A Proposal for Cognitive Gameplay Requirements

David Callele
Department of Computer Science
University of Saskatchewan
Saskatchewan, Canada
callele@cs.usask.ca

Eric Neufeld
Department of Computer Science
University of Saskatchewan
Saskatchewan, Canada
neufeld@cs.usask.ca

Kevin Schneider
Department of Computer Science
University of Saskatchewan
Saskatchewan, Canada
kas@cs.usask.ca

***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.

I. 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 [2]. 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 [2].

We define *experience requirements* [6] 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 [3] 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.

978-1-4244-8765-3/10/$26.00 ©2010 IEEE

II. 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 [4] 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 [2] for examples of the difficulties faced) and the cognitive gameplay requirements described in greater detail in Section VI are aimed at addressing some of these challenges.

III. Related Work

The trade press associated with game design is rich with pragmatic advice. Game designers like Rollings and Adams [15], Crawford [7] and Koster [11] and academics like Salen and Zimmerman [16] 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 [17] provides significant anecdotal evidence of the issues involved in video game production. In prior work [2] 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 [12] 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 [10] 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" [14]. For example, Loucopoulos and Garfield [13] 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 [9] 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. [13]

We shall see in Section V 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 [18] 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 [1] 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.

IV. 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 cognitive engagement process in "A Theory of Fun" [11]. 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 [8] 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.

V. 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 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

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

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

output they produced to formulate a definition for cognitive gameplay requirements.

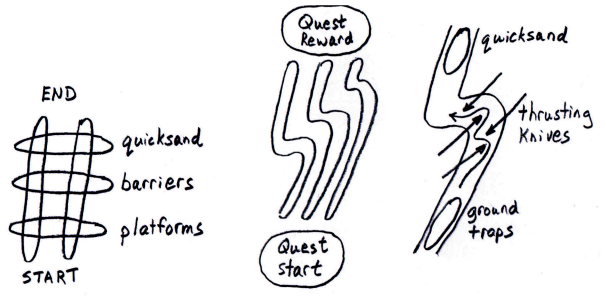
As part of the response to the RFP, the game designer generated a gameplay scenario to meet the following requirements.

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.

A. 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 1(a)).



(a) Desert Puzzle concept sketch 1 (b) Desert Puzzle concept sketch 2

Figure 1. Desert Puzzle concept sketches

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 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 1(b). 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.

B. 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, hand-written 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.

C. Gameplay 1: Platform Maze

This puzzle is simple in concept and a sketch of the platform puzzle region was developed *in situ* (Figure 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.

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

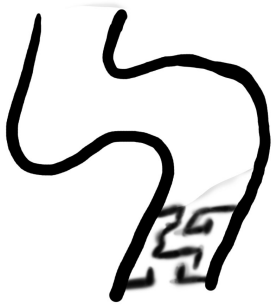


Figure 2. Platform maze concept sketch

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.

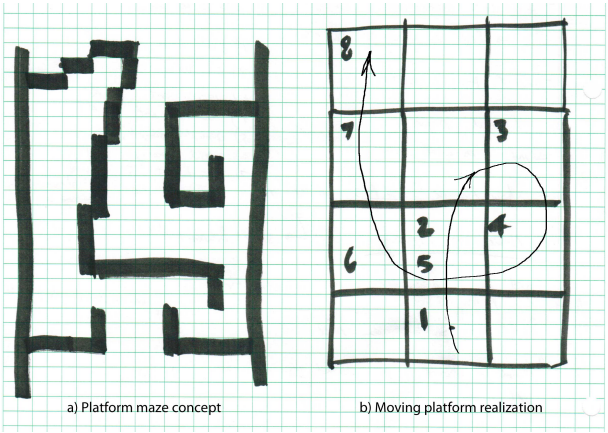


Figure 3. Platform maze design sketches

Figure 3(a) provides a view of the first detailed description of the platform maze. 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 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 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 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 playtested. From a classic, productivity-application-oriented requirements engineering perspective, the system was severely under-specified.

D. Gameplay 2: Sliding Walls

The sliding walls puzzle 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 1(a)) and later annotated to "thrusting knives" (Figure 1(b)). 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.

E. Gameplay 3: Shifting Sands

The shifting sands puzzle is similar to the moving platform puzzle at the start of the scenario. 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.

F. 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.

VI. 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.

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 [11] – 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 V-C) or via manipulation of in-game artifacts in order to continue to progress through the game (Section V-D). 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 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/failure at the task of demolishing the brick wall.

For each element 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 V-D.

A. Preconditions (Pre_i)

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 V-D.*
   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

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

B. Cognitive Challenge (Cog_i)

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 'unlearn' 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 [5]. *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 V-D.*
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 V-D.*

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

$$Cog_i = < Clues_i, Challenge_i, VandV_i(SolutionStrategy_i, SideEffects_i)$$

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

Many of the defined elements exist to ensure that the production team can develop appropriate design, verification, and validation strategies.

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 [4] or an otherwise satisfied player can turn into an individual intent upon disrupting the play experience for themselves and others.

C. Post-Conditions ($Post_i$)

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 VI-A. *Updated to reflect player performance.*
2) *World State:* See Section VI-A. *Updated to reflect player performance.*
3) *Puzzle State:* See Section VI-A. *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

$$Post_i = < PlayerState, WorldState, PuzzleState\ PlayerKnowledge, GameEngineKnowledge, SideEffects_i >$$

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

VII. 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.

VIII. 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.

References

[1] 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.

[2] David Callele, Eric Neufeld, and Kevin Schneider. Requirements engineering and the creative process in the video game industry. In *Proceedings of the 13th IEEE International Conference on Requirements Engineering (RE 2005)*, pages 240–250, Washington, DC, USA, 2005. IEEE Computer Society.

[3] David Callele, Eric Neufeld, and Kevin Schneider. Emotional requirements in video games. In *RE '06: Proceedings of the 14th IEEE International Requirements Engineering Conference (RE'06)*, pages 292–295, Washington, DC, USA, 2006. IEEE Computer Society.

[4] 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, Washington, DC, USA, 9-9 2008. IEEE Computer Society.

[5] David Callele, Eric Neufeld, and Kevin Schneider. Visualizing emotional requirements. In *Requirements Engineering Visualization (REV), 2009 Fourth International Workshop on*, pages 1 –10, Washington, DC, USA, 1-1 2009. IEEE Computer Society.

[6] David Callele, Eric Neufeld, and Kevin Schneider. Introducing experience requirements *accepted*. In *RE '10: Proceedings of the 2010 18th IEEE International Requirements Engineering Conference*, Washington, DC, USA, 2010. IEEE Computer Society.

[7] Chris Crawford. *Chris Crawford on Game Design*. New Riders Publishing, 2003.

[8] Peter E. Doolittle. Constructivism and Online Education. In *1999 Online Conference on Teaching Online in Higher Education*, pages 1–13, 1999.

[9] F. O. Gehry. Reflections on designing and architectural practice. *Managing as Designing*, 2004.

[10] Martin Glinz. On non-functional requirements. *Requirements Engineering, IEEE International Conference on*, 0:21–26, 2007.

[11] Raph Koster. *A Theory of Fun*. Paraglyph Press, 2005.

[12] Francois Dominic Laramee, editor. *Game Design Perspectives*. Charles River Media, Inc., 2002.

[13] 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.

[14] 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.

[15] Andrew Rollings and Ernest Adams. *Andrew Rollings and Ernest Adams on Game Design"*. New Riders Publishing, 2003.

[16] Katie Salen and Eric Zimmerman. *Rules of Play: Game Design Fundamentals*. MIT Press, 2004.

[17] Marc Saltzzman, editor. *Game Design Secrets of the Sages*. Macmillan Publishing USA, 2000.

[18] 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.