers*, players who participate in the game only for the purpose of interacting with other players in a negative manner.

Due consideration of security goals is costly and challenging. Furthermore, security requirements can also conflict with the emotional requirements of an immersive play experience. For example, authentication can be an intrusive operation. However, if a constructive player perceives that the game is prone to attack by destructive players, they may feel that there is sufficient justification for the authentication measures.

Moffet *et al* [74] state that it is not necessary to know the goals of the individual attackers when performing risk analysis, just what kind of attack they will mount. We look at the motivation (the why behind the threats, and security in general) and try to determine if there are emotional requirements that can be met that mitigate the risk factors. Unlike the general practice of attempting to resolve all conflicting requirements, emotional requirements may not be resolvable – all that may be achieved is a set of requirements that lead to a state of constant, small-scale skirmishes between constructive and destructive stakeholders.

The dominant security goal for most video games is ensuring the integrity of the playing experience. This goal is shared by all constructive stakeholders; Consalvo [30] and others [111, 1, 25] have shown that players need to trust the integrity of the game – the same rules must apply to all participants and their playing experience should never include attacks by other players unless they have agreed to that playing mode.

In practice, "ensuring the integrity of the playing experience" is excessively vague. More likely, the actual security requirements would be defined by a document of considerable size defining legal play, and, unlikely to capture every possible scenario. In the end, the problem may be best resolved by an economic model that justifies the cost of the security requirement by its corresponding emotional benefit.

This work further contributes to the domain by showing that emotional requirements can assist the development of security requirements by identifying the motivation behind security threats. The emotional irritants that motivate the attacks can be addressed proactively, potentially reducing the magnitude of the risk. Emotional requirements can also be used to help prioritize security requirements; strong emotional irritants that require low effort to overcome are the most likely attack vectors. Addressing the high-risk security requirements identified in this manner should be prioritized during development.

Failure to meet the player's emotional requirements can lead to market forces that override security requirements; details are provided in the published work. If the emotional requirement failures are as a result of cheating or other threats to the integrity of the game experience, we have suggested that in-game justice systems could allow the players to act as a self-correcting mechanism in the face of these security failures. The justice system places emotional requirement negotiation in the hands of the players, providing them with a framework wherein their own community values can develop.

1.4 Extending the Experience Requirements Framework: Chapters 7, 8

Investigating the relationship between security requirements and emotional requirements on an appropriate scale was not feasible, and subsequent research returned to the basic experience requirements formalism and extensions.

Field work done earlier with Far Vista Studios was reviewed, the participants revisited, and we found that the adoption of emotional requirements at the studio was lower than expected [21, 22]. While the combination of emotional requirements and emotional intensity maps were useful, the media production team did not find them sufficiently useful to adopt them because the emotional intensity maps did not indicate how the target emotion was to be induced or where the inducing elements were located.

We investigated film-making for potential solutions. Storyboards are a well-known technique that draw heavily from the comic strip genre as a way of sketching key events in a movie. We considered the possibility of annotating storyboard frames with both text and emotional intensity maps to better guide the production team. We also reviewed the film-studies works presented by Plantinga and Smith (*eds.*) [84], works that looked at film studies from a cognitive psychology perspective, where practitioners tend to "discuss emotion states in terms of goals, objects, characteristics, behaviors, judgments, and motivations." Smith further notes that the "concepts such as pleasure, and displeasure, and desire used in film studies are too broad to provide specific insight into how a particular film makes its emotional appeal at any given moment", motivating his work toward gaining the desired precision.

These perspectives have strong parallels with our work and we used Smith [100] as the exemplar for the application of cognitive psychology to film studies and, by extension, to our work. Much of Smith's work that is referenced herein is aimed at performing critical analysis of the emotions in film in a *post hoc* manner. One of our goals was to use the same or similar concepts *a priori*, in the requirements and design phase.

Smith posits that cognitivists believe that we recognize emotions by pattern matching against emotion prototypes [84, 100]. Emotion prototypes have three characteristics. They have an object orientation; the emotion is cued, or triggered, by an object or the action taken by an object. They demonstrate an action tendency; the emotion spurs us to take some action. Finally, they demonstrate a goal orientation; there is some purpose to the action that we take. Smith also identifies an emotion marker as something that will engender a brief burst of emotion but probably does not affect the narrative or underlying story. Emotion markers can take any form; they may be sounds, scenes, or even dialog. There may be more than one emotion marker in a given scene and it is expected that one or more of the emotion markers is the cue or trigger in the emotion prototype.

As a result, the ER formalism was extended to include Smith's emotion markers as triggers for intended emotions, and Smith's emotion prototypes to provide further guidance to a production team.

This resulted in extending the ER formalism to a (cue, action, goal) triple, and also to the identification of explicit locations for markers, and the development of techniques for eliciting, capturing and visualizing the requirements. For this work, we collaborated closely with industry team members, in the spirit of an action research approach.

Although games have less narrative than film, these ideas proved to inform useful refinements of experience requirements. The emotion marker generalizes our earlier use of emoticons by providing information we had previously identified as belong to the artistic context, better meeting the guidance needs of the production team. The emotion prototype and the related emotion marker do cause some issues with representation - the extra information is not part of the emoticon, it is not necessarily co-located with the emoticon, and does not appear to have a suitable, generalizable visualization. It may be that integration with the storyboard, as discussed earlier, is most appropriate.

As a first step, we began to standardize emotion-specific terminology on a per-project basis. Emoticons and text are both straightforward and compact ways to represent emotions on diagrams. We used Parrott's [83] classification as a starting point, but expect practitioners to adopt whatever classification meets their needs. Game designers were presented with this classification and asked to identify the intended emotions for a scenario. Not surprisingly, experience with the team showed that what Parrott calls basic and secondary emotions are relatively easy to work with, but the so-called tertiary emotions require significant context, and probably some experience, for proper interpretation. However, this constraint does not seem to undermine the promise of the (cue, action, goal) emotional prototype as a means of concisely and compactly transmitting information across the preproduction/production boundary. In fact, it confirms that this kind of information is complex, and difficult to communicate, and offers an argument in support of improving the communication of this kind of information across the boundary, even if only the primary and secondary aspects are confidently transmitted.

The final emotional intensity map (which ideally would accompany a (low resolution) rendering of the actual scene) uses intensities to represent emotional states at places in the terrain, and uses shading to indicate transitions between these emotional states. We encountered an interesting limitation. Grayscale maps only allow a pair of emotions to be represented in a single map. One solution is color, but training a team to associate colors with emotions, and, to then recognize intermediate colors as transitions between a specific pair of emotions is more problematic still. This is merely a manifestation of the curse of dimensionality [10] in our particular setting. Humans seem to be able to comfortably interpret only a relatively small number of dimensions of information from 2D images.

Another contribution of this work is that it is the boundaries between regions that are of greatest import for the media production process. It is at the boundaries that some form of emotion marker must be placed to act as a trigger to induce the desired emotion state. For example, safe zones exist at the boundaries of the emotions of fear and relief. Knowing this, we used a luminosity thresholding algorithm to identify possible locations for these emotion marker(s), providing necessary production guidance in a lightweight manner.

Ultimately, however, the sketches used for the layout of the virtual world during the requirements process were used more as inspiration to the art department than as hard requirements, and practitioners should be prepared to accept this. The final requirements specification demonstrated a variety of interesting cue mechanisms. For example, in the analyzed sniper-runner scenario, brightly lit windows are cues and clues to the runner – used to draw their attention to the source(s) of danger. These same cues simultaneously draw attention away from the barrels and boxes scattered about the street that promise a refuge, however brief, to the runner as they attempt to escape attack from the sniper positions. The game designer has deliberately created a conflict between the cues.

This work also developed an iterative elicitation process. First, a gameplay experience is defined in general terms using a text summary and a few sketches of the virtual world. Then, iterate as follows. Define the actions a player can take, what assets the player can utilize and the legal interactions of players. Provide enough artistic context that production can develop the media assets. Finally, define the emotional requirements with the techniques developed thus far, using the (cue, action, goal) emotional prototypes to help ensure that production understands the desired player experience.

1.5 Experience Requirements and Cognitive Gameplay Requirements: Chapters 9, 10

The formal aspect of this work culminates with the development of an ontology of experience requirements [24]. An initial investigation of cognitive gameplay requirements was performed [23] and a formalism for representing emotional requirements was provided (see Appendix A for further details).

In this work we explore the contribution that experience requirements can make to the domain of videogame development. Experience requirements are descriptions of user, player, and customer experiences that must be met (functional experiences) or are satisfaction goals (non-functional experiences) for products or services. Experience requirements may be constructed using generally accepted requirements engineering principles and techniques or they may use less traditional techniques such as concept art or sound effect samples. Experience requirements are not software requirements, although they may result in software requirements or may be met by software artifacts.

The following ontology of experience requirements for the videogame domain is based on the interactions between what the underlying game system can deliver as part of the experience and what the player can sense and internalize:

1. Emotional requirements (the heart)
2. Gameplay requirements (the intellect)
   (a) Cognitive (the head)
   (b) Mechanical (the hands)
3. Sensory requirements (the senses)
   (a) Visual (the eyes)
   (b) Auditory (the ears)
   (c) Haptic (if available) (touch)

These experience requirements are expected to be contextually situated within their domain. For example, while gameplay requirements are appropriate for the videogame domain they are probably not relevant to the movie domain. However, the screenplay or shooting script for a movie will likely contain elements that can be represented as emotional requirements, sensory requirements, and cognitive requirements and thus shares these aspects with videogames.

In the videogame domain, defining and capturing the intended player experience as experience requirements that are influenced and informed by established requirements engineering principles and techniques can help practitioners bridge the communications chasm between preproduction and production. Applying requirements engineering principles, in a manner tailored to the domain, should help the game design document assume more of the attributes of a software requirements specification, improving the communication between preproduction and production without negatively impacting the preproduction effort. Rather than defining a formal language with say, first-order, semantics that may limit the practical applicability of the methodology, we present an encapsulation of the experience requirements formalism that provides practitioners with a useful perspective and guide to action.

Maintaining a constructivist stance, we assume a stimulus-perception-response model guides the design of the user experience: First, the desired user response is specified. A stimulus, that is (to be) perceived by the user, is then designed to engender the desired response. The stimulus-perception-response model is a representation of the ways in which the designer can affect the user – informally, via the emotions, the intellect, and the senses. We find that, for each element of this model, there are tangible and intangible elements. As regards stimuli, tangible elements are physical objects in time, intangible elements can arise through the interaction of physical objects, sound and perhaps force feedback. As regards perception, it is straightforward that humans receive certain kinds of information through sensory receptors, but these also combine in ways that remain difficult to capture. Finally, the response of a player may be tangible (measurable) to the game in some ways, but intangible in others (experience).

There would be a clear benefit in being able to create a mechanistic definition of (for example) fear that would let us induce exactly a certain amount of fear, or precisely measure the degree of fear. However, the current state of the art in most theories of measurement suggests that we must settle for crude proxies - for example, facial expresssion as a measure of degree of fear. This does not prevent us from having meaningful discourse about such topics, however, and that is what we propose to do here. Thus, we produced an ontology of types of experiences for the video game domain that reflect what we believe to be shared understandings or common intuitions of these experiences. This ontology begins with emotional requirements (experiences of fear or joy), gameplay requirements (intellectual satisfaction obtained from solving puzzles, or from receiving a good cognitive experience), and sensory requirements (visual, auditory, and increasingly, tactile experiences of playing.)

There is a considerable literature on this referenced in our published work. The trade press associated with game design is rich with pragmatic advice. Game designers like Rollings and Adams [93], Crawford [31] and Koster [61] and academics like Salen and Zimmerman [95] present their perspectives on a field that is generally considered to be more of an art than a science. Each author presents their perspective on the act of game design, but none of the authors comments on software engineering processes that could support the activity. The anthology of project *post mortem* reports presented by Saltzzman [96] provides significant anecdotal evidence of the issues involved in video game production. The anthology of commentaries by well-known industry professionals compiled by Laramee [64] also provides further insight into video game production. Despite the breadth of these works, none of the authors advocates a structured approach to capturing gameplay as requirements or utilizing requirement engineering principles.

In general, the work in the requirements engineering research literature is only slightly related. A traditional perspective on requirements is likely to consider cognitive gameplay requirements to be some form of non-functional requirements. In his analysis of non-functional requirements, Glinz [49] notes that there are significant issues with defining, representing, and classifying non-functional requirements. He proposes a solution based on the concept of a concern, defined as "a matter of interest in a system". A concerns-based taxonomy is presented, along with a series of questions that can be applied by the practitioner to guide them in applying the taxonomy. It is unclear whether this taxonomy covers, for example, cognitive gameplay requirements or emotional requirements. While "matters of interest", whether they qualify as 'concerns' would depend upon whether one accepts cognitive gameplay as an appropriate target for requirements efforts.

The preproduction requirements that define the player experience are conceptually more like design requirements, as discussed in the collected works of "Design Requirements Engineering: A Ten-Year Perspective" [67].

Finally, our own works made a modest, but practically situated, contribution to this body of knowledge. In another informal field study with Far Vista Studios, we participated in a preproduction effort response to a third-party request for proposal for a Massively Multiplayer Online Role Playing Game (MMORPG) situated in ancient Egypt. The proposed gameplay scenario had the following requirements. The scenario must require the player to solve one or more puzzles. The game is located in a desert. The gameplay must support individual and team play, and the player navigates using a click-on-destination paradigm.

The included publication [23] reports considerable detail on the design process, both on the concept and on the details surrounding traps, puzzles, and penalties of gameplay components. However, the experience confirmed again the problems of capturing sufficient information during preproduction, the practical technical problems (for example, rendering load) that must be addressed and the complexity of the experience issues (for example, planning for, and measuring, the enjoyability of repeat play).

These field observations helped us to identify elements that are necessary to elicit, capture, and represent during the requirements specification activity for cognitive gameplay requirements. To facilitate the expression and discussion of the elements of cognitive gameplay requirements, these are presented as a definition. Consistent with the exploratory nature of this work, this definition is not formal in a semantic sense, nor is it complete. Confirming the observations with other teams is necessary before the results can be generalized.

Summarizing the proposed conceptual framework, we defined cognitive gameplay requirements as challenges, bracketed by pre- and post-conditions. The preconditions consist of assets, clues, infrastructure, player state and puzzle state, possibly annotated by a gameplay context and possibly a link to a narrative. These are tangible objects that can be tracked in straightforward fashion.

A cognitive challenge is cast as a learning exercise. If the pre-conditions are satisfied, the challenge is described with text or symbolic description, typically using flow charts or finite state machines. Solving the problem generates side effects - mostly changes to the player, puzzle, and world state. The post-conditions include the updated states mentioned above, and changes to player's knowledge, and changes to the game engine's knowledge of the player.

1.6 Implementation: Chapter 11

The ideas presented herein are particularly challenging to validate in any short term study. If they are adopted, requirements engineers will likely find new problems during deployment. These problems may then be addressed by both industry experts, who tend to be situated in and have a deep experience with a fixed work culture, and academics, who have the opportunity to study a variety of experiences and share some of them, but who may not be able to experience long hours in production.

Nevertheless, experience requirements appear to be a useful starting point for further work in the field. In the style of paper prototyping, we demonstrated a process for expressing experience requirements using ordinary office materials at an interactive session [15], where the requirements for a side scrolling game racing game were demonstrated. In our Future Work (Chapter 12), we illustrate the potential of the formalism for more complex games.

1.7 Guide to the Document

This introductory chapter provides an overview of the work from the inception of a new idea, its growth in the course of exploratory field work with a small game development firm to a conceptual framework, and a proof of concept. Details appear in the publications that follow. The final chapter presents conclusions and illustrates the future potential of this work in other settings.

Chapter 2
Requirements Engineering and the Creative Process in the Video Game Industry

To quantify our understanding of this problem, we began with the *Postmortem* columns in Game Developer magazine [8], some of which are extracted in *POSTMORTEMS from Game Developer* [39], as a source of observational reports. We analyzed 50 such reports, published between May 1999 and June 2004 to identify factors that led to success or failure. The analysis suggested that project management issues are the greatest contributors to success or failure, and that in the case of failure, many of these issues can be traced back to inadequate requirements engineering during the transition from preproduction to production. Our study also identified the need to solve three problems:

1. Documentation must be transformed from its preproduction form to a form that can be used as a basis for production.

2. It must be possible to identify implicit information in preproduction documents and make it sufficiently explicit that production decisions can be made.

3. It is necessary to be able to apply domain knowledge to preproduction without hindering the creative process, particularly knowledge of constraints.

Three examples from real video games provided further evidence of the importance of properly managing the transition from preproduction to production. They illustrated the challenges associated with transforming preproduction documents to production documents, the importance of detecting implied information as early as possible, and the effects of applying a priori knowledge from the production domain to the transition from preproduction to production.

The observations within this work motivated the rest of the dissertation. Originally published as an extra-length paper at the request of the Program Committee.

David Callele, Eric Neufeld, and Kevin Schneider. Requirements Engineering and the Creative Process in the Video Game Industry. In RE 05: Proceedings of the 13th IEEE International Conference on Requirements Engineering (RE 2005), pages 240 250, Paris, France, 2005. IEEE Computer Society [16].

Abstract: The software engineering process in video game development is not clearly understood, hindering the development of reliable practices and processes for this field. An investigation of factors leading to success or failure in video game development suggests that many failures can be traced to problems with the transition from preproduction to production. Three examples, drawn from real video games, illustrate specific problems: 1) how to transform documentation from its preproduction form to a form that can be used as a basis for production, 2) how to identify implied information in preproduction documents, and 3) how to apply domain knowledge without hindering the creative process. We identify 3 levels of implication and show that there is a strong correlation between experience and the ability to identify issues at each level.

The accumulated evidence clearly identifies the need to extend traditional requirements engineering techniques to support the creative process in video game development.

Keywords: Non-functional requirements, elicitation, video game development, game design document, preproduction, production, domain-specific terminology.

2.1 Introduction

Video games are a special type of multimedia application – an entertainment product that requires active participation by the user. Developed by a multi-disciplinary team, non-functional requirements such as entertaining the user create special demands on the requirements engineering process. Requirements like *fun* and *absorbing* are not well understood from the perspective of requirements engineering, compounding communication issues between game designers and software engineers. Game designers may not understand, for example, the limitations of artificial intelligence when designing non-player characters while software engineers may not understand the creative vision or they may be too willing to compromise that vision in the rush to ship the product.

It may be that nothing can qualitatively change this. However, it should be possible to decrease the cost of delays caused by communication errors in such a heterogeneous group. As a first step toward the development of a formal process, we have attempted to locate the causes of the most costly errors. By way of background, we first review the requirements engineering literature applied to multimedia development and introduce the video game industry and the video game development process, with attention to the roles of preproduction and the game design document (as a deliverable artifact of the preproduction process). We analyze the observational reports from the Postmortem column in Game Developer magazine, categorize the information therein, and present the results. Three examples, drawn from real video games, illustrate particular issues that must be addressed in a formal process. We follow with our conclusions, an analysis of the role of requirements engineering in video game development, and directions for future work.

2.2 Background

Requirements engineering within a community of common interest is difficult – the ability to precisely communicate and capture stakeholder wants and needs is rare. Traditional requirements engineering techniques [51, 92] assume these communications issues can be overcome in a few iterations. However, we are unaware of any work that directly addresses the validity of this assumption in a multi-disciplinary development effort. While goal [5, 34] and scenario [54] based techniques can be used to alleviate communications issues, their efficacy when development efforts include a strong artistic or inventive element [91] (such as in video game design, multimedia web sites, or the movie industry) remains unproven.

Members of video game development teams include practitioners from such diverse backgrounds as art, music, graphics, human factors, psychology, computer science, and engineering. Individuals who, in other circumstances, would be unlikely to interact with each other on a professional basis unite in their economic goal of creating a commercially successful product. Requirements engineering in the face of such diversity requires the creation of a common (domain) language (and implied world model) specific to the task at hand. Once all stakeholders fully commit to the domain language, then a set of requirements that captures the stakeholders wants and needs can be generated.

Given the dearth of directly related work, we performed a more extensive literature review, focussing on: (1) requirements engineering and emotional factors (including fun in games), (2) issues of language and the creation of a common language or domain ontology, and (3) requirements elicitation and the effects of feedback on emergent requirements, particularly in multimedia development.

2.2.1 Emotional Factors

While emotion in human-computer interaction is coming under ever increasing scrutiny [68], few researchers have investigated emotional factors in requirements engineering. Draper [36] looked at fun as a candidate software requirement, attempting to identify what it is that makes play *fun*. He concluded that "fun is not a property of software, but a relationship between the software and the users goals at that moment" and that "providing enjoyment is now a defining requirement of an important class of software, and this has not been sufficiently recognized in our analyses and design methods". These conclusions are consistent with our experience.

Hassenzahl *et al.* [55] introduced *hedonic* qualities (those that are unrelated to the current task but present for emotional reasons) and associated repertory grid techniques for measuring them. Bentley *et al.* [11] investigated emotional (affective) factors in computer games, noting that "software requirements for these and other affective factors are never truly captured in an official manner". In particular, usability, immersion, and motivation were considered via a user survey mechanism. They note that there are no established techniques for eliciting emotional requirements. Even Chung, in his detailed analysis of non-functional requirements [27], does not substantively address emotional issues.

At their best, video games stimulate a state of *flow* in the player, engendering concentration so intense that their perception of time and sense of self become distorted or forgotten [33]. In the field of game design, Salen and Zimmerman [95], Laramee [64], and Saltzzman

[96] address issues of emotions and emotional response in game players. While these works do not directly address requirements engineering practices, the techniques that they describe for game design and eliciting feedback from players may increase the range of elicitation techniques available to practitioners. In a more general sense, Norman [80] describes numerous human factors practices that could be readily incorporated into requirements engineering for video games.

2.2.2 Language and Ontology

Zave [113] classifies the problems addressed by requirements engineering, defining the domain, in part, as "...translation from informal observations of the real world to mathematical specification languages." In game development, this is only partially true. In many cases, the game designer, an individual who may have little or no interest in a mathematical representation, is also tasked with generating the requirements. Unable to generate the requirements in isolation, the game designer works with the production team to translate the vision to requirements – usually stated in natural language complete with domain specific terminology. Once captured, the requirements may be formalized in place or, more likely, formalized as they are translated into specifications.

A common language [52, 105], ontology [60, 105, 14], or vision [60] is often mentioned as the solution to communications issues between disparate stakeholders. Natt och Dag *et al.* [81] have demonstrated the application of statistical natural language processing techniques to managing and understanding requirements generated by a multitude of sources. Their results may be applicable to the documentation transition issues studied further in Section 2.6.1.

2.2.3 Elicitation, Feedback and Emergence

Goguen [52] emphasizes that "feedback and feedforward go on all the time, at least in successful large projects" and that "requirements are emergent". Emergent requirements discovered during the transition from preproduction to production are a significant aspect of the creative design process.

Zave [113] presents a classification scheme that assumes that "... as software engineers, we can seek to understand social factors but we can only hope to influence technical practices." We posit that requirements engineering can be more proactive in video game development by providing feedback from production to preproduction in response to a feedforward of early versions of preproduction documentation. The resultant influence on the creative process escapes Zave's technical practices restriction. Specific feedforward and feedback examples appear in Section 2.6.2.

2.3 Video Game Development

Video games are a significant element of the entertainment industry. The Consumer Electronics Association [7] reports that entertainment software sales rose from \$5.1 billion in 1999 to \$7.7 billion in 2003 and that hardware sales increased from \$2.3 billion in 1999 to \$3.2 billion in 2003. Combined hardware and software sales in the video game industry exceed the 2003 \$9.42 billion gate receipts of theatrical release movies in North America [46].

However, for every advertisement for a newly released game, the trade press reports a disproportionately large number of projects that fail to reach the market. The present work begins an investigation into the causes of these failures. The multidisciplinary nature of the video game development process – with art, sound, gameplay, control systems, human factors (and many others) interacting with traditional software development creates complexities that may recommend a specialized software engineering methodology for this domain.

2.3.1 Development Process

Figure 2.1 models the game development process as two consecutive efforts. The left hand side of the diagram depicts the preproduction phase, resulting in a Game Design Document (GDD). Preproduction loosely corresponds to a customer's internal efforts to define their wants and needs before meeting with the development team.

The right hand side of the diagram, derived from Medvidovich and Rosenblum [71], depicts the production phase. Requirements engineering, with the assistance of the game designer(s), transforms the GDD to a specification (see Section 2.3.2). Once the specification

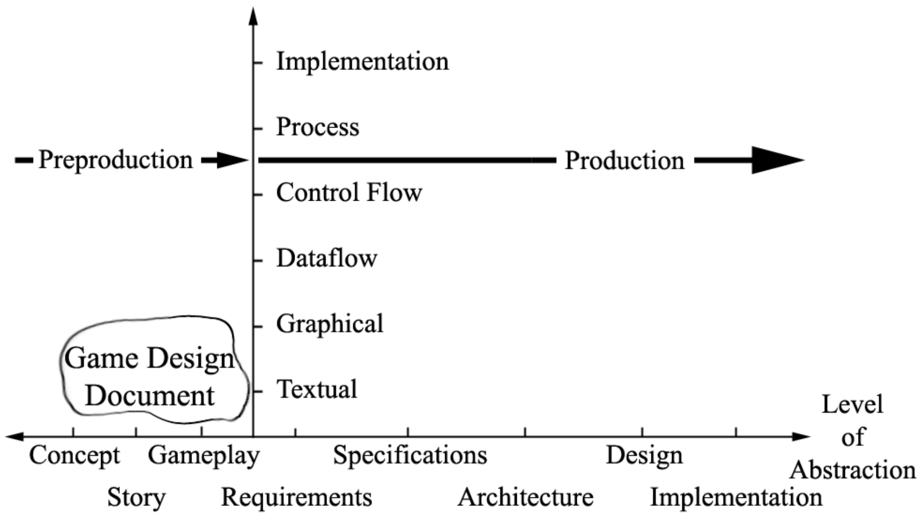


Figure 2.1: Video game development

is complete, a traditional software development process begins (often using an iterative development effort of some form), resulting in the game artifact.

Moving from preproduction to production is particularly difficult in video game development. A wide range of factors (*e.g.* artistic, emotive, and immersive factors) must be addressed by the requirements engineering effort. These factors are captured in the game design document.

2.3.2 The Game Design Document

The game design document is a creative work written by the game designer (or game design team). The GDD must be thorough, but not necessarily formal (in the sense of structure or from a mathematical perspective). In fact, one could argue that imposing too much structure on the creative process may be highly detrimental – constraining expression, reducing creativity, and impairing the intangibles that create an enjoyable experience for the customer. In a sense, the GDD is the requirements document as defined by the preproduction team.

The form of the game design document varies widely across genres and studios. Typically, a GDD (drawing loosely from Bethke [12]) includes a concept statement and tagline, the genre of the game, the story behind the game, the characters within the game, and the character dialogue. It will also include descriptions of how the game is played, the look, feel, and sound of the game, the levels or missions, the cutscenes (short animated movie clips), puzzles, animations, special effects, and other elements as required.

A game design document is a preproduction artifact designed to capture a creative vision. It is not designed to meet the needs of a production effort. If a GDD is being used as a source document in the production phase, there are two possible explanations. The game design document may contain the information required for the production phase. In this case, the game design document is malformed and should be restructured and maintained as independent preproduction and production documents. Or, it may be that, even though the game design document does not contain production information, the production team is performing requirements engineering, specification, and possibly even design, on an *ad hoc* basis. The greatest danger associated with such *ad hoc* activities is the dependence on human memory for capturing decisions and their justifications.

There are issues associated with managing the game design document to requirements document transition. Two sets of documentation must be created and maintained. The writing styles associated with the two sets of documentation are very different – is it reasonable to expect that a single individual can perform both tasks in an efficient and acceptable manner, particularly in the absence of generally accepted practices for performing this translation? In general, we found little evidence of structured application of generally accepted requirements engineering principles in our review of observational reports on industrial practices (Section 2.5).

2.4 The Transition from Preproduction to Production

Requirements errors are some of the most costly to fix; Boehm and Basili [13] estimate that errors of this type can cost up to 100 times more to fix after delivery than if caught at the start of the project. Despite the available evidence and accumulated experience, many projects still suffer from failures due to inadequate requirements engineering.

Game designer and producer Eric Bethke [12] states

> …too many projects violate their preproduction phases and move straight

> to production. ...In my opinion, preproduction is the most important stage of the project. I would like to see the day when a project spends a full 25 to 40% of its overall prerelease time in preproduction. During production there should to be relatively few surprises. [p.26]

He promotes the use of UML based tools as a way to manage the transition but a formal (or semi-formal) transition process is not presented. Many of the requisite elements for production management (such as requirements capture, requirement analysis, task analysis, time estimation, project plans and technical design) are discussed in an informal manner.

Other producers and consultants, such as Rollings [94] and Michael [73], also identify many of the requisite elements for production management but do not provide formal or semi-formal guidelines for managing the transition.

When discussing game design documents, Bethke [12] comments "...I have never seen a completed design document, and one of the reasons is that game design documents need to be maintained through the course of production." With time-to-market pressures so prevalent, it is easy to see how documentation maintenance is given low priority.

Despite the recognized need, we have discovered no evidence that a process for managing the transition from preproduction to production has been proposed (recognizing that such a process may exist within an organization but remain unreported in the literature).

2.5 Review of Postmortem Columns

The video game industry is competitive and management processes are significant corporate assets and generally inaccessible to the researcher. Therefore, we use the *Postmortems* columns in Game Developer magazine [8], some of which are extracted in *POSTMORTEMS from Game Developer* [39], as a source of observational reports on this issue.

From the author's guide provided by the publisher:

> ...Explain what 5 goals, features or aspects of the project went off without a hitch or better than planned. ...Explain what 5 goals, features or aspects of the project were problematic or failed completely. ...Important: try to come up with things that went right/wrong during project that are likely unique to your project. Stay away from common and well understood problems and solutions (*e.g.*, "communication between the team members wasn't good" – that's been true of most games), and focus on what made your project different from others.

The reports presented in the Postmortem column potentially capture what makes video game development unique. They are typically attributed to members of the project management team or middle to upper management within the development organization. As such, one can reasonably assume that the reports reflect issues of particular import to the authors. While there may be an observer effect, particularly with respect to those items that went wrong, we assume the information presented has a strong basis in fact.

Fifty postmortem reports [8], published between May 1999 and June 2004, were analyzed in an attempt to identify factors that lead to success or failure in video game development. Each report contained 5 entries in the "what went right" and "what went wrong" sections. These entries were reviewed and classified according to the following scheme[1].

The classifications scheme has five categories: (1) *preproduction*, issues outside of the traditional software development process such as inadequate game design or inadequate storyboarding, (2) *internal*, issues related to project management and personnel, (3) *external*, issues outside of the control of the development team such as changes in the marketplace and financial conditions, (4) *technology*, issues related to the creation or adoption of new technologies, and (5) *schedule*, issues related to time estimates and overruns. Schedule issues are a subset of internal issues, but were uniquely identified in an effort to determine if scheduling was a significant issue. Any pair of the five categories was also possible (*e.g.*, "internal and technology") if the entry was that precise.

Figure 2.2 is a normalized representation of the results of the categorization process; of the 15 possible categories (singles and pairs), only those categories that represent 10% or more of the final result are shown. Internal factors dominate any other category by a factor of approximately 300%.

Closer inspection of points classified as internal or schedule factors reveals that many, if not most, of the entries are related to classic project management issues. For example, PM4W5[2] notes "inadequate planning", PM20W3 claims a lack of success due to "underestimating the scope of tasks" PM9W3 calls their schedule "too aggressive", PM18W2 states

[1]In an attempt to reduce possible bias, entries were reviewed with minimal identifying information and categorization of the "what went right" entries was performed independently of the "what went wrong" entries.

[2]Project coding: **PM**[Project number **1..50**][**R**ight | **W**rong][Entry **1..5**]

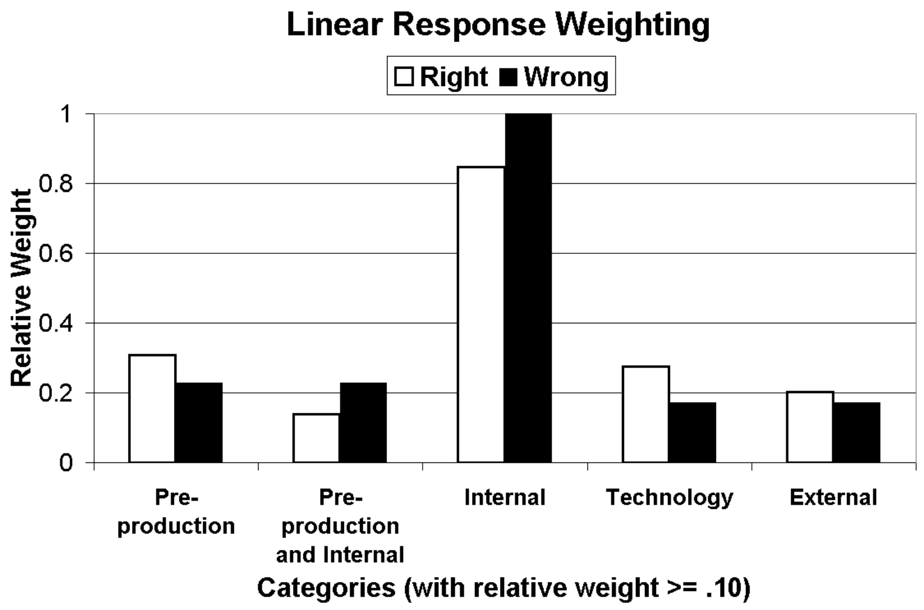


Figure 2.2: Observational Report Analysis

that "clear goals are great - when they are realistic" and PM21W1 states that "an unrealistic schedule can't be saved without pain". It appears that these issues could be addressed by a RE process that better manages the transition from preproduction to production.

Of interest is the balance in the categorization results. Across all categories, *across all projects*, the maximum deviation from the mean is only 7.7% – a category was perceived as likely to contribute to the success of a project as it was to the failure of the project. The high degree of correlation between the "what went right" and "what went wrong" entries could be a result of the granularity of the categorization scheme – approximately 60% of all entries are categorized within the (major) *internal* category or related minor categories.

In general, the management of different aspects of the production process was often listed both as an element that went right and an element that went wrong within a given project. For example, in the internal category, PM3 considered their ability to focus on the task at hand as a success, while stating that "inadequate planning" caused significant issues. PM21 felt that experienced personnel and internal communication contributed to success, yet stated that their "Conventions should have been better documented, communicated, and adhered to." PM27 had "strong quality assurance" yet asserted that they were weak when documenting their internal standards and processes, claiming that the "Design document (was) not implemented effectively". These apparent contradictions (such as strong QA but weak internal standards) are a common theme in the observational reports.

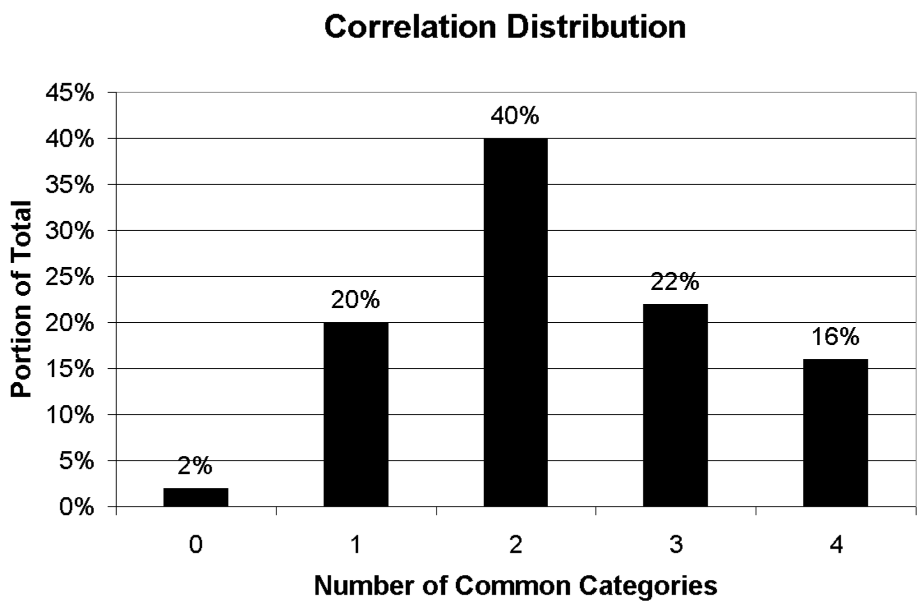


Figure 2.3: Correlation Within a Project

The degree of correlation between the "what went right" and "what went wrong" entries *within a given project* is also significant. We assumed that the order in which the entries were presented was irrelevant and then cross-checked the results of the categorization process to see if the same categories were being reported as success and as failures. It appears (Figure 2.3) that individual categories are just as likely to be viewed, *within a given project*, as a contributor to success as to failure.

In an effort to determine whether these strong correlations are related to the categorization process or are inherent within the data, we are currently performing a more detailed analysis of these reports. The current analysis has identified particular challenges for requirements engineering in this domain, presented for discussion in Section 2.8.

For the interested reader, the postmortem columns provide further details. Domain specific successes included: PM11R1 "maintained . . . style of gameplay", PM27R2 has " gameplay driven design", PM28R1 "created *deep* characters", PM40R1 focused on "great

Table 2.1: Documentation Transformation

1	Story	After her father, Bernard, died, Crystal did not know which way to turn – paralyzed by her loss until the fateful day when his Will was read.
2	Gameplay	The Player must visit Anna the Lawyer to receive a copy of Bernard's Last Will and Testament, thereby obtaining the information necessary to progress to the next goal.
3	Requirements	The Player must be represented by an avatar. Female Non Player Character required: Anna the Lawyer Inventory Item: Last Will and Testament (LWT) Player can not progress beyond Game State XYZ until LWT added to Inventory
4	Specifications	Could easily reach 50 pages

art". Examples of issues in preproduction included: PM2W1 "(there was a) lack of up-front design", PM6W2 "(the) game was too hard", and PM28W3 "(too much) gee-whiz factor".

2.6 Examples From Real Games

The initial results from our analysis of the Postmortem columns led us to conclude that weak management of the transition from preproduction to production was a source of many issues in video game development. We now look at some examples from real games that have either been published or are currently in development[3] to establish further support for this conclusion. We look at 3 issues in particular: documentation transformation, implication creating emergent requirements, and the effects of *a priori* knowledge, to situate them within the domain and within the larger realm of requirements engineering.

2.6.1 Documentation Transformation

A microcosm of the documentation transformation issue is shown in Table 2.1. The game designer begins (1) with a story written in a narrative style. That story is then translated (elsewhere in the game design document) to a more formal form (2) that describes the

[3]In the first example, minor changes have been made to the material to obfuscate the source.

action as a task and a justification for that task. The requirements engineer analyzes this information, in context (3), to determine a set of requirements: identifying in-game assets such as the player avatar, Anna (a Non-Player Character (NPC)) and an inventory item. A state that controls the player's progress through the game is also identified and captured. Depending on the in-house process used, the detailed description (4) of these in-game assets may be part of the requirements document or part of a specification document. Independent of where the detailed descriptions are located, they could easily reach 50 pages once issues like artistic style, animation, and game state are included.

Performing and managing this transformation is complex. Each of these documents requires a different writing style and a single individual may not have the requisite writing skills to author materials for all purposes. In addition, creating the requirements document or specification document often requires considerable *a priori* knowledge of the available technology so that the requirements can be presented in context. There is also a multiplicative effect: each successive document is larger than the prior document as the author(s) attempt to precisely capture the required information. The authors must manage multiple stakeholder viewpoints, synthesizing a common domain language, numerous nonfunctional requirements, and inconsistencies as the project evolves.

The list of required skills is long (*e.g.* game design, requirements engineering, and technical communications) and implies a team effort. The associated costs are significant, leading to a strong management bias toward minimizing the documentation effort.

2.6.2 Implication

By its nature as a creative work, a game design document is replete with implied information. *Identifying* these implications requires careful analysis, *understanding* the ramifications of the implications requires significant domain knowledge.

To expand on the importance of domain knowledge, we revisit Table 2.1. This table captures what we call first-level implications: those implications that can be derived directly from the materials presented. Almost all development teams, independent of their experience levels, capture these implications. Missing implications at this level is usually an oversight on the part of the team.

The second level of implication requires general knowledge of the domain – in this case, the adventure game genre. These implications are generally captured by teams with members who have experience with non-trivial software development projects in the domain. In this case, the description contains significant implications regarding the game world: the characters must be situated within the appropriate environment(s). Therefore, there is an environment surrounding the player when they receive the information, there is Anna's office, perhaps an office building with other office interiors, background sounds, and possibly even other NPCs in the office areas. And, if there are other NPCs, do the NPCs interact with the Player?

These second level implications could easily amount to many person-months of development effort by modelers, artists, animators, and other members of the production team.

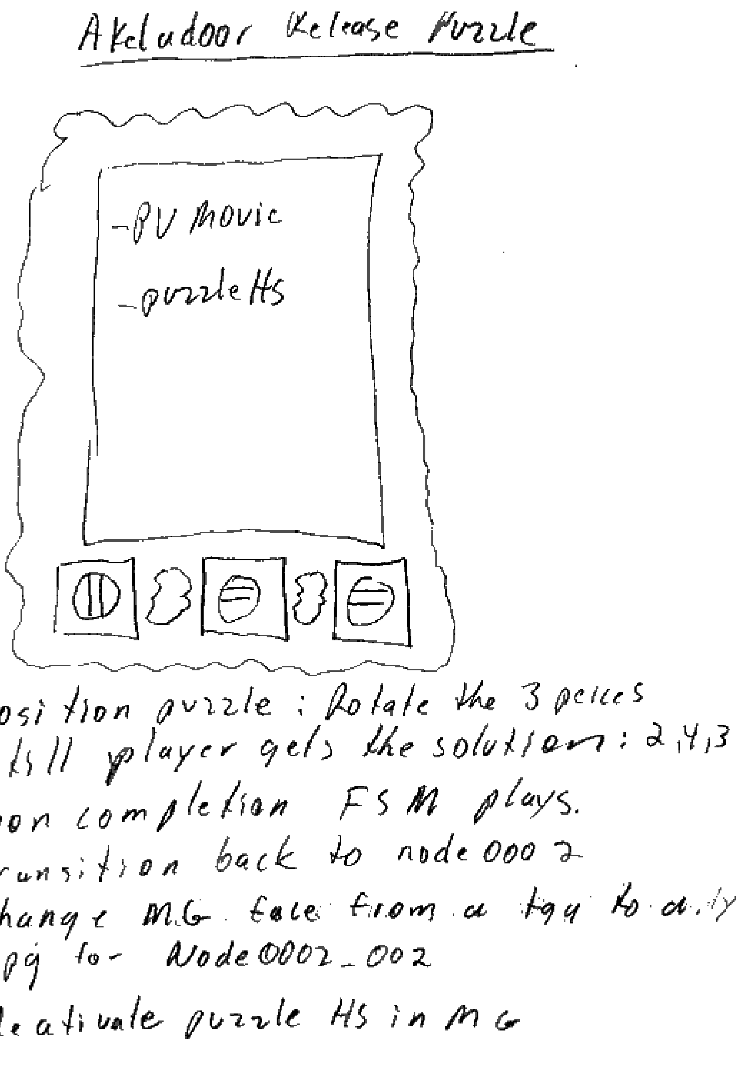
The third level of implication requires knowledge of implementation details such as the target architecture. These implications are captured by experienced teams, particularly when the present project is a sequel of some form. The requirement for the player to visit Anna raises questions about the connectivity between the elements (locales) of the virtual world – is there more than one way the Player can get to Anna the Lawyer? How does the player experience the journey – via a scene change? Or, must they guide their avatar through the virtual world (implying the creation of all the media assets to represent the world)?

Perhaps more importantly, does the connectivity change over time? Dynamic connectivity has significant implications for representing game state (the current state of the world simulation). Designing, verifying, and maintaining a *stateful* world is more complex than a *stateless* world.

A question is raised by identifying these three levels of implication: Is it more appropriate to follow a traditional iterative process and allow these issues to surface later, or should this feedback be applied as early as possible in the process? Intuitively, early feedback is better. However, early feedback could have a negative effect on the creative process: if the game design team feels that the production team is going to reject their proposals then they may become conditioned to be less creative. The effects of early production feedback on the preproduction process merits further investigation.

A Priori Knowledge

Building on the analysis of the prior section, we now look more closely at the effect of *a priori* knowledge on the requirements engineering process.



Akeladoor Release Puzzle

-PV movie

-puzzle HS

4 position puzzle : Rotate the 3 peices untill player gets the solution : 2,4,3

-Upon completion FSM plays.

-Transition back to node 0002

-Change MG file from a tga to a.jpg for Node0002_002

-deactivate puzzle HS in MG

Figure 2.4: Akeladoor Puzzle Description, *used with permission*

Domain specific terms, particularly abbreviations and acronyms, are common in working papers. Figure 2.4 is the game designer's description of the *Akeladoor Release Puzzle* from the game *Apocalypse Spell,* currently under development by Far Vista Studios. Upon inspection, we see *PV Movie:* Partial Video, a less than full screen video clip, *puzzle HS:* a puzzle Hot Spot, an interaction point for the player, *FSM:* Finite State Machine, *MG:* Master Guidelines (the game uses a model driven architecture whose repository is called the *Master Guidelines* by the team).

If one attempts to formalize this document, they must understand large portions of both the preproduction and production realms. In a typical studio, this implies senior personnel from the preproduction or production staff but they are usually "too busy" to perform the task. Documentation is often assigned to a junior staff member with the rationalization that this task will "bring them up to speed".

Another alternative is to add professional technical writing resources to the projects. However, there is often a perception that it takes more time to explain it to the technical writer than it does to just write it oneself. Once this excuse is in place, no writer is hired, and soon, little or no documentation is maintained.

Significant elements of the game design documentation are informal, often with substantial visual content. Visual content is particularly difficult to represent in a formal manner: iterations are often sketched as shown in the *Pyramid Puzzle* description of Figure 2.5. Careful examination of Figure 2.5 reveals evidence of prior iterations that were simply erased. Maintaining an iteration history of sketches, such as this working paper, is challenging. An electronic form of the working paper may have captured the revisions, but probably would not have captured the justifications for making the changes – often an important piece of information later in the development cycle. These justifications could lead to evolutionary changes in the game engine, perhaps even to a product family architecture.

A detailed explanation of the puzzle is beyond the scope of this paper – suffice it to say that it is a combinational puzzle that requires the player to generate the correct sequence of symbols on the screens below the pyramid, one sequence for each corner of the base of the pyramid. However, application of domain knowledge during the requirements capture phase led to significant changes in the design of the puzzle.

The first issue was puzzle complexity. Solution hints were provided in the form of inventory items that looked like papyrus scrolls but there was no way for the player to show the scroll and the puzzle at the same time – the game engine simply did not support simultaneous operation of inventory inspection and puzzle modes.

The game designer was informed of this restriction and it was suggested that a place be

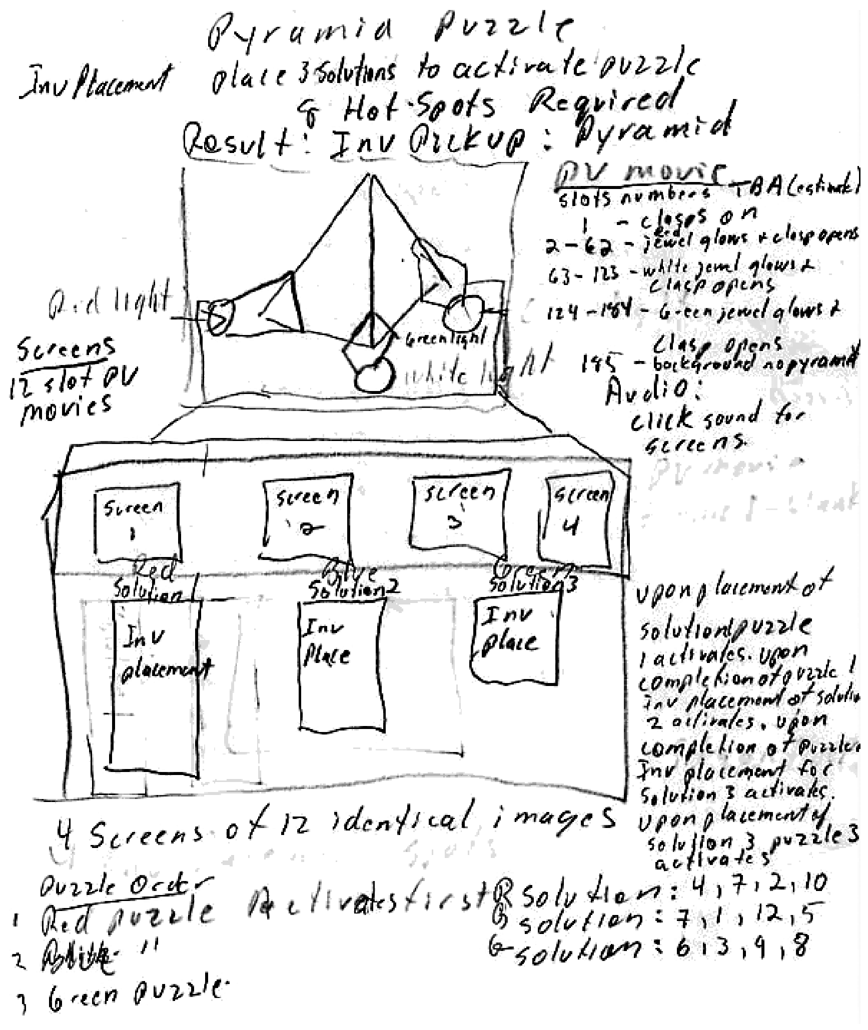

Pyramid Puzzle
Inv Placement
place 3 Solutions to activate puzzle
4 Hot-Spots Required
Result: Inv Pickup: Pyramid
PV movie
slots numbers TBA (estimate)
1 - clasps on
2-62 - Red jewel glows & clasp opens
63-123 - white jewel glows & clasp opens
124-184 - Green jewel glows & clasp opens
185 - background no pyramid
Audio:
click sound for screens
Red light
Green light
white light
Screens 12 slot PV movies
Screen 1
Screen 2
Screen 3
Screen 4
Red Solution 1
Blue Solution 2
Green Solution 3
Inv placement
Inv Place
Inv place
upon placement of solution puzzle 1 activates. upon completion of puzzle 1 Inv placement of solution 2 activates. upon completion of puzzle Inv placement for Solution 3 activates. upon placement of solution 3 puzzle 3 activates
4 Screens of 12 identical images
Puzzle Order
1 Red puzzle Activates first
2 ~~White~~ Blue "
3 Green puzzle
R solution: 4, 7, 2, 10
B solution: 7, 1, 12, 5
G solution: 6, 3, 9, 8

Figure 2.5: Pyramid Puzzle Description, *used with permission*

made on the puzzle for the player to "hang" the scrolls so that they could see them while playing the puzzle. The result of this feedback was the layout of Figure 2.5 where the scrolls for each corner had a specific location (shown as *Inv Placement Blocks*).

This new layout raised an issue of screen resolution. The puzzle design called for an upper region for special effects, a middle region for puzzle input, and a lower region for puzzle solution hints. Unfortunately, this layout was beyond the resolution of the target platform so an alternative layout was required.

The final layout, shown in Figure 2.6, is a compromise between the game designer's vision, the technical capabilities of the game engine, and the technology constraints of the target platform. Only one hints scroll is visible at a time, requiring the player to shift between inventory and puzzle modes for each corner of the pyramid – not an ideal solution from a human factors perspective, but the best that could be achieved within the constraints.

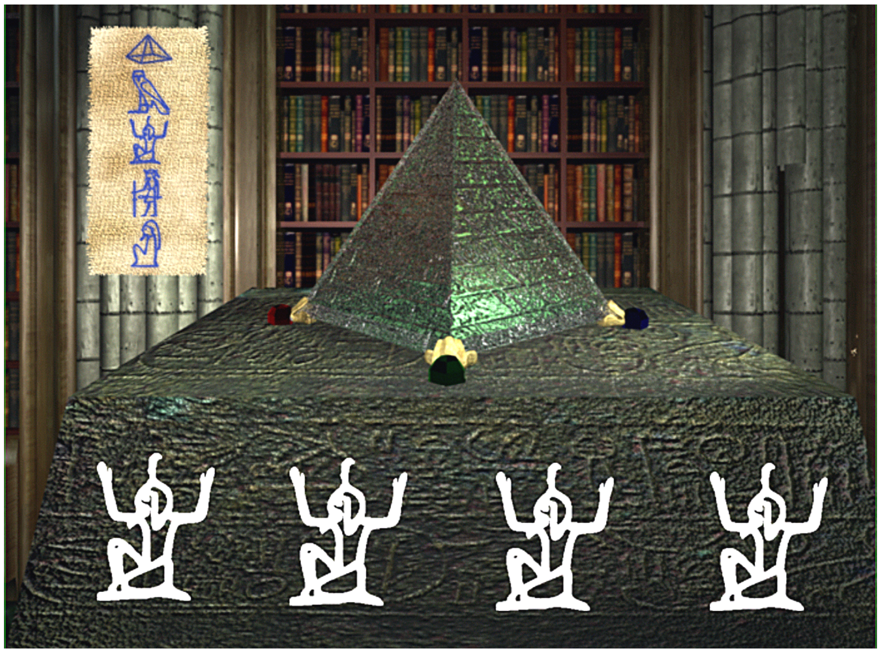


Figure 2.6: Pyramid puzzle prototype, *used with permission*

In this example, success was achieved through dialog between team members. Unfortunately, the revised requirements and specifications for the final product were never formally captured. Given that this is one of approximately 100 puzzles in the game, the cost of formal capture for all puzzles is significant.

The single sheet description of the puzzle resulted in the creation of the following assets: four new inventory items, 12 secondary screen elements for user interaction, three animation sequences of four seconds duration, and sound effects for user interaction and animation support. On the software side, four state machines for validating user input and three state machines for the individual corner puzzles were required. Interactions with the game world state, the current player state, inventory management, and the save game subsystem also had to be managed. None of these assets were explicitly identified by the designer; rather, they were implied in the description of the puzzle. It can be argued that identifying these implied assets if a function of the design process. However, accurately predicting the magnitude of the production effort requires their identification at the earliest possible stage in the process.

Given that this was just was one of approximately 100 puzzles in the game, it is highly desirable that the process for identifying the implied assets and side-effects be efficient. However, we are unaware of any work in this area.

2.6.3 Evaluation

The challenges associated with the Pyramid Puzzle are typical of the issues reported in the Postmortem columns. Using the same categories as Section 2.5, the terse puzzle description (assuming significant domain knowledge) is a *preproduction* issue. The puzzle description called for features that the underlying *technology* could not deliver. The technology constraints of the target platform (as defined by *external* market forces) caused a number of game design iterations. *Internal* issues, such as design complexity, design iteration, and emergent requirements interfering with test plan development, made it difficult to predict a *schedule* for this task with reasonable accuracy. The interactions are non-trivial and, when coupled with the complexities of media production, bring a unique flavor to requirements engineering in this domain.

2.7 Summary and Conclusions

We have analyzed the video game development process from the perspective of requirements engineering, presented a model for video game development that integrates preproduction with production, and situated the game design document as an artifact of the preproduction process. Our analysis of 50 observational reports from the Postmortem column in Game Developer magazine showed that project management issues are the greatest contributors to success or failure in video game development. In the case of failure, many of these issues can be traced back to inadequate requirements engineering during the transition from preproduction to production.

Three examples from real video games provide further evidence of the importance of properly managing the transition from preproduction to production. These examples illustrate the challenges associated with transforming preproduction documents to production documents, the importance of detecting implied information as early as possible, and the effects of applying *a priori* knowledge from the production domain to the transition from preproduction to production.

The Pyramid Puzzle example showed that, if early versions of preproduction documentation are fed forward to the production team then the production team can provide important feedback to the preproduction team. This communication cycle enables earlier identification of emergent requirements and production constraints and may improve the reliability of the transition from preproduction to production. However, the introduction of production personnel into the preproduction process may have a negative effect on the creativity of the preproduction team.

We show that requirements engineering practitioners can identify at least three levels of implication: (1) those implications that can be derived directly from the materials presented, (2) those implications that can only be derived with the introduction of general knowledge of the domain, and (3) those implications that can only be derived with the introduction of implementation details such as the target architecture. There is a strong relationship between experience and the ability to identify issues at each level of implication – indicating that a formal process for identifying implied information would not necessarily enable individuals with lesser experience to handle higher levels of implication without further guidance.

We postulate that the exploratory nature of attempts to capture the game design vision and the consequent number of production iterations is due to a lack of formal process for managing the preproduction to production transition. As project complexity increases, we predict that studios will shift to more formal processes to increase the probability of success in their development efforts despite internal resistance to this formalization.

We conclude that creating documentation to support the transition from game design document through formal requirements and specifications is difficult, requiring significant preproduction and production domain knowledge to perform successfully. A formal process to support this transition would likely increase the reliability of the process.

2.8 Challenges for Requirements Engineering

Is requirements engineering for video games unique? Analysis of the postmortem documents and game examples reveals that the video game industry could learn a great deal from current research and practice in requirements engineering and project management. Issues particularly notable due to their significance to game development success, and their relevance to requirements engineering, include: (1) communication between stakeholders of disparate background, (2) remaining focused on the goal and resisting feature creep, (3) influence of prior work (*e.g.*, building a new game on top of an existing game), (4) media and technology interaction and integration, (5) the importance of non-functional requirements, and (6) gameplay requirements.

Communication, focus, and prior work issues are relatively common in requirements engineering. Media and technology interaction, and the dominance of NFRs are also experienced (to a lesser extent) in other multimedia development efforts. Gameplay requirements are unique to video games.

2.8.1 Media and Technology

Creating a video game requires the creation of numerous software artifacts. Not only must the game engine be developed but a media production pipeline is also required. The pipeline must be designed and the tools associated with the pipeline must be built while keeping in mind that these are tools for artists and animators as well as for technical personnel.

Technology requirements often emerge as media assets are integrated into the game engine. The actual player experience delivered by the game engine may not meet the requirements of the game designer and publishers. Minimum platform targets may change due to technological advances and marketplace pressures - it is not uncommon to have to rework media assets developed early in a project to make them appear less dated by the end of the project.

Requirements engineering for media production in video game development is particularly challenging due to the interactions between the requirements of the video game artifact, the requirements of the tools needed to create the video game artifact, and the strongly differentiated user groups.

2.8.2 The Importance of NFRs

Video games are designed to entertain. Therefore, non-functional requirements such as *fun*, storyline, continuity, aesthetics, and flow must dominate their requirements specification. However, there are no established practices for capturing and specifying such NFRs – requirements engineering can make a significant contribution in this area.

Validation of gaming NFRs is very complex. Generally, an abstract NFR like *fun* is highly dependent on the target market - something that is fun for a young child may be annoying to an adult. The link between NFRs and target markets or user demographics has not yet been explored by RE in this domain.

Verification of gaming NFRs, and functional requirements related to media assets, is also complex. Requirement verification via test is particularly difficult when the requirement is to engender emotions in the user.

2.8.3 Gameplay

It is usually through gameplay that NFRs like *fun* and *flow* are achieved. One can argue that it is the NFRs and gameplay that make each video game unique. For example, the dominant video game genre is the first person shooter, made famous by the *Doom* and *Quake* series from id Software. All first person shooter video games share a common set of core technologies required by that genre: protagonist avatar(s), antagonist(s), the ability to move the protagonist avatar within the virtual world in an acceptably realistic manner, and the ability for the protagonist avatar to choose and use a weapon to wreak mayhem upon the antagonists. It is the presentation of these core technologies to the user (via gameplay, storyline, and aesthetic elements such as art and sound) that makes each game unique.

Storyboards in video game development are more closely related to storyboards in animated movie production (evaluating aesthetics and storyline) than the typical user-interaction scenario development in productivity application software development. Storyboards are also used by some developers as a first step in prototyping gameplay – a means for assessing the player experience.

Prototyping gameplay is particularly challenging. It is difficult to assess the player experience early in the development cycle for significant progress must be made on building the underlying game engine infrastructure before gameplay testing can begin. This is a particularly high-risk scenario due to the likelihood that new requirements will emerge as gameplay testing continues, new requirements that must be tracked, and for which test plans must be developed. The emerging requirements may even force significant changes to the fundamental architecture of the system that, in extreme cases, may cause project failure.

2.9 Future Work

We are currently performing a more detailed analysis of observational reports from Game Developer magazine and other sources. We expect this information to further guide the development of a process for managing the transition between preproduction and production. Mechanisms for capturing and stating non-functional requirements, such as *fun*, in a manner that can be validated, measured, and verified are also required.

Involving production personnel in the preproduction process may lead to more efficient development or it may lead to reduced creativity. Further investigation is needed to quantify the tradeoffs.

2.10 Acknowledgements

We wish to thank Game Developer magazine for continuing the *Postmortem* column over the years. The first author thanks Electronic Arts for the opportunity to present an early version of this (and related) work at their British Columbia studios and for the valuable feedback he received. We also thank Richard Buckley of Far Vista Studios for access to internal game design documentation prior to release of their game.

Chapter 3
Emotional Requirements in Video Games

This paper introduced the novel concept of *emotional requirements* to the field of requirements engineering. Requirements engineering practitioners were challenged by the work, asked to consider non-functional requirements from far outside their traditional domain. This work begins suggest that requirements engineering techniques could be more-widely applicable than previously considered, pushing them into realms such as human factors and industrial design.

Originally published as follows.

David Callele, Eric Neufeld, and Kevin Schneider. Emotional Requirements in Video Games. In Proceedings of the 14th IEEE International Requirements Engineering Conference (RE 2006), pages 292 295, Minneapolis, MN, USA, 2006. IEEE Computer Society [17].

Abstract: Requirements engineering for video games must address a wide range of functional and non-functional requirements. Video game designers are most concerned with capturing and representing the player experience: the means by which the player's consciousness is cognitively engaged while simultaneously inducing emotional responses. We show that emotional requirements can be expressed in two parts: as the emotional intent of the designer and the means by which the designer expects to induce the target emotional state. Spatial and temporal qualifiers on intent and means may also be required.

We introduce emotional terrain maps, emotional intensity maps, and emotion timelines as visual mechanisms for capturing and expressing emotional requirements. Using a first-person shooter example, we show that these mechanisms can express the desired emotional requirements while providing support for spatial and temporal qualifiers.

Keywords: Non-functional requirements, emotion, emotional requirements, video game.

Schneider, Emotional Requirements in Video Games, Proceedings of the 14th IEEE International Conference on Requirements Engineering, September 2006.

3.1 Introduction

In prior work [16] we showed that the most probable source of failures in video game development is in the transition from pre-production (capturing the game designer's vision in a game design document) to production (implementation). The game design document is often mis-used as a system design document even though it is principally a set of requirements that the software artifacts and media assets must meet in order to capture the game designer's vision.

Video games are unique among software artifacts in the breadth of their requirements. The software implementation of a video game has a set of functional requirements. The "game" within the video game also has a set of functional requirements governing the rules that create the player's cognitive engagement. Finally, there is a set of non-functional requirements for the emotions that are to be induced in the player at each stage of the game or by each game element.

Emotions and emotional impact are not generally considered critical to the implementation of a software artifact. Their highly subjective nature resists quantification; yet, from the player's perspective, the induced emotional state is the most important deliverable requirement. Any functionality that exists, exists only to further that emotional goal: player acceptance, and market success, is driven by the realization of this set of non-functional requirements.

This paper investigates the application of requirements engineering techniques to emotions in video game design. We characterize emotional requirements and, using examples, we investigate issues associated with representing emotional requirements. We identify mechanisms for expressing emotional requirements in context and conclude with a summary and directions for future work.

3.2 Related Work

There has been relatively little work on emotion in the requirements engineering literature [11, 36, 55]. The fields of HCI [68], industrial design [80] and emotionally intelligent software agents [9] have pursued emotion more aggressively.

Representing or specifying non-functional requirements is challenging. Ekman's Facial Action Coding System (FACS) [40] is exemplary of a visual approach to representing emotion. The Cognitive Affinity project at the University of Birmingham [41] presents an ontological and architectural alternative.

3.3 Emotional Requirements

The subjective nature of emotions makes identifying, specifying, and representing them inherently difficult. Providing appropriate support for emotional requirements means we must understand what the game designer needs to capture and represent. Figures 3.1(a) and 3.1(b) are game design working drawings illustrative of the design process used at Far Vista Studios[1]. Figure 3.1(a) is a floor plan view of a portion of a virtual world. An internal room, in the lower left, contains two enemy characters, a directional sound source, and a lighting source that varies with time. A window to the outside world (labeled EXT) provides spatial orientation clues and ambient light for the scene. Figure 3.1(b) is a fragment of an aerial view of a racing track.

Note the high degree of abstraction in both diagrams: significant domain knowledge is assumed. In fact, both diagrams are more detailed than those usually used – many of the annotations were only added by the game designer in response to our questions.

To be useful to the game designer, we assert that an emotional requirement must capture:

1. the intent of the designer: *I want the player to feel apprehensive as they approach the entry to this room.*

2. the means by which the designer expects to induce the target emotional state, the

[1]These diagrams have been redrawn for publication

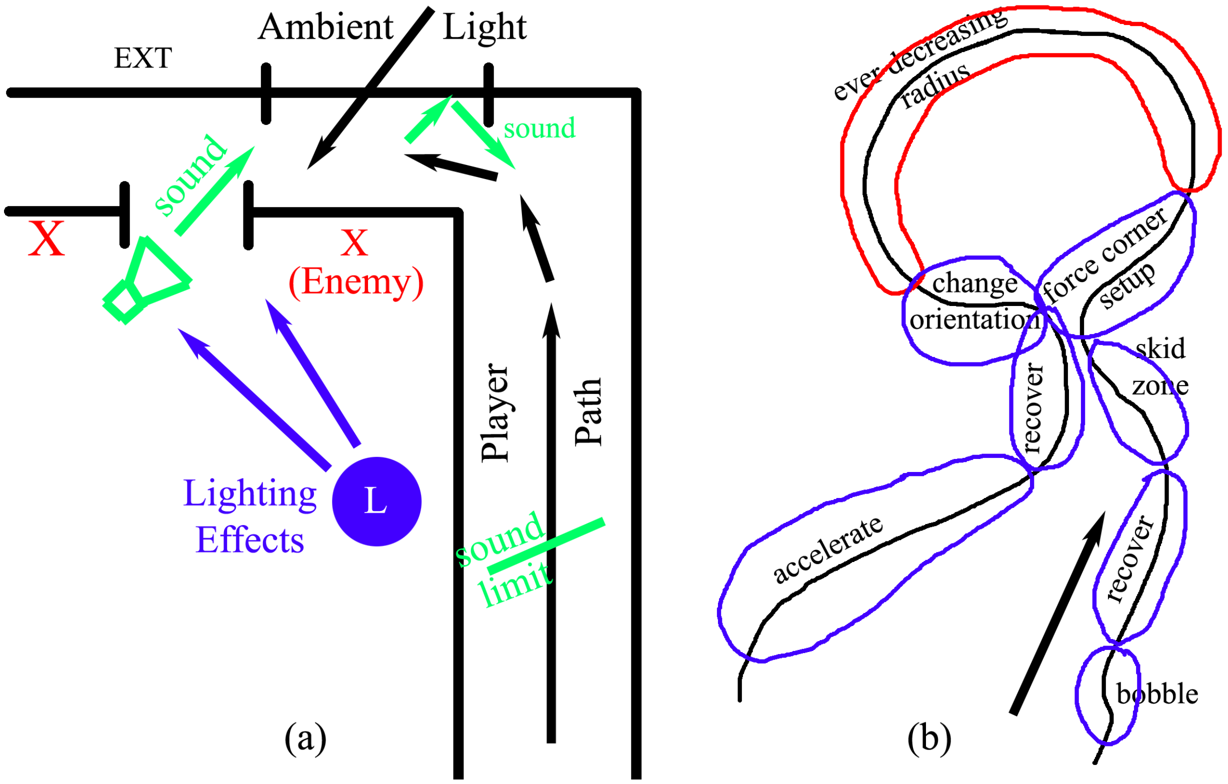


Figure 3.1: First-Person Shooter (a) and Racing Game (b) designs, *used with permission*

artistic context: *The player will feel apprehensive because the lighting is very dim and throbbing slowly. A soft, but deep and menacing sound fades in and out as the player nears the entrance to the room.*

3.3.1 Designer Intent

The designer's intent expresses a target emotional state to be induced in the player; this is the primary goal, the reason for the existence of this scenario. The intent may also express a (physical) location in virtual reality (a spatial qualifier) and/or a temporal qualifier of some form. In the example of Figure 3.1(a), the target emotional state is apprehension, with a spatial qualifier of "the entry to the specified room"; no temporal qualifier is provided.

Temporal qualifiers could be provided: *If the player reaches the entry to this room within 5 minutes of the start of gameplay, the player is obviously very skilled so increase the intensity.* However, this is still not a traditional software requirement. We can, however, restructure the combined statement of the designer's intent in a more traditional manner: *If the player reaches the entry to this room within 5 minutes of starting the game, induce a highly apprehensive emotional state in the player. Otherwise, induce a state of mild apprehension in the player.*

The emotional requirement has now been stated in a relatively quantitative manner. However, there are still qualitative descriptors for the intensity of the emotional state. In production, the design and development team can replace the qualitative descriptors with quantitative constraints. For example, the team could use well-established techniques for focus group testing as a replacement metric. *During focus group post-gameplay interviews, a minimum of 70% of the players that reach the room within five minutes of starting the game must indicate that either their emotional state was "scared" with a minimum rating of five on a scale of one to 10, or that their tension level exceeded five on the same scale.* However, this level of precision may be more than required and simpler metrics may be sufficient.

Using requirements engineering techniques, we have translated the game designer's intent for the emotional experience into a traditional functional software requirement. From this example we may conclude that capturing the designer's intent requires a mechanism for representing the induced emotional state. We should also be able to express spatial and temporal qualifiers.

3.3.2 Artistic Context

The designer must also be able to express the means by which the emotional state is induced as part of the emotional requirements. Traditionally, this might be viewed as a design or implementation detail that has no place in the requirements engineering process. However, within this context, we are capturing the game designer's vision. As such, the means by which a specific emotion is to be induced is a requirement and not a design or implementation detail.

Temporal and spatial qualifiers also apply to artistic context. In Figure 3.1(a), when asked about immersion in the context of the sound source, the game designer added the rebound path for the sound in the top right corner of the figure and indicated an approximate limit for the distance the sound could travel (and, therefore, be detected by the player). The designer was indicating an experiential requirement (here, a spatial qualifier on the intensity of the sound source), a means of capturing their vision, and not an implementation detail.

In contrast, Figure 3.1(b) is a portion of a race track for a racing game. The game designer began by tracing the path of the racetrack as shown and concluded the design as soon as the heavy black line was drawn. Discussion indicated that the designer felt that this particular geometry would be greatly enjoyed by the game player. There was no indication of direction of travel or purpose for any of the geometric elements. When asked, the designer indicated the direction of travel and seemed to feel that the purpose of each of the regions was obvious. Upon request, the designer added the annotations visible in the figure. For this segment of the race track the designer had, by implication, created eight separate experiential regions.

The game designer was then asked to assign an intensity level to each of the experiential regions. The results are shown in Figure 3.2, where the raw data is presented as a smoothed curve. The game designer explained that, in his opinion, it was very important to provide periods of increasing intensity followed by recovery periods: "...you can only stay on the edge of your seat for so long." The overall design of this segment of the race track clearly follows this philosophy – brief periods of tension followed by recovery periods with an overall trend toward ever increasing levels of tension.

Temporal qualifiers are very important in this racing game design: the purpose of the game is to complete the traversal of the virtual world in the minimum time. The intent of the designer is to control the intensity of the induced emotion (tension) as the player proceeds around the track. The means by which the designer controls the intensity of the emotional experience is through the nature of the challenges in each experiential region.

A first-person shooter and a racing game are fundamentally different constructs. In a first-person shooter, the player has much greater control over the time spent in each section of the world. Therefore, the spatial qualifiers tend to have greater relevance to the emotional requirements – it is only once a player enters a given experiential region that the rate at which the player's interactions with the game world unfold can be locally controlled. However, it may not be possible to force the player to enter an experiential region without irreparably damaging the sense of immersion. As noted above, in a racing game the temporal qualifiers tend to dominate the emotional requirements – the player has very little choice over where they travel within the virtual world.

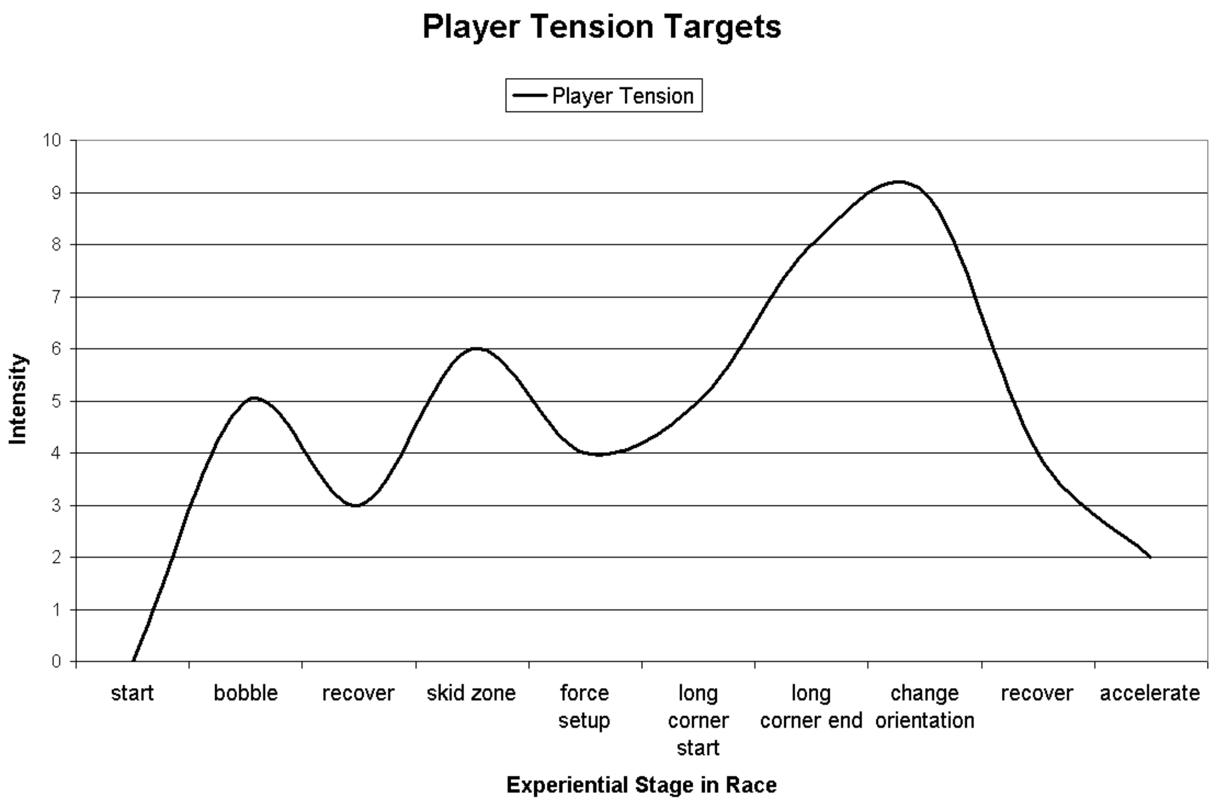


Figure 3.2: Emotional intensity timeline

3.4 Representing Emotional Requirements

Applying requirements engineering techniques to video game design can lead to significant resistance from game designers. From their perspective, we propose to take a essentially artistic activity and convert it to an engineering process. However, we can take advantage of the intensely graphical nature of the final software artifact to overcome this resistance by building on the familiar scenario [101] and storyboard [4] paradigms.

We initially proposed the *emotional terrain* (Figure 3.3(a)) for requirements whose artistic expression is strongly affected by spatial qualifiers. In an emotional terrain, the target emotion is linked to a spatial representation of the world, the emotion is color-coded, and the intensity of the emotion is associated with the luminance or perceived intensity of that color. Stereotypically, red can be used for danger, green for safety, and black or gray for neutral. In practice, color alone was insufficient to capture the necessary information.

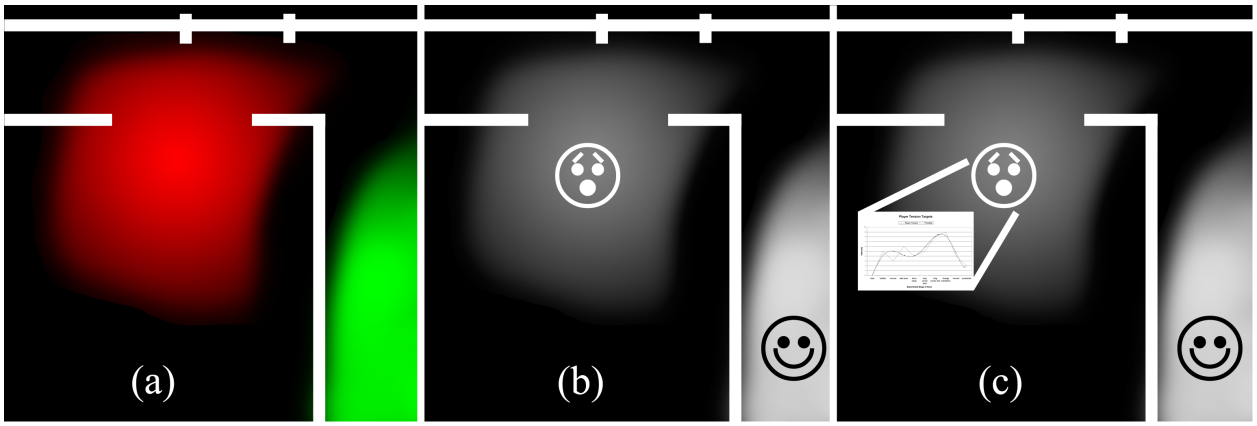


Figure 3.3: Representing emotional requirements

As an alternative, we then proposed the *emotional intensity map* (Figure 3.3(b)). Luminance (rather than color) is used to quantify intensity while the identity of the local emotion is indicated via a graphic symbol like an emoticon, a Chernoff face [26], or some derivative of Ekman's Facial Action Coding System [40]. The addition of the facial icon allows the artist to quickly express the desired emotion in a way that transcends typical societal barriers since most facial expressions are (effectively) universal [40].

For both emotional terrains and emotional intensity maps, spatial regions are quickly sketched and intensity can be quickly approximated with an airbrush style graphics tool. The graphic symbol for the emotion can be sketched or instantiated as text from a special symbol set.

We can also draw upon the film industry paradigm of the video editing timeline as an alternative for games that are dominated by their temporal qualifiers. Rather than applying audio tracks to a filmstrip, we can apply *emotion tracks* to an *emotion timeline* instead. Each track can capture the designers intent for a given emotion. For example, emotion tracks for tension, frustration, fear, relief, accomplishment, etc. could be associated with progress through the game. The timeline can be sketched as a simple graph within an experiential region, as shown in Figure 3.3(c). In a special-purpose software tool, a timeline editor could be accessed by interacting with the symbol and overlays could be used for display purposes.

3.5 Summary and Future Work

We have shown that video game design requires the capture and expression of emotional requirements: how the player is supposed to feel while playing the game. Emotional requirements express the emotional intent of the designer and the means by which the designer expects to induce the target emotional state. Analyzing game design fragments illustrated the further need for spatial and temporal qualifiers on both intent and means.

We introduced emotional terrain maps, emotional intensity maps, and emotion timelines as in-context visual mechanisms for capturing and expressing emotional requirements. Using a first-person shooter example, we showed that emotional requirements for intent can be readily expressed, including spatial and temporal qualifiers, using this low-fidelity mechanism.

In the future, we plan to evaluate the proposed techniques with a larger body of game designers. We shall continue our efforts to apply requirements engineering techniques to all aspects of video game design and the pre-production phase of video game development. Techniques for capturing artistic context, such as an artistic "look and feel" are still needed, as is their integration with emotional intensity maps.

Chapter 4
Emotional Requirements

The following paper was an invited contribution by the editor of the Requirements column and appeared in the January/February 2008 issue of IEEE Software. The IEEE Software paper greatly increased the awareness of the work.

Originally published as follows.

> David Callele, Eric Neufeld, and Kevin Schneider. Emotional Requirements. IEEE Software, 25(1):43 45, 2008 [19] *(with permission)*.

4.1 Introduction

Imagine that you are a software developer working on a video game. One morning, your boss comes in and says, "Make sure the new game is fun or were all out of a job! Our last game just got savaged by the reviewers!" Now, what can you as a developer do to help make this happen? Before you panic, begin by remembering that video game software exists to entertain, to actively engage the players cognitive and emotive processes while delivering a satisfying playing experience. This experience is what customers are purchasing. They care about the games functional aspects, such as the engine and control interfaces, only so much as they affect the player experience. The functional aspects are simply the expected minimum requirements that must be met before delivering the game.

The game designer crafts the playing experience – how the player should feel at certain points in the game. In "Requirements Engineering and the Creative Process in the Video Game Industry," a paper for the 2005 Requirements Engineering Conference, we showed that its not easy to effectively communicate the game design vision to the production team. New techniques were needed to ensure that the production team captured, understood, and implemented the intended player experience.

Like a movie director instructing the technical crew on implementing nuanced set design, lighting, sound, and acting, a game development team must work together to implement the game designer's vision. In "Emotional Requirements in Video Games," a paper for the 2006 Requirements Engineering Conference, we introduced emotional requirements to assist game developers with this task. Just as with functional requirements, emotional requirements have attributes that you must describe and model, and those attributes sometimes require careful balancing.

4.2 Requirements Challenges

Emotional requirements must contain at least two elements: the game designers intent (that is, the target emotional state) and the means by which the game designer expects (requires) the production team to induce that emotional state in the player. We can consider an emotional state such as happiness as universal, but the way you induce happiness isn't. Emotional requirements need context: classic pratfalls from vaudevillian theater can induce gales of laughter in a viewer who also feels horror at seeing a loved one fall. Unanticipated interactions between what the player sees, hears, and feels before or during the game can also affect the players emotional response to stimulus, which is further conditioned by the individuals personality, culture, and life experiences.

Emotional requirements blur the lines between requirement and specification. They require significant contextual information, possibly more than any other form of requirement. It's not as simple as stating "The player should be scared." In this domain, vaguely understood emotions interact with well-understood engineering constructs, generating requirements engineering challenges.

(a) Fear **(b)** Relief **(c)** Elation

Figure 4.1: Far Vista Studios (www.farvistastudios.com) Run the Gauntlet in-game promotional scenes: The players environment might promote feelings of fear, relief, or elation. (used with permission)

4.3 Induced Emotional Requirements

Its easy to generate emotional requirements that seem reasonable to the game designer but have no value to the player. For example, you might decide to dynamically adjust the games difficulty. In a real-world example, the designers modified a video games control systems to adapt to the users skill level, effectively enabling two players of significantly different skill levels to achieve the same score. This sparked accusations of cheating and unfair play from the player community. The game designer and developers had set an emotional requirement of "Make the player feel successful, independent of their skill level," but the player audience had a conflicting emotional requirement: "Validate my self-worth on the basis of my performance in this video game – relative to others." Game developers should validate such complex emotional requirements through user testing to ensure that the player shares (or derives value from) the game designers goals.

4.3.1 Representation

How do you represent the emotional state envisioned by the game designer? In the "Basic Emotions" chapter of the *Handbook of Cognition and Emotion* (John Wiley, 1999), Paul Ekman provides a list of culture-independent (universal) emotions (amusement, anger, contempt, contentment, disgust, embarrassment, excitement, fear, guilt, pride in achievement, relief, sadness/distress, satisfaction, sensory pleasure, and shame) that you can use as a basic emotional vocabulary for representing these states.

To illustrate, Figure 4.1 shows three scenes from Far Vista Studios game *Run the Gauntlet*. In Figure 4.1a, the central player is totally exposed to attack from all sides; the game designer wants the player to be nervous or fearful at this location. In Figure 4.1b, the central player has found a structure to hide behind; the game designer wants the player to feel relief from gaining some degree of safety from attack. While purely textual descriptions suffice, theyre difficult to maintain and don't align with the game's graphical paradigm. Previously, we successfully overlaid cartoon faces and emoticons on the game graphics (see Figures 4.1a and 4.1b) as a simple, graphical representation of the desired emotion and recommend this practice for economy and ease of use.

4.3.2 Cultural Conditioning

Emotional requirements can require localization efforts. International audiences (and members of international teams) might interpret the same symbols and events differently. Test your scenarios on your target markets to ensure that they don't elicit inadvertent interpretations. Perhaps the most cited example is the color red, which in North America means danger, whereas in China, red means good fortune.

4.4 Contextual Information Requirements

You should also consider how abstract contextual elements produce or affect player emotions.

4.4.1 Positional

An effective emotional requirement identifies where, in the virtual world, the player should feel a given emotion. In Figure 4.1c, we see a player leaping off a roof to perform an aerial attack on the street-level players below. Discovering this tactical advantage, which exists only at this location within the game, is intended to induce feelings of success (pride in achievement). Because it's expensive to create a virtual world, use emotional requirements to ensure that every element contributes to how the player is supposed to feel in that setting.

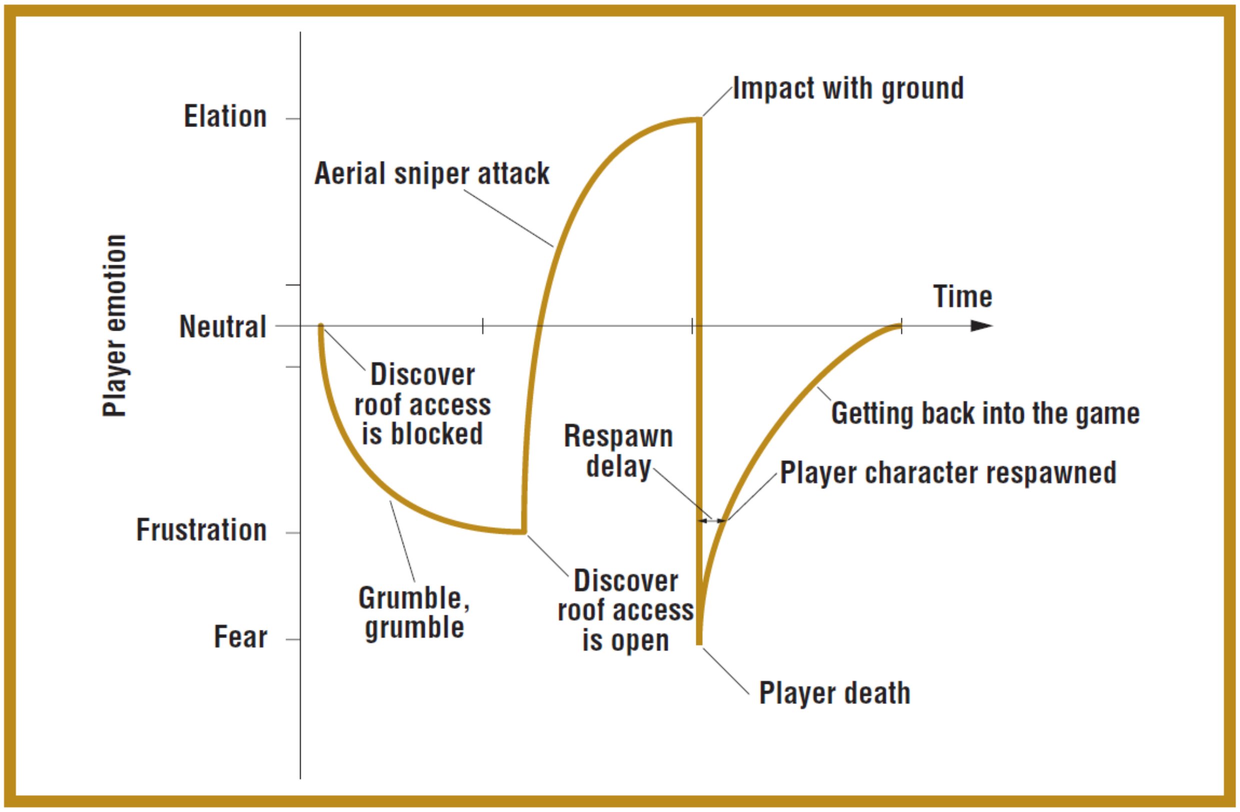


Figure 4.2: An emotional timeline for Far Vista Studios Run the Gauntlet. (used with permission)

4.4.2 Temporal

Context can vary with time. In Figure 4.1c, jumping off the roof elates the player. This feeling of success increases as the player approaches street level because the distance between the player and the enemy decreases and the probability of hitting the enemy consequently improves. However, this elation ends on landing because the game designer has set a trap – the fall impact actually causes the character to die and regenerate elsewhere. This emotional roller coaster is a staple of the action and suspense genres and delivers great satisfaction to the target audience.

Supporting the development of the game's story arc over time, the game designer could also block access to the roof for certain periods, deliberately inducing frustration in the players as they attempt to use the roof as a sniping position and find that they can't. The designer could then use the roof access as a trap to force a transition from frustration to fear.

Figure 4.2 shows the emotional timeline, which depicts players' emotions as they experience the game's story arc. Effective use of emotional requirements requires understanding how your audience reacts to emotional intensity. Players, just like movie-goers, generally react better to a change in intensity than to long-term exposure to high-intensity emotions. For example, you can only expose players to high-intensity emotions for a relatively brief period before they begin to become immune to the stimulus.

4.4.3 Relational

Gamers play because they want to have particular emotional experiences. If a game delivers those experiences, they play it again and again until they no longer achieve their emotional fix. Game designers must remember that this emotional experience is the player's definition of a successful game, and it might not match the emotions they've specified in their game design document. As gameplay progresses, players accumulate experiences that can lead to positive and negative prejudices. Development teams should attempt to consider these accumulated experiences. Which would you rather hear: "I am so angry; I just can't get past that enemy!" or "This game is worth every minute that you invest in it!" You can use extensive play testing to identify these emotional biases, but consider whether player perceptions are being skewed by prior experiences with the game.

4.5 Conclusions

Emotional-requirements techniques can help improve player experience and reduce development uncertainties. Focusing on entertainment software is a logical starting point, but we plan to extend our application of emotional requirements to other areas. Emotional requirements add the human element to engineering practice. This combination helps and encourages us to better understand those around us and might even help those around us better understand our practice.

Chapter 5

Balancing Security Requirements and Emotional Requirements in Video Games

In the traditional requirements engineering demarcation between Functional Requirements (FRs) and Non Functional Requirements (NFRs), emotional requirements are considers NFRs. Identifying and managing interactions between the categories, and prioritizing within and between categories, are common tasks for the domain practitioner. Videogame NFRs such as *fun* and *entertainment* tend to dominate all other requirements for this application domain. In this work, we investigate the effects of introducing a new (and dominant) class of NFRs into the requirements process. In particular, we look at interactions between emotional requirements and security requirements.

Emotional requirements, as originally envisioned, were a constructively motivated, creative affordance, used to help deliver the intended emotional experience to a willing audience. All stakeholders were assumed to be similarly constructively motivated. However, not all players are the 'good guys'; not everyone plays fair. A significant number cheat and, even worse, some of them actively attempt to disrupt or destroy the game experience for other players. We can model these destructive stakeholders as security threats of a particular type: they are willing, but apparently hostile users. This model led to an exploration of the application of emotional requirements to this unusual stakeholder class.

The poster *Balancing security requirements and emotional requirements in video games* won the Best Poster Award for Requirements Engineering 2008. Of particular interest to the judging panel was the introduction of in-game justice systems as a mechanism for negotiating requirements at runtime.

Originally published as follows.

David Callele, Eric Neufeld, and Kevin Schneider. Balancing Security Requirements and Emotional Requirements in Video Games. In RE 08: Proceedings of the 2008 16th IEEE International Requirements Engineering Conference, pages 319 320, Barcelona, Spain, 2008. IEEE Computer Society [18].

Abstract: A fundamental conflict exists between designers, players, and cheaters: *Who has control over how the game is played?* Resolving this conflict, by balancing the associated emotional and security requirements is challenging.

Emotional requirements can assist the development of security requirements and to prioritize their development. Failure to meet the player's emotional requirements can lead to market forces that override security requirements. We suggest that in-game justice systems would allow the players to act as a self-correcting mechanism for emotional requirement failures that lead to cheating or other threats to the integrity of the game experience. Further investigation into this form of just-in-time requirements negotiation is ongoing.

Keywords: Non-functional requirements, emotion, emotional requirements, security, security requirements, video game.

5.1 Introduction

The dominant security goal for most video games is ensuring the integrity of the playing experience. This goal is shared by all constructive stakeholders; Consalvo [30] and others have shown that players need to trust the integrity of the game – the same rules must apply to all participants and no player should have an unfair advantage over another.

A developer might define the corresponding security requirement as "All players shall play the game only as the designer intended the game to be played." However, if players don't enjoy the approved method of gameplay they will turn to cheating in an attempt to get at least some value from their investment. And as Consalvo [*ibid.*] notes "...cheating isn't just about subverting the (game) system; it's also about augmenting the system. It's a way for individuals to keep playing through boredom, difficulty, limited scenarios, rough patches or just bad games."

These comments illuminate the fundamental conflict between the player and the designer: *Who has control over how the game is played?* Is the player only allowed to play the game as designed? Or does the player control how the game is played? Perhaps the answer lies somewhere in-between.

In this work, we explore the intersection of security requirements and emotional requirements, investigating the use of emotional requirements to assist in the development of security requirements and identifying justifications for overriding security requirements with emotional requirements.

5.2 Related Work

While there is no universal definition for cheating [62, 30], Yan provides a cheating classification [109] that builds on prior work in security issues, extending Pritchard's work [86] while relating it to more traditional mechanisms for understanding and defining security requirements and security design. The most detailed analyses of cheating in video games is that by Consalvo [30], broadening our understanding of player motivations for cheating. The motivating factors for grief play (play that deliberately disrupts other players) have been most extensively studied by Foo [44], and to a lesser extent, Consalvo [30] while justice systems [97] remain relatively ignored.

5.3 Evaluating Threats

Players are not usually a traditional security threat such as a disgruntled employee in search of revenge. They are more like a frustrated employee who attempts to use unauthorized (if not illegal) means to bypass what they perceive to be obstacles to the proper discharge of their duties.

Emotional requirements can assist in evaluating the risk levels of these security threats. Identifying the sources of greatest frustration to the player, the *emotional irritants* (a negative emotional requirement, or a failure to meet an emotional requirement) will identify those issues most likely to sufficiently motivate the player to attack the game. The risk factor for an emotional irritant can be expressed as:

$$\text{Risk Factor(Emotional Irritant)} = \frac{\text{Level of Irritation}}{\text{Cost of Attack}}$$

Those security requirements associated with high risk emotional irritants should receive high priority in the development plan.

5.4 Resolving Requirement Conflicts

If the 'approved' method of gameplay is not actually fun for the players, the game may be a sales disaster. At this point, the developer and publisher will be strongly motivated to salvage some form of revenue stream – even if it means arbitrarily relaxing the security restrictions via a source code patch, or the publication of means for accessing alternative operating modes (*e.g.* developer shortcuts, *a.k.a* cheat codes). The initial security requirements are overridden in an attempt to salvage a flawed game – demonstrating that there do exist situations where emotional requirements can override security requirements.

There does not appear to be an optimal resolution to the conflict between security requirements and emotional requirements – the security goal of ensuring the integrity of gameplay is unlikely to be achieved. Instead, a negotiation process is needed that eliminates the requirement to identify and resolve all problems *a priori* yet allows them to be resolved as they occur and in a manner that addresses the emotional requirements of the stakeholders.

This resolution can be provided *just-in-time* by introducing an in-game justice system to apply corrective action to those who corrupt the integrity of the game experience. Sanderson [97] provides an informative view of such justice systems. For an in-game justice system to be effective, it must address issues of judicial authority, the penalties associated with various 'crimes', enforcement mechanisms, and whether enforcement has real-world consequences.

An in-game justice system can be used as a fall-back, catching those cases that were not considered in the requirements. Determining the requirements for a justice system, then developing and implementing it is expensive but we expect that some of the cost may be offset by reducing the number or scope of the initial security requirements.

Experience has shown that if griefing is not addressed, the griefers will come. Placing justice in the hands of the players means that they can act as dynamic systems that are able to adapt to, and counter, griefing tactics. We expect that the griefers will tire of victims that fight back and will move on to easier prey (in other systems).

5.5 Summary and Future Work

We have shown that emotional requirements can assist the development of security requirements by identifying the motivation behind security threats. The emotional irritants that motivate the attacks can be addressed proactively, potentially reducing the magnitude of the risk. Emotional requirements can also be used to help prioritize security requirements; strong emotional irritants that require low effort to overcome are the most likely attack vectors. The high-risk security requirements identified in this manner should be prioritized during development.

Failure to meet the player's emotional requirements can lead to market forces that override security requirements. If the emotional requirement failures are as a result of cheating or other threats to the integrity of the game experience, we suggest that in-game justice systems would allow the players to act as a self-correcting mechanism in the face of these security failures. The justice system places further requirement negotiation in the hands of the players, providing them with a framework wherein their own community values can develop.

Further investigation into the use of in-game justice systems as a form of just-in-time requirements negotiation is warranted and ongoing. The role of the community as a self-policing entity is worthy of further investigation, particularly with respect to the effects on the stringency necessary for the security requirements for that community: if the players will self-correct, it may not be necessary to invest as heavily in security infrastructure.

Chapter 6

Requirements in Conflict: Player *vs.* Designer *vs.* Cheater

The related workshop paper looked more deeply at the issues associated with the interactions between emotional requirements and other requirements, security requirements in particular.

Originally published as follows.

David Callele, Eric Neufeld, and Kevin Schneider. Requirements in Conflict: Player vs. Designer vs. Cheater. In Multimedia and Enjoyable Requirements Engineering - Beyond Mere Descriptions and with More Fun and Games, 2008. MERE 08. Third International Workshop on, pages 12 21, Barcelona, Spain, 2008. IEEE Computer Society [20].

Abstract: There are significant interactions between video game stakeholder emotional requirements and security requirements. Counter-intuitively, some traditional security requirements are not necessarily met by the game implementation – some forms of security breaches are condoned by the stakeholders (if not actually demanded by them) and the requirements engineering process must support these contradictions.

We present an overview of security requirements for video games and show how stakeholder diversity introduces significant complexities to the requirements negotiation process. Our analysis of certain security threats, and their emotional motivations, shows that these motivations form an important element of the emotional requirements and that significant context is necessary for properly capturing the emotional requirements related to security. Finally, we show how emotional requirements can be used to guide security goal development for this domain and propose the use of in-game justice systems to allow players to address security violations in realtime.

Keywords: Non-functional requirements, emotion, emotional requirements, security, security requirements, video game.

6.1 Introduction

Stakeholders often have differing opinions on the relative importance of a requirement. For productivity applications, the set of stakeholders is typically dominated by the users of the application and their immediate management. The stakeholder domain for video games must be more diverse. The entertainment aspect of the product means that emotions are involved and that there may be interactions that are not necessarily logical.

The stakeholders range from financiers to players and they share the common desire for a great game. However, great games are not easy to create. In prior work we showed that capturing the game designer's vision was difficult [16] and we introduced emotional requirements, emotion timelines, and emotion terrains [17, 19] to help capture that vision. Using a detailed case study performed with an industrial partner, Alves *et al* [3] elaborate on the challenges faced in requirements engineering for mobile video games, extending our work on requirements engineering for video games[16].

Emotional requirements, as originally envisioned, were a constructively motivated, creative affordance, used to help deliver the intended emotional experience to a willing audience. All stakeholders were assumed to be similarly constructively motivated, desiring the best entertainment experience that they could achieve.

But not all players are the 'good guys'. Not everyone plays fair. In fact, a significant number cheat and, even worse, some of them actively attempt to disrupt or destroy the game experience for other players. We can model these stakeholders as security threats of a very particular type: they are willing, but apparently hostile and destructive, users.

These destructive stakeholders are motivated in some way to act as they do. It is these motivations that we capture as (perhaps unfulfilled) emotional requirements. Rather than adopting a simple "all destructive stakeholders are the enemy" attitude, in this paper we explore the intersection of security requirements and emotional requirements, asking:

- Can we use emotional requirements to assist in the development of security requirements?
- Can emotional requirements be used proactively, to identify the motivations behind the security threat?
- Can emotional requirements be used to ameliorate security risks by providing insight into threat motivation?
- Are there situations where emotional requirements can override security requirements? And if so, with what justification?

To begin, we identify the constructive and destructive stakeholders and briefly explore their motivations. We review the related work and then present an overview of a generic security model for a typical multiplayer video game. A threat analysis is performed and possible attack vectors are explored to demonstrate how the related security requirements can be enhanced by emotional requirements. Conflicting emotional requirements are investigated to determine how they impact security requirements, and how the various demands can be balanced. We conclude with a demonstration of guiding security goal development for this domain with emotional requirements by deploying an in-game justice system then provide directions for future work.

6.2 Stakeholders

Consalvo [30] introduced the concept of *gaming capital* as a motivator for, and means of valuing, interactions between stakeholders in the gaming domain. Gaming capital is based upon economic principles, with *capital* an abstract representation of value or worth that has the (potential) ability to be exchanged between stakeholders. We motivate the stakeholder identification process with a simple producer / consumer model.

6.2.1 Producers

A functional analysis of the production process identifies the following primary contributors. The *Designer* works for a *Developer* that has a publication agreement with a *Publisher*. The Publisher arranges distribution with a *Distributor* that delivers the game to a *Vendor* who sells it to the final consumer. The secondary contributors include the *Financier*, who provides the necessary financial capital to all parties, the *Marketer*, who stimulates demand for the product, often working closely with the *Media*, who report upon the product. There are also *After-market suppliers*, providers of information, software, and hardware that interacts with the product and *Regulators*, that ensure compliance with regulations imposed by *Society*, an abstraction that exhibits (often contradictory and unpredictable) emotional responses to the product.

The supply side of the model is relatively straightforward and follows well-established free-market principles. The supply chain starts with a designer that we shall denote as principally artistically motivated. The developer, representing the studio that transforms the game from concept to product is denoted as motivated both artistically and economically, with the economic motivation dominant (those studios that are principally artistically motivated rarely survive for long). It is worth noting that, in the domain of multiplayer gaming, the Developer also has a long-term economic commitment to operating a game related service of some form.

The remainder of the primary supply side contributors are fundamentally economically motivated. So are the secondary supply side contributors, with the exception of *society* which we denote as emotionally motivated, but in a manner that we can not predict.

The economically motivated stakeholders are modeled here as perfect capitalists. Their sole emotional requirement is for success, as measured by their ability to make a profit. While this abstraction may ignore the contributions of their other emotional factors, it is sufficient for this work.

6.2.2 Consumers

The consumer stakeholders do not always follow the traditional user behavior patterns. Their behavior is complicated by the fact that they expect that their (non-functional) emotional requirements for entertainment will be satisfied.

However, when the playing experience goes poorly then player emotions, attitudes, motivations, and actions change dramatically. The player finds that their emotional requirement for *fun* is not being met. For example, they may perceive that their efforts to play are being thwarted, they may feel betrayed by the game, or they may even feel threatened by other players. The player now views some element(s) of the game playing experience in an adversarial manner.

Independent of the reason for the shift, while the player maintains this attitude we consider them a destructive stakeholder. However, the player still wants to play – it is just that they can not find satisfaction so they turn to alternatives that many would consider cheating.

Since the player still wants to fulfill their emotional requirement for fun, they should not be considered a traditional security threat. They are not, for example, a disgruntled employee in search of revenge. They are more like a frustrated employee who attempts to use unauthorized (if not illegal) means to bypass what they perceive to be obstacles to the proper discharge of their duties.

Within our economic model, these players are consumers that are willing to invest in purchasing the product. However, if the product fails to deliver its promised utility, they are willing to 'do what it takes' to get what they perceive to be value for their money (even if that means bending or breaking the 'law' as a last resort).

6.2.3 An Example

The relative nature of the definition of a destructive stakeholder (judged by intent rather than perception) means that we are exposed to observational error. However, in this work, we are looking for ways to be proactive, not reactive, so the temporal accuracy of an observation is not as important, just whether or not the player entered an adversarial attitude.

For example, assume that the player is unable to progress beyond a certain point in a game because they can not solve a particular puzzle[16]. They are unable to find any clues in the game as to how to proceed and their frustration level is very high. From their perspective, the game is a failure. In order to salvage their investment, they turn to the Internet for help. After a few minutes of searching, they find a guide (commonly termed a *walkthrough*) that explains how to get past this puzzle. Using this advice, they are finally able to continue with the game.

Did the player cheat?

For now, at least, the answer is irrelevant. What is relevant is the difference between intent and perception. The player perceives that the game is flawed: the puzzle was too difficult, and the designer neglected to provide a support mechanism of some form. Their emotional requirements for "fun", and for "receiving value for my money" dictate that they go out-of-game to meet their requirements.

Contrast this with the perception of the game designer: the player cheated, they went outside of the game for help and the player has broken the implied contract to "play the game as intended". The player has violated the designer's emotional requirement for maintaining the integrity of their artistic vision.

The emotional requirements for the designer and the player are in conflict; the designer now considers the player to be a cheater and the player feels that the designer has betrayed them. The *reason* for the conflict is always important, knowing *when* the conflict occurred is important only if the designer wants to reduce or eliminate a specific conflict.

6.2.4 The Exception to the Rule

It should be noted that there exists a class of players that do not fit well into this economic model. These are the *griefers* (players who participate in the game for the purpose of interacting with other players in a negative manner, see Section 6.3.6 for further details). They have no apparent rational economic basis for their actions; their behavior appears to be a manifestation of an emotional requirement for power (over others). As such, they appear willing to perform a direct exchange of game capital for emotional capital (gratification) – a currency exchange not willingly shared by the other stakeholders.

6.3 Related Work

We now review the security requirements literature, literature on player types, motivations, and their attributes as destructive stakeholders. We close with related work on negotiating conflicting requirements.

6.3.1 Security Requirements

Due consideration of security goals within the requirements engineering process is expensive and an informed cost-benefit decision is strongly recommended. Crafting security requirements is challenging [58, 43, 70] and many factors [74, 82] should be considered. Prioritizing security requirements [70] for video games is made even more difficult by problems with determining the economic value of play.

Security requirements also conflict with the emotional requirements of an immersive play experience. For example, authentication can be an intrusive operation. However, if a constructive player perceives that the game is prone to attack by destructive players, they may feel that there is sufficient justification for the authentication measures.

Moffet *et al* [74] state that it is not necessary to know the goals of the individual attackers when performing risk analysis, just what kind of attack they will mount. We substantively differ in this work: we look directly at motivation (the *why* behind the threats, and security in general) and try to determine if there are emotional requirements that can be met that mitigate the risk factors. Unlike the general practice of attempting to resolve all conflicting requirements, emotional requirements may not be resolvable – all that may be achieved is a set of requirements that lead to a state of constant, small-scale skirmishes between constructive and destructive stakeholders.

6.3.2 Misuse, Abuse, and Anti-Requirements

We have identified destructive stakeholders by their behavior patterns. These behavior patterns have strong parallels in the requirements engineering literature. Misuse and abuse cases [69, 99, 2, 57] are use cases that explicitly identify threatening scenarios (such as cheating) so that they may be proactively considered during systems design. The role of griefers and grief play is most similar to anti-requirements [32], a "requirement of a malicious user that subverts an existing requirement."

Emotional requirement failures are closely related to Pott's obstacles [85, 106]. The player's goal (to have fun) is blocked by failures in the game design [1].

6.3.3 Threats and Attacks

Attacks can be modeled [76] in many ways – most commonly in scenario form [77]. Difficulties [76] include organizing, managing, and prioritizing.

Resources detailing known attacks on games are readily available [25, 56]. These attacks have been roughly categorized [66] as cheating against the provider, other players, or the virtual society. There are active efforts [86, 53] to reduce the effectiveness of such attacks. However, Golle [53] notes that "...no defense appears possible against an adversary who has more intrinsic utility for using a bot than for winning the game." In other words, if the player is simply being destructive, with no rationale consistent with the game, then there is no protection against their actions.

6.3.4 Player Types

Numerous researchers have studied players in attempts to provide taxonomies of player types and player motivations. In particular, Yee [111, 110] has extensively studied players in massively-multiplayer online games. While most of the player types and their motivations are constructive, his analysis has confirmed the presence (and disruptive capabilities) of grief players but does not investigate cheating behaviors in detail.

6.3.5 Cheating

While there is no universal definition for cheating [63, 62], Yan provides a cheating classification [109] that builds on prior work in security issues [108, 107]. Yan extends Pritchard's work [86], relating it to more traditional mechanisms for understanding and defining security requirements and security design.

The most detailed analyses of cheating in video games is that by Consalvo [28, 29, 30] broadening our understanding of player motivations for cheating. These motivations have strong parallels with emotional requirements and form the basis for parts of this work

Most research into cheating does not address justice systems [97] although some do report on common practice at the time [107, 30].

Issues of morality and ethics have been addressed [90, 78, 30] and it is instructive to understand what mechanisms can be used to evaluate *good vs. bad* or *right vs. wrong.* Only then can questions like this be answered: If there are no consequences to breaking the rules, can an act be called cheating? Can a player cheat in a single-player game?

6.3.6 Grief Play and Griefers

Grief play (play that is intentionally disruptive of the game and other player's game experiences) and griefers (those who perpetuate grief play) have come to significantly greater attention with the advent of multiplayer online games. While there have always been those who play in this manner, they could only affect those in physical proximity; their actions were self-limiting. With the Internet, griefers can disrupt players anywhere in the world.

The motivating factors for grief play have been most extensively studied by Foo, and to a lesser extent, Consalvo [30]. Foo [44] reviews the prior work, compares it with other research into bullying and teasing and identifies four motivating influences: game (and game management) influenced, player influenced, (other) griefer influenced and self (griefer) influenced.

In later work, Foo [45] presents a more detailed analysis of the concept of grief play, looking at intention, perception, and side-effects. Three rule classes are identified: those in the code, those in the service contract (including game rules), and those implied by the game community via social etiquette. The importance of perception, particularly of motivation, is brought forward and is related to our work on stakeholder differences. As above, there remains an open question of rule enforcement and justice systems: Who has authority and what are the penalties?

6.3.7 Emotions in Requirements

Now that we have broadened the applicability of emotional requirements, we note that Ramos *et al* [88] performed earlier investigations into the interactions between change (organizational transformation, most particularly in Information Technology systems), requirements engineering, and the emotions of those affected by the change(s). In particular, they address the issue of including the users emotional responses into deployment planning, attempting to mitigate any issues before they become blockers – a proactive approach similar to our current work.

6.3.8 Negotiating Requirements

Easterbrook *et al* [38] describe a development environment quite similar to game development. The different viewpoints utilized in this work correspond to the perspectives of the constructive stakeholders in this work and consistency checking within a viewpoint is analogous to resolving emotional requirement conflicts. Of particular interest is the observation that requirements inconsistencies are not failures, they only need to be resolved if the owner of one of the inconsistencies requires resolution.

Menzies *et al* [72] describe a negotiation process whereby the mutually agreeable set of common viewpoints are identified such that a group can constructively work together. They note the importance of *buy-in*, an emotional commitment to the success of the endeavor. By adjusting perspectives, they show that the common set of requirements can be larger than anticipated.

The multi-disciplinary nature of the present work has been reflected in the breadth of the related work. It represents a synthesis of security analysis, security requirements, emotional requirements, requirements negotiation, and motivational psychology.

6.4 Security and Video Games

The dominant security goal for most video games is ensuring the integrity of the playing experience. This goal is shared by all constructive stakeholders; Consalvo and others (see

Table 6.1: Emotional Requirements

Emotional Requirement	**Description**	**Player Comments**
Escape, Experience	Distraction from (pressures and influences of) physical reality	I like to explore, especially in God mode. I want to be anonymous. I want to do things I can't do in real life.
Reward	Need for immediate feedback (of success or failure)	I *love* finding hidden rooms! The feeling when I finally mastered that move...
Posture, Image	How the player believes they are perceived by others	I love being the hero; It's cool to be bad!
Acceptance	Finding and becoming part of a community	These are my real friends...
Power, Control	Exercise power, control and influence	I love being able to dispense justice!
Accomplishment	Long-term accumulation of experience and reward	Figuring out how to advance my character, all the way to the highest levels – that's what I play for...

Section 6.3.5) have shown that players need to trust the integrity of the game – the same rules must apply to all participants and their playing experience should never include attacks by other players unless they have agreed to that playing mode[1]. We restrict our current analysis to this single goal.

In practice, "ensuring the integrity of the playing experience" is an excessively vague goal. The Developer may define the corresponding security requirement as:

> The player shall play the game as the Designer intended the game to be played.

Once the requirements engineering team is done with it, the same requirement might look like:

> All inputs to the system must be validated according to the Input Validation Rules as defined in Appendix A: Acceptable Game Play.

where Appendix A is a formidable document that, given the complexities of the typical video game, would be unlikely to provide complete coverage of all potential interactions.

[1]In gaming parlance, the two most common playing styles are *PvG*: Player *vs.* Game and *PvP*: Player *vs.* Player. PvP can also allow *Pk*: Player killing by other players.

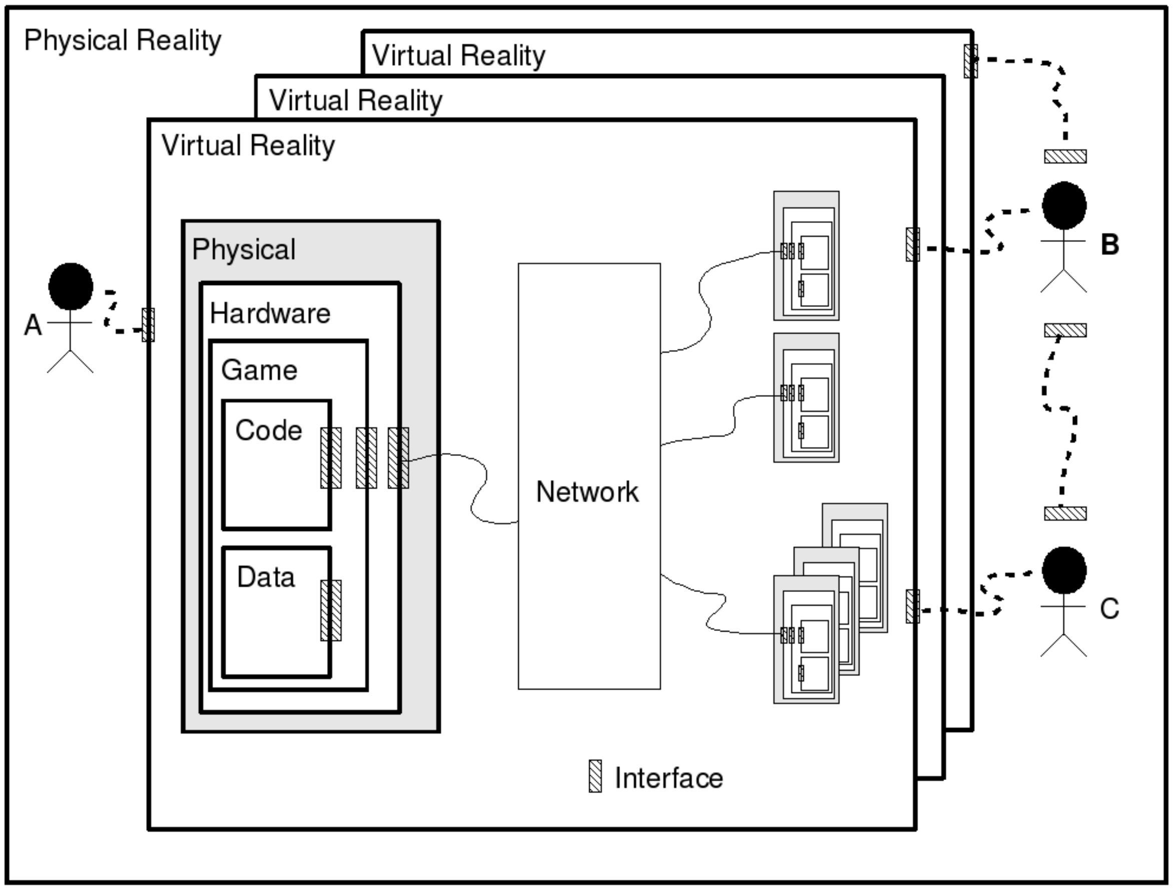


Figure 6.1: Generic Video Game Architecture

The cost/benefit analysis for this scenario is complex. What is the value associated with ensuring that the game is played only in the intended manner? Such a constraint might benefit the rest of the production channel by reducing support costs after the game is sold. However, if the approved method of gameplay is not actually fun for the players then the game may be a sales disaster. At this point, the Developer and Publisher will be strongly motivated to salvage some form of revenue stream – even if it means arbitrarily relaxing the security restrictions via a source code patch, or the publication of means for accessing alternative operating modes (*e.g.* developer shortcuts).

The player, of course, doesn't care about any of this, they only care about having fun. If relaxing the security restrictions to allow alternative gameplay means that players can enjoy the game, it may become a success simply by word of mouth about the "player-first" attitude of the Developer and Publisher.

6.5 The Player's Perspective

A sample of typical player emotional requirements for this context, condensed from experience and the cited literature, is presented in Table 6.1. The underlying emotions are heavily abstracted and are best interpreted as a motivating factor, as a *need* that the player attempts to satisfy by playing the game.

When these (and other) emotional requirements are not met, then otherwise constructive players may become destructive stakeholders. They are more likely to turn to some form of cheating and their observable actions may become beligerent toward other players. As Consalvo notes "…cheating isn't just about subverting the (game) system; it's also about augmenting the system. It's a way for individuals to keep playing through boredom, difficulty, limited scenarios, rough patches or just bad games". Some players make comments that indicate they cheated only because they were stuck, that they wanted to play the game in a different way (for the alternative experience), or because they were facing time constraints (they couldn't spend hours performing repetitive tasks just to meet the Designer's vision).

We note that the player comments indicate conflict between the Player and the Designer. The players do not share the designer's vision and demand the freedom to play the game in a manner of their choice. Thus we arrive at the fundamental requirements conflict for all video games:

> *Who has control over how the game is played?* Is the player only allowed to play the game as designed? Or does the player control how the game is played? Or is the answer somewhere in-between?

Ensuring that the game can be played, only as designed, requires significant investment in security infrastructure (and the associated demands on algorithmic correctness). As noted earlier, market forces can override the Designer's intent. Because a game is experiential, absolute control over the player's experience is only a goal, it can not be guaranteed and there may be real world moral or legal issues associated with fine control over the player experience – blatantly manipulating the player may not be socially acceptable.

6.5.1 Alternative Play as Threat

For discussion, we classify any deviation from the intended play experience as a threat. Not all threats must be addressed; their severity can vary greatly.

From the gameplay perspective, these threats manifest as follows:

- Conferring an unfair advantage; (deliberately) breaking a rule *and* deriving a benefit[2]. Examples include using cheat codes to bypass parts of the gameplay and programmatic assistance such as macros or *bots* (from ro*bot*, an automated assistant).

- Using information from outside the game. Examples include employing hints, guides, and walkthroughs. The game is played as intended, but the player doesn't do the 'work'.

- Exploiting the implementation, gaming the game. Includes breaking the rules of the game (because the rule was not enforced), taking advantage of bugs, emergent gameplay (particularly taking advantage in a covert *vs.* an overt manner)

- Technological Cheating. The hardware or communication channels are modified in some manner.

- Hacks. The binary expression of the rules, communications, the game engine itself, are modified to change the play experience.

Consalvo [29] notes that "...much of the time, cheating actually implies a player is actively engaged in a game and wants to do well, even when the game fails *them*." Given that the the player's feelings convert them to a security threat, emotional requirements are a useful means for capturing player motivation, even when it is destructively focused. We conclude that emotional requirements can be used to ameliorate security risks by providing insight into threat motivation.

[2]Implies a zero-sum game, that one player cheating causes another player harm. This is not always true: what about single-player games? Is it even possible to cheat against a "game as opponent"?

6.6 The Developer's Perspective

There are many ways that the integrity of the play experience can be compromised. We classify these threats as follows.

1. Physical. Attacks upon, or requiring access to, the physical devices used in gameplay. This can include player computers, controllers, communications links, servers, *etc.*

2. Logical. Attacks upon the rules of the game, principally exploits on unexpected interactions.

3. Temporal. Attacks that manipulate time, calibration, or sequencing – both within the virtual reality as well as in physical reality.

4. State (information). Attacks upon the information used to control the game. Typically performed in realtime but may also be attacks upon data repositories between games.

5. Social (player). Out-of-game attacks upon the players themselves. Also known as social-engineering attacks.

All of these threats distort the virtual reality in some way.

To better understand these threats, they shall be addressed in the context of the generic, multiplayer video game architecture shown in Figure 6.1 (loosely based upon the OSI network model). One or more *Virtual Realities* are situated within a *Physical Reality.* Each computer is composed of a *Physical* aspect that includes the associated operating system, and these computers are connected via some form of communications *Network.* The *Game* is a software artifact composed of *Code* and *Data* elements.

The integrity of the game can be threatened in many ways (see Sections 6.3.5, 6.3.6). In practice, the actual techniques most often used compromise the integrity of one or more of the interfaces shown in Figure 6.1. Hoglund and McGraw [56] use a set of prepositions (*over, under, into, outside*) to denote how the attack is made. For example, the *Game* relies upon the *Hardware* to perform certain tasks such as rendering the virtual world. If the expected video drivers are replaced with new video drivers that render all walls translucent (thus enabling the player to see oncoming attackers through the walls) then this attack occurs *under* the game…the integrity of the *under*lying infrastructure was compromised.

The four attack modes exploit assumptions about the integrity of the system on the other side of the interface. An attack positioned *over* a layer manipulates the input channels, *under* a layer manipulates the output channels. An *outside* attack is directed at a specific layer but is launched from at least one layer away (*e.g.* an outside attack on a *Game* is launched from the *Network* or from another *Physical* location). The final attack position, *into*, refers to manipulating the internals of the *Game* as represented by the *Code* and *Data*.

We also extend this paradigm to include an *inside* attack. An inside attack "games the game" by attempting to exploit the rules of the game in some way. As such, an *inside* attack is a meta form of an attack that gets *into* the game internals but does not rely upon direct manipulation of code or data representations in memory.

Given this context for attacks upon this architectural model, security requirements are most likely to focus on the integrity of the interfaces. A complete analysis would result in security requirements for each interface, addressing each attack mode.

Implementing such a thorough set of requirements may be prohibitively expensive. While emotional requirements do not appear to have a role in formulating security requirements (security requirements should manifest from security goals), they can be used to guide the prioritization process. Identifying the sources of greatest frustration to the player, the *emotional irritants* (a negative emotional requirement, or a failure to meet an emotional requirement will identify those issues most likely to sufficiently motivate the player to become a destructive stakeholder.

Removing or defeating these emotional irritants will require attacking the system in some way. If the player is willing to pay the cost of successfully attacking the system, they will do so. It follows that inexpensive attacks on significant irritants are the most likely to proceed so the risk factor for an emotional irritant is expressed as

$$\text{Risk Factor(Emotional Irritant)} = \frac{\text{Irritant Factor}}{\text{Cost of Attack}}$$

We recognize that it is difficult to be precise in a matter such as this. However, we feel that even informed estimates based on prior experience are better than no guidance at all. Those security requirements associated with emotional irritants with high risk factors should receive high priority in the development plan.

6.7 A Process

The model presented herein has been relatively simple: Play is going well but something bad happens and the player becomes a threat to the integrity of the game. If the irritant is sufficiently large, and the player is willing to pay the cost of attacking the game, then they will do so.

The provider of the game can attempt to ensure that this scenario does not come to pass by following this process.

1. Identify the player's generic emotional requirements – for your studio and/or the genre of the game.
2. Quantify the relative importance of each emotional requirement, even if it is just an informed estimate.
3. Identify corresponding emotional irritants, failures to meet the emotional requirements.
4. Identify the emotional irritants associated with gameplay elements specific and/or unique to this game.
5. Quantify the magnitude of the irritants and determine the associated risk factors.
6. Identify security requirements corresponding to the emotional irritants.
7. Prioritize corresponding security requirements according to the risk factors.

6.8 Resolving Requirement Conflicts

We noted earlier that a Developer or Publisher may override security requirements in order to salvage a flawed game. Given that this action is in response to the player's reactions to the game, there exist situations where emotional requirements can override security requirements. From a pragmatic financial standpoint, positive emotional reactions sell the game while negative reactions will kill it.

The addition of mechanisms that satisfy the player's emotional requirement for instant gratification via reward systems, by definition, weaken the security requirements for the main game by introducing complexity.

As noted earlier, there does not appear to be an optimal resolution to these conflicts – the security goal of ensuring the integrity of gameplay is unlikely to be achieved. Instead, a negotiation process is needed that eliminates the requirement to resolve all problems *a priori* yet allows the problems to be resolved as they occur and in a manner that addresses the emotional requirements of the stakeholders.

This just-in-time conflict resolution can be provided by introducing an in-game justice system. This justice system is then used to apply corrective action to those who corrupt the integrity of the game experience.

Sanderson [97] provides an informative view of such justice systems. These systems range from those where the Developer is also a Service Provider who acts as judge, jury, and executioner through to player policing systems wherein players are allowed to administer 'justice' upon each other.

Unfortunately, in-game justice systems suffer from the same issues as our real justice systems including false accusation, atonement, recidivism, and the need for appellate review. However, linking the justice system to the game capital system has been shown [97] to act as a deterrent to abuse, particularly if the cost is proportional to a player's wealth (to prevent the wealthy from preying upon the poor).

For an in-game justice system to be effective, it must address these issues:

- Who has judicial authority?
- What are the *penalties* associated with various *crimes*?
- What enforcement mechanisms are available?
- Does enforcement have real-world consequences?

A justice system can be used as a fall-back, catching those cases that were not considered in the requirements. Determining the requirements for a justice system, then developing and implementing it is expensive but we expect that some of the cost may be offset by reducing the number or scope of the security requirements.

Griefers and grief play remain an issue for it is difficult to judge the value of protecting other players. If there are few incidents of grief play then it may not be worth investing heavily. However, past experience [63] has shown that if griefing is not addressed, the griefers will come. Placing justice in the hands of the players means that they can act as dynamic systems that are able to adapt to griefing tactics. We expect that, eventually, the griefers will tire of victims that fight back and will move on to easier prey.

6.9 Summary and Future Work

We have shown that emotional requirements can assist the development of security requirements by identifying the motivation behind security threats. The emotional irritants that motivate the attacks can be addressed proactively, potentially reducing the magnitude of the risk. Emotional requirements can also be used to help prioritize security requirements; strong emotional irritants that require low effort to overcome are the most likely attack vectors. The high-risk security requirements identified in this manner should be prioritized during development.

Failure to meet the player's emotional requirements can lead to market forces that override security requirements. If the emotional requirement failures are as a result of cheating or other threats to the integrity of the game experience, we have suggested that in-game justice systems would allow the players to act as a self-correcting mechanism in the face of these security failures. The justice system places emotional requirement negotiation in the hands of the players, providing them with a framework wherein their own community values can develop.

In the future, we hope to extend the economic model begun here to provide more concrete mechanisms for valuing *fun* and *irritation*. Interactions with gambling research and decision-theoretic frameworks appear promising.

The risk factor analysis mechanism could then be extended by a detailed case study that compares the predicted attacks against actual attacks once the game is in production.

The role of the community as a self-policing entity is worthy of further investigation, particularly with respect to the effects on the stringency necessary for the security requirements for that community: if the players will self-correct, it may not be necessary to invest so heavily in security infrastructure.

Chapter 7

Augmenting Emotional Requirements with Emotion Markers and Emotion Prototypes

Field work done earlier with Far Vista Studios was reviewed, the participants revisited, and we found that the adoption of emotional requirements at the studio was lower than expected. While the combination of emotional requirements and emotional intensity maps were useful, the media production team did not find them sufficiently useful to adopt them, because the emotional intensity maps did not indicate how the target emotion was to be induced or where the inducing elements were located.

Drawing upon the work of Smith in the application of cognitive psychology to film-making [84, 100], we adopted Smith's emotion prototypes to provide further guidance to the production team and emotion markers as triggers for intended emotions. Emotion prototypes have three characteristics: They have an object orientation; the emotion is cued, or triggered, by an object or the action taken by an object. They demonstrate an action tendency; the emotion spurs us to take some action. Finally, they demonstrate a goal orientation; there is some purpose to the action that we take. Smith also identifies an emotion marker as something that will engender a brief burst of emotion but probably does not affect the narrative or underlying story. Emotion markers can take any form; they may be sounds, scenes, or even dialog. There may be more than one emotion marker in a given scene and it is expected that one or more of the emotion markers is the cue or trigger in the emotion prototype.

This resulted in extending the ER formalism to a (cue, action, goal) triple, to the identification of explicit locations for markers, and to the development of techniques for eliciting, capturing and visualizing the requirements. For this work, we collaborated closely with Far

Vista Studios team members, in the spirit of an action research approach.

This work was originally published in two parts. The poster is presented first, followed by the paper.

> David Callele, Eric Neufeld, and Kevin Schneider. Augmenting Emotional Requirements with Emotion Markers and Emotion Prototypes. In RE 09: Proceedings of the 2009 17th IEEE International Requirements Engineering Conference, pages 373 374, Atlanta, GA, USA, 2009. IEEE Computer Society [21].
>
> *Abstract:*A production-phase weakness in emotional requirements was identified and resolved during a follow-up study. The definition of emotional requirements was extended to include emotion prototypes and emotion markers. Improved practices for identifying media assets for emotional requirements were developed, enhancing their utility to the production process.
>
> Keywords: Non-functional requirements, emotional requirements, emotion, video game.

7.1 Introduction

Our research program is motivated by a desire to reduce the risks in video game development. An evaluation of development processes in the video game industry identified a problem with the transition between the pre-production and production phases of game development [16]. Emotional requirements (capturing the intended emotional experience for the player) and emotional intensity maps (a graphical representation of the intended player experience within the game world) were developed [17] to improve the transition and then introduced into the development process for the game *Run the Gauntlet* by Far Vista Studios.

A follow-up investigation with the lead game designer at Far Vista Studios identified that the adoption of emotional requirements at the studio had been lower than expected. While the combination of emotional requirements and emotional intensity maps were useful to the game designer, the media production team did not find them sufficiently useful to trigger adoption – the emotional intensity maps did not indicate *how* the target emotion was to be induced or *where* the inducing elements were located.

The production team identified that some form of indicator must be placed within the game world to act as a trigger to induce the desired emotional state in the player. In response to this observation, the definition of emotional requirements was extended to include Smith's emotion markers (as triggers for the intended emotion) and the associated documentation was enhanced to include the three characteristics (cue, action, and goal) of Smith's emotion prototype [100] to provide further guidance to the media production team.

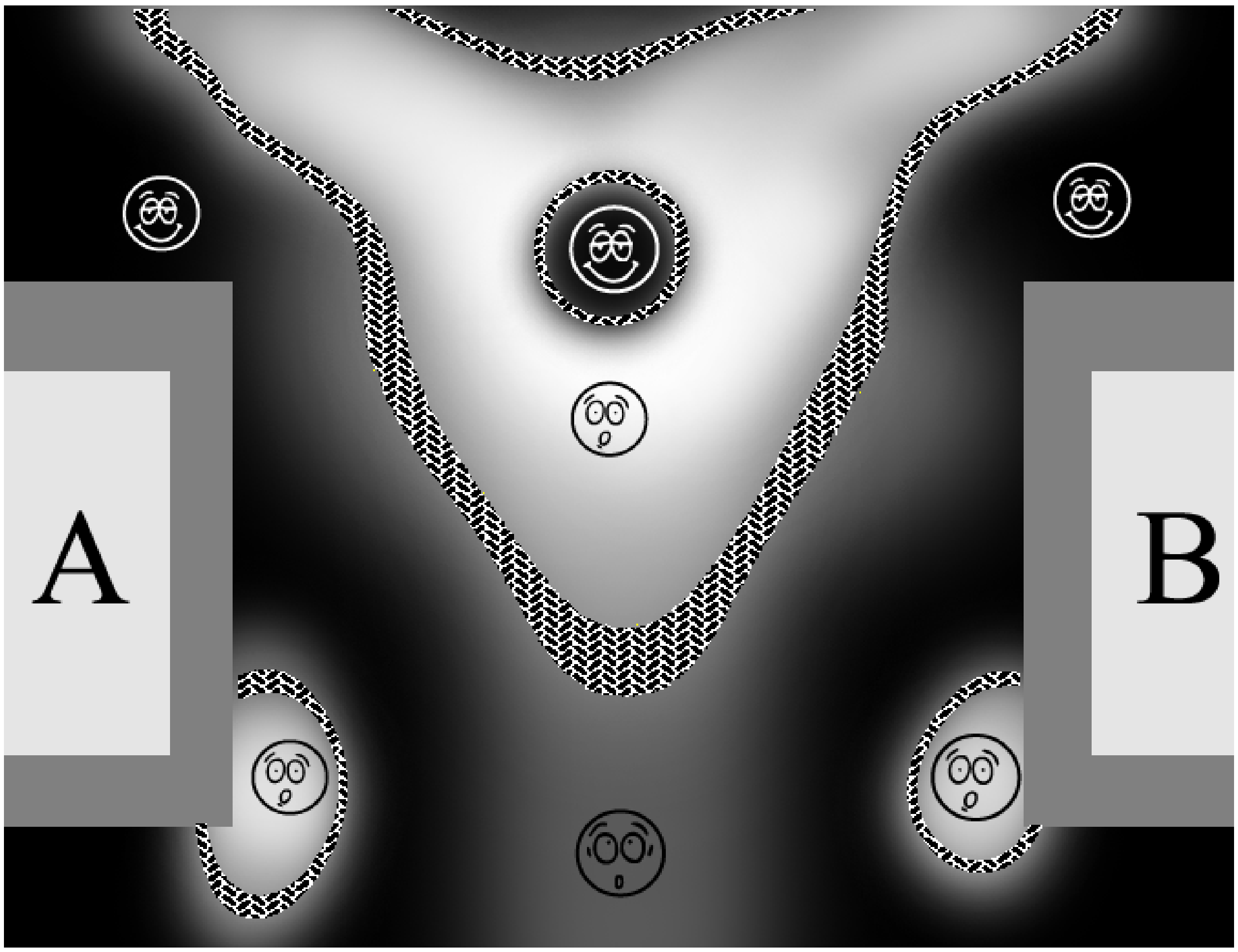


Figure 7.1: Sniper scenario emotional intensity map. Black fill denotes safety, white fill denotes danger, pattern fill indicates possible locations for emotion markers. Emoticons represent player emotion.

Figure 7.1 is a plan view diagram for a sniper scenario in the combat game *Run The*

Gauntlet, constructed by the game designer using a simple graphics editor. The Sniper may take a position in the buildings marked A and B while the Runner must attempt to pass through the shaded region between the buildings. The safe zones (darker means safer) imply that there are aspects of the virtual world that make these zones safe – most likely physical constructs used as emotion markers. In this figure, a luminosity thresholding algorithm was used on the shaded region to identify possible locations for the emotion marker(s). These locations, identified with a pattern fill, thereby provide the location guidance for the media production team. Accompanying notes identified the nature of the cue (*e.g.* barriers to hide behind), the expected player action (hide behind the barriers), and the player's goal (to rest in safety while trying to decide what to do next).

7.2 Elicitation and Capture

The elicitation and capture process is as follows.

1. Create the scenario concept and capture a textual summary and a few sketches of the virtual world.
2. Iterate as necessary:
   (a) Define the gameplay experience. What actions can each player take, what assets can the players utilize, how can the players interact?
   (b) Define the artistic context, the virtual world, in sufficient detail that the definition can be given to the media department for implementation.
   (c) Capture the emotional requirements using the definition of Section 7.3.
   (d) Iterate as necessary, within the context of (cue, action, goal), to ensure that the desired player experience will be created:
      i. Evaluate interactions with the artistic context.
      ii. Evaluate interactions with the defined gameplay experience.

Evaluating the interactions with gameplay and artistic elements often presented opportunities to define new interactions – in essence, to invent new requirements. The following interaction patterns were identified: Spatial – The location of the element within the virtual world; Temporal – The interaction pattern is dependent only on time, or on a (readily) discernible or deducible function that includes time; Engine Attributes – manipulating the physics of the virtual world; Game Attributes – those aspects that are unique or defining elements of the game.

After exploring the possibilities for realizing the scenario, a pseudo-verification phase should be performed to verify that the value proposition for each artifact is sufficient to justify expending the necessary resources.

7.3 Specifying Emotional Requirements

An emotional requirement, a guideline that provides sufficient information about the relationship between the intended emotion and the virtual world such that the communication and specification needs of the game designer and the media production team are met, is defined as follows.

1. The intended emotion. Use of a reference list or ontology (*e.g.* [83]), standardized for the project or organization, is recommended. Emoticons, or local artwork, can be used as placeholders.

2. The artistic context, *e.g.* the look-and-feel.

3. The emotion prototype.

    (a) The cue, or trigger (emotion prototype). The objects, animations, sounds, lighting changes, or other elements of the virtual world that are used to trigger the player's emotional response.

    (b) The action that the player is expected to take. This action is specified relative to the cue.

    (c) The purpose or goal of the player's response. The goal integrates the emotional requirement with the gameplay requirements and design.

4. The emotion timeline. One or more elements of the emotion prototype may be time-dependent [17].

7.4 Conclusions

Weaknesses in the prior definition of emotional requirements and emotional intensity maps were addressed by the addition of (cue, action, goal) information to the emotional requirements and by the explicit identification of potential locations for the emotion markers (cues).

An enhanced elicitation, capture and specification process for emotional requirements, that better meets the needs of the production team, was developed and tested. Four interaction patterns that may be used to identify new opportunities for enhancing the player experience were also identified.

Chapter 8

Visualizing Emotional Requirements

The poster from the previous chapter was also published as a full length paper, presented here.

The definition for emotionally requirements continues to evolve as we gain field experience with the techniques. The current (working) definition for emotional requirements is given in Appendix A.

David Callele, Eric Neufeld, and Kevin Schneider. Visualizing Emotional Requirements. In Requirements Engineering Visualization (REV), 2009 Fourth International Workshop on, pages 110, Atlanta, GA, USA, 2009. IEEE Computer Society [22].

Abstract: Emotional requirements capture the game designer's vision for the player's emotional experience and are used to facilitate communication between pre-production and production teams. However, production-phase deficiencies in emotional requirements have been identified. In this work, we extend the definition of emotional requirements to include emotion prototypes and emotion markers and present improved techniques for eliciting, capturing and visualizing emotional requirements. A detailed investigation of one gameplay scenario is presented, with a focus on evaluating visualization techniques for emotional requirements. The solutions developed in this work met the needs of all development team members and appear to be general solutions for the domain.

Keywords: Requirements visualization, non-functional requirements, emotion, emotional requirements, video game.

8.1 Introduction

Our research program is motivated by a desire to reduce the risks in video game development. Our evaluation of development processes in the video game industry [16] identified communicating the game designer's vision across the transition between the pre-production and production phases of game development as a source of development risk. We developed emotional requirements and emotional intensity maps [17] to capture the intended emotional experience for this vision and showed how they can alleviate some of the communication challenges that occur between pre-production and production teams.

An emotional requirement captures the emotional state that the designer intends to induce in the player, and the artistic context (the look and feel) within which the emotional experience is to occur. An emotional intensity map is a lightweight visualization technique that allows the game designer to situate the intended emotional experience within the virtual world (see Figure 8.3 for an example). Emoticons are used to identify the player's intended emotional state within simplified representations of the virtual world and grayscale shading is used to describe the changes in the player's emotional state within different parts of the virtual world.

Emotional requirements were introduced into the development process for the game *Run the Gauntlet* by Far Vista Studios for further evaluation within an industrial setting. During a follow-up interview with the lead game designer at Far Vista Studios, we learned that the adoption of emotional requirements at the studio had been lower than expected. The game designer self-identified the principal reason for the reduced usage as "they [emotional requirements] did not provide the expected benefits". When queried further, he noted that the combination of emotional requirements and emotional intensity maps were useful in his role as a game designer but that the media production team did not find them sufficiently useful to trigger adoption. Followup with the production team identified two weaknesses: the emotional intensity maps did not identify *how* the target emotion was to be induced in the player nor did the map identify *where* the inducing elements were located. In this work we show how how we have augmented the definition of emotional requirements with emotion prototypes and emotion markers to address these shortcomings and report on our investigations into visualizing these emotional requirements.

We used an action research approach [37] for our investigation so that we could collaborate closely with the team members to critically study how they used emotional requirements. While this is not a strict action-research study due to resource constraints, we followed the guidelines for this approach as much as possible for our work. We observed the issues hindering greater adoption by the production team and actively engaged both the game designer and members of the production team in the refinement of the proposed solution. This study was limited to a single development team and the results met all of their needs. The proposed solution appears to be general and has support from prior work in other fields but has not yet been validated with other teams.

The study focused on a single scenario used in the game *Run the Gauntlet* by Far Vista Studios, from conception through to virtual world implementation and the initial stages of gameplay testing and balancing. The requirements portion of this study was performed over a two week period. There were two in-person sessions of approximately six hours each and numerous telephone conversations of more than 10 hours total length. Ongoing media production lasted approximately three more weeks.

In the remainder of this work, we present a review of related work and the context for the study[1]. We report upon the extensions made to emotional requirements as a result of the study and techniques for visualizing the augmented emotional requirements. We review the results of a design effort that utilized the revised emotional requirements and propose mechanisms for integrating emotional requirements into the development workflow. Finally, we present a revised definition for an emotional requirement then present our conclusions and make recommendations for future work.

[1]A poster version of related elements of the same study is available in the Proceedings of Requirements Engineering 2009

8.2 Related Work

8.2.1 Storyboards

Storyboards are a well-recognized prototyping tool, originally developed for the movie industry where they are used for planning and communication as a film is prepared for production. These prototypes, quick sketches that draw heavily from comic strip techniques to convey the sense of the planned shots and their sequence, are also in common use in the game industry. Storyboards have also been used in requirements engineering, Andriole [4] introduced them to the domain over 20 years ago as a tool for requirements verification during customer sessions. More recently, Thronesbery [104] proposed the use of storyboards during requirements elicitation to also capture design knowledge from the domain experts. Reeder [89] showed how the requirements engineering phase of an industrial design process can be enhanced by the application of storyboarding techniques based on photographic images rather than the work of a sketch artist.

Extending storyboard tools such that they support emotional requirements could assist game designers that utilize storyboards in their work. For example, the image captured within the storyboard could be the emotional intensity map. The accompanying textual information could capture the specific instructions of the game designer as to how to induce the desired emotional state in the player and provide further guidance regarding the artistic context.

8.2.2 Film Studies and Cognitive Psychology

In their collection of film studies essays written from a cognitive psychology perspective, Plantinga and Smith (Eds.) [84] note that cognitive psychology practitioners tend to "discuss emotion states in terms of goals, objects, characteristics, behaviors, judgments, and motivations." Smith further notes in a later essay in the same work that the "concepts such as pleasure, and displeasure, and desire used in film studies are too broad to provide specific insight into how a particular film makes its emotional appeal at any given moment", motivating his work toward gaining the desired precision.

These perspectives have strong parallels with our work and we are able to use Smith's work as the exemplar for the application of cognitive psychology to film studies and, by extension, to our work. Much of Smith's work that is referenced herein is aimed at performing critical analysis of the emotions in film in a *post hoc* manner – one of our goals is to use the same or similar concepts *a priori*, in the requirements and design phase.

In games, we are limited by a lack of controlled dialog between players, particularly in Player *versus* Player (PvP) games. Typically, the only player to player communication is unmoderated, via some form of textual or vocal chat channel. Therefore, the artistic context is relatively more important in games than in film and Smith's perspective has greater immediate relevance than the work of the other researchers.

Smith posits that cognitivists believe that we recognize emotions by pattern matching against *emotion prototypes* [84, 100]. Emotion prototypes have three characteristics. They have an object orientation; the emotion is cued, or triggered, by an object or the action taken by an object. They demonstrate an action tendency; the emotion spurs us to take some action. Finally, they demonstrate a goal orientation; there is some purpose to the action that we take.

Smith identifies an *emotion marker* as something that will engender a brief burst of emotion but probably does not affect the narrative or underlying story. Emotion markers can take any form; they may be sounds, scenes, or even dialogue. There may be more than one emotion marker in a given scene and it is expected that one or more of the emotion markers is the cue or trigger in the emotion prototype[2].

Emotion prototypes and emotion markers, as described by Smith, illustrate how the questions raised by the production team have been addressed in other fields.

8.3 Emotion Prototypes and Markers

The three characteristics of the emotion prototype are useful refinements upon the emotional requirement; explicitly providing this information could provide some of the guidance

[2]For the remainder of this work, we shall refer to the emotion marker as singular, understanding that the plural is also supported

requested by the production team as to *how* the emotion is to be induced. The characteristics can also be used in formulating metrics for evaluating the quality of the emotional requirements: Does the emotional requirement sufficiently exhibit these characteristics such that the requirement provides appropriate guidance for the production team?

While our earlier work [17] proposed the use of emoticons as a lightweight mechanism for identifying the intended emotion, the concept of the emotion marker augments the emoticon by providing specific information that we had generally identified as belonging to the artistic context. The addition of a direct link between the emoticon (target emotion) and the emotion marker (emotion-inducing element) meets the guidance needs of the production team.

The emotion prototype and the related emotion marker do cause some issues with representation - the extra information is not part of the emoticon, it is not necessarily co-located with the emoticon, and does not appear to have a suitable visualization. It may be that integration with the storyboard, as discussed earlier, is most appropriate and this issue remains under investigation.

Despite the additional complexity that was introduced, we decided to address the identified weaknesses in the prior definition of emotional requirements by extending their definition to include Smith's emotion markers and enhancing their internal documentation using the three characteristics (cue, action, and goal) of the emotion prototype.

While we gain insight from Smith's work, we must also remember that games are less narrative than film; the game designer is unable to absolutely control the narrative journey in the same manner as a writer/director for film. This lack of narrative control is replaced by interaction with a set of designed experiences that are used by the game designer to emotionally manipulate the player into the path of the desired emotional and narrative journeys.

8.4 Scenario Concept

Run The Gauntlet is a player *vs.* player combat game. One of the most important scenarios in this genre is the sniper scenario: There are two players, the Sniper and the Runner, and they embody the classic antagonist *v.s.* protagonist narrative model. The fundamental goal of the Runner (protagonist) is to survive contact with the Sniper (antagonist) while the fundamental goal of the Sniper is to prevent the Runner from achieving their goal. Both of the players also share the goals of surviving the scenario, maximizing their gains, and minimizing their losses.

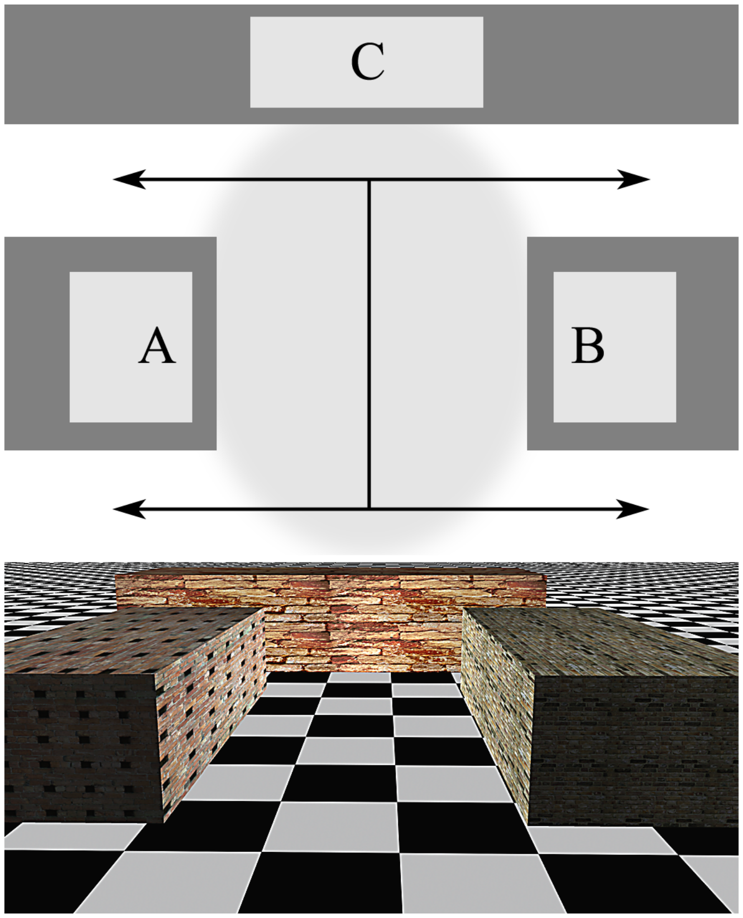


Figure 8.1: Sniper scenario pre-visualizations. The upper image is a plan view of the scene, the lower image is a minimum cost 3D rendering.

In the study scenario, the Sniper can take a position in one of the buildings marked A, B or C in the top half of Figure 8.1, a plan view of the relevant portion of the game world. The Runner is free to move about on the streets below and must successfully transit the shaded danger zone, while under attack from the Sniper, to continue onward to other gameplay regions and other gameplay experiences. The bottom half of Figure 8.1 is a low-fidelity 3D prototype of the same scene, typical of those used at the beginning of what is referred to as the “pre-visualization phase” of game design.

Capturing an intended emotional experience in the form of requirements is challenging. For example, the intended emotional experience for the Runner can be described in narrative form as follows.

> I approach the area with nervousness because I expect to be surprised in some way. When the attacks begin, I am afraid but I am optimistic that I can survive and I am excited by the challenge. I recognize a puzzle that I have to solve and when I do survive, I feel relieved and satisfied. When I fail, I am disappointed and if I fail too often then my annoyance can turn to frustration and anger.

Mechanisms for performing this conversion are part of our current research effort. Compared to functional requirements, a significant challenge with emotional requirements is precision: what emotions exist within the game design, what do we call them, and do the labels for the emotions mean the same thing to all members of the development team? Another challenge is to convert this description of an emotional experience to emotional requirements that are situated within the game world, under the assumption that doing so will reduce the chance of miscommunication between the team members. Finally, are there elements of emotional requirements that can and should be visually expressed and some that are more appropriately expressed via another mechanism?

8.5 Designing the Player Experience

For practical reasons, we need to standardize emotion-specific terminology – at least on a per-project basis. Emotions and their categorization have been intensively studied and there are many results available. In this study, we used the (primary, secondary, tertiary) categorization developed by Parrott [83]. We note that we employ this categorization as a language reference and ontology without making judgment as to its absolute accuracy since it’s role is to facilitate communication within a team and it is not used to perform comparative analyses between teams.

In Parrott’s classification, the primary emotions are the primitive emotion states. Secondary emotions are those generated by some form of deliberation upon the primary emotion and the generating stimuli. Tertiary emotions are similar to secondary emotions but there is also an element of loss of control or attention. . . to some degree, an involuntary response.

The game designer was presented with this categorization and asked to identify the intended emotions for this scenario. The chosen emotions included (among others) elation, satisfaction, joy, excitement, exhilaration, relief, surprise, disappointment, nervousness, sadness, and fear. A review of the chosen emotions identified a pairing between emotions such as nervousness and surprise and also between joy and anger/sadness. While not truly opposites, these pairings are strongly contrasting and we feel that there is an underlying principle at work: Satisfactory player experiences for this genre appear to be derived from enforced shifts between strongly contrasting emotional states. In this scenario, as the player becomes aware of the obstacle, they become nervous. When the threat is finally exposed, they are surprised. If the player overcomes the obstacle, their mood is directed toward an emotion in the joy category. If they fail to overcome the obstacle, they experience one of the emotions identified in the anger or sadness categories and they tend in that direction.

8.5.1 Emotions and Emoticons

A visual representation for emotions is necessary to meet the goal of situating the requirements within the game world. Further, given the traditional business requirement for archival storage of production documentation, we would prefer a mechanism that survives the transition from interactive query or inspection within the game world to print media. Emoticons can act as useful placeholders but they do not necessarily provide the resolution required to support the designer's choices – we found ourselves searching for an expression that was "just right".

In Figure 8.2 we see two examples of abstraction. In the top half of the figure, we see a selection of emoticons from a font freely available on the Internet. In general, there is insufficient information in these images to be able to readily discern the difference between any but the primary emotions. In the bottom half of the image, we see a selection of sketch artist cartoon faces. If sketches of this quality were converted to a font, then there would be less opportunity for misunderstanding (there were some issues associated with members of the production team relating the emoticon to the underlying emotion across all aspects of the production process). Detailed and expressive emoticons, implemented as a font, would greatly facilitate inclusion or adoption within standard production tools used in the production process, at little or no cost.

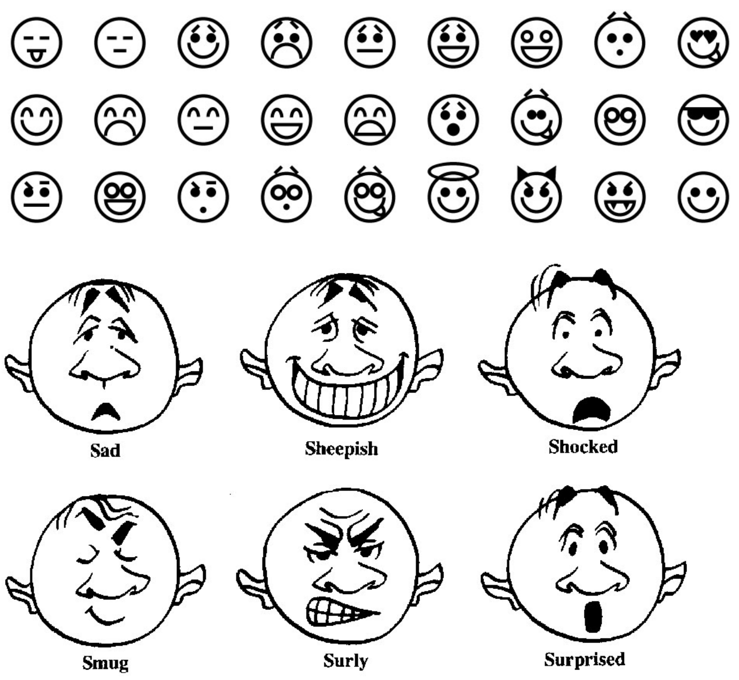


Figure 8.2: Top: A selection of typical emoticons from the EmotRG font. Bottom: Sketch artist cartoon face samples. Authors unknown.

We recommend that each team develop their own emoticons, or equivalent images, for their use so that all team members instantly recognize the intended emotions. If this proves impractical, a simple text label could be used but a text-based approach suffers when images are significantly reduced in size.

Our experience has shown that we can capture and represent the primary emotions, and some secondary emotions, with a well-designed set of emoticons. However, tertiary emotions require significant context for proper interpretation and may not be amenable to this technique. If game design progresses to the point where designers are actively identifying tertiary emotions, we expect that the emoticon would be used somewhat like the actor symbol in a use-case diagram – as a placeholder that indicates the presence of further information, such as our earlier suggestion [17] to bind the emotion timeline (a graph that illustrates the relationship between emotional intensity and time) to the emoticon. Given that we are now extending the emotional requirement to include the characteristics drawn from the emotion prototype (a *cue* that identifies the relevant emotion marker(s), an *action* describing the expected player response, and a *goal* that captures the interactions with the world), this information should also be bound to the emoticon.

Once the desired emotions were identified from the categorization, the corresponding emotional intensity map (Figure 8.3) was generated for this scenario using standard graphics tools on the same plan-view template used in the upper image of Figure 8.1. The dark zones represent relatively safe areas (HAPPY, RELIEF) for the Runner while the white zones are the zones of highest danger (FEAR). The game designer's intended player emotion is identified by an embedded emoticon within each zone.

8.6 Generating the Emotional Intensity Map

In Figure 8.3, the game designer has specified that there is a large Y-shaped region where the Runner is in greatest danger and two smaller regions near the corners of buildings A and B that are almost as dangerous. The designer has also specified that there is a region of safety within the Y-shaped zone. This safe zone implies that there is something in the virtual world that makes this zone safe – most likely a physical construct.

Grayscale shading is used to characterize transitions between two emotional states such as FEAR and RELIEF. The luminosity indicates the relative strengths of the states and the background mood is assumed (externally documented). However, this model breaks down if we want to indicate transitions between more than two states. For example, a transition between FEAR and a combination of RELIEF and RESENTMENT (*e.g.* I made it, but curse the game designer for making it so hard!) is not supported for we have no way to allocate the respective contribution of RELIEF and RESENTMENT.

A multi-layer compositing technique, using controls for translucency and visibility could be used to simultaneously support multiple emotions. However, such a technique may only be feasible in the interactive medium of the computer and does not readily support the transition to print media, especially grayscale printing.

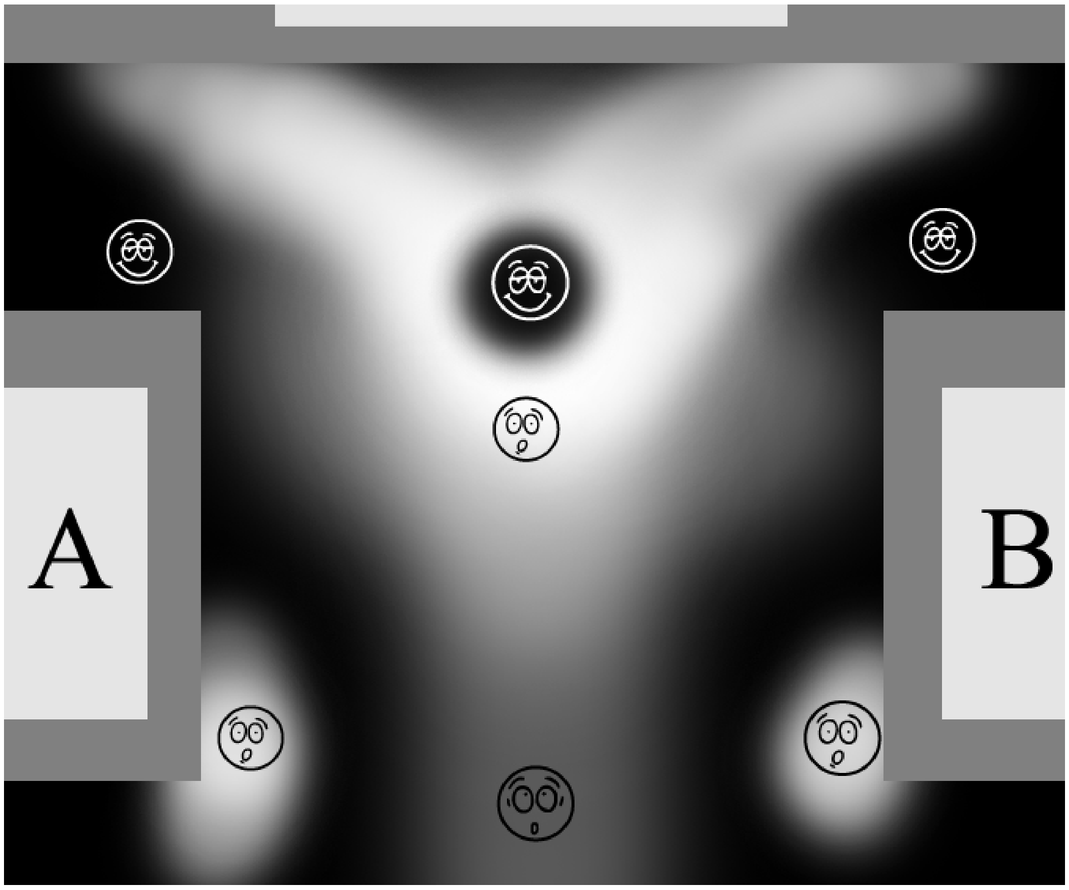


Figure 8.3: Sniper scenario emotional intensity map. Black denotes safety, white denotes danger.

Even with interactive support, multiple color encoding for requirements is problematic. Figure 8.4 illustrates some of the issues. In the color portion of Figure 8.4, red denotes FEAR, yellow denotes RESENTMENT and blue denotes RELIEF. The saturation of each color denotes the intensity of the emotion: red and blue are at 100%, yellow is at 50%. In words, the goal is to transition the player's emotions from very afraid to very relieved while deliberately engendering a mid-level feeling of resentment toward the game designer - possibly to create a feeling of competition between the player and the designer.

The top color bar is for the transition from FEAR to RESENTMENT. The bottom color bar is for the transition for FEAR to RELIEF. The middle color bar is the luminance blend of the two transitions. For the color blend to act as an effective media for requirements capture and representation, all users of the representation must be capable of recognizing that the

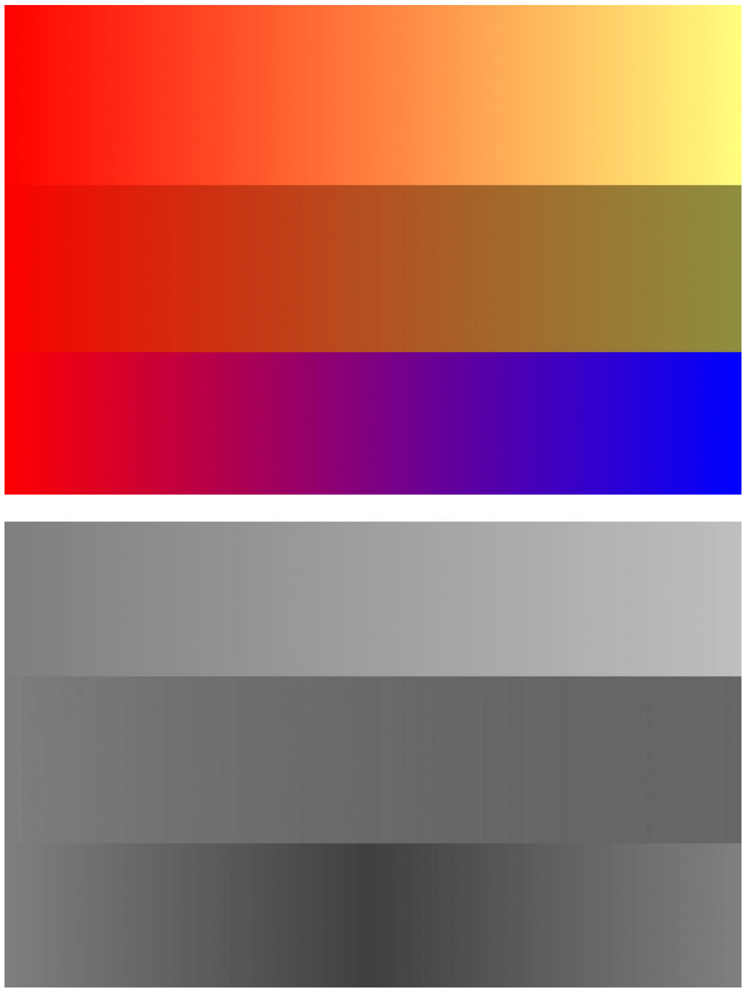


Figure 8.4: Color in emotional intensity maps

blended color at the right hand side of the middle bar of the image is a composite of 50% yellow and 100% blue. While this may be possible given that we have deliberately chosen primary colors for illustration, what happens when other colors are used?

It is reasonable to expect that, if every emotion is assigned a unique color, the team could come to recognize those colors – in isolation. However, given that there are only three primary and three secondary colors, we are less comfortable in believing that the combinations of the colors will be easy to recognize and translate to appropriately precise specifications.

The grayscale portion of Figure 8.4 shows the same transitions between emotions after they have have been converted to luminance. It may be dangerous to assume that the typical asset developer could successfully reason from the grayscale image back to the information contained within the color image. Further complications are illustrated in the bottom band of the grayscale image – there is a significantly darker band at approximately the midpoint in the transition. This occurs because we are collapsing three degrees of freedom (Red, Green, Blue (RGB)) down to one (Luminance, L). As a result, there is no longer a one-one mapping between the visual representation and the information it is meant to convey; many combinations of (RGB) map to the same value of L. Further, there is more than one algorithm for the conversion of color images to grayscale images which can also lead to problems with interpretation.

8.7 From Requirements to Design and Implementation

Prior to this study, our focus was on capturing the regions and on the emotions themselves. However, we now understand that it is the boundaries between regions that are of greatest import for the media production process. It is at the boundaries that some form of emotion marker must be placed to act as a trigger to induce the desired emotion state. The safe zones of Figure 8.3 imply that there are aspects of the virtual world that make these zones safe – most likely physical constructs that act as emotion markers. In Figure 8.5, a luminosity thresholding algorithm was used to identify possible locations for these emotion marker(s), identified here with a pattern fill, thereby providing the appropriate guidance for the media production team.

The game designer would typically explicitly identify the emotion markers, and their locations, during the requirements phase to ensure that the desired control over the artistic context is maintained. As a consequence of this specification effort, appropriate guidance is provided for media production from the beginning of the development process, potentially reducing the number of development iterations. However, the exact mechanism for implementing the emotion marker could be left to the production team – thereby allowing them to make their own creative contribution, within the constraints of the production budget and the artistic context.

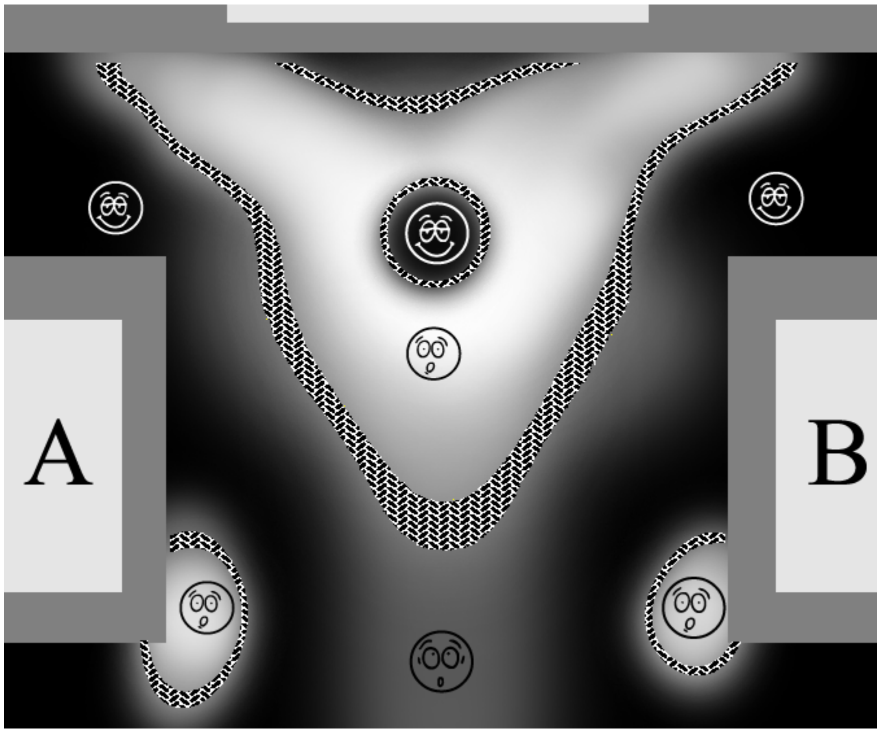


Figure 8.5: Sniper scenario emotional intensity map. Pattern fill indicates possible locations for emotion markers.

8.7.1 Difficulty and Emotional State

While *difficulty* is actually a gameplay requirement, the difficulty level has strong emotional interactions, engendering emotions such as FRUSTRATION and ACCOMPLISHMENT. Figure 8.6 illustrates how the game designer could, if there is a one-one mapping between emotion and difficulty, also indicate relative difficulty using an emotional intensity map. There are three gradients shown; in each case white denotes FEAR, black denotes RELIEF, and the player experience starts from the left and proceeds to the right within each gradient. This overloading allows the emotional intensity map to serve a dual purpose. The top gradient represents an easy path – the luminance of much of the path is closer to black than to white. The middle gradient is a relatively difficult path; only near the end of the path does the intended emotion substantially move from fear to relief. The bottom gradient represents a relatively complex, yet safe, path; a path with some obstacles and one point of greater perceived danger, but without significant elements of fear over most of the path.

The bottom path might represent, for example, the emotions of a player as they traverse a maze-like scenario.

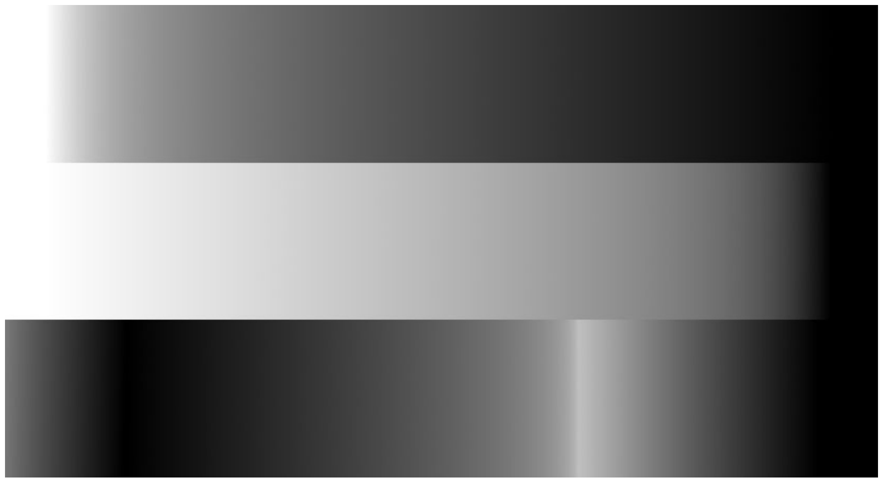


Figure 8.6: Gameplay difficulty in the emotional intensity map. White denotes danger, black denotes safety. From the top, an easy path, a difficult path, and a complex path.

Difficulty can also be controlled, to some degree, via the threshold setting for the technique of Figure 8.5. In this example, increasing the threshold value would move the potential locations for the emotion markers closer to the central safety zone. As a result, the Runner would remain exposed to the Sniper for more of the playing area and the Runner player would perceive this as increased difficulty.

8.7.2 Constraints on Emotional Requirements

A noteworthy characteristic of the sniper scenario is the lack of face-to-face, personal contact between the players (or rather, their avatars). While traditional cognitive psychology draws heavily upon the concept of facial feedback (the presentation of facial cues indicating the internal emotional state of an individual), video games do not always have this option. The low resolution of the player avatars and the lack of feedback paths between the players and their avatars in the world make facial feedback difficult; the game designer is deprived of this most-familiar interaction mechanism. In addition, there are many scenarios, such as the one studied in detail in this work, where one player cannot necessarily see or perceive the location of the other player. Finally, interactivity allows the game designer to introduce the faceless opponent – the rules embedded in the game engine can act as the replacement for a visible character, further complicating matters.

Therefore, constructing emotional requirements that include facial cues may be inappropriate, if not impossible. It may be better for the game designer to assume that they must design the emotional experience within the constraints of employing only environmental visuals, sounds, and observable actions within the virtual world.

8.8 The Final Product

The sketches used for the layout of the virtual world during the requirements process acted more as inspiration to the art department than as hard requirements and practitioners should be prepared to accept this behavior pattern. Figure 8.7 is a plan view of the final prototype from the artists and modelers, prior to the placement of the emotion markers, and is an innovative interpretation of the requirements created during the study. The black arrow in the image points to the danger zone for the Runner and the white arrows point to the Sniper positions in the three buildings.

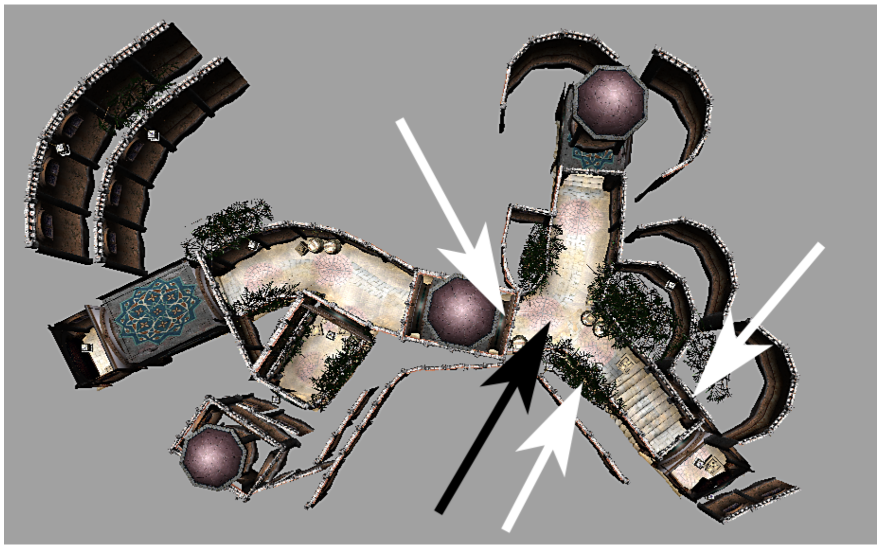


Figure 8.7: Late prototype, plan view. Black arrow indicates Runner danger zone, white arrows indicate possible Sniper positions.

In Figure 8.8 we see, from three perspectives, a late-stage prototype of the region of interest that is almost ready for play testing. The top row of images is the scene rendered with full illumination and no special effects. The bottom row of images is what the players perceive in-game, under low illumination and with active fog effects, a significantly different experience.

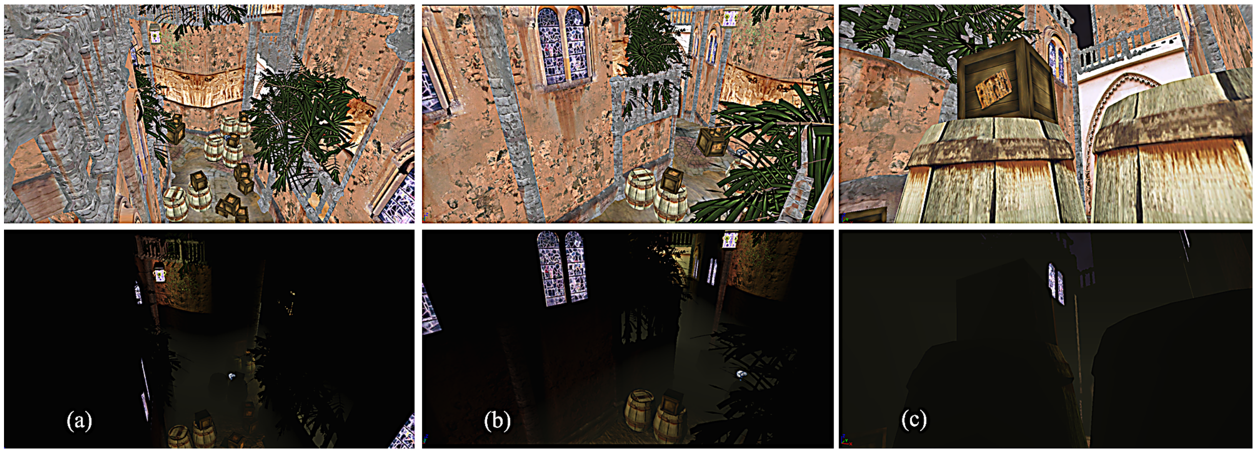


Figure 8.8: Three perspectives: (a)Isometric View, (b) Sniper View, (c) Runner View

There are numerous emotion markers in the scene. The brightly lit windows are cues and clues for the Runner, used to draw their attention to the source(s) of danger, while drawing their attention away from the barrels and boxes scattered about the street that promise a refuge, however brief, to the Runner as they attempt to escape attack from the Sniper positions. Column (a) is a view of the scene from an arbitrarily placed camera. Column (b) is the scene from the Sniper perspective in building C while column (c) is the Runner's perspective as they (carefully) look back toward the Sniper position of column (b).

8.9 Elicitation and Capture

In earlier work with emotional requirements, we were including the scenario concept information (a summary of the scene) within the emotional requirement. However, we found it useful to extract this information since it was shared by all emotional requirements within a given scene. The final form of the process is as follows.

1. Create the scenario concept. Capture a textual summary and a few sketches of the virtual world.

2. Iterate as necessary:

   (a) Define the gameplay experience. What actions can each player take, what assets can the players utilize, how can the players interact?

   (b) Define the artistic context, the virtual world, in sufficient detail that the definition can be given to the media department for implementation.

   (c) Define the emotional requirements using the suggested techniques.

   (d) Iterate as necessary, within the context of (cue, action, goal), to ensure that the desired player experience will be created:

      i. Evaluate interactions with the artistic context.

      ii. Evaluate interactions with the defined gameplay experience.

When evaluating the interactions with gameplay and artistic elements, we found that we were often presented with opportunities to define new interactions – in essence, inventing new requirements. After exploring the possibilities, we abstracted the following interaction patterns.

- Spatial: The location of the element within the virtual world. For example, spatial interaction patterns were typically based upon the relative position between players or between a player and an artifact in the virtual world.

- Temporal: The interaction pattern is dependent only on time, or on a (readily) discernible or deducible function that includes time.

- Engine attributes: Engine services such as collision detection and visibility calculation – the *physics* of the virtual world can be defined and manipulated.

- Game attributes: Those things that are unique or defining elements of the game; for example, the slow-motion effect now known as *bullet-time*, popularized in the *Matrix* movie series.

Spatial and temporal interactions can probably be captured by static visuals in the requirements process but engine and game attributes are more likely to require additional textual information.

8.10 Specifying Emotional Requirements

The complexity of an emotional requirement is greater than we realized during our prior work. An emotional requirement that provides sufficient information about the relationship between the emotional requirement and the virtual world, such that the needs of the game designer and the media production team are met, is specified as follows.

1. The intended emotion. Use of a reference list, standardized for the project or organization, is recommended (Section 8.5.1). The emotion can be situated within the virtual wold using an emoticon or other abstraction.

2. The artistic context (discussed further below).

3. The emotion prototype.

   (a) The cue (trigger), the emotion marker. The objects, animations, sounds, lighting changes, or other elements of the virtual world that are used to trigger the player's emotional response.

   (b) The action that the player is expected to take. This action is specified relative to the cue.

   (c) The purpose or goal of the player's response. The goal integrates the emotional requirement with the gameplay requirements and design.

4. A means, such as an emotion intensity map, to situate the emotion within the virtual world and to provide guidance as to the spatial relationships between the emotion and the virtual world.

5. A means, such as an emotion timeline, to provide guidance as to the temporal relationships between the emotion and the virtual world.

The original designation of the "artistic context" proved to be too vague for production purposes. While the artistic context describes the look and feel for a given setting (including such information as the period, genre, architectural style, the color palette used, the lighting conditions, and descriptions of the ambient sounds), this information was already available within the general artistic guidelines used by the media production team for each scene. Including the same information within the emotional requirement, rather than simply referencing the relevant artistic guidelines, may lead to greater specification maintenance costs as document versions must remain synchronized.

As in the current scenario, there may be multiple cues. If there are multiple cues, there may also be multiple goals. Further, some of the cues may be intended for emotional requirements and some of the cues may be intended for gameplay requirements. The requirements must clearly identify whether a given cue is related to an emotional requirement or to a gameplay requirement.

The game designer must be careful to avoid creating too many trigger requirements. The player may not recognize the triggers as *triggers*, which can lead to player frustration. If there are too many simultaneous triggers, the player may not be able to respond to them in a timely manner, leading to a far different experience than expected. Finally, triggers that are unused or malformed waste development and testing effort.

8.11 Conclusions

The study identified weaknesses in the prior definition of emotional requirements and emotional intensity maps that were impediments to adoption at the development studio participant. The addition of emotion prototype (cue, action, goal) information to the emotional requirements, and the explicit identification of potential locations for the cues (emotion markers), have addressed the known issues and the study participants expressed satisfaction with the results. Independent corroboration of the principles underlying our work was also identified in the work of Smith and other members of the cognitive psychology and film studies communities.

The use of color in emotional intensity maps was investigated. Techniques that support interactive queries show some promise but there is significant potential for misinterpretation by the user. Further, the need for archival storage in black and white print media is a significant barrier to the adoption of color in emotional intensity maps.

The study has elaborated an enhanced elicitation and capture process that better meets the needs of the production team. Additionally, four interaction patterns were derived then used to identify new opportunities for enhancing the player experience.

The action research methodology was well-suited to this problem. Engaging the game designer and media production team as an integral part of the research program helped to ensure that proposed solutions received timely critical feedback and progress was very rapid. Unfortunately, this rapid progress came with reduced control over the research agenda. While we feel that we effectively addressed the immediate issues with emotional requirements, we did not explicitly answer all of our motivating questions and the study would have been strengthened with more participants.

8.12 Future Work

A stronger understanding of emotion prototypes and emotion markers has the potential to further reduce risk in the production process. If we can identify libraries of emotion prototypes and emotion markers that are known to induce the desired emotional response, effectively identifying *emotion patterns*, then we can reduce the associated risks. However, such a library can quickly become recognizable by the target audience and, therefore, useless. In addition, any form of risk reduction must not impair the creative process or it will do more harm than good.

We have shown that we can readily identify the desired emotions from a reference categorization. However, we do not feel that we have sufficient control to discriminate between these emotions beyond some instances of the secondary emotions. Further work is necessary to identify techniques that will allow us to craft experiences that are defined in terms of all of the secondary emotions and even the tertiary emotions. Increasing the number of emotions simultaneously supported in an emotional intensity map would allow us to remain with a relatively simple, graphical representation for emotional requirements. Techniques that survive the transition between the virtual world and print media are most desired.

Developing extensions to one or more artists tools, level-editing tools, or storyboarding tools such that they provide integrated support for emotional requirements would allow us to broaden our research base and enhance adoption of the research outcomes.

Chapter 9
An Introduction To Experience Requirements

Knowledge developed over the course of the research effort led to the definition of *experience requirements* and an accompanying ontology of types of experience requirements in videogames. Experience requirements are descriptions of user, player, and customer experiences that must be met (functional experiences) or are satisfaction goals (non-functional experiences) for products or services. Experience requirements may be constructed using generally accepted requirements engineering principles and techniques or they may use less traditional techniques such as concept art or sound effect samples. Experience requirements are not software requirements, although they may result in software requirements or may be met by software artifacts.

Originally published as follows.

> David Callele, Eric Neufeld, and Kevin Schneider. Introducing Experience Requirements. In RE 10: Proceedings of the 2010 18th IEEE International Requirements Engineering Conference, Sydney, Australia, 2010. IEEE Computer Society [24].

Abstract: We consider the application of requirements engineering principles and techniques to the elicitation, capture, and representation of the output of the user experience design process. A stimulus-perception-response model is used to motivate experience requirements, defined as descriptions of user experiences that must be met (functional experiences) or are satisfaction goals (non-functional experiences). We identify potential benefits and look at experience requirements in video games.

Keywords: Experience requirements, user experience design, non-functional requirements.

tional Requirements Engineering Conference, September 2010.

9.1 Introduction

User eXperience Design (UXD) is the deliberate creation of one or more aspects of the user experience. An intersection of many schools of design, one could consider UXD a superset of HCI, industrial design, and the fine arts.

The field of Requirements Engineering (RE) is based, in part, upon the premise that if one sets out to design something (anything), it is best to know *a priori* what that thing is for and what it should do – *i.e* the requirements are defined. In this work, we present early results of our investigations into the intersection between UXD and RE.

9.2 Experience Requirements

We define the application of RE to UXD as *experience requirements*. Experience requirements are descriptions of user, player, and customer experiences that must be met (functional experiences) or are satisfaction goals (non-functional experiences), for products or services. These experience descriptions may be constructed using generally accepted requirements engineering principles and techniques or they may use less traditional techniques such as concept art or sound effect samples. We note that even though the customer experience is considered a basic tenet of product quality (for example, as *aesthetics* in Garvin's "Eight Dimensions of Product Quality" [47]), it is not addressed by the section on software quality in the ISO 9126 standard [59] (except, perhaps as an element of usability) despite the standard's attempts to be exhaustive.

It appears that a new model could be useful. We follow a relatively strict constructivist approach, rooted in classic engineering principles, when defining this model: we identify the domain of interaction, then use decomposition and refinement to create the model. Then, using a thought experiment process, we apply the model to a domain to see how well it works, possibly identifying particular strengths and weaknesses. To date, we have investigated elements of expressiveness – the ability to elicit, capture, and represent the user experience. The model's success as an expressive medium will determine whether deeper investigation is warranted.

9.3 A Model

Maintaining the constructivist stance, we assume a stimulus-perception-response model guides the design of the user experience: First, the desired user response is specified. A stimulus, that is (to be) perceived by the user, is then designed to engender the desired response. We note that the stimulus-perception-response model is a representation of the ways in which the designer can affect the user – informally, we could say via the emotions, the intellect, and the senses. We find that, for each element of this model, there are *tangible* and *intangible* elements, examples are noted below.

STIMULUS Tangible stimuli exist in the world around us – they can have physical and temporal aspects. Intangible stimuli affect our conscious and unconscious selves, cognitive and emotional responses are examples. While tangible stimuli exist in the four dimensions that we perceive in the world around us, we note that intangible stimuli allow an effectively infinite expansion of the dimensions.

PERCEPTION The user can only perceive the stimulus via their five senses. However, the stimuli may generate a meta-level perception. For example, the stimulus may be a block of text that is read by the user. The block of text actually contains a message to the user – it is this message that we want the user to perceive and not the block of text *as a block of text.*

RESPONSE The user may generate a tangible (observable) response. They may also have an intangible response such as learning a new fact, or entering a new emotional state; responses that we can not directly observe.

Experience requirements could be considered a type of non-functional requirement (even though our definition notes that experience requirements can capture functional experiences). Non-functional requirements are often referred to as the "-ilities": reliability, usability, maintainability, *etc.* Except for usability, even the most extensive investigations into non-functional requirements such as that by Chung [27] do not appear to address the intended user experience. It appears that the field of requirements engineering could take a more holistic approach to the user experience and that experience requirements may be a worthwhile addition to the domain.

9.4 Potential Benefits

Extending existing development practices to include experience requirements could provide a number of benefits. For example, any media or software element identified in an experience requirement is an element that must be implemented by the production team – experience requirements could reduce the risk that the existence of the element has only been inferred in the specifications or can only be identified by implication. Any element identified in an experience requirements is also a *mission-critical* element, necessary for creating the intended user experience, and the implementation of the element can be prioritized.

Design reviews are facilitated by the explicit identification of the critical elements and appropriate test plans can be devised earlier in the process and with greater certainty. Experience requirements can also provide guidance for play testing and player satisfaction testing. By more explicitly capturing the designer's intent for an experience, we enable greater certainty in design reviews, and the design and development of tests for both verification and validation. (For example, if the designer has specified that a particular use-case is expected to make the user laugh, then the test team can monitor users for the expected response.) The documented experience requirements may reduce production's dependence upon the designer's availability and we anticipate that the more structured representation could enable greater certainty in development planning, project estimation, and project scheduling. Finally, prioritizing experience requirements during requirements negotiation efforts may also lead to increased customer satisfaction, potentially improving the quality of the user experience.

Focusing on experience requirements during product conceptualization and design means that experience requirements will be captured before other (more traditional) requirements (*i.e.* functional or non-functional). We expect that this temporal precedence will tend to subordinate the traditional production requirements during requirements negotiation activities.

9.5 Experience Requirements in Video Games

We have been exploring the use of experience requirements as a mechanism for capturing the game designer's vision for the player's experience. To date, our work has shown that the user experience for this domain, when captured as experience requirements, will address one or more of the following items:

1. Emotional experience
2. Gameplay experience
    (a) Cognitive experience (*e.g.* puzzles or quests)
    (b) Mechanical experience (*e.g.* command sequences, combat 'combos')
3. Sensory experience
    (a) Visual experience
    (b) Auditory experience
    (c) Haptic experience (if available)

Our prior reported work focused on the emotional experience, represented by emotional requirements [17]. We have found that our emotional requirement techniques have been effective for capturing the intended player experience in a side-scrolling platform-jumping game, in a racing game, and in multiple scenarios within a first-person-shooter game. There are some restrictions: our visualization techniques only support transitions between one emotion at a time (for example, from happy to sad and do not support, for example, transitions from a mix of anger and fear to sad. Our current work is addressing the gameplay experience. While results to date are showing promise, the complexities of capturing and representing cognitive gameplay experiences are greater than anticipated.

9.6 Conclusions and Future Work

Our initial results with experience requirements indicate that it may be possible to use experience requirements in user experience design efforts. Experience requirements apply requirements engineering techniques very early in the typical product definition process and, as with all methodologies, their application to the creative phase should be carefully managed to ensure that creativity is not negatively impacted. Initial results in the video game domain show promise, but the complexities are greater than anticipated. Further work appears warranted, based on results to date.

Chapter 10
A Proposal for Cognitive Gameplay Requirements

In another informal field study with Far Vista Studios, we participated in a preproduction effort response to a third-party request for proposal for a massively multiplayer online role playing game (MMORPG) situated in ancient Egypt. The work reports considerable detail on the design process, both on the concept and on the details surrounding traps, puzzles, and penalties of gameplay components. However, the experience confirmed again the problems of capturing sufficient information during preproduction, the practical technical problems (for example, rendering load) that must be addressed and the complexity of the experience issues (for example, planning for, and measuring, the enjoyability of repeat play).

These field observations helped us to identify elements that are necessary to elicit, capture, and represent during the requirements specification activity for cognitive gameplay requirements. To facilitate the expression and discussion of the elements of cognitive gameplay requirements, these are presented as a definition. Consistent with the exploratory nature of this work, this definition is not formal in a semantic sense, nor is it complete. Confirming the observations with other teams is necessary before the results can be generalized.

Originally published as follows.

> David Callele, Eric Neufeld, and Kevin Schneider. Cognitive Gameplay Requirements. In MERE 10: Proceedings of the 4th International Workshop on Multimedia and Enjoyable Requirements Engineering, Sydney, Australia, 2010. IEEE Computer Society [23].

Abstract: In cognitive gameplay, players must identify inputs, classify and integrate them in a contextually appropriate manner, then draw conclusions and provide feedback to the game engine to demonstrate their mastery of the challenge. Established requirements practices do not exist for this domain and game development teams rely upon *ad hoc* approaches to specification and iterative requirements-through-implementation-and-test techniques to achieve their goals.

In this work we report our observations of a game development team as they prepared a game design in response to a third-party commercial request for proposal. We report upon three examples of cognitive gameplay definition and propose a definition for cognitive gameplay requirements, capable of capturing the requirements from within the case study, that can be used as the basis for further investigations.

Keywords: Experience requirements, design requirements, non-functional requirements, gameplay requirements, cognitive requirements, videogame.

10.1 Introduction

Game development is typically a two phase effort consisting of iterations between a preproduction phase in which the game is designed and elements prototyped followed by a production phase in which the game is implemented. Production is guided by a game design document, the output of the preproduction efforts that focuses on telling the story behind the game and describing the game itself (the look and feel, the gameplay). The game design document does not usually explicitly elaborate all of the details of the intended player experience, particularly with respect to how the player is intended to feel as the game progresses. Details of the intended experience tend to be communicated verbally, on an as-needed basis during iterations of the production effort.

In prior work, our analysis of post-mortem project reports from the video game industry showed that game development is difficult; the two phase, multi-disciplinary task is complex and fraught with opportunity for error [16]. We posit that focusing on mechanisms for defining and capturing the player experience will lead to improvements in the preproduction process and in the transition from preproduction to production, reducing, for example, the threats associated with implication in communication [16].

We define *experience requirements* [24] as descriptions of user, player, and customer experiences that must be met (functional experiences) or descriptions of satisfaction goals (non-functional experiences), for products or services. We believe that, in the video game domain, defining and capturing the intended player experience as experience requirements that are influenced and informed by established requirements engineering principles and techniques will help practitioners bridge the communications chasm between preproduction and production. Our goal is to extend our work on requirements in videogame development into a more general experience requirements methodology composed of:

1. a model for the elements that compose experience requirements,
2. a framework that provides guidance for expressing experience requirements, and
3. an exemplary process for the elicitation, capture, and negotiation of experience requirements.

that can complement and extend traditional requirements engineering techniques such as goals and scenarios.

We developed the following ontology of types of experience requirements for the video game domain based on the interactions between what the underlying game system can deliver as part of the experience and what the player can sense and internalize.

1. Emotional requirements (the heart)
2. Gameplay requirements (the intellect)
    (a) Cognitive (the head)
    (b) Mechanical (the hands)
3. Sensory (the senses)
    (a) Visual (the eyes)
    (b) Auditory (the ears)
    (c) Haptic (if available) (touch)

In prior work, we focused on capturing and representing the intended emotional experience for the player [17] via the emotional requirement. In the current work, we turn our attention to issues associated with gameplay requirements. We provide a definition for gameplay requirements and review the related work then look more closely at cognitive engagement in games. We report our field observations of three gameplay definition examples from an industry case study then present our proposal for the elements that compose cognitive gameplay requirements. Each element is accompanied by an example from one of the gameplay definitions. We conclude with final comments and suggestions for further work.

10.2 Gameplay Requirements

We define *gameplay requirements* as requirements that identify, capture, and represent those elements critical to crafting the intended gameplay experience. These requirements include elements as diverse as game rules, commands for various actions, sequences of actions that the player must master for success, and puzzles and their associated clues. In traditional requirements engineering terms, experience requirements encompass both functional and non-functional aspects – even though most practitioners would likely consider gameplay requirements a type of non-functional requirement. However, within the context of a game, one can argue that gameplay requirements are the functional requirements for the game itself. The interested reader can also review our prior work on the interactions between emotional requirements and security requirements [20] which provides example scenarios where emotional requirements can dominate and override security requirements – even after the game has been released.

To simplify our analytic efforts, we choose to classify gameplay requirements into mechanical and cognitive aspects. This physical-intellectual classification roughly follows the two dominant gameplay styles. While we choose to classify these requirements into two categories for our research, they are synergistic in practice and do not exist in isolation from each other.

Gameplay requirements are not expected to be formal in the mathematical sense, at least as developed by the game designer. Rather, gameplay requirements are descriptions of the intended player experience. By more explicitly capturing the game designer's intent for an experience, we expect greater certainty in design reviews, project estimation, play testing, player satisfaction testing, and test design and development for both verification and validation.

In mechanical gameplay, the focus is on mechanical operations upon the game controller that players must perform in response to visual and auditory stimuli. Games where one plays simulated instruments, performs dance moves, racing games, and the combat elements within many games, all emphasize mechanical gameplay.

The cognitive aspects of gameplay include facts and rules about the game world and the game challenge(s) faced by the player. In cognitive gameplay, players must identify inputs, classify and integrate them in a contextually appropriate manner, then draw conclusions and provide feedback to the game to demonstrate their mastery.

Quest and puzzle games emphasize cognitive gameplay. For example, in a typical puzzle from a quest game the player must interact with non-player characters within the game to obtain clues as to the locations of items within the game world and the purpose to which these items are to be put. Locating, identifying, and obtaining these items are supporting cognitive puzzles within the larger context of the cognitive puzzle of determining the motivating purpose behind the items. Finally, the player must also solve the cognitive puzzle of how to successfully utilize the items for their intended purpose in order to gain their reward. Successfully structuring these puzzles is challenging (see [16] for examples of the difficulties faced) and the cognitive gameplay requirements described in greater detail in Section 10.6 are aimed at addressing some of these challenges.

10.3 Related Work

The trade press associated with game design is rich with pragmatic advice. Game designers like Rollings and Adams [93], Crawford [31] and Koster [61] and academics like Salen and Zimmerman [95] present their perspectives on a field that is generally considered to be more of an art than a science. Each author presents their perspective on the act of game design, but none of the authors comments on software engineering processes that could support the activity. The anthology of project *post mortem* reports presented by Saltzzman [96] provides significant anecdotal evidence of the issues involved in video game production. In prior work [16] we analyzed these reports and concluded that there were significant issues associated with capturing the game designers's vision and communicating it to the production team. The anthology of commentaries by well-known industry professionals compiled by Laramee [64] also provides further insight into video game production. Despite the breadth of these works, none of the authors advocates a structured approach to capturing gameplay as requirements or utilizing requirement engineering principles.

In general, the work in the requirements research literature is not strongly related. A traditional perspective on requirements is likely to consider cognitive gameplay requirements to be some form of non-functional requirements. In his analysis of non-functional requirements, Glinz [50] notes that there are significant issues with defining, representing, and classifying non-functional requirements. He proposes a solution based on the concept of a *concern*, defined as "a matter of interest in a system". A concerns-based taxonomy is presented, along with a series of questions that can be applied by the practitioner to guide them in applying the taxonomy. It is unclear whether this taxonomy covers, for example, cognitive gameplay requirements or emotional requirements. While "matters of interest", whether they qualify as 'concerns' would depend upon whether one accepts cognitive gameplay as an appropriate target for requirements efforts.

The preproduction requirements that define the player experience are conceptually more like *design requirements*, as discussed in the collected works of "Design Requirements Engineering: A Ten-Year Perspective" [79]. For example, Loucopoulos and Garfield [65] describe requirements engineering practices and principles within the context of the overall enterprise strategy. They note that the interaction between strategy and requirements contains elements of co-design and co-development, and that maintaining the "designing stance" described by Gehry [48] is critical when considering requirements in this domain:

> The term designing stance is used in this chapter to mean that the process should involve reflection, exploration, negotiation, compromise and revision. It seems that these are the activities in which top class designers engage when considering complex projects in uncertain situations. [65]

We shall see in Section 10.5 that the observed behavior patterns in preproduction for video game development are similarly suggestive of those reportedly observed in co-design and co-development efforts.

Scacchi [98] investigates a number of open-source software development efforts, including an example from the *mod*[1] community for first-person shooter games. He notes that the requirements engineering challenges in mod development are significant: the aspiring mod developer must harvest information from many sources that tend to present technical information as narratives in order to set their requirements within the appropriate context. The mod developer must also deduce the requirements for the existing game infrastructure. He also notes that challenges associated with creating a viable mod are not just technical, but social as well, and that the expectations of the player community, as stakeholders, must be met if the mod is to be a success. These observations are consistent with our own observations of both open-source and commercial game developers over the years but the work does not provide any direct guidance applicable to cognitive gameplay requirements.

Finally, Aoyama [6] investigates the use of personality constructs (*personas*) as surrogates for unknown stakeholders in the context of mass-market consumer electronic devices. These personas may be useful in the context of videogame development, particularly when evaluating requirements and designs for acceptance by the target market. In the current context, if the personas included information about the cognitive skills of the target market then they could be used in puzzle design and in requirements validation and verification efforts.

In summary, the related research work is sparse and only generally related to the focus of this work.

10.4 Cognitive Engagement in Games

Cognitive gameplay requirements are a mechanism for capturing the designed and intended cognitive engagement for the player. To set the context from the perspective of the game designer, we present comments from Raph Koster's introductory critical analysis of the

[1] *mod* – for *mod*ification, or extension, of the video game through the use of scripting, changes to artwork and animation, rules, *etc.*

cognitive engagement process in “A Theory of Fun” [61]. A leading game designer, Koster [2] maintains that the primary motivator for the cognitive engagement between the player and the virtual reality created by the game designer is the learning process:

> Fun is primarily about practicing and learning, not about exercising mastery. [p.96]
>
> Once you learn something, it’s over. You don’t get to learn it again. [p.126]
>
> The definition of a good game is therefore “one that teaches everything it has to offer before the player stops playing.” [p.46]

A game designer wants to keep the player learning about the cognitive elements of the game for as long as possible (and have the player desire to continue learning) because when the learning is done, much of the motivation for remaining cognitively engaged with the game is gone (as compared to improving the performance of mechanical gameplay elements over numerous practice and training sessions).

Koster follows a relatively constructivist learning philosophy [35] when he asserts that the player’s cognitive engagement is driven by the brain seeking to identify patterns. It follows that the game designer should be able to explicitly identify all of these patterns as part of the requirements process. But it is still unknown as to whether they *need* to do so and this will be the subject of future work. In the next section, we investigate the manner in which one game designer addresses the issue and in following sections present a model for identifying the necessary elements for capturing cognitive gameplay requirements.

10.5 Field Observations of Gameplay Design

In this section we present slightly redacted elements of a preproduction effort by Far Vista Studios in response to a third-party Request-For-Proposal (RFP) for a Massively Multiplayer Online Role Playing Game (MMORPG). We report here on our opportunity to observe a game designer as they designed three cognitive gameplay elements (puzzles) within a given scenario. The observations were gathered during approximately 15 hours of meetings, held on three separate days across a two week period in a meeting room at the game company. In

[2]While Koster’s theory is not the only one available, it appears to work well within a requirements engineering framework.

addition to the cognitive gameplay elements reported here, the participants generated many concept sketches and exerted effort developing the story behind the game.

Any sketches developed by the game designer and presented here were redrawn by the first author to protect certain confidential information; an effort was made to capture the look and feel of the original diagrams. The preproduction process is reported in chronological order to give the reader a sense of the evolving effort and result. We use the results of our analysis of the process that they followed and the output they produced to formulate a definition for cognitive gameplay requirements.

As part of the response to the RFP, the game designer generated a gameplay scenario to meet the following requirements.

1. The scenario must require the player(s) to solve one or more puzzles.
2. Gameplay is located in a desert setting.
3. The puzzle(s) must support play modes for individual and team play.
4. The player navigates their avatar through the world using a click-on-destination paradigm: The player clicks on the destination and the player's avatar automatically moves to that location, traversing the virtual world in a context-appropriate manner.

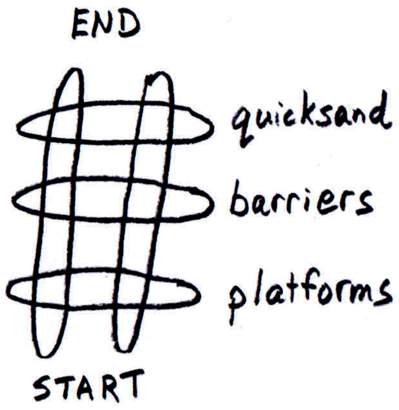
The overall artistic context was set by the third-party, but only in the most general sense: Egypt, in the time of the Pharaohs.

10.5.1 General Design

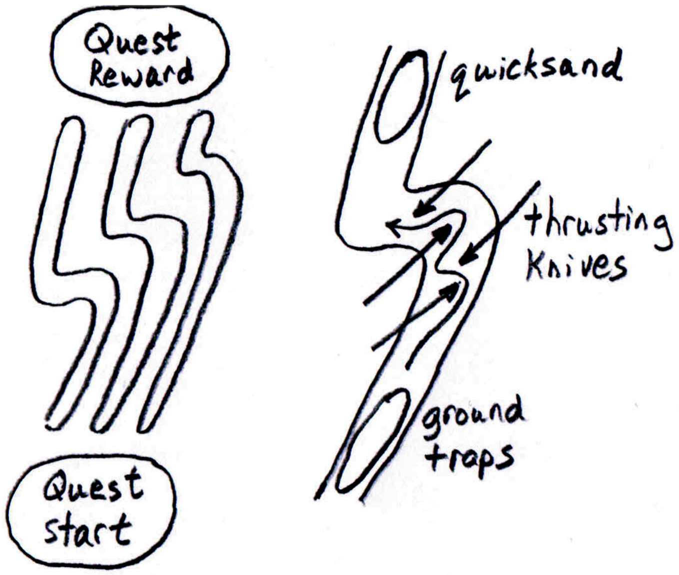
The game designer approached the puzzle design in a relatively ordered manner. At the beginning, the game designer quickly sketched a plan view of the environment for a single player puzzle (Figure 10.1a).

The environment contained the start location, an end location, and three experiential regions bounded on the sides by barriers.

The first experiential region was identified as a moving platform maze. The second experiential region was identified as a series of moving barriers that significantly changed



(a) Desert Puzzle concept sketch 1



(b) Desert Puzzle concept sketch 2

Figure 10.1: Desert Puzzle concept sketches

the apparent length of a player's path. The third experiential region was identified as a series of quicksand traps.

The designer then created a revised version of the puzzle layout, shown in Figure 10.1b. Comments were added to the diagram to provide further guidance to the production team and as reminders to the game designer (not shown on this simplified diagram).

Note that the shape of the path was changed to an exaggerated **S** shape. When asked to explain, the game designer stated that it was necessary to restrict how much of the scene the camera could see at one time in order to keep the scene rendering rates acceptable. The game designer used the rock barriers on each side to act as artificial clipping planes to manage scene rendering complexity – a case of implementation constraints being fed forward to the conceptual design phase of preproduction and directly impacting the creative process.

The game designer then turned their attention to the scoring/reward structure for the scenario. Only two notes were made. The first note dealt with the penalty structure: what happens when the player fails to solve the puzzle? In this case, the note stated that the player was simply sent back to the starting position for the scenario. The second note dealt with scoring: how do players receive feedback about their performance or compare their performance with others? The note stated that this was a combination race/accuracy test – the lowest number of moves wins.

Further design details, elaborated in the section associated with each puzzle, were added and the game designer completed this revision of the game design document – dominated by annotated sketches of this type with relatively little prose. A design review meeting was held a few days later with representatives of the production team, production feedback was received by the game designer, design changes were discussed and verbally agreed upon, and a revised version of the preproduction design was promised. Our direct observations ceased at that time but we were assured that the same process would be followed in subsequent iterations.

10.5.2 Observations on the Review Meeting

The review meeting was informative for our purposes because it clearly identified numerous instances where the current revision of the game design document was insufficient to the needs of production. It also identified numerous instances where an internal review, by the preproduction team, of the elements of the game design against known production constraints would have made much of the meeting unnecessary (thereby saving meeting costs and reducing review efforts). For example, questions like the following are typical (these questions are highly abstracted; the observed questions were more specifically focused):

1. *Is the rendering load budget met? What can be built within the polygon-count restriction?* The production reviewer is evaluating the requirements, attempting to identify performance constraints. It is important to note that the effect of performance constraints upon algorithm design in video games is significantly different from that in many productivity applications. It is often unnecessary to have an algorithm that delivers an absolutely correct answer for many problems (An analogy: In a videogame, a phonetic spelling may be acceptable, whereas in a spelling-checker, the spelling must be correct.). Instead, iterative algorithms that converge upon an acceptable (good enough) solution are often used and are, in some cases, the only viable means of managing computational loads. Production reviewers were actively looking for this issue

and identifying high-risk areas.

2. *Is the gameplay repeatable?* Repeatability is necessary for customer satisfaction, otherwise the player does not feel like they are making progress. The game designer must ensure that there is consistency within the game world in order to maintain the player's sense of immersion.

   Testing for repeatability requires identifying gameplay pre-conditions and post-conditions *and* ensuring that all other elements are ignored. Emergent behavior is particularly difficult to manage, especially if it is the result of unintended interactions between subsystems.

3. *What do we have to build to support this [concept]?* The production team is investigating overall project feasibility and performing rudimentary project management. Estimates of production effort can reduce waste in preproduction efforts – before committing excessive resources to refinement and decomposition activities in preproduction, ensure that there is an acceptable probability that the concept will make it past the requirements phase.

These review meetings are highly interactive, with many differing perspectives, handwritten notes, and verbal commitments. There are few (if any) formal minutes and traceability is very difficult. However, how much traceability is necessary remains an open question. This is a relatively small group, with the major design decisions effectively dictated by the designer, and the triumvirate of designer, director and producer shares near absolute authority and responsibility in their respective domains. With a larger development team, particularly if geographically distributed, we expect that traceability would be more important.

10.5.3 Gameplay 1: Platform Maze

This puzzle is simple in concept and a sketch of the platform puzzle region was developed *in situ* (Figure 10.2). The player must traverse a maze to be able to proceed onward in the game. However, there is nothing obvious that makes this region a maze. Visually, it is simply a flat region between the rocks – until the player tries to cross it. Then, the region becomes a series of platforms that may suddenly drop down beneath the level of the remainder of the region. If the player is located on a platform as it drops down, their avatar is transported back to the beginning of the area and forced to start again.

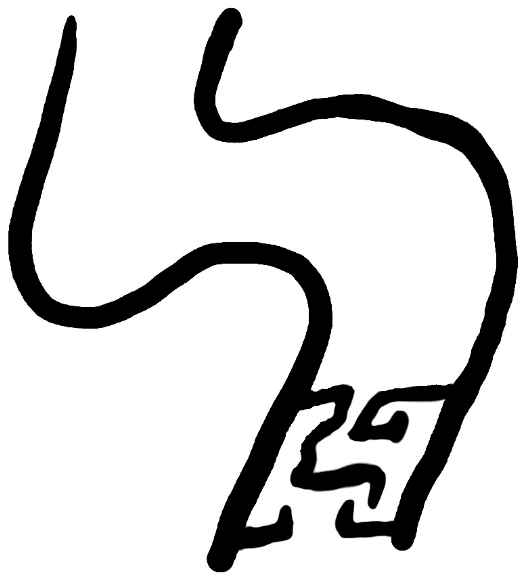


Figure 10.2: Platform maze concept sketch

The game designer noted that there are many ways to modify this cognitive puzzle. Examples include changing the rate at which the platforms descend to allow the alert player time to attempt to escape the platform, changing the mobility of the player, modifying the location at which the player is forced to restart, and modifying the time delay (penalty) before the restart occurs. Combinations of these, and other modifications are also possible. These observations may illustrate the need to ensure that the requirements process pays particular attention to exposing the attributes that control the gameplay experience and to facilitating their ongoing modification as development progresses.

It is the combinatorial explosion of the combinations made possible by the gameplay attributes that can lead to high re-playability. However, it can also lead to unexpected emergent behaviors that can put the integrity of the gameplay experience at risk.

Figure 10.3(a) provides a view of the first detailed description of the platform maze.

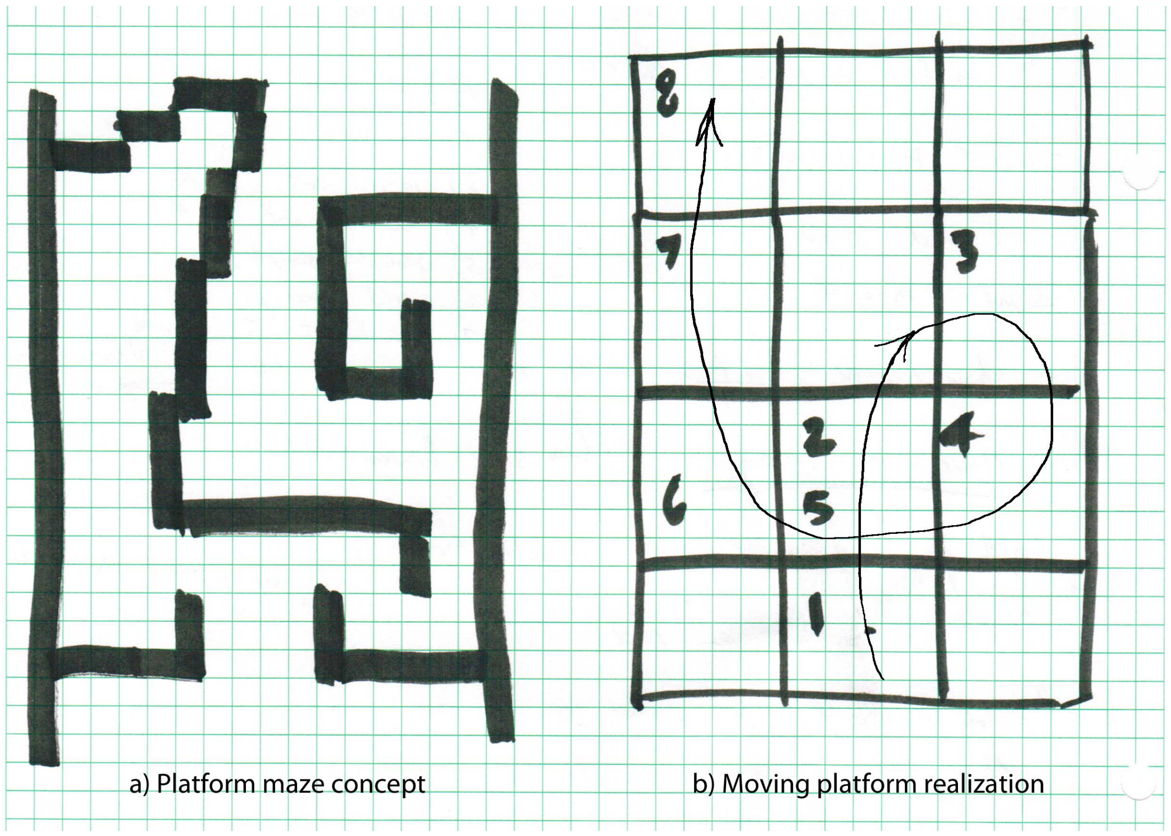


Figure 10.3: Platform maze design sketches

Each of the squares in the grid underlying the image can be considered a "platform unit". The puzzle area is approximately 17 units wide by 24 units deep – 408 platform units in total. Only those platform units without black fill are actually capable of motion. The black regions denote invisible barriers that constrain the player's motion; the player must navigate the maze using the visible walls as reference points.

Review of the design by the preproduction and production teams raised concerns about development costs and computational complexity. The teams investigated implementation strategies other than moving platforms but none delivered the desired player experience.

The game designer then proposed the alternative presented in Figure 10.3(b). There are no invisible barriers in this alternative and there are only 12 platform units in total, all of which can move. In this version, the game designer explicitly identified the path that the player must follow to successfully traverse the region.

Capturing this puzzle design as a set of cognitive requirements is facilitated with appropriate use of visualizations. For example, Figure 10.3(b) captures the number of puzzle elements and the proposed puzzle solution. However, this visualization needs to be augmented with significant further details. These details include identifying the fundamental building blocks of the puzzle (the moving platforms) and their characteristics, the clues that the player will receive (including the spatial and temporal locations of the clues) and what the player should learn from these clues (for guiding verification efforts). It should also include information about degree of difficulty, and interaction with mechanical gameplay requirements (such as the speed at which the player can command their avatar *vs.* the rate at which the platforms descend).

The final preproduction document for the game design was not significantly more detailed than Figure 10.3(b). Approximately 15 point-form notes were made to accompany the description given above. The general form of the process was to describe the experience then refine the design with the assistance of the other team members, particularly employing knowledge of known production constraints. Many of the details reached (undocumented) consensus through discussion or were left to the production team to resolve at the time the puzzle elements were implemented and play-tested. From a classic, productivity-application-oriented requirements engineering perspective, the system was severely under-specified.

10.5.4 Gameplay 2: Sliding Walls

The sliding walls puzzle (Figure 10.1b) is composed of sliding barriers that impede the player's forward motion. The barriers come out of the side walls and cross the player's forward path. By observation, the player can deduce that there is an interlocking pattern to the barrier paths. The player can pass through the region by traversing from side wall to side wall in a serpentine path, slowly advancing toward their destination but costing them valuable time in their race to the finish. The game designer wants the player to experience fear when in the path of any of the barriers; as the barriers move in and out of the walls the accompanying sound effects should exude a sense of menace or danger.

Under close examination, the player can identify a symbol etched into the side wall at the end of travel for the first barrier. If the player clicks on the symbol with the pointer, the barriers retract into the walls, leaving the path clear for a limited time, a time sufficient for a player to traverse the danger region if they react quickly enough.

We note that the middle experiential region is initially captioned “barriers” (Figure 10.1a) and later annotated to “thrusting knives” (Figure 10.1b). How did the requirement for “barriers” become “knives”, and why?

The answer lies in a production constraint. The game designer explained that if the puzzle was left with sliding barriers, like hidden walls that slide out of the rocks to block passage, then the game engine must support the case where the player avatar is stationary and in the path of the leading edge of the sliding wall. In this case, when the wall touches the avatar, the avatar should be pushed along the ground in a believable manner. The believability requirement would require either a physics model for the character (and the world) to force translation along an appropriate vector or the introduction of some kind of special effect to knock the character out of the way. The alternative, knives, simply kill the character. Avatar death and re-spawn (forcing the avatar to restart at a re-spawn location) is a well-established videogame paradigm and as such, is deemed a ‘believable’ alternative (to the sliding barriers) that can be utilized by the game designer to achieve their experience goal at significantly lower production cost (yet another implementation constraint). We note also that knives are more in keeping with the emotional states expressed in the puzzle description – changing from barriers to knives is a refinement of these experience requirements that addresses a realization cost constraint.

10.5.5 Gameplay 3: Shifting Sands

The shifting sands puzzle (Figure 10.1b) is similar to the moving platform puzzle at the start of the scenario (Figure 10.2). The player must traverse an invisible maze that is full of quicksand traps. The ground is not composed of platforms; instead if the player steps into a trap they slowly sink out of sight, swallowed by the shifting sands.

To differentiate between the two puzzles, the game designer allows the player to toss inventory items onto the sand to help probe their way. If the inventory item lands on a quicksand trap, it will be swallowed by the shifting sands rather than the player. Once the item is swallowed by the sands, it is lost forever – therefore the player must not waste valuable items by using them as probes and the game must contain items that can support this design.

During review with the production team, numerous options were discussed such as special effects and total number of quicksand traps. The topic that caused the most concern was inventory management. Questions were raised about the need to create multiple copies of certain items that would now have to be ‘throw-away’ and serve no other purpose than probing the shifting sands. Rather than confusing the player even further, the game designer suggested that one of the inventory items be a bag of figs that could be tossed out, one at a time.

As a result of the design of this puzzle, the implications and repercussions are many. Some of the comments, concerns, and questions include:

- The inventory item class of game elements must now support sub-items and quantity management.
- The player must be able to add and extract sub-items one at a time. Is extracting n items at a time also supported?
- The sub-items require independent models. What fidelity is required?
- Must we provide visual feedback to the player (can they look in the sack of figs)?
- Can the player inspect the container (bag of figs), perhaps to get the quantity remaining in the sack?

Such a simple concept, with such significant production consequences.

10.5.6 Summary

There were many more requirements, such as sensory requirements for the look-and-feel of the game world and the mechanical gameplay requirements for the controllers, that were developed during the puzzle design phase but these requirements were not developed within a framework informed by requirements engineering principles and practices. We observed elements of co-design and co-development behavior throughout the process *e.g.* the application of engine and production constraints on the game design, by the game designer, in response to production feedback in review meetings.

The strong impact of production needs and opinions, particularly with respect to feasibility, scope, and testability, is considered in our definition of the elements of cognitive gameplay requirements in the next Section.

10.6 The Elements of Cognitive Gameplay Requirements

In this section, we elaborate the elements that we have identified in our field observations as necessary to elicit, capture, and represent during the requirements specification activity. To meet the needs of practitioners in the videogame domain, cognitive gameplay requirements should be lightweight, situated within existing workflows, and must not unduly disturb the highly creative, highly iterative workplace. Many of the defined elements exist to ensure that the production team can develop appropriate design, verification, and validation strategies.

To facilitate the expression and discussion of the elements of cognitive gameplay requirements, we present them in the form of a definition, however, we note that this definition is neither formally correct nor necessarily complete and is used only as a matter of convenience. This work reports on experiences with a single development team; confirming the observations with other teams is necessary before we can claim to be able to generalize these results.

While we do not explicitly capture the following information in cognitive gameplay requirements, we must remain aware that game design operates on two levels. The first level is the software artifact that implements the functionality that presents the cognitive challenge. The second level is the *game* part of cognitive gameplay. Particular elements of the virtual world are overloaded with meanings contextually significant only within the context of the cognitive gameplay. Gameplay occurs in the interaction between these players and these contextually significant elements (such as clues or weapons). The remaining elements are part of the context for the gameplay and do not directly contribute to gameplay.

We shall continue to use the learning paradigm as espoused by Koster [61] – we shall speak of the player learning a lesson or solving a puzzle, attempting to solve a cognitive challenge of some form. This is a matter of convenience, and does not affect the definition. For example, the player may need to solve a puzzle by identifying a path through a maze (Section 10.5.3) or via manipulation of in-game artifacts in order to continue to progress through the game (Section 10.5.4). The cognitive challenge can be relatively passive, such as observation only, or relatively active (guide the avatar through the maze) – it does not appear to be necessary to discriminate across the range of activities.

Given our current knowledge, we define the i^{th} cognitive gameplay requirement CGR_i as a vector composed of elements of three types:

1. Pre-Conditions

2. Cognitive Challenge

3. Post-Conditions

such that

$$CGR_i \quad = \quad < Pre_i, Cog_i, Post_i >$$

We choose a vector representation to allow the user to specify the gameplay requirements with as many elements of each type as they feel is necessary, but this decision may change with greater experience.

We define each element of the vector in turn. The presented definitions are pragmatic, constructed in a manner that reflects the observed practices, and may be modified to meet the needs of a given project.

We now present an example to help understand why we do not yet see a need for the definition to be mathematically optimal or mathematically correct. Imagine that one part of a cognitive gameplay scenario requires that the player knock down a brick wall. Further, as the bricks tumble down, their paths are probabilistically determined. One of the elements that forms the post-condition is the Player State. Another element of the post condition is

Side Effect. As software developers, we would generally expect any side effects to be captured by the world state, yet we have chosen to separate the two aspects. We choose to capture the player's success or failure at the task of knocking down the wall via an attribute in the Player State. However, since the final configuration of the bricks is probabilistic, it is unrealistic to expect the game designer to specify the location and orientation of every brick in the wall. Instead, we note that there is an expected side effect: that the wall collapses in an acceptably realistic manner and that the final configuration of the bricks, whatever that may be, is also acceptable as long as the final configuration is also acceptably realistic. Loosely, one could consider the Side Effect as a quality requirement compared to the functional requirement for success or failure at the task of demolishing the brick wall.

For each element of the definition, at the end of the definition we give an abbreviated example (in italics) of the captured requirements information within the context of the Sliding Walls puzzle in Section 10.5.4.

10.6.1 Preconditions

Cognitive gameplay requirements are an integral part of a learning exercise that is designed to challenge the player, a learning exercise that the player is expected or required to master. We must ensure that the player has the necessary elements in place to be able to address the cognitive challenge.

A clear definition of the preconditions for a cognitive challenge helps to ensure that the production team constructs the necessary assets and that the software team can develop appropriate design, verification, and validation strategies. The preconditions for a cognitive gameplay requirement are defined as follows.

1. *Assets:* Those specific elements of the game world that can be perceived by the player and that are necessary components of the cognitive challenge. Assets may include assets that the player has accumulated in their inventory, visual elements, auditory elements, and in some cases, haptic elements. *Sliding walls, side-wall symbol, sounds of wall sliding.*

2. *Clues:* The cognitive-level meaning associated with assets in the game world; a class of

assets that have special meaning to the gameplay and are not just part of the *ambiance.* For example, a sign in the game world can advertise the location of a item needed in a quest. *The sliding walls are a barrier to progress. The symbol etched into the side-wall disables the threat.*

3. *Game Infrastructure:* Hardware or software elements that the player must have, such as specialized controllers or subscriptions to pay-to-play services. *No elements specific to this cognitive gameplay requirement.*

4. *Player State:* Player-specific attributes that the game engine is tracking, controlling, and manipulating. Specific attributes, such as health, skills, or puzzles successfully completed, and their values, are typical. *Player must have successfully completed the Platform Maze puzzle.*

5. *World State:* Attributes that the game engine can track, control, and manipulate, other than those of the player. *No elements specific to this cognitive gameplay requirement.*

6. *Puzzle State:* This may not be the first time that the player has attempted this cognitive challenge. Records the puzzle state as a consequence of attempting the cognitive challenge. It is critical to identify positive, negative, and intermediate outcomes for testing purposes. *Current puzzle state = Old puzzle state.*

7. The following terms are optional, but recommended. There may be significant costs associated with managing these items. *As per description in Section 10.5.4.*
   (a) Description of the game world context that the cognitive gameplay requirement expects to exist.
   (b) Link to narrative (backstory); can help in the design of test routines.

such that

$$\begin{aligned} Pre_i &= < Assets_i, Clues_i, Infrastructure_i, \\ & \quad PlayerState, WorldState, PuzzleState, \\ & \quad [Context_i, Narrative_i] > \end{aligned}$$

10.6.2 Cognitive Challenge

We describe the cognitive challenge in terms of a learning exercise. The player is required to observe the world, deduce the nature of the cognitive challenge, devise a solution to the cognitive challenge, and perform experiments to validate their proposed solution by taking appropriate actions in the virtual world via their avatar. The process iterates until the player solves the cognitive challenge and continues onward in the game, or until the player tires of attempting to solve the cognitive challenge. We note that the act of solving the challenge 'consumes' the puzzle – the player can not 'un-learn' the solution, but the speed at which they solve the puzzle can increase with practice.

The three observed gameplay designs illustrated that flow charts and finite state machine representations are typical mechanisms already in common use by this game designer for capturing the cognitive challenge and appear to suffice for the task. Possible reasons include information density (they are efficient mechanisms for capturing the gameplay), their inherent self-limits (on diagrammatic complexity) help to ensure that gameplay is acceptably complex (but not overly complex), and they are readily accepted by members of the production team since they are already familiar with these representations.

The cognitive challenge element of the cognitive gameplay requirement is defined as follows.

1. *Clues:* The inputs that the player must recognize as relevant to solving the cognitive challenge. Clues bear strong resemblance to the *cues* in emotional requirements [22]. *Pattern and paths of sliding walls. Side-wall symbol to disable. Sound-effects.*

2. *Challenge:* A description or symbolic representation of the cognitive challenge. Flow chart and finite state machine representations are typical. The description should also describe player feedback mechanisms (such as clues), if they are available. *As per description in Section 10.5.4.*

3. *Verification and Validation:*

   (a) *Solution Strategy:* Description for design review and test. Includes descriptions of the winning condition(s), the optimal solution strategy, and the algorithm used for evaluating partial success (if supported). *As per description in Section 10.5.4.*

(b) *Side Effects:* Explicit identification of the expected side effects on the player and world states that are as a consequence of attempting the cognitive challenge, but do not affect the cognitive challenge. If necessary, explicitly identifies those attributes that must not be modified as a result of this cognitive challenge.

i. Player State *If player is touched by a sliding wall, player health is decremented 10 points and player location is set to the respawn point between the Platform Maze and Sliding Walls puzzles. If successful, player points incremented by 100.*

ii. World State *All aspects of the Sliding Wall puzzle are reset to their initial state.*

iii. Puzzle State – Puzzles can be left in intermediate states. *Mark puzzle state as one of Completed, Attempted, Failed.*

such that

$$\begin{aligned} Cog_i =< Clues_i, Challenge_i, \\ VandV_i\left(SolutionStrategy_i, SideEffects_i\right) \end{aligned}$$

where $SideEffects_i$ is composed of side effects on the player, world, and game states, as a result of attempting this cognitive challenge.

The complexity of the cognitive challenge must be carefully managed. Excessive complexity, through combinatorial explosion of possible solutions, is a typical issue that must be addressed. Game designers are cautioned to ensure that the player's emotional needs for accomplishment are met [20] or an otherwise satisfied player can turn into an individual intent upon disrupting the play experience for themselves and others.

10.6.3 Post-Conditions

Defining post conditions helps to ensure that the player state and the world state are known, and consistent with design expectations, in the period between cognitive challenges. If these states are not carefully managed, a cascading error effect can occur that can be very difficult to trace and address.

Some games keep an explicit model of the player, most commonly to manage adaptive gameplay – gameplay that adjusts in difficulty according to the perceived skill level of the player. The next most common reason is for the game itself to perform a type of self-policing effort to ensure that the player is not cheating.

The post-conditions for a cognitive gameplay requirement are defined as follows.

1. *Player State:* See Section 10.6.1. *Updated to reflect player performance.*
2. *World State:* See Section 10.6.1. *Updated to reflect player performance.*
3. *Puzzle State:* See Section 10.6.1. *Updated to reflect player performance.*
4. *Player Knowledge:* Player knowledge includes what the player has learned from this puzzle, what is the expected learning outcome. *Recognize that elements inconsistent with their context, such as symbols etched onto walls, may have special meaning.*
5. *Game engine knowledge of the player:* Game engine knowledge of the player includes what the game engine has learned about the player from this puzzle, how has the player model been updated? Includes metrics visible to the player (*e.g.* health, abilities) and hidden metrics (*e.g.* performance on tasks to date). *Health, score.*
6. *Side effects:*
   (a) Player state *Updated to reflect player performance.*
   (b) World state *Updated to reflect player performance.*
   (c) Puzzle state *Updated to reflect player performance.*

such that

$$\begin{aligned} Post_i &= < PlayerState, WorldState, PuzzleState \\ &\quad PlayerKnowledge, GameEngineKnowledge, \\ &\quad SideEffects_i > \end{aligned}$$

where $SideEffects_i$ is composed of side effects on the player, world, and game state, as a result of attempting this computational challenge.

10.7 Conclusions

Our observations of a game development team as they prepared a game design in response to a third-party commercial request for proposal have lead to a better understanding of the game design process. We note that the gameplay definition process is highly iterative, with extensive use of top-down and bottom-up analysis and design patterns, and with team interactions and work patterns suggestive of those observed in co-design and co-development efforts.

Three examples of cognitive gameplay definition were observed and a pragmatic definition for cognitive gameplay requirements, capable of capturing the requirements from within the case study, was derived. Cognitive gameplay requirements captured using this definition should more explicitly capture the game designer's intent for cognitive gameplay than observed practice. Further studies with other game designers and multiple game designs are needed to further mitigate this single-source threat to validity.

The strong impact of production needs and opinions, particularly with respect to feasibility, scope, and testability, was addressed in our definition of cognitive gameplay requirements but the effects of this impact upon the requirements process need further investigation. We expect that cognitive gameplay requirements will enable greater certainty in design reviews, project estimation, play testing, player satisfaction testing, and test design and development for both verification and validation.

10.8 Future Work

The cognitive and emotional issues identified in the first three sections of this work are relatively open domains for requirements engineering research. Investigations into the their role in the requirements process and their return on investment are some of the directions that could be pursued. Further, could the same techniques be applicable to the design of other experience artifacts such as movies or advertising?

To be able to generalize our results, we need to observe other game developers and teams to determine whether our initial observations are upheld. We can then formalize the defined attributes for cognitive gameplay requirements and verify the suitability of the approach by using it with other teams. Ideally, some elements of a production game could be specified using cognitive gameplay requirements and the various production artifacts could be inspected to determine the validity of our hypothesis that using cognitive gameplay requirements will reduce production issues and improve the quality of the delivered artifact.

There are many opportunities to develop tools to support this domain. Of particular interest are tools that support traceability (although the degree of traceability that is needed is unknown) and capture rationale (for making design choices) without unduly disturbing the creative process. Other tools could provide support for early evaluation of the development effort (*e.g.* computational and rendering complexity) associated with a given requirement: creativity without a reality check on production constraints can lead to features (and chains of dependencies) that are not technically feasible.

Acknowledgments

The first author would like to thank Krzysztof Wnuk and Birgit Penzenstadler for their feedback on this paper.

Chapter 11

Physualization: Going Beyond Paper Prototyping

The ideas presented to this point are difficult to validate in any short term study. If they are adopted in the market, requirements engineers will find new problems in deployment that must be addressed by both industry experts, who tend to be situated in and have a deep experience with a fixed work culture, and academics, who have the opportunity to study a variety of experiences and share some of them, but who may not be able to experience long hours in production.

Nevertheless, experience requirements appear to be a useful starting point for further work in the field. We demonstrated a process for expressing requirements of all types using ordinary office materials at an interactive session at the Requirements Engineering Visualization Workshop 2010. We further demonstrated the techniques at Requirements Engineering 2010, where experience requirements, the process, and the final side-scrolling racing game were all presented.

Demonstrations were not published but are available upon request. Originally published as follows.

> David Callele. Physualization: Going Beyond Paper Prototyping. In RE 10: Proceedings of the 2010 18th IEEE International Requirements Engineering Conference, Sydney, Australia, 2010. IEEE Computer Society [15].

Abstract: We present physualization, the deliberate physical manipulation of visualization entities, as a means of helping stakeholders explore possibilities in the requirement and design spaces. By engaging more of the stakeholder's sensory and cognitive processes, our goal is to provide a means to enhance the requirements process and the resulting artifacts. Physualization relies upon readily available materials and *ad hoc* techniques to facilitate a lightweight requirements process.

This work provides guidance for an interactive session that explores physualization support for specific requirements engineering topics; developing paradigms for supporting these tasks using materials like stickies, transparencies, markers, and sketchpads as building blocks.

Keywords: Requirements process, requirements methodology, requirements visualization.

11.1 Introduction

Over the years we have investigated many rapid prototyping techniques for their utility in the requirements process: storyboards [4], paper prototyping [87], rich pictures [75] and rich media [112] to name a few.

Paper prototyping is possibly the most common form of rapid prototyping technique and is particularly adept at rapid exploration of the visual aspects of software applications, particularly user interfaces. However, there are more aspects to requirements than the user interface – support for aspects of requirements such as negotiation, traceability and rationale is needed and mechanisms for the rapid capture and representation of spatial options and temporal activities is desirable. These aspects have varying degrees of support in tools such as *Doors* and *RequisitePro* but what about lighter-weight alternatives?

11.2 Physualization

We define *physualization* as the physical manipulation of visualization entities – this is not just visualization for the sake of communicating or creating a record. Physualization actively promotes physical manipulation to help participants explore possibilities in the requirement and design spaces by engaging more of their sensory and cognitive processes – possibly leading to improvements in the requirements process and resulting artifacts. Because of its reliance on materials at hand and *ad hoc* techniques, physualization is most likely to be considered a form of agile requirements process.

This interactive session explores the extension of traditional paper prototyping to physualization with the goal of improving support for requirements activities such as those listed in Table 11.1. The session explores support for specific requirements engineering topics, developing paradigms for supporting these tasks using materials like stickies, transparencies, markers, and sketchpads as building blocks.

Elicitation	Capture	Representation
Specification	Verification	Validation
Triage	Negotiation	Prioritization
Traceability	Rationale	Invention
Revisions	Modeling	Constraints

Table 11.1: Requirements Tasks

In Section 11.6 we provide suggestions for possible techniques and in Section 11.8 we document example physualization output.

11.3 Session Goals

Participants in a physualization session explore how to use common office materials to symbolically represent many of the design paradigms and patterns used in their domain. Participants are challenged to develop physical visualization metaphors to support requirements activities such as those listed in Table 11.1. That is, what tools and techniques can be used to capture and represent RE tasks and principles? These metaphors can be broadly grouped in artifacts (what can be produced, captured) and activities (how are they represented e.g. how to represent negotiation, prioritization).

Participants are further encouraged to explore whether other computing concepts and tasks such as objects, database records, or even the database normalization process can be readily supported.

11.4 Session Results

Session participants are expected to

- Develop specific techniques to support their requirements activities.
- Develop increased appreciation for the utility of common office materials in support of their requirements activities.
- Develop a shared language and methodology for communication using these materials.
- Develop a better understanding of how increasing the number of sensory inputs that are actively engaged in a process can enhance creativity and improve participation.

A typical session will occupy 60 to 90 minutes for the participants – be prepared to have pressure to continue, some groups have kept investigating for far longer!

11.5 Resources and Tools

The suggested resources and tools for physualization are typical office materials.

- Large sketchpads for use as a work surface.
- Sticky notes of different colors and sizes – we have found extra large sticky notes to be quite useful.
- Writing instruments of various colors and sizes.
- Transparency sheets of the type used for overhead projection.
- Permanent and washable markers of the type used for transparency sheets.

Any other items that may be available may also be employed.

11.6 Questions for Consideration

The following points suggest some questions that participants can keep in mind when attempting to generate new techniques.

- What meanings can be encoded into color, size, and other visual attributes?
- Are the X and Y dimensions on the work surface the only ones available? Consider stacking elements rather than replacing them.
- What can transparency sheets be used for?
- How to represent invariates *vs.* variates?
- Can items (such as stickies) be reused?
- Are there patterns or building blocks inherent in what you are attempting to accomplish? Can the pattens be abstracted? Into the materials?
- How to communicate that items are associated?
- How to communicate that items are part of a collection?
- Is there value in generating a record of Work In Progress (WIP)?
- How are you going to generate a record of WIP and final results?
- How to express the elements of a given modeling language?

11.7 Leading a Session

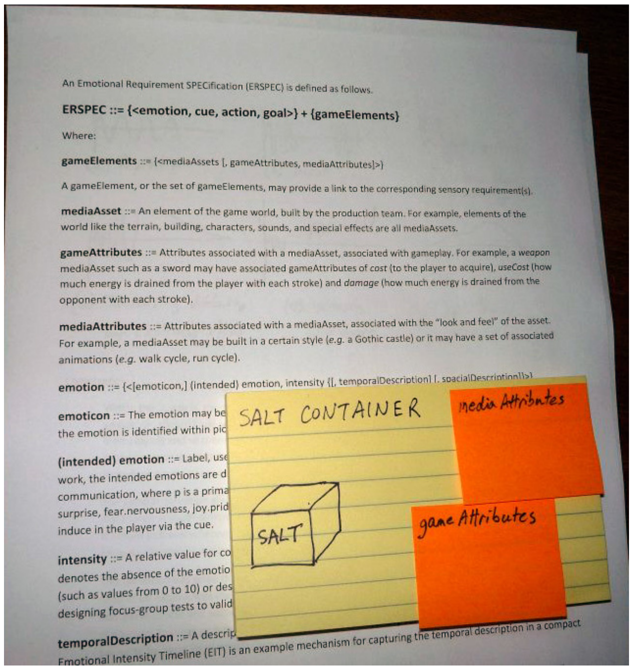
The following outline should help organizers to lead a session.

1. Prepare working materials such as sketchpads, stickies, transparencies, etc. and partition materials into packages for distribution to each working group. Ensure that contribution recording forms, used to capture submissions, are part of each package.

2. Introduce concepts to the participants.
3. Demonstrate sample metaphors.
4. Partition attendees into working groups.
5. Assign topics to working groups.
6. Distribute materials to working groups.
7. Distribute contribution recording forms to working groups. Explain how to record contributions.
8. Allow work period. Attempt to record intermediate results via camera or video.
9. Allow each working group to present and demonstrate their results, allowing time for discussion of each group's results.
10. Present summary comments.
11. Mediate discussion of strengths, weaknesses, suggestions for improvement. Record comments.
12. Gather recording forms to composite summary record.
13. Distribute copies of summary report to participants.

11.8 Some Examples

The following examples are taken from work performed in gathering requirements for video games. The focus in these sessions was on capturing the intended user experience, in general, and the intended emotional experience in particular. In Figure 11.1a we see a template description for emotional requirements and a covering sticky note with layers of stickies and handwriting. The yellow sticky note is for a *gameElement* titled SALT CONTAINER and it has an associated image to provide artistic guidance to the production team. From the background template description, we see that a gameElement has associated *mediaAttributes* and *gameAttributes*. These attributes are on secondary, supporting stickies that are themselves color-coded. The use of secondary stickies allows the requirements elicitation process to be very dynamic - there are no concerns with rapid iterations and complexities of erasing and replacing, simply peel off and replace with a new iteration.



An Emotional Requirement SPECification (ERSPEC) is defined as follows.

ERSPEC ::= {<emotion, cue, action, goal>} + {gameElements}

Where:

gameElements ::= {<mediaAssets [, gameAttributes, mediaAttributes]>}

A gameElement, or the set of gameElements, may provide a link to the corresponding sensory requirement(s).

mediaAsset ::= An element of the game world, built by the production team. For example, elements of the world like the terrain, building, characters, sounds, and special effects are all mediaAssets.

gameAttributes ::= Attributes associated with a mediaAsset, associated with gameplay. For example, a *weapon* mediaAsset such as a sword may have associated gameAttributes of *cost* (to the player to acquire), *useCost* (how much energy is drained from the player with each stroke) and *damage* (how much energy is drained from the opponent with each stroke).

mediaAttributes ::= Attributes associated with a mediaAsset, associated with the "look and feel" of the asset. For example, a mediaAsset may be built in a certain style (*e.g.* a Gothic castle) or it may have a set of associated animations (*e.g.* walk cycle, run cycle).

emotion ::= {<[emoticon,] (intended) emotion, intensity {[, temporalDescription] [, spacialDescription]}>}

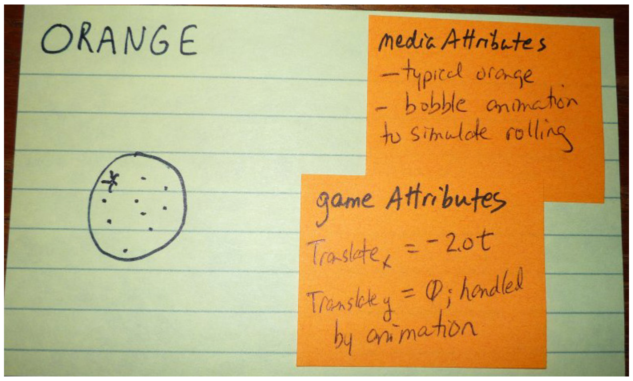
emoticon ::= The emotion may be
the emotion is identified within pic

(intended) emotion ::= Label, use
work, the intended emotions are d
communication, where p is a prima
surprise, fear.nervousness, joy.prid
induce in the player via the cue.

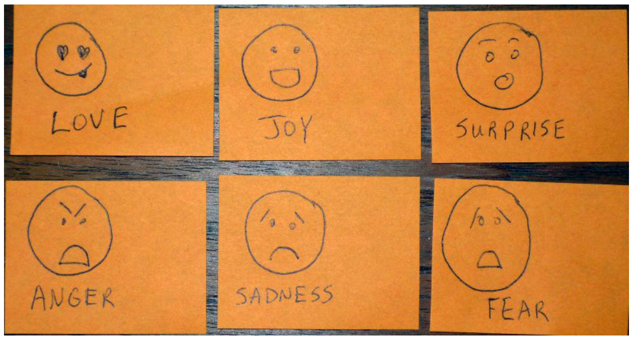
intensity ::= A relative value for co
denotes the absence of the emotio
(such as values from 0 to 10) or des
designing focus-group tests to valid

temporalDescription ::= A descrip the temporal description in a compact
Emotional Intensity Timeline (EIT) is an example mechanism for capturing

(a) Emotional requirement specification format, getting started


ORANGE
media Attributes
– typical orange
– bobble animation to simulate rolling
game Attributes
Translate$_x$ = -2.0t
Translate$_y$ = Φ; handled by animation

(b) A closer look at a gameElement



(c) Reusable emotion icons

Figure 11.1: Starting to physualize

The sticky notes also allow us to bind together requirements elements; the mediaAttributes and gameAttributes are clearly bound to the larger gameElement. Figure 11.1c shows a selection of intended emotional states. The iconic nature of the elements shows that they are intended to act as library elements, promoting reuse during sessions.

In Figure 2, we see portions of workspaces associated with simple gameplay elements from a 2D side-scrolling game. These scenarios make use of a number of principles. The invariate element is sketched on a background workspace. The player avatar is iconified in various actions (jumping, in this example) and the intended emotional states for the player are drawn from the library previously introduced. Patterns for gameplay activity, such as *repetition* and *challenge followed by mastery* are also used and shown in contextually appropriate locations across the bottom of the background workspace. Success (Figure 11.2a) and failure (Figure 11.2b) modes are shown and the player's emotional state is indicated. For example, the player experiences an alternating emotional state, passing between JOY and FEAR as the oranges roll toward them. JOY is associated with successfully jumping over the orange, FEAR is associated with the recognition that the orange is rolling ever closer and that the player must soon successfully jump, or have their player killed. The Challenge followed by Mastery pattern identifies the type of challenge – the player must repeatedly meet the challenge but they eventually master the technique. The amplitude of the the sine wave represents the decreasing intensity of the experience.

Figure 11.2c illustrates alternative gameplay in the same scenario. In this example, two elements have changed: the challenge is now a banana which has more difficult gameplay than an orange (Increasing Challenge sticky) and the player is punching the banana rather than jumping over it. Note that the banana sticky is layered on top of the (partially hidden) orange sticky. The underlying sticky is deliberately exposed to indicate that the banana and orange are options, each of which can occur during gameplay. If the orange sticky was completely hidden by the banana, then this would indicate that the original design decision to use an orange has been changed to that of a banana.

The metaphor chosen: partial *vs.* full overlap to indicate the difference between runtime gameplay options and gameplay design history was arrived at by the participants in an earlier session. The design history metaphor supports the common requirements task of maintaining a revision history. Partial overlap is a concise representation of the conjunction of requirements.

The basic principle of placing the invariates on the background of the workspace are also illustrated in Figure 3. Figures 11.3b and 11.3c illustrate the use of transparencies to present gameplay requirements for different gameplay scenarios. Each gameplay scenario is described on the transparent overlay and different options can be explored with ease. Figure 11.3b also illustrates that validation and verification activities can be added with another color of sticky.

11.9 Other Work of Interest

In addition to the traditional bibliography, we include links to a small selection of related materials on the Internet and links to a selection of YouTube video clips presenting related work on the use of paper prototyping.

11.9.1 Printed Materials

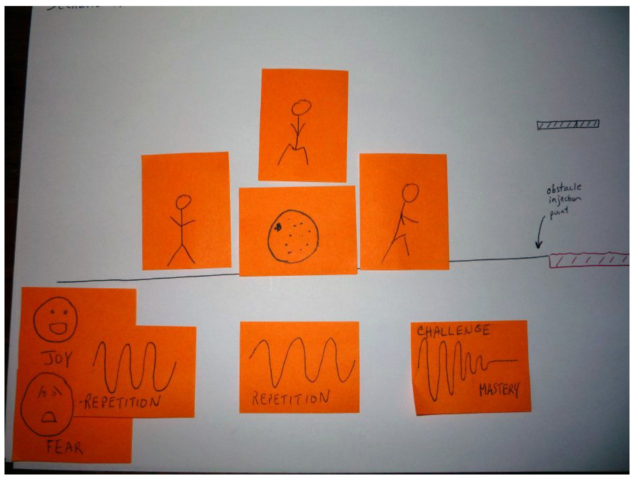
- *Paper Prototyping by Carolyn Snyder* http://www.paperprototyping.com/index.html
- *Paper prototyping (a general introduction to the process)* http://www.usabilitynet.org/tools/prototyping.htm
- *Hipster PDA* http://en.wikipedia.org/wiki/Hipster_PDA
- *Post-it Note Design Docs* http://www.lostgarden.com/2008/12/post-it-note-design-docs.html
- *Paper Prototyping by Shawn Medero, A Basic Introduction* http://www.alistapart.com/articles/paperprototyping/
- *Considering Prototypes* http://www.uxbooth.com/blog/considering-prototypes/
- *Data Sculpture* Zhao, Jack and Moere, Andrew Vande. Embodiment in data sculpture: a model of the physical visualization of information. In DIMEA '08: Proceedings of the 3rd international conference on Digital Interactive Media in Entertainment and Arts, 2008, pp. 343-350, ACM, New York, NY, USA

11.9.2 Physical Visualizations

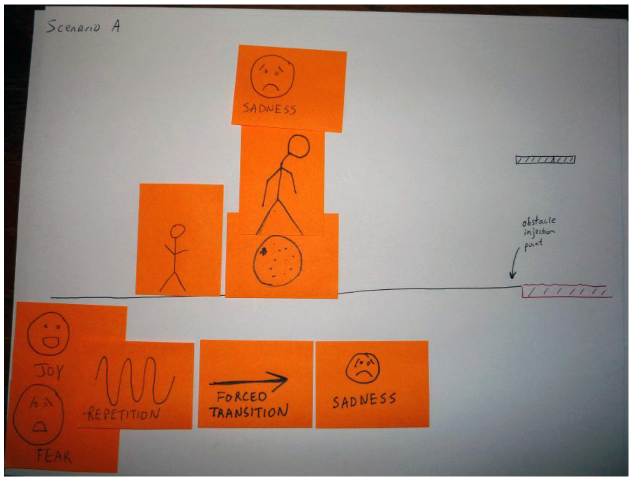
- *Glowing temperature sink fixtures*
  http://www.boingboing.net/2005/07/13/glowing_temperatures.html
- *Waveform display of a musical piece*
  http://well-formed-data.net/archives/150/physical-visualization
- *Visualization Problems? Get Physical!*
  http://ezinearticles.com/?Visualization-Problems?–Get-Physical!&id=1383153
- *Physical Data Art by Willem Besselink (by Maria Popova)*
  http://www.brainpickings.org/index.php/2009/11/11/willem-besselink/ and
  http://www.willembesselink.nl/read/willem_besselink–portfolio

11.9.3 YouTube Videos

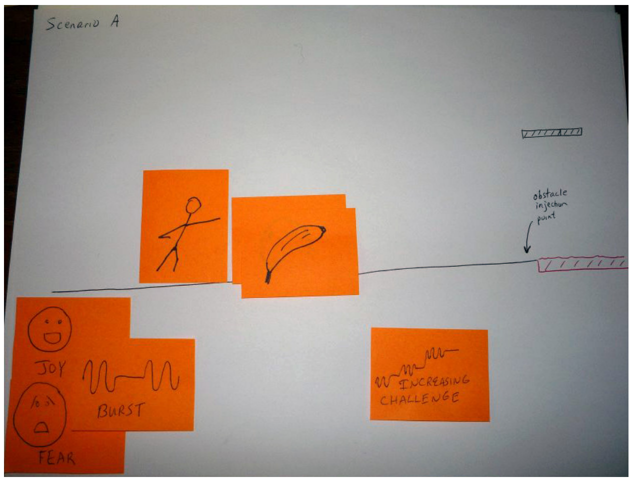
- *iPhone Paper Prototype Post-it* http://www.youtube.com/watch?v=If2iRj1GWzk
- *Trouble (Game) Paper Prototyping*
  http://www.youtube.com/watch?v=dTR7gbsF7Os
- *Paper Prototype for Mobile Journalism*
  http://www.youtube.com/watch?v=3-UWIVMhYkA
- *Paper prototype created by using the Scrum process.*
  http://www.youtube.com/watch?v=ykJ60H4Qkvg
- *IAT 410 paper prototype, game design*
  http://www.youtube.com/watch?v=4ROZqOwHyWo
- *DAC 300 Paper Prototype - Tap That!*
  http://www.youtube.com/watch?v=EiMyMk10d0I
- *Paper prototyping: Game design* http://www.youtube.com/watch?v=k-9pkB05IlQ
- *Have Paper, Will Prototype* http://www.youtube.com/watch?v=L3yl9vaJuFE



(a) A simple gameplay scenario

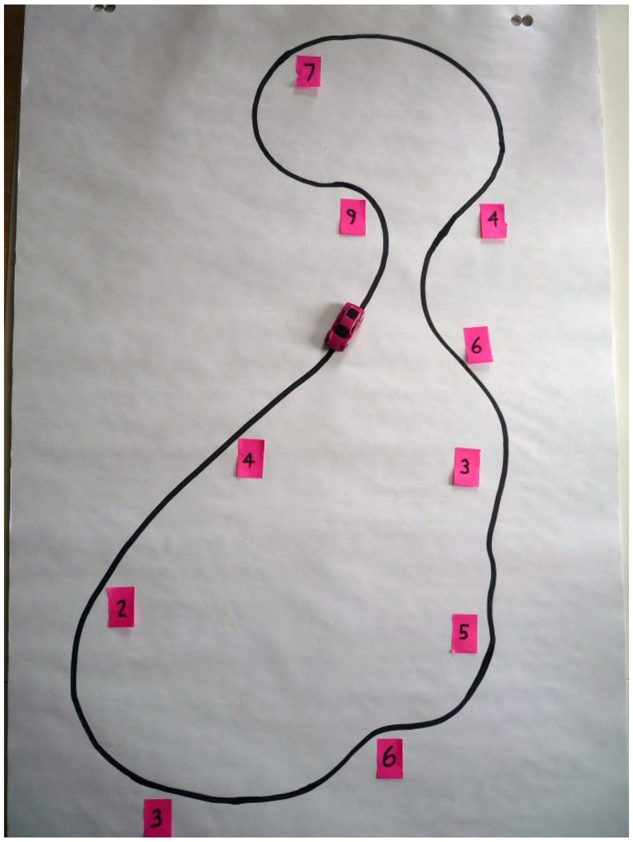


(b) Visualization of a failure mode for the same gameplay scenario

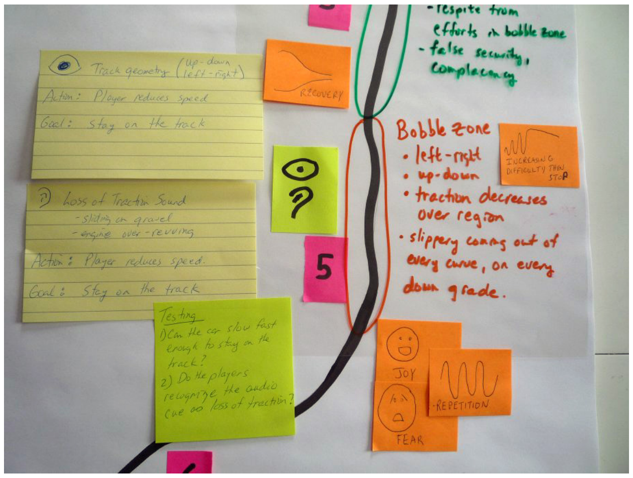


(c) Visualization of alternate gameplay for the same scenario

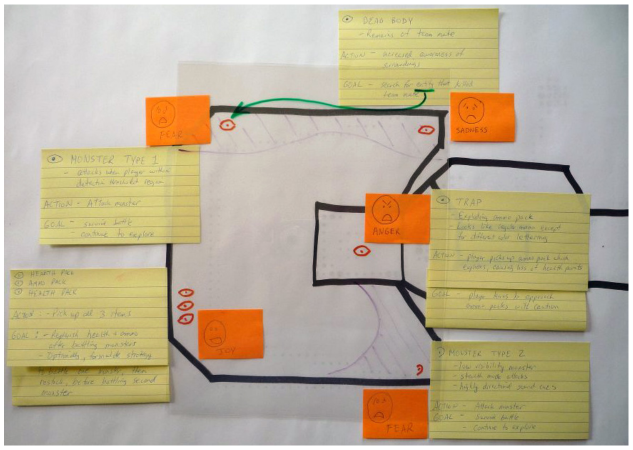
Figure 11.2: Simple gameplay scenarios



(a) Racing game track with experience region identifiers


- respite from efforts in bobble zone
- false security, complacency
Track geometry (up-down, left-right)
Action: Player reduces speed
Goal: Stay on the track
RECOVERY
Bobble zone
• left-right
• up-down
• traction decreases over region
• slippery coming out of every curve, on every down grade.
INCREASING DIFFICULTY THEN STOP
1) Loss of Traction Sound
- sliding on gravel
- engine over-revving
Action: Player reduces speed.
Goal: Stay on the track
5
Testing
1) Can the car slow fast enough to stay on the track?
2) Do the players recognize the audio cue as loss of traction?
JOY
REPETITION
FEAR

(b) A closer look at experience requirements in a racing game scenario



(c) A closer look at experience requirements in a first person shooter game scenario

Figure 11.3: Other game requirements

Chapter 12
Applying Experience Requirements and Conclusions

The models and techniques presented in this document have the potential to help cross the communications chasm between preproduction and production identified in Chapter 2. To illustrate the potential, we present preliminary details of work on two games not discussed in our published articles, The Windblown Adventure and a scenario from a published first person shooter, DOOM™ by id Software.

These games were chosen because both games have published online game design documents that we can analyze and use to compare and contrast with our efforts. In both cases, we have annotated or extended parts of these documents using the formalisms and techniques presented in the body of this work, concentrating on emotional requirements and the player experience.

This preliminary work is presented to provide the reader demonstrations of how emotional requirements can be used in practice and the benefits that they can deliver.

After the Doom™ example, we then take a closer look at negotiating requirements at runtime as proposed in Chapter 6, providing examples for both player *vs.* griefer and player *vs.* publisher conflicts.

12.1 Windblown Adventure

The Windblown Adventure was created by students at DADIU, The National Academy of Digital Interactive Entertainment in Denmark. It is a 2.5D side-scrolling platform jumping game targeted at a (primarily female) 8 year old demographic.

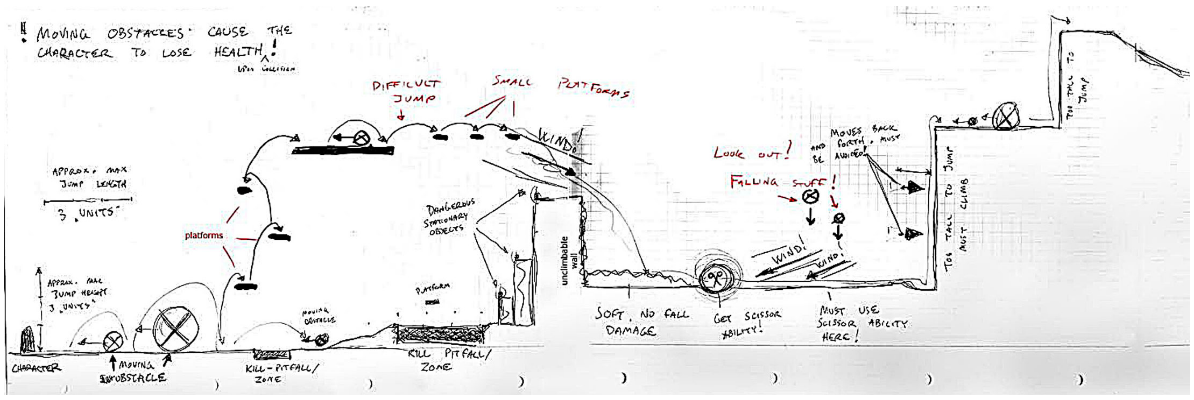

MOVING OBSTACLES CAUSE THE CHARACTER TO LOSE HEALTH! UPON COLLISION
DIFFICULT JUMP
SMALL PLATFORMS
WIND
APPROX. MAX JUMP LENGTH 3 UNITS
platforms
APPROX. MAX JUMP HEIGHT 3 UNITS
CHARACTER
MOVING OBSTACLE
KILL-PITFALL/ZONE
MOVING OBSTACLE
PLATFORM
DANGEROUS STATIONARY OBJECTS
KILL PITFALL/ZONE
unclimbable wall
SOFT, NO FALL DAMAGE
GET SCISSOR ABILITY!
LOOK OUT!
FALLING STUFF!
WIND!
MOVES BACK AND FORTH, MUST BE AVOIDED
MUST USE SCISSOR ABILITY HERE!
TOO TALL TO JUMP MUST CLIMB
TOO TALL TO JUMP

Figure 12.1: Windblown Game Level Design

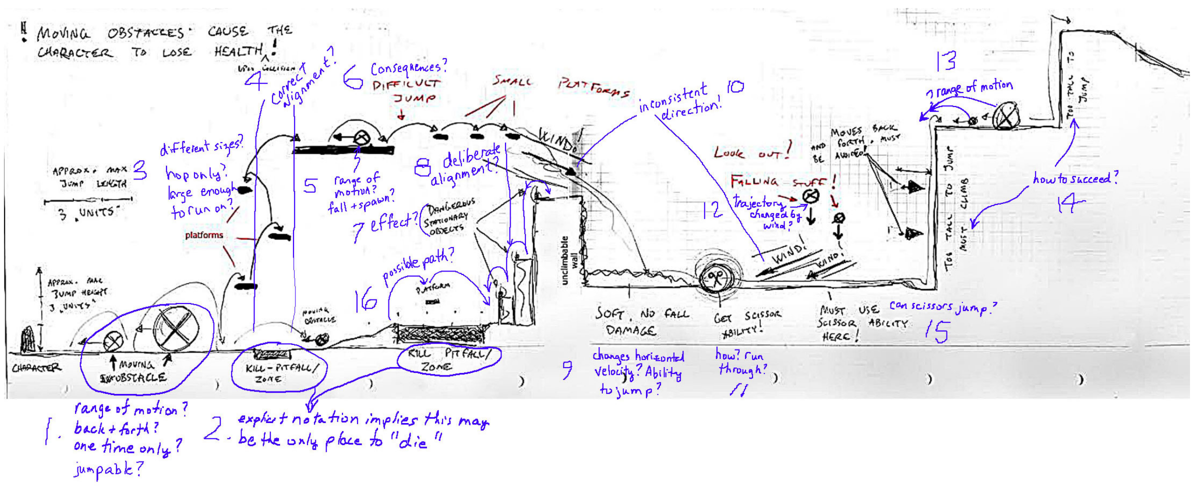

1. range of motion? back + forth? one time only? jumpable?
2. explicit notation implies this may be the only place to "die"
3 different sizes? hop only? large enough to run on?
4 correct alignment?
5
6 Consequences?
range of motion? fall + spawn?
7 effect?
8 deliberate alignment?
9 changes horizontal velocity? Ability to jump?
10 inconsistent direction!
11 how? run through?
12 trajectory changed by wind?
13 range of motion
14 how to succeed?
15 can scissors jump?
16 possible path?

Figure 12.2: Windblown Game Design Review

The game concept is straightforward. The player is represented by a rag-doll-like character situated within a kitchen environment. The player is building a castle out of dry flour when their creation is threatened by a strong breeze coming through an open kitchen window. The player's goal is to cross the kitchen and close the window before their castle is destroyed by the wind - a task complicated by the fact that their avatar is quite small so crossing the kitchen becomes an obstacle race against time.

This example was chosen for the following reasons.

1. The game development effort had a published game design document [102].

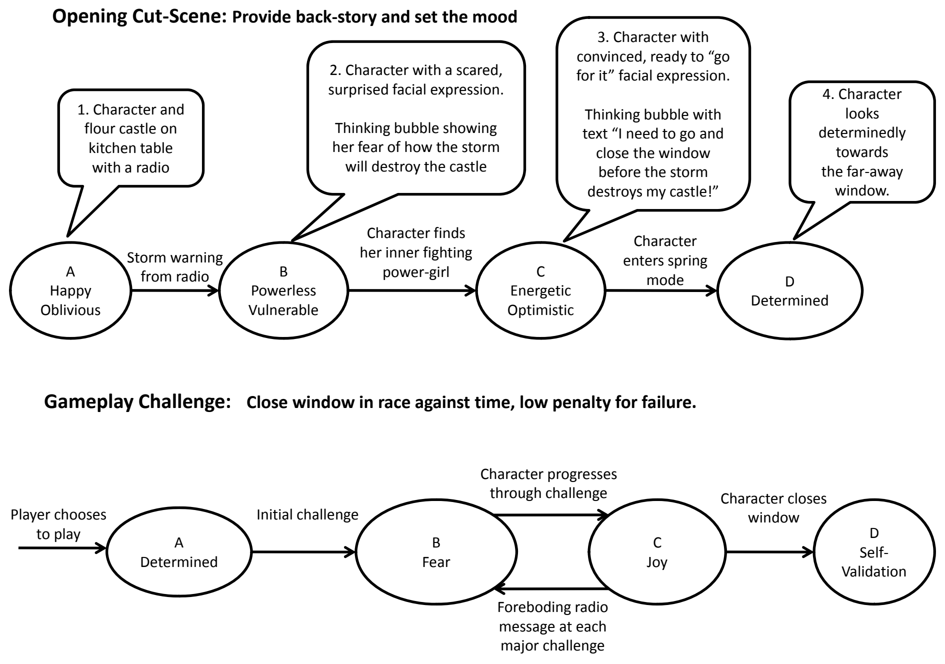

Opening Cut-Scene: Provide back-story and set the mood
1. Character and flour castle on kitchen table with a radio
2. Character with a scared, surprised facial expression. Thinking bubble showing her fear of how the storm will destroy the castle
3. Character with convinced, ready to "go for it" facial expression. Thinking bubble with text "I need to go and close the window before the storm destroys my castle!"
4. Character looks determinedly towards the far-away window.
A Happy Oblivious
Storm warning from radio
B Powerless Vulnerable
Character finds her inner fighting power-girl
C Energetic Optimistic
Character enters spring mode
D Determined
Gameplay Challenge: Close window in race against time, low penalty for failure.
Player chooses to play
A Determined
Initial challenge
B Fear
Character progresses through challenge
Foreboding radio message at each major challenge
C Joy
Character closes window
D Self-Validation

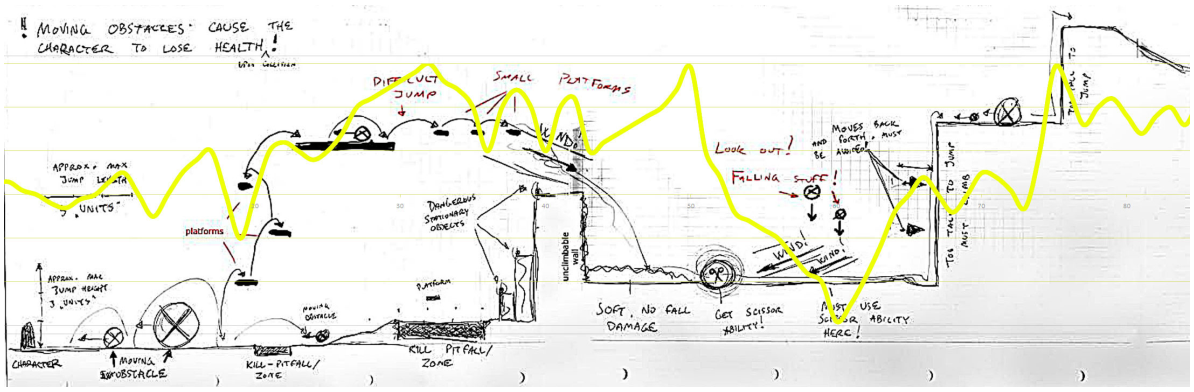
Figure 12.3: WindBlown Adventure Emotional Journey

2. The game development effort completed and a demonstration of the final product was available (in the form of a gameplay video [103]) that could be compared to the game design document.

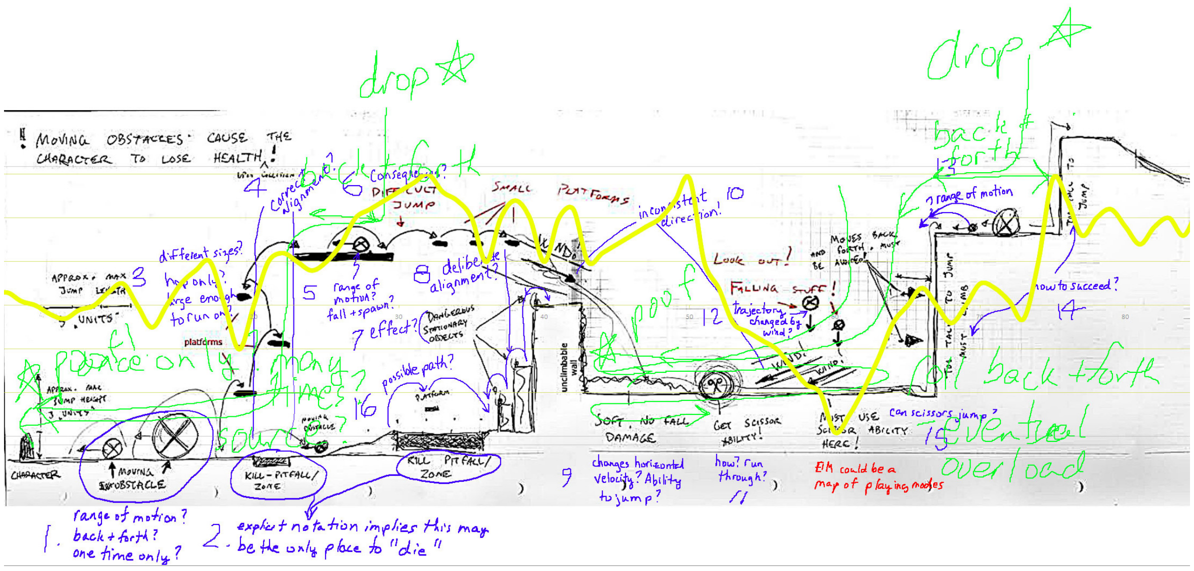
3. The game genre is a side-scrolling platform-jumper, one of the simplest game genres to define. The game is intended to induce strong emotional experiences and contains requirements for mastery of both mechanical gameplay and cognitive gameplay challenges.

The project was successfully completed and that the development team addressed every issue noted in our review. The methodologies used by the development team are typical of those we have observed in active use in other commercial development efforts and do not represent inherent deficiencies in the team that threaten the validity of the results.

Figure 12.1 is copied from The Windblown Adventure Design Document and represents



(a) Windblown Emotional Intensity Timeline Overlay


drop ☆
drop ☆
back + forth
back + forth
consequences?
correct alignment?
different sizes?
how only large enough to run on?
range of motion? fall + spawn?
8 deliberate alignment?
7 effect?
inconsistent direction! 10
poof
trajectory changed by wind?
range of motion
how to succeed?
14
place only
many times?
possible path?
16
sound?
can scissors jump?
15
eventual overload
changes horizontal velocity? Ability to jump?
9
how? run through?
11
EM could be a map of playing modes
1. range of motion? back + forth? one time only?
2. explicit notation implies this may be the only place to "die"
3
4
5
6
12
13

(b) Design review, all layers enabled

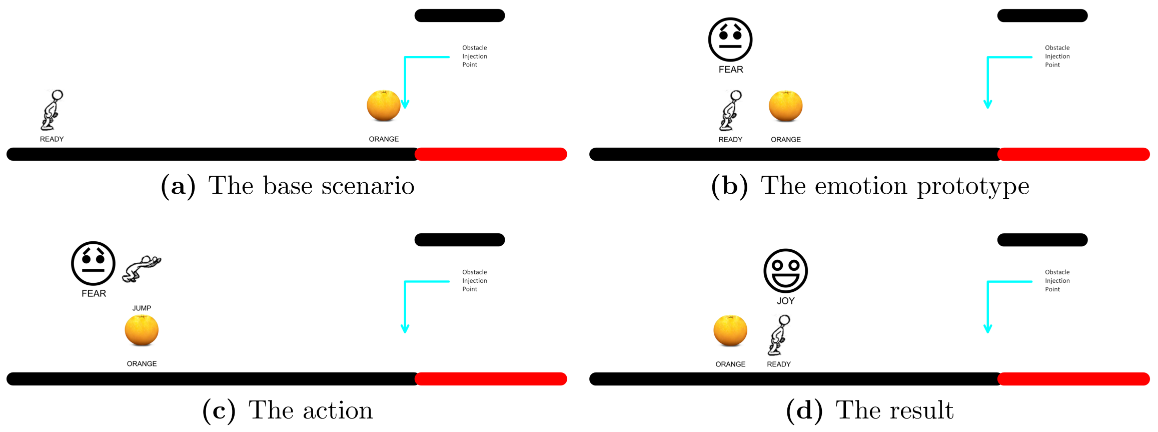
Figure 12.4: Windblown Design Review Details

the level design concept art for a portion of the gameplay environment [1].

Figure 12.2 has the same content, marked up as would occur in a typical production design review process. The list of questions is lengthy, identifying challenges typical of the issues identified in our analysis of the post mortem reports associated with other game development projects [16].

The first challenge faced by the game development documentation process is demonstrated in these two figures - a large part of the specification for a videogame is visual, yet

[1]The gameplay environment is commonly partitioned into scenarios related by the storyline where these partitions are individually referred to as a *level*.



(a) The base scenario **(b)** The emotion prototype

(c) The action **(d)** The result

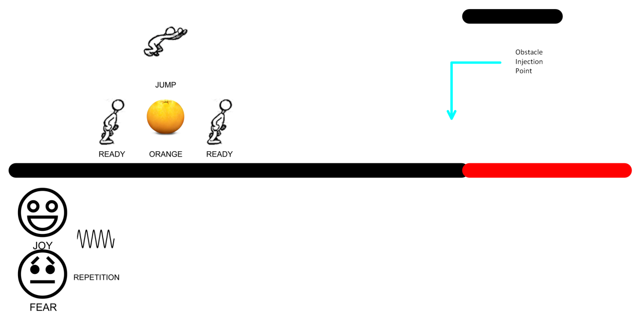
Figure 12.5: Capturing the gameplay experience

the visualization by itself is insufficient to capture the information needed by production. Extensive supporting information is required and automatically maintaining the links between the visualizations and supporting text through tool support may be a solution. However, these tools must be as unobtrusive as possible during preproduction to ensure that they do not have a negative impact upon the creative process and the nature of an appropriate user interface remains an open question.

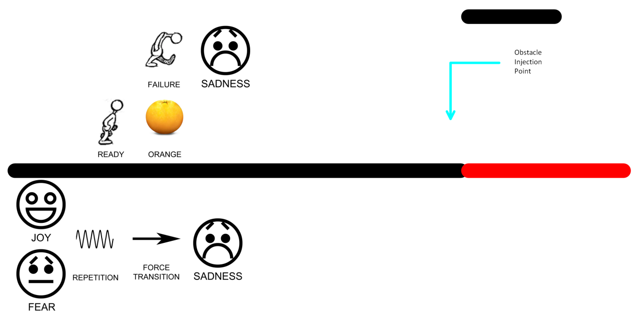
Emotional requirements are situated within the game world via an emotional intensity map and communicate the designer's intended emotion at that location. For (relatively linear) games like side-scrollers or races, the emotional intensity map can be visualized in 2D using an X-Y graph where the X axis represents location and the Y axis illustrates intensity of the emotional experience. The map can be annotated to ensure that the viewer can clearly identify the intended emotion at that time.

Identifying and resolving inconsistencies is a core requirements engineering task and responsibility. For example, Figure 12.1 shows various arrows that indicate wind direction. Unfortunately, these arrows are pointing in different directions (see Question 10 on Figure 12.2) and a review of the text in the game design document shows that the document does not contain any further clarification as to why the arrows point in different directions.

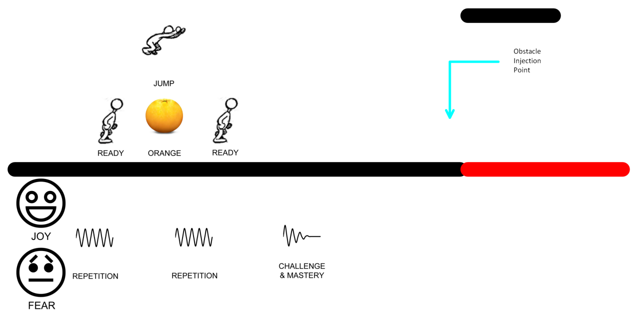
The opening cut-scene is expected to help the player to identify with their on-screen avatar. The player's intended emotional experience is shown in Figure 12.3, a visual representation of the experience that was derived from the quoted text. If this were a screen-



(a) Summary for success



(b) Failure mode



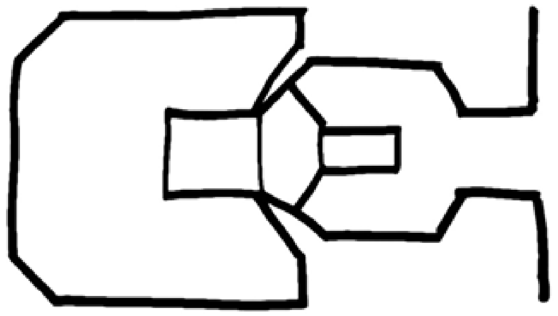
(c) Adding further production guidance

Figure 12.6: Experience requirement production guidance

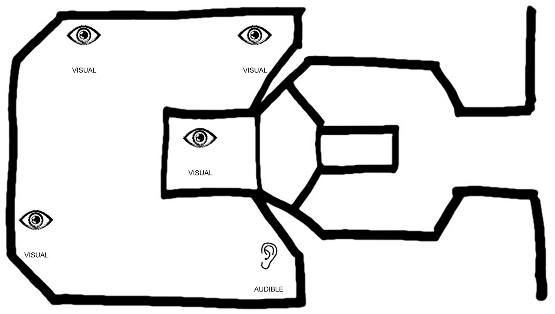
play, or stage play, this diagram would represent the player's emotional journey during the preamble [42]. While not explicitly identified in the design document, we also visualize our interpretation of the intended gameplay emotional experience in the lower half of the diagram.

The diagram facilitates design review, provides production guidance, and also facilitates the identification of tests for states and tests for transitions. For example, the test team could choose to evaluate whether the preamble succeeded at inducing the desire emotional state in the player as they start the game.

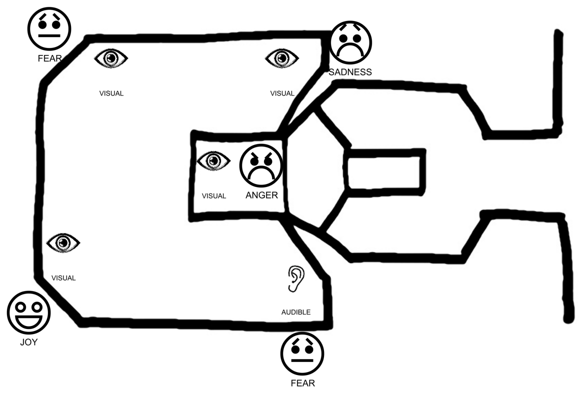
The intensity of the emotional experience can be visualized using an emotional intensity map. In this genre, a 2D representation suffices to represent the designer's intent. Unfortunately, the game design document does not contain the information necessary for us to construct the designer's intended emotional intensity map, so we assumed the role of designer for that purpose. Based upon the known design details, we inferred the hypothetical emotional intensity map shown in Figure 12.4a and overlay it upon the level diagram. In Figure 12.4b we enable all comments and design guidance.



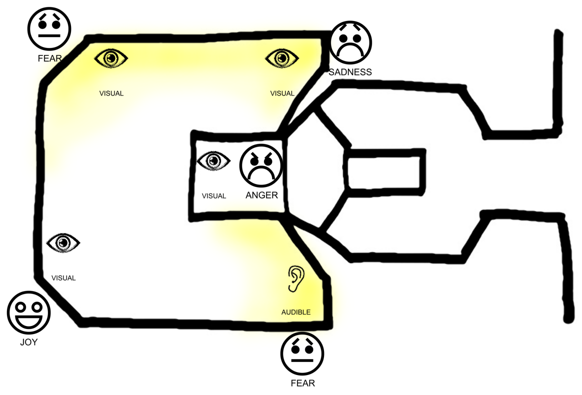
(a) Floor plan of scenario



(b) Identifying the type and location of cues



(c) Intended emotions added



(d) Emotional intensity map

Figure 12.7: First-person shooter experience requirements

Figures 12.5a through 12.5d illustrate stages in capturing the gameplay experience. Figure 12.5a captures the base scenario. In the background we identify the obstacle injection point and provide points of reference (the pit in red and platform in black) to situate the location within the larger level. The orange is shown at the injection point and the player is shown at the start point. Figure 12.5b captures the emotion (Fear of the orange) and the cue (the orange) elements of the emotion prototype. The action could be captured as shown in Figure 12.5c; if necessary the player trajectory could be shown using arrows. Finally, the expected player emotional state is captured in Figure 12.5d, Joy at successfully overcoming the challenge

Figure 12.6a captures further details of the gameplay experience. It shows that the player is expected to alternate between emotional states of Fear and Joy, in a Repetition pattern, where the emotional states are induced by the presentation of obstacles that induce Fear in the player as they approach and induce Joy as they are overcome. Figure 12.6b illustrates a failure mode for the player. The player experiences Sadness upon contact with the obstacle because their avatar is 'killed' and they must restart the gameplay scenario. The designer specifically identifies that the emotional state change is deliberate via the Forced Transition token.

More subtle gameplay experiences can be captured as shown in Figure 12.6c. In this case, production is advised that the overall gameplay pattern is one of Challenge then Mastery. In other words, continue to inject obstacles into the scenario until the game engine has determined that the player has mastered the necessary skills (*e.g.* by successfully jumping over N successive obstacles), then taper off the injection of obstacles to zero to allow the player to go on to other scenarios.

12.2 First Person Shooter (FPS)

This example is loosely based upon the E1M1 map from the original DOOM $^{\mathrm{TM}}$game by iD Software. Figure 12.7a shows a plan view of the location for the experience scenario. In Figure 12.7b, the types and locations of the cues are added – note that all but one of the cues is a visual cue. The emotions that the cues are expected to trigger in the player are added in Figure 12.7c. Figure 12.7d shows the beginning of adding the emotional intensity map to the scenario. One of the basic tenets of the emotional intensity map is the use of grayscale shading to indicate relative intensity between emotions in a region (done in yellow in this example for publication clarity). These regions do not have to follow any form of real-world physics, so arbitrary shapes like those shown are possible.

Figure 12.8 adds the remaining information necessary to capture the emotional requirements, the details for the actions and goals. Note how the relationship between the two cues at the top of the diagram is explicitly identified: In the accompanying narrative description, while the player is inspecting the remains of their dead teammate, the monster responsible for their demise is stalking them.

12.3 Negotiating Requirements at Runtime

In Chapter 6 we looked at the interactions between emotional requirements and security requirements, noting that there are cases where the emotional requirements can dominate the security requirements. In other words, there are cases where security requirements may be relaxed, or even set aside, to accommodate the emotional needs of the stakeholder. We also observed that there may not be an optimal resolution to these conflicting requirements and that negotiation may be ongoing. For example, there may be a security goal for ensuring the integrity of gameplay. However, the player (customer) may find that they do not enjoy playing the game as designed – but with some (unauthorized) gameplay changes their satisfaction is greatly increased. The publisher can authorize a change in gameplay to meet the immediate player requirements but this may not be the end; there may be further changes demanded at a later time.

It appears that a negotiation process that eliminates the requirement to resolve all requirements conflicts *a priori* yet allows the conflicts to be resolved as they occur and in a manner that addresses the emotional requirements of the stakeholders could be very useful.

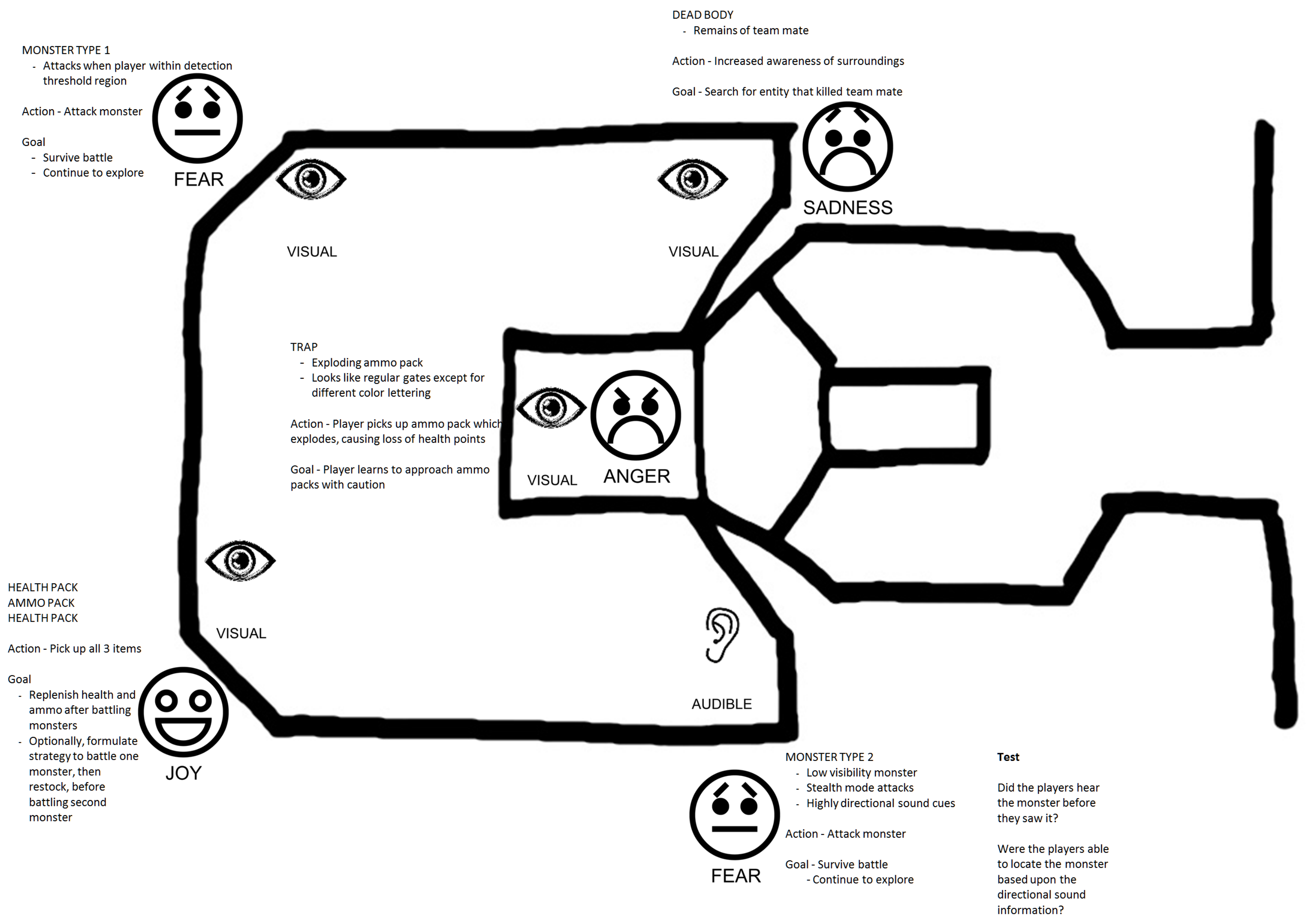

MONSTER TYPE 1
- Attacks when player within detection threshold region
Action - Attack monster
Goal
- Survive battle
- Continue to explore
FEAR
DEAD BODY
- Remains of team mate
Action - Increased awareness of surroundings
Goal - Search for entity that killed team mate
SADNESS
VISUAL
VISUAL
TRAP
- Exploding ammo pack
- Looks like regular gates except for different color lettering
Action - Player picks up ammo pack which explodes, causing loss of health points
Goal - Player learns to approach ammo packs with caution
VISUAL
ANGER
VISUAL
HEALTH PACK
AMMO PACK
HEALTH PACK
Action - Pick up all 3 items
Goal
- Replenish health and ammo after battling monsters
- Optionally, formulate strategy to battle one monster, then restock, before battling second monster
JOY
AUDIBLE
MONSTER TYPE 2
- Low visibility monster
- Stealth mode attacks
- Highly directional sound cues
Action - Attack monster
Goal - Survive battle
- Continue to explore
FEAR
Test
Did the players hear the monster before they saw it?
Were the players able to locate the monster based upon the directional sound information?

Figure 12.8: Emotional requirements

12.3.1 An Exemplary Dispute Resolution Process

We have proposed that just-in-time resolution of these conflicting requirements can be provided by introducing the metaphor of an in-game justice system [20]. While this metaphor is used in some games, in this discussion we are going to use the less emotionally-loaded term dispute resolution.

We discuss herein two example scenarios: conflicting requirements between a player and a griefer and conflicting requirements between the player community and the publisher. For these discussions to have meaning, we must assume that there is some form of deciding authority to which the conflict can be referred for resolution. In other words, there was a requirement for a dispute resolution mechanism as part of the experience requirements developed during preproduction. In each case, the discussion shall address the following points.

1. Accusation
2. Identification
3. Advocacy
4. Resolution
5. Implementation
6. Implications: Virtual World and Real World

Dispute resolution mechanisms tend to rely upon a recognized authority that has the final say on disputes; examples include the publisher, developer, and other game players. In the case where other game players form the recognized authority, their selection may be random (like real-world jury selection) or deliberate such as by peer nomination or even formal elections.

12.3.2 Player *vs.* Griefer

This scenario addresses the relatively common griefing behavior pattern where one player is accused of repeatedly stalking and killing another player. We refer to the participants as the griefer and the player.

Accusation The player accuses the griefer of taking actions that are unacceptable. The arguments are that the behavior pattern is against the rules of the game, either real rules as defined in the game rules or Terms Of Service (the participation contract) or rules that have been implied and accepted by the greater community of players.

Identification While the player and griefer are obvious stakeholders, every other participant in the game ecosystem may also be affected by a decision made in this particular case. While the desired consequences may be achieved, the possibility of emergent behavior in the game world may result in unintended consequences.

Advocacy Advocacy on behalf of the stakeholders is a significant challenge and parallels those challenges faced in real world legal systems: How is this advocacy to be achieved? Is it via some form of realtime communication between the parties? Are there anonymity and privacy concerns? Can alternative (possibly professional) advocates be employed?

Resolution Someone or something must have the ability and authority to force a resolution of the matter.

Implementation If we assume that the dispute resolution system incorporates real-world concepts such as punishment, restitution and rehabilitation then there may be parallels in the game world. Punishments could include banishing a player from a particular game area, perhaps for a given period, or they may be assessed fines in terms of their in-game wealth or experience points. Restitution could also be economic or it could be achieved by putting the player on a "watch list" and identifying them, in some way, to other players in the game world.

Implications: Virtual World and Real World The points just presented represent only a small sample of the things that *could* be done in the scenario. However, requirements engineering is a pragmatic discipline and there are significant constraints on any dispute resolution mechanism that must be addressed.

The first constraint is that the griefer must be a willing participant in any corrective action that is designed to illustrate (to them) why the behavior was unacceptable and to punish them for this behavior with the expectation that they will modify their behavior in the future. While this approach may work for some participants, it is unlikely to work for others.

Taken further, such an approach may only work within communities of interest, in-game communities where the participants have some motivating reason to stay, and potentially succeed, within the community. Under this assumption, we can then turn our attention to the challenge of advocacy.

Given the potentially geographically distributed nature of the player community, practical implementations imply that realtime resolutions will be difficult. Therefore, there may be a need for support for written submissions, *etc.* if anyone but those players "in the area at the time of the incident" are involved. Further, if this is a paid service, then there are real world contract issues that may be involved (there may even be issues for an unpaid service) – who decides whether there is a breach of contract on the terms-of-service? Can that authority be (legally) delegated to anyone other than the service provider?

Example We now present a more definite example for a player *vs.* player dispute scenario.

Accusation	Camping. The griefer is stationed near the new player spawn (start) point and killing the other players as they enter the world.
Identification	Victims of the griefer (usually novice players). The griefer.
Advocacy	Individual stakeholders act as their own advocates.
Resolution	Publisher issues a rule change: If camping is detected, the camper will be penalized with the loss of all weapons and ammunition. Camping is defined as the killing of players as they enter (re-enter) the world at the spawn point, and within 100m of the spawn point, in the virtual world.
Implementation	The rules of the game are embedded in configuration files since the developer planned for runtime dispute resolution. New configuration files are pushed to the community but no software patches are required.
Implications	Griefers / campers lose their desired experience of an *easy kill* as players spawn. Player satisfaction in the rest of the community improves.

12.3.3 Player *vs.* Publisher

In comparison, we now look at conflict between the player and the publisher. Player *vs.* publisher conflicts are typically over who has the right to control the gaming experience. Must the game be played as designed? Can the game be played any way the player wants to play since they bought the game? Or, may there be instances where the answer is somewhere between the two extremes? In the following discussion, we will assume that it is the player (customer) that wants to force a change.

Accusation For some reason, the customer is unhappy with the game that they purchased. It could be that the opponents are too difficult to overcome. Or, the player feels that they have to spend too much time performing tasks (such as farming) that they do not want to do before they are allowed to go and do what they want to do (*e.g.* go on epic quests for riches and enlightenment). In both cases, we assume that the player is satisfied with the game engine and the game implementation – the player is dissatisfied with the game experience. We shall look at the case of gameplay imbalance (opponents are too difficult) for the remainder of the discussion.

Identification The game experience is unsatisfactory to the player. If there is just one player that feels this way, or even a small number of like-minded players then the publisher can safely ignore them. However, when a sufficiently large portion of their customer base shares this negative opinion then the publisher may be forced to act before substantial damage is done to their reputation or to sales of the game.

Advocacy Advocacy for change can be swift and obvious. Submissions from the players directly to the publisher, complaints to support lines and discussions on user groups are all widely available.

Resolution In the absence of an in-game resolution mechanism, the publisher is forced to try to determine what changes to make by observing the customer base. However, the same mechanics that allow a "jury of one's peers" to vote on whether a player should be banned from the community can also be used to allow the community to vote on the type and magnitude of the changes.

Implementation One implementation option is to simply allow the community to make their desired modifications to the rules and attempt to play within the new environment. Alternatively, the player community could choose modifications to the rules then forward the change requests to the publisher for implementation.

Implications: Virtual World and Real World The player community, as a whole, is unlikely to be familiar with large-scale simulation theory and practice or with the challenges associated with managing unexpected emergent behavior in these systems. By allowing the player community to change how gameplay is ruled, it could be that the entire game world is ruined rather than improved. Who would 'take the blame' at that point is unclear.

From the experience requirements perspective, the game designer should identify the attributes that control the experience during the requirements phase. These attributes can then be governed by the runtime dispute resolution mechanism. If the game designer is aware of the potential for emergent behavior causing failure, they can also specify requirements for checkpointing game state before the attributes are modified. As a result of this backup, if the proposed changes are a failure, the system can be restored to its prior state and alternatives pursued.

Example We now present a more definite example for a player community *vs.* publisher dispute scenario.

Accusation	The game isn't any fun for anyone who isn't already at level 20 or higher. Given that higher level players (level 20 or higher) can engage in player *vs.* player combat, how do those of us who want to just explore and quest continue to participate in the game after we reach level 20? As it is, if we don't want to enter combat, we just get killed and we are being forced to restart.
Identification	Game designer, developer and publisher have a vision for a world that graduates players from quests to combat. One group of players strongly endorses this vision, one group of players accepts (or is ambivalent about) this vision, and a third group of players strongly opposes the resulting experience.
Advocacy	The situation was severe enough that the Publisher commissioned focus groups to evaluate possible solutions. The focus groups were principally drawn from the active player population but were also seeded with members of the game design team to gain an insider's perspective.
Resolution	Analysis of the results of the focus group lead to a new requirement for a future release of the code base.
Implementation	Game engine was modified to support partitioning players into N classes, with separate rule sets for conduct within a class and between that class and other classes.
Implications	Player population is divided into recognizable (visually identifiable) classes of players that form their own sub-communities. The first three communities are (1) those who participate in player *vs.* player combat, (2) those who participate in player *vs.* game (*e.g.* monster) combat, and (3) those who participate in puzzle related quests without any combat modes. New codebase had to be developed, tested, and pushed out to the customers. Customer support costs surged during the transition.

12.3.4 Summary

Negotiating requirements at run-time poses a number of challenges, including practical concerns around advocacy and real-time resolution, and needs to be constrained. In this section we focused on a dispute resolution model to provide some opportunity to balance requirements at run-time.

In the case of player *vs.* griefer conflict, experience requirements have shown us that we should support communities of interest, and provide players with the ability to self-identify. There are risks – the feeling of a larger community may break down, and the community may fragment into special interest groups that may not be large enough to be self-sustaining.

The discussion has shown that, if a runtime dispute resolution system is desired, that the requirements process must support requirements (at least for some aspects of the system) that are malleable, even after delivery. For example, these requirements might look like the following, where each requirement depends upon the prior requirements.

1. All requirements that affect the player experience give priority to player satisfaction as long as player satisfaction does contradict corporate (developer) policy.
2. All requirements that affect the player experience are written to support dynamic modification.
3. There shall be a set of rules that govern player behavior.
4. There shall be a set of rules that govern player *vs.* player combat.
5. A player shall have a set of attributes that capture the player state. The player state shall include, at least, the player's health, strength and agility.

Then, the relationships between the player attributes of health, strength and agility and player's chance of success in the game can be negotiated within the player community and adjusted after delivery. The space in which the modifications can occur has been constrained and the risks associated with emergent behavior have been managed, at least to a degree.

12.4 Conclusions

The publications collected in this manuscript thesis have described the introduction and development of a new requirements engineering methodology called experience requirements.

Originally called emotional requirements, the concept arose from observations that many failures in game design arise due to problems in project management. In particular, there are issues with the transmission of information across the boundary from preproduction, mostly the domain of artists, to production, mostly the domain of scientists and engineers.

Collocating preproduction and production throughout the lifetime of the project may resolve these communications issues, but this may be infeasible. For example, different parts of a game project may be subcontracted. Thus, an alternative is to transmit better information across that boundary via experience requirements.

Field experience permitted several development directions of the formalism for experience requirements. An experience requirement began as a pair - an emotion, and the means of inducing that emotion. We developed the concepts of emotional terrains, emotional intensity maps, and emotional timelines as mechanisms for visualizing these requirements.

Upon further feedback from industry, the work was extended to include emotion markers (elements of the game world that trigger the experience, situated within the world) and emotional prototypes, which extended the earlier emotional requirement to a (cue, action, goal) triple. Emotion markers typically form the cue in the triple, and the player is expected to take some action in response to the stimuli in pursuit of a goal. When combined with testing suggestions from preproduction, significant critical information is captured for the production team.

Further work showed that there can be significant interactions between emotional requirements and security requirements – experience requirements can even dominate security requirements in some cases. Experience requirements can also be used to capture player motivations, their attitudes toward the game, and their attitudes toward other players. In each example, experience requirements can be used to proactively address player needs before they become issues.

Emotional requirements were recognized as a specific type of a more general concept of experience requirements and this work has proposed an initial definition for experience requirements as well as an accompanying ontology of experience requirements for the videogame domain. Initial investigations into another type of experience requirement, the cognitive gameplay requirement, have been performed and a proposed definition has been presented.

Finally, a framework and methodology for expressing emotional requirements has been developed and presented in a manner that facilitates adoption with little or no investment in software tool support. Extending paper prototyping, physualization employs common office materials to support lightweight capture of emotional requirements.

The exploratory research captured herein has provided a greater understand of problem domain and has provided both a rich conceptual framework and initial results for the area. The proposed solution, the experience requirements methodology, is a novel contribution composed of:

- a model for the elements that compose experience requirements,
- a framework that provides guidance for expressing experience requirements, and
- an exemplary process for the elicitation, capture, and negotiation of experience requirements.

This is a rich start for an exciting domain. As our understanding continues to evolve there will be many interesting opportunities for future work. For example, breadth investigations with other game developers will help us to evaluate generalizations of the methodology. The other experience requirements that have been identified can also be investigated to develop models and frameworks for their areas.

Tool support is a promising research direction and detailed case studies of expressiveness, completeness, workflow and usability would advance our knowledge. Finally, using experience requirements to automatically implement specific player experiences, or to tune these experiences at runtime, would be challenging but rewarding.

References

[1] Ernest Adams. The No Twinkie Database. *http://www.designersnotebook.com/Design_Resources/No_Twinkie_Database/no_twinkie_database.htm*, Accessed January, 2008.

[2] Ian Alexander. Misuse cases: Use cases with hostile intent. *IEEE Software*, 20(1):58–66, 2003.

[3] Carina Alves, Geber Ramalho, and Alexandre Damasceno. Challenges in requirements engineering for mobile games development: The meantime case study. In *Proceedings of the 15th IEEE International Conference on Requirements Engineering (RE 2007)*, pages 275–280, Washington, DC, USA, 2007. IEEE Computer Society.

[4] S. Andriole. Storyboard prototyping for requirements verification. *Large Scale Systems in Information and Decision Technologies*, 12(3):231–247, 1987.

[5] Annie I. Anton. Goal-based requirements analysis. In *ICRE '96: Proceedings of the 2nd International Conference on Requirements Engineering (ICRE '96)*, page 136, Washington, DC, USA, 1996. IEEE Computer Society.

[6] M. Aoyama. Persona-and-scenario based requirements engineering for software embedded in digital consumer products. In *Requirements Engineering, 2005. Proceedings. 13th IEEE International Conference on*, pages 85 – 94, 29 2005.

[7] Consumer Electronics Association. *Digital America.* Published electronically at http://www.ce.org, 2003.

[8] Various Authors. Postmortem column. *Game Developer*, 6(5) through 11(6), May 1999 - June 2004.

[9] Joseph Bates. The role of emotion in believable agents. *Communications of the ACM*, 37:122–125, 1994.

[10] R. Bellman. *Adaptive Control Processes: A Guided Tour.* Princeton University Press, 1961.

[11] Todd Bentley, Lorraine Johnston, and Karola von Baggo. Putting some emotion into requirements engineering. In *Proceedings of the 7th Australian Workshop on Requirements Engineering*, 2002.

[12] Eric Bethke. *Game Development and Production.* Wordware Publishing, Inc., 2003.

[13] B. Boehm and V. Basili. Software defect reduction top 10 list. *IEEE Computer*, 34(1):135–137, January 2001.

[14] Karin Breitman and Julio Cesar Sampaio do Prado Leite. Ontology as a requirements engineering product. In *Requirements Engineering*, pages 309–319, 2003.

[15] David Callele. Physualization: Going beyond paper prototyping. In *MERE '10: Proceedings of the 5th International Workshop on Multimedia and Enjoyable Requirements Engineering*, pages 1–10, Sydney, Australia, 1-1 2010. IEEE Computer Society.

[16] David Callele, Eric Neufeld, and Kevin Schneider. Requirements engineering and the creative process in the video game industry. In *RE '05: Proceedings of the 2005 13th IEEE International Requirements Engineering Conference*, pages 240–250, Paris, France, 2005. IEEE Computer Society.

[17] David Callele, Eric Neufeld, and Kevin Schneider. Emotional requirements in video games. In *RE '06: Proceedings of the 2006 14th IEEE International Requirements Engineering Conference*, pages 292–295, Minneapolis, MN, USA, 2006. IEEE Computer Society.

[18] David Callele, Eric Neufeld, and Kevin Schneider. Balancing security requirements and emotional requirements in video games. In *RE '08: Proceedings of the 2008 16th IEEE International Requirements Engineering Conference*, pages 319–320, Barcelona, Spain, 2008. IEEE Computer Society.

[19] David Callele, Eric Neufeld, and Kevin Schneider. Emotional Requirements. *IEEE Software*, 25(1):43–45, 2008.

[20] David Callele, Eric Neufeld, and Kevin Schneider. Requirements in conflict: Player vs. designer vs. cheater. In *MERE '08: Proceedings of the 3rd International Workshop on Multimedia and Enjoyable Requirements Engineering*, pages 12 –21, Barcelona, Spain, 9-9 2008. IEEE Computer Society.

[21] David Callele, Eric Neufeld, and Kevin Schneider. Augmenting emotional requirements with emotion markers and emotion prototypes. In *RE '09: Proceedings of the 2009 17th IEEE International Requirements Engineering Conference*, pages 373 –374, Atlanta, GA, USA, August 2009. IEEE Computer Society.

[22] David Callele, Eric Neufeld, and Kevin Schneider. Visualizing emotional requirements. In *Requirements Engineering Visualization (REV), 2009 Fourth International Workshop on*, pages 1 –10, Atlanta, GA, USA, 1-1 2009. IEEE Computer Society.

[23] David Callele, Eric Neufeld, and Kevin Schneider. Cognitive gameplay requirements. In *MERE '10: Proceedings of the 5th International Workshop on Multimedia and Enjoyable Requirements Engineering*, Sydney, Australia, 2010. IEEE Computer Society.

[24] David Callele, Eric Neufeld, and Kevin Schneider. Introducing experience requirements. In *RE '10: Proceedings of the 2010 18th IEEE International Requirements Engineering Conference*, Sydney, Australia, 2010. IEEE Computer Society.

[25] Simon Carless. *Gaming Hacks*. O'Reilly Media, Inc., 2004.

[26] Herman Chernoff. The use of faces to represent points in k-dimensional space graphically. *Journal of the American Stat. Association*, 68:361–368, 1973.

[27] Lawrence Chung. *Non-Functional Requirements in Software Engineering*. Kluwer Academic Publishers, 2000.

[28] Mia Consalvo. Gaining Advantage: How videogame players define and negotiate cheating. In *Proceedings of the Second Annual conference of the Digital Games Research Association*, page Online. IT University of Copenhagen, 2005.

[29] Mia Consalvo. Cheating Is Good For You. *Forbes Magazine*, 12, 2006.

[30] Mia Consalvo. *Cheating: Gaining Advantage in Videogames*. The MIT Press, 2007.

[31] Chris Crawford. *Chris Crawford on Game Design*. New Riders Publishing, 2003.

[32] Robert Crook, Darrel Ince, Luncheng Lin, and Bashar Nuseibeh. Security requirements engineering: When anti-requirements hit the fan. In *Proceedings of IEEE International Requirements Engineering Conference (RE 2002)*, Washington, DC, USA, 2002. IEEE Computer Society.

[33] Mihaly Csikszenthmihalyi. *Flow: The Psychology of Optimal Experience*. Harper Perennial, 1990.

[34] Anne Dardenne, Axel van Lamsweerde, and Stephen Fickas. Goal-directed requirements acquisition. In *6IWSSD: Selected Papers of the Sixth International Workshop on Software Specification and Design*, pages 3–50, Amsterdam, The Netherlands, The Netherlands, 1993. Elsevier Science Publishers B. V.

[35] Peter E. Doolittle. Constructivism and Online Education. In *1999 Online Conference on Teaching Online in Higher Education*, pages 1–13, 1999.

[36] Stephen W. Draper. Analysing fun as a candidate software requirement. *Personal Technology*, 3(1):1–6, 1999.

[37] S. Easterbrook, S. Singer, J. Storey, and D. Damian. Selecting empirical methods for software engineering research. In *Guide to Advanced Empirical Software Engineering, F. Shull and J. Singer Eds.* Springer, 2007.

[38] Steve Easterbrook, Anthony Finkelstein, Jeff Kramer, and Bashar Nuseibeh. Coordinating distributed viewpoints: The anatomy of a consistency check. *Concurrent Engineering: Research and Applications*, 2(3):209–222, 1994.

[39] Auston Grossman Editor. *POSTMORTEMS from Game Developer*. CMP Books, 2003.

[40] Paul Ekman and W Friesen. Measuring facial movement. *Environmental Psychology and Nonverbal Behavior*, 1(1):56–75, 1976.

[41] Andrew Sloman et al. The Cognitive Affinity Project. http://www.cs.bham.ac.uk/research/cogaff/cogaff.html.

[42] S. Field. *Screenplay: the foundations of screenwriting.* A Delta book. Delta Trade Paperbacks, 2005.

[43] Donald Firesmith. Engineering security requirements. *Journal of Object Technology*, 2(1):53–68, 2003.

[44] Chek Yang Foo and Elina Koivisto. Grief Player Motivations. In *Proceedings of the Other Players conference*, page Online, Copenhagen, Denmark, 2004. IT University of Copenhagen.

[45] Chek Yang Foo and Elina Koivisto. Redefining Grief Play. In *Proceedings of the Other Players conference*, page Online, Copenhagen, Denmark, 2004. IT University of Copenhagen.

[46] Brian Fuson. *2003 Top Boxoffice.* Published electronically at http://www.hollywoodreporter.com, 2003.

[47] David A. Garvin. Competing on the eight dimensions of quality. *Harvard Business Review*, 65(6):101–119, 1987.

[48] F. O. Gehry. Reflections on designing and architectural practice. *Managing as Designing*, 2004.

[49] M. Glinz. On non-functional requirements. In *Proc. RE*, volume 7, pages 21–26. Springer, 2007.

[50] Martin Glinz. On non-functional requirements. *Requirements Engineering, IEEE International Conference on*, 0:21–26, 2007.

[51] J. A. Goguen and C. Linde. Techniques for requirements elicitation. In *Proceedings of the International Symposium on Requirements Engineering*, pages 152–164, Los Alamitos, California, 1993. IEEE CS Press.

[52] Joseph A. Goguen. The dry and the wet. In *ISCO*, pages 1–17, 1992.

[53] Philippe Golle and Nicolas Ducheneaut. Keeping bots out of online games. In *ACE '05: Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*, pages 262–265, New York, NY, USA, 2005. ACM.

[54] III H. Holbrook. A scenario-based methodology for conducting requirements elicitation. *SIGSOFT Softw. Eng. Notes*, 15(1):95–104, 1990.

[55] Marc Hassenzahl, Andreas Beu, and Michael Burmester. Engineering joy. *IEEE Software*, 18(1):70–76, 2001.

[56] Greg Hoglund and Gary McGraw. *Exploiting Online Games: Cheating Massively Distributed Systems (Addison-Wesley Software Security Series)*. Addison-Wesley Professional, 2007.

[57] Paco Hope, Gary McGraw, and Annie I. Antón. Misuse and abuse cases: Getting past the positive. *IEEE Security and Privacy*, 2(3):90–92, 2004.

[58] Cynthia Irvine, Timothy Levin, Jeffery Wilsonz, David Shifflett, and Barbara Pereira. A Case Study in Security Requirements Engineering for a High Assurance System. In *Symposium on Requirements Engineering for Information Security, Held in conjunction with the 9th IEEE International Conference on Requirements Engineering (RE'01)*. Online, 2001.

[59] ISO/IEC. *ISO/IEC 9126: Information technology – Software product evaluation – Quality characteristics and guidelines for their use.* International Organization for Standardization, International Electrotechnical Commission, Geneva, Switzerland, 2006. 1991.

[60] M. Jarke, K. Pohl, R. Doemges, S. Jacobs, and H. Nissen. Requirements information management: The NATURE approach. *Ingenerie des Systemes d'Informations*, 2(6):609–637, 1994.

[61] Raph Koster. *A Theory of Fun.* Paraglyph Press, 2005.

[62] Julian Kuecklich. Other playings: cheating in computer games. In *Proceedings of the Other Players conference*, page Online, Copenhagen, Denmark, 2004. IT University of Copenhagen.

[63] Andy Kuo. A (very) brief history of cheating. *http://shl.stanford.edu/Game_archive/StudentPapers /BySubject/AI/ C/Cheating/Kuo_Andy.pdf*, 2001.

[64] Francois Dominic Laramee, editor. *Game Design Perspectives.* Charles River Media, Inc., 2002.

[65] Pericles Loucopoulos and Joy Garfield. The intertwining of enterprise strategy and requirements. In John Mylopoulos, Kalle Lyytinen, Nicholas Berente, Pericles Loucopoulos, and Sean Hansen, editors, *Design Requirements Engineering: A Ten-Year Perspective*, volume 14, Geneva, Switzerland, 2009.

[66] Y Lyhyaoui, A Lyhyaoui, and S Natkin. Online games: Categorization of attacks. In *Proceedings of EUROCON 2005*, pages 1340–1343, Washington, DC, USA, 2005. IEEE Computer Society.

[67] K. Lyytinen, P. Loucopoulos, J. Mylopoulos, and B. Robinson, editors. *Design Requirements Engineering: A Ten-Year Perspective.* Lecture Notes in Business Information Processing. Springer, 2009.

[68] Aaron Marcus. The emotion commotion. *interactions*, 10(6):28–34, 2003.

[69] John McDermott and Chris Fox. Using abuse case models for security requirements analysis. In *ACSAC '99: Proceedings of the 15th Annual Computer Security Applications Conference*, page 55, Washington, DC, USA, 1999. IEEE Computer Society.

[70] Nancy Mead and Ted Stehney. Security quality requirements engineering (square) methodology. In *SESS '05: Proceedings of the 2005 workshop on Software engineering for secure systems – building trustworthy applications*, pages 1–7, New York, NY, USA, 2005. ACM Press.

[71] Nenad Medvidovic and David S. Rosenblum. Domains of concern in software architectures and architecture description languages. In *Proceedings of the 1997 USENIX Conference on Domain-Specific Languages*, 1997.

[72] Tim Menzies, Steve M. Easterbrook, Bashar Nuseibeh, and Sam Waugh. An empirical investigation of multiple viewpoint reasoning in requirements engineering. In *RE '99: Proceedings of the 4th IEEE International Symposium on Requirements Engineering*, page 100, Washington, DC, USA, 1999. IEEE Computer Society.

[73] David Michael. *The Indie Game Development Survival Guide.* Charles River Media, Inc., 2003.

[74] Jonathan Moffett, Charles Haley, and Bashar Nuseibeh. Core security requirements artefacts. Spiral Development Workshop Technical Report 2004/23, The Open University, Department of Computing, June 2004. Edited by Wilfred J. Hansen.

[75] Andrew Monk and Steve Howard. Methods & tools: the rich picture: a tool for reasoning about work context. *interactions*, 5(2):21–30, 1998.

[76] A. Moore, R. Ellison, and R. Linger. Attack modeling for information security and survivability. Technical Report CMU/SEI-2001-TN-001, CMU/SEI, 2001.

[77] Andrew Moore. Security Requirements Engineering through Iterative Intrusion-Aware Design. In *Symposium on Requirements Engineering for Information Security, Held in conjunction with the 9th IEEE International Conference on Requirements Engineering (RE'01).* Online, 2001.

[78] David Myers. What's good about bad play? In *IE2005: Proceedings of the second Australasian conference on Interactive entertainment*, pages 133–140, Sydney, Australia, Australia, 2005. Creativity & Cognition Studios Press.

[79] John Mylopoulos, Kalle Lyytinen, Nicholas Berente, Pericles Loucopoulos, and Sean Hansen. *Design Requirements Engineering: A Ten-Year Perspective*, volume 14. Springer Berlin Heidelberg, Geneva, Switzerland, 2009.

[80] Donald A. Norman. *The Design of Everyday Things.* Doubleday Books by permission of Basic Books, 1988.

[81] Johan Natt och Dag, Vincenzo Gervasi, Sjaak Brinkkemper, and Björn Regnell. Speeding up requirements management in a product software company: Linking customer wishes to product requirements through linguistic engineering. In *RE*, pages 283–294, 2004.

[82] U.S. Department of Homeland Security. Build security in: Setting a higher standard for software assurance, Accessed January, 2008.

[83] W. Gerrod Parrott, editor. *Emotions in Social Psychology*. Psychology Press, Philadelphia, 2000.

[84] Carl Plantinga and Greg M. Smith, editors. *Passionate Views: Film, Cognition, and Emotion*. Johns Hopkins University Press, 1999.

[85] Colin Potts. Using schematic scenarios to understand user needs. In *DIS '95: Proceedings of the 1st conference on Designing interactive systems*, pages 247–256, New York, NY, USA, 1995. ACM.

[86] Matt Pritchard. How to Hurt the Hackers: The Scoop on Internet Cheating and How You Can Combat It. *Gamasutra*, 2000.

[87] Paper Prototyping. *Carolyn Snyder*. Morgan Kaufmann, 2003.

[88] Isabel Ramos, Daniel M. Berry, and João Alvaro Carvalho. Requirements engineering for organizational transformation. *Information & Software Technology*, 47(7):479–495, 2005.

[89] Kevin Reeder. Visual storyboarding: Anthropometrics, innovation, and designing the process. In *Proceedings of the 2004 National Education Conference*, Dulles, VA, USA, 2004. Industrial Designers Society of America.

[90] Ben Reynolds. Playing a "good" game: A philosophical approach to understanding the morality of games. *http://www.igda.org/articles/rreynolds_ethics.php*, 2002.

[91] James Robertson. Point/counterpoint: Requirements analysts must also be inventors. *IEEE Software*, 22(1):48–51, January 2005.

[92] Suzane Robertson. Requirements trawling: techniques for discovering requirements. *Published electronically at http://www.systemsguild.com/GuildSite/Robs/trawling.html*, Accessed January 2008.

[93] Andrew Rollings and Ernest Adams. *Andrew Rollings and Ernest Adams on Game Design"*. New Riders Publishing, 2003.

[94] Andrew Rollings and Dave Morris. *Game Architecture and Design, A New Edition*. New Riders Publishing, 2004.

[95] Katie Salen and Eric Zimmerman. *Rules of Play: Game Design Fundamentals*. MIT Press, 2004.

[96] Marc Saltzzman, editor. *Game Design Secrets of the Sages*. Macmillan Publishing USA, 2000.

[97] Derek Sanderson. Online Justice Systems. *http://www.gamasutra.com/features/20000321/sanderson_01.htm*, 2001.

[98] Walt Scacchi. Understanding requirements for open source software. In John Mylopoulos, Kalle Lyytinen, Nicholas Berente, Pericles Loucopoulos, and Sean Hansen, editors, *Design Requirements Engineering: A Ten-Year Perspective*, volume 14, Geneva, Switzerland, 2009.

[99] Guttorm Sindre and Andreas L. Opdahl. Eliciting security requirements by misuse cases. In *TOOLS (37)*, pages 120–131, 2000.

[100] Greg M. Smith. *Film Structure and the Emotion System*. Cambridge University Press, 2007.

[101] Alistair G. Sutcliffe, Neil A.M. Maiden, Shailey Minocha, and Darrel Manuel. Supporting Scenario-Based Requirements Engineering. *Communications of the ACM*, 24(12):1072–1088, 1998.

[102] Windblown Adventure Production Team. *The Windblown Adventure, Game Design Document*. Windblown Adventure Production Team, 2008, Last access July 11, 2010.

[103] Windblown Adventure Production Team. The Windblown Adventure, YouTube video upload, http://www.youtube.com/watch?v=Gs5hEq21lQI, 2008, Last access July 11, 2010.

[104] Carroll Thronesbery, Debra Schreckenghost, and Arthur Molin. Storyboards: A medium for stealth knowledge capture. National Aeronautics and Space Administration, Knowledge Management, Internal Presentation, 2006.

[105] Axel van Lamsweerde. Requirements engineering in the year 00: a research perspective. In *International Conference on Software Engineering*, pages 5–19, 2000.

[106] Axel van Lamsweerde and Emmanuel Letier. Handling obstacles in goal-oriented requirements engineering. *Software Engineering*, 26(10):978–1005, 2000.

[107] Jeff Yan. Security design in online games. In *ACSAC '03: Proceedings of the 19th Annual Computer Security Applications Conference*, page 286, Washington, DC, USA, 2003. IEEE Computer Society.

[108] Jeff Yan and Hyun-Jin Choi. Security issues in online games. *The Electronic Library*, 20(2), 2002.

[109] Jeff Yan and Brian Randell. A systematic classification of cheating in online games. In *NetGames '05: Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games*, pages 1–9, New York, NY, USA, 2005. ACM.

[110] Nick Yee. Unmasking the Avatar: The Demographics of MMO Player Motivations, In-Game Preferences, and Attrition. *Gamasutra*, 2004.

[111] Nick Yee. Motivations of Play in MMORPGs: Results from a Factor Analytic Approach. *http://www.nickyee.com/daedalus/motivations.pdf*, Accessed January, 2008.

[112] Konstantinos Zachos, Neil Maiden, and Amit Tosar. Rich-media scenarios for discovering requirements. *IEEE Software*, 22(5):89–97, 2005.

[113] Pamela Zave. Classification of research efforts in requirements engineering. *ACM Computing Surveys*, 29(4):315–321, 1997.

Appendix A

Emotional Requirements Definition

An Emotional Requirement SPECification (ERSPEC) is defined as:

$ERSPEC$	=	$\langle \mathit{emotionalRequirements}, \mathit{gameElements} \rangle$

Where the individual components are defined as follows.

The emotional requirements are defined as:

$\mathit{emotionalRequirements}$	=	$\{\langle \mathit{emotion}, \mathit{cues}, \mathit{action}, \mathit{goal}\,[, \mathit{rationale}]\,[, \mathit{authorID}] \rangle\}$

The emotion is defined as:

$\mathit{emotion}$	=	$\langle [\mathit{emoticon},]\, \mathit{intendedEmotion}, \mathit{intensity}\,[, \mathit{temporalDescription}]\,[, \mathit{spatialDescription}] \rangle$
$\mathit{emoticon}$	=	The emotion may be depicted in pictorial form, such as via an emoticon, particularly when the emotion is identified within pictorial representations of the game world.
$\mathit{intendedEmotion}$	=	Textual label, used for communication between team members and project management. In this work, the intended emotions are drawn from Parrott's classification. The syntax p[.s[.t]] is used for textual communication, where p is a primary emotion, s is a secondary emotion, and t is a tertiary emotion (e.g. surprise, fear.nervousness, joy.pride.triumph). This label identifies the emotion that the designer intends to induce in the player via the cue.
$\mathit{intensity}$	=	A relative value for communicating about the intensity of the player experience. A value of 0 denotes the absence of the emotion, or a lack of concern for the intensity of the emotion. Scales can be numeric (such as values from 0 to 10) or descriptive (e.g. low, medium, high). The intensity is particularly useful when designing focus-group tests to validate that the player experience is as intended.
$\mathit{temporalDescription}$	=	A description of how the emotional intensity varies with time, in the scenario. The Emotional Intensity Timeline (EIT) is an example mechanism for capturing the temporal description in a compact visual representation.
$\mathit{spatialDescription}$	=	A description of how the emotional intensity varies with location, in the scenario. The Emotional Intensity Map (EIM) is an example mechanism for capturing the spatial description in a compact visual representation.

The cues are defined as:

cues	=	$\{\langle [\mathit{cueType},]\, \mathit{gameElement}\,[, \mathit{gameAttributes}]\,[, \mathit{mediaAttributes}] \rangle\}$ Any gameElement may be used as a cue to induce an emotional response in the player. Each gameElement has associated attributes for which the game designer may provide specific direction (*e.g.* the game designer may direct that a door in the scenario must have a specific openDoor sound and a specific closeDoor sound). There may be more than one cue in a given scenario.
$\mathit{cueType}$	=	An indicator of the way in which the gameElement is recognized as a cue. The mechanism may be via any one or more of the senses. In practice, for videogames, this means visual or auditory recognition - touch (haptic) may be employed but support for haptic feedback is relatively rare.

		The action and goal are defined as:
action	=	$\langle actionDescription, actionTestDescription \rangle$
actionDescription	=	A textual description of the action that the game designer expects that player to take in response to the cue. The action may provide a link to the corresponding mechanical gameplay requirement(s). (See Chapter 9 for further details of mechanical gameplay requirements.) There may not be an expected action, but if that is the case then the requirement is likely being misused to provide artistic direction. Cues without actions should be avoided.
actionTestDescription	=	A textual description of one or more example tests that can be used to validate that the specified actions are taken by the play testers during gameplay testing.
goal	=	$\langle goalDescription, goalTestDescription \rangle$
goalDescription	=	A textual description of the goal that the player is expected to formulate in response to the cue. The goal is typically related to the action via the cue (*e.g.* dodge (action) the oncoming vehicle (cue) to reach the finish line (goal)). The goal may provide a link to the corresponding cognitive gameplay requirement(s).(See Chapter 9 for further details of cognitive gameplay requirement.)
goalTestDescription	=	A textual description of one or more example tests that can be used to validate that the specified goals are being formulated by the play testers during gameplay testing. Passive (observation only) testing of goal formulation may not be possible. Invasive techniques, such as play-tester interviews, may be required.
		Other attributes:
rationale	=	*Optional* Provides the ability to explicitly capture the rationale associated with the requirement, most likely the rationale associated with the action or goal elements.
authorID	=	*Optional* If the requirement is part of a specification generated by multiple authors, then each requirement can have an associated author. The authorID is should identify the individual responsible for this requirement.
		The game elements are defined as:
gameElements	=	$\{\langle mediaAsset\,[, gameAttributes]\,[, mediaAttributes]\rangle\}$ A gameElement, or the set of gameElements, may provide a link to the corresponding sensory requirement(s). (See Chapter 9 for further details of sensory requirements.)
mediaAsset	=	An element of the game world, built by the production team. For example, elements of the world like the terrain, building, characters, sounds, and special effects are all mediaAssets.
gameAttributes	=	Attributes associated with a mediaAsset, associated with gameplay. For example, a weapon mediaAsset such as a sword may have associated gameAttributes of cost (to the player to acquire), useCost (how much energy is drained from the player with each stroke) and damage (how much energy is drained from the opponent with each stroke).
mediaAttributes	=	Attributes associated with a mediaAsset, associated with the "look and feel" of the asset. For example, a mediaAsset may be built in a certain style (*e.g.* a Gothic castle) or it may have a set of associated animations (e.g. walk cycle, run cycle).