Requirements for game design tools

A Systematic Survey

Marcos S. O. Almeida
Computer Science Coordination
Federal University of Technology – Paraná (UTFPR)
Campo Mourão, PR, Brazil
marcossilvano@utfpr.edu.br

Flávio S. C. da Silva
Computer Science Department
University of São Paulo (USP)
São Paulo, SP, Brazil
fcs@ime.usp.br

***Abstract*—Although the computerized entertainment has shown a splendid growth in the last decades, the knowledge base and formal techniques of game design is still restricted if compared to filmmaking and software development. While games production software have clearly evolved, the game conception process still relies too much on each designer's capabilities. In this sense, efforts have been made towards establishing standardization through proposals of design toolsets and formal methods, but only few have gained attention from designer's community and none have succeeded as real production tools. While valuable, the existing implementations of these approaches have been serving only as reference to future works. In this context, this paper presents a systematization over the contributions and failures of researchers and designers aiming to elicit the requirements for game design tools. At the end, we propose a list of features for future tools and methodologies in the form or requirements.**

Keywords—Game design; game design methods; game design tools.

I. INTRODUCTION

Game designers are responsible for a game outcome, whether it succeeds or fails. It is the designer's task to ensure the game's playability. However, they are the least served in terms of tools. Development tools and methods have fairly evolved. Software developers use modern engines and state of the art techniques in order to create complex gaming experiences. Drawing and 3D modeling software have shown a huge leap in improvements, as well as sound creation support tools. Overall, the emergence of new computing technologies and processing power brought room for the evolution of production tools. However, little has been made to improve the support to game design.

Although the game industry has seen a continuous growth, designers still made primarily use of the same instruments from the earlier days of the area: text and paper. While some designers have developed alternative methods, they haven't been widely adopted. Researchers and professionals consider this lack of standard tools an important issue, a barrier to the evolution of design tools and methods. They agree that the lack of tools, whether conceptual or software, prevents the possibility of a standardization in the area and hinders the knowledge transfer between generations of designers.

Conceptual and concrete software tools have been proposed by many authors in order to complement or replace the current design tools and methods, aiming improvements to the games creation process. Mostly, they have focused efforts in some specific design threads, such as design vocabularies, collections of good design principles, libraries of reusable design concepts and visual languages for game design through visual modeling. While none of these approaches have succeeded as tools of practical use, they clearly bring some strong design needs.

This paper presents a systematization of these efforts through a chronological overview of the main approaches and their implementations, in order to elicit requirements to game design tools and methods. The main objective is to clearly propose a list of features for future tools and methodologies in the light of the current tools and previous work under analysis. In the end of each section, we present a partial requirements discussion. The full requirements list is presented at the end of this paper.

II. GAME DESIGN: THE CRAFT AND THE TOOLS

Since the earlier days of digital entertainment, game creation has been a process with strong reliance on the designer's creative skills. Fun is not an aspect achievable only through a strict formal process and a huge amount of financial resources. Contemporary low budget independent games has shown a strong success in sales and critics, most notability by its simple, yet artistically beautiful presentation and deep gameplay experience, provided by clever design. However, while good games cannot be conceived through a factory style "line production process", the use of standards and unified knowledge bases can aid the design process.

Inspiration for new games is usually crafted from books, movies, music and real world facts and culture. Furthermore, existing games typically act as a foundation for new ideas. Designers usually analyze the aesthetics of existing games while searching for elements that may contribute to their projects. Thus, experimentation is an essential part of the design process, which can be currently summarized into a three steps process: design, documentation and prototyping.

A. The Game Design Document

While designers do use physical prototypes in the form of board games and some visual aids to help them conceiving the

game, the main artifact produced by game designers is still, the Game Design Document (GDD). The main objective of the GDD is to communicate the designer's vision to the development team. As a project document, it acts as a guide to the whole development process and for this reason, it is considered by many as a production method [1].

Considered the major artifact of game design and the primary designer's task, the GDD has been subject of many discussions. Some efforts have been made in order to establish more standardized content and format, but no major step was achieved. While some authors strongly advocates the use of the document in its most complete (and long) form, many others have argued the opposite [1]. Dormans [2] states that the main inhibitor of the creation of a universal design methodology is the lack of standard in the design documents. For Keith [3] many designers agree that the document usually becomes too long, hard to used, being rarely used by developers in later development stages, serving for contractual purposes only. He also states that the size and format of the document can be factors that lead to such practice.

Many authors agree that the GDD hasn't evolved like other development tools did and the document is still used by contractual reasons. Costykian [4] advocates that even with the use of visual aids, such as sketches and storyboards, the GDD is not enough to describe the design. The static nature of a document is often highlighted as issue. Changes in the design of a game are common during the development process, due to the idiosyncratic nature of the games conception. However, the massive documentation style of the traditional GDD makes the continuous updating of the document an unproductive task as the project progresses [2]. To address this, Demachy [5] suggests the use of lightweight documentation and diagramming, and a development process that improves responsiveness to design changes. He then discusses the application of Extreme Programming, an agile software development method, for production of games.

Visual artifacts applied to the design of games is the subject of others works. Librande [6], during a presentation at GDC 2010 (Game Developers Conference), emphasized that people involved in production feel little motivation to read the GDD. He advocated the use of visual languages on game design, more expressive and compact, by presenting results of his own experience as a designer. More recently, Cerny [7] presented his own experience as a designer at the GameLab 2013 narrating the transition between heavy, immutable GDD, to lightweight, agile documentation.

B. Game Prototyping

Game development projects usually build testing prototypes of the games being produced. Those prototypes are often created during preliminary project stages, after the design documents are mostly done. Prototypes have and intense focus on gameplay and usually disregards artistic presentation. They are commonly used as a proof of design concepts and an experimentation environment to evaluate and evolve the gameplay described in the GDD [8].

Game designers have unanimous agreement about the value of experimentation through prototyping, regarded a critical part of the game development process and considered the only reliable method of verifying the design quality [9]. However, with rare exceptions, they are not able to build the game prototypes by themselves. Although there are reports of the use of game creation tools for rapid creation of game prototypes, they are considered too restrictive as they present a limited set of construction options, often aimed at end users. Another attempt for prototyping is the so called "analog prototypes". Built as board games, they simples representation of some game rules, mostly numeric based mechanics, and are made by designers with paper, glue and scissor. While valuable when analysis situations of numerical balance in games, such as in Real Time Strategy attack units or RPG combats, they are ineffective for games with interaction mechanisms heavily based on real time actions [10].

Game prototypes are usually built by software developers and graphic designers, guided by the GDD. Even with the guidance of the game designers, the prototypes are post-constructed, after a substantial part of the GDD is complete. This creates a gap between the conception of the game concepts, which are conceived during the documentation of the GDD, and the experimentation process that could attest them [9]. Prototyping becomes a costly and slow process, as it needs the allocation of specialized people to build it and as it's away from the direct control of the designer.

The importance of the prototype is especially impaired during the game conception phase, once the prototype is usually built only after the completion of a significant part of the GDD. This makes impossible for designers to do instant gameplay experiments during the game conception phase.

III. EARLY DISCUSSIONS TOWARDS NEW DESIGN TOOLS

Although widely adopted as the mainstream tool in the industry, there are lots of criticisms about the use and production of the GDD. Despite the industry success, researchers and professionals have long discussed that the lack of tools in game designing hinders the knowledge transfer and the evolution of design process [9].

In 1994, Costikyan [4] pointed the need for greater formalism on game design. He suggested the creation of a common vocabulary for game design. According to his arguments, this tool would allow designers to analyze and describe games. He advocated that designers should have a way to analyze games, understand them and identify the elements that make them good or bad. He was clearly not only referring to an ontology, but to a collection of design concepts that could be used to analyze and build game designs. While he didn't propose the tool, his speech has been echoed by many others researchers and designers.

In a later work, Church [11] pointed the lack of a common design vocabulary to describe games concepts as the main inhibitor of the design methods evolution. For him, this issue could hinder the transfer of knowledge between generations of designers. Fullerton [12] also expressed the same opinion, pointing out the lack of a common vocabulary as one of the biggest problems faced by the games production industry of its time. Over the years, practitioners and researchers have identified the need for formal models and tools to support game

design. In this context, "formal" does not refer to mathematical models, but to organized, standardized and structured models and tools to aid the game design process.

The need for a vocabulary of game design that not only standardizes names and meanings in the area, but allows the usage of a collection of design concepts is an important indication of a relevant requirement for game design. It does not only defines a tool, such as the vocabulary itself, but leads to a different approach, a concepts-oriented game design, where designers would be able to dissect a game, identify and separate its forming components, understand how they fit and balance together, and analyze which ones benefit or harm certain games or game genres.

IV. GAME DESIGN VOCABULARY APPROACHES

The approaches towards the constitution of a vocabulary of design had split into several slightly different directions. Four main branches can be summarized: collections of design concepts, game taxonomies, dictionaries of design terms and project and design guidelines. Through the following two subsections we will discuss each of these approaches in order to and summarize what design tolls requirements can be elicited from them.

A. FADT: Formal Abstract Design Tools

One of the approaches based on design vocabularies is the collection of design concepts. Following the speech of Costikyan [4], they believe that we can identify, organize and reuse the parts that compose games. Church [11] presented the FADT. The Formal Abstract Design Tools (FADT) was the first attempt to bring a vocabulary of recurring game design concepts. By the author definition, each FADT should be precise (i.e. “Formal”), unambiguous and yet applicable to as many situations as possible (i.e. “Abstract”) as a “Design Tool”. Although a “formal” approach, the FADT structure is rather simple, being composed by only two field: name and description. Also, the FADT defines very abstract concepts and designers and not game parts, as initially desired by the author. This can be noticed in one of only three terms defined by the author: "Perceivable Consequence: a clear reaction from the game world to the action of the player."

The FADT structure doesn’t allow relations with games, genres and other FADT. It also lacks usage guidelines with information about the implications of use of each FADT in a game design and it’s up to the designer to decide how to use it. In fact, Church [11] doesn’t try to expand the collection of concepts in every aspect. It rather discusses the general idea behind it. In Church’s discussion, three FADT were presented and in the end, only twenty five were documented at total [1]. Between 1999 and 2002, the Gamasutra[1] website hosted a forum where people discussed and expanded Church's tool, although there are no reports of its use on real world projects [2].

B. Patterns in Game Design

Although simple and short lived, the FADT was the first attempt to discuss the idea of a collection of design concepts. Later approaches employed more structured models to document and relate the design concepts. Kreimeier [1] suggested the application of the Design Patterns model [13] from software engineering, to document of design concepts as pairs “problem-solution”, allowing relations between patterns. As a step forward, Björk, Lundgren and Holopainen [14] used the design patterns model as inspiration to the Game Design Patterns project (GDP). Although similar to the original work of [13], they introduced significant modifications in order to better adequate it to game design. Most notably, they don’t follow the approach of pairs "problem-solution" when structuring patterns, once game concepts are not solutions to design problems: they are tools to build game concepts. The GDP project document recurring concepts of game design, representing games mechanics as well as high level concepts, like Church's FADT.

The GDP model document each design pattern by a well-defined structure, which includes name, definition, usage examples, application instructions, narrative aspects, consequences of use, and relationships with other patterns. The objective is to allow analysis and design of games through the patterns they may contain. The project has a wiki[2], in which there are almost hour hundred patterns documented mostly written by the tool authors summed to later contributions of designers. The project is referenced in a number of publications listed in the wiki.

The GDP project represents a clear evolution over the FADT. It uses the design patterns documenting structure to organize reusable design concepts, which enables the highlighting of relevant aspects of each pattern and their interrelationships. However, shortcomings hinder his usage in real projects. Like a problem pointed in the FADT, there is not enough correspondence between patterns and the games or genres that use them in order to allow a games-centered analysis. Also, by using collaborative documentation, there is a need for a moderation mechanism. But the GDP doesn’t have it. It is common to find incomplete and contradictory documentation on patterns, with disagreements between title, definition and usage examples.

Finally, FADT and GDP models lack graphical models to facilitate the design concepts understanding and to allow the visualization of the hierarchy and relationships between concepts and games. Still, just like FADT, there is no guidance on how to design games with the GDPs. Together, all cited issues makes the GDP in a little intuitive scattered collection that requires a high learning curve.

C. Mechanics and MDA

Church[11], Kreimeier [1] and Björk, Lundgren and Holopainen [14] proposed approaches based on collections of design concept, focusing on defining the structure of the collection and, at some extent, on populating the collection. On the other hand, LeBlanc, Hunicke and Zubek [15] chose a

1 http://www.gamasutra.com/ (visited on 2013/03/13)

2 http://gdp2.tii.se/index.php/Main_Page (visited on 2013/03/13)

different path. They proposed a framework to describe games through a three layer framework of interrelated components. The MDA framework separates game concepts into three dimensions: mechanics, dynamics and aesthetics. The mechanics describe the static rules of the game system, usually documented in the design document. The dynamics represent the run-time behavior of the mechanics when implemented in prototypes or games. Finally, the aesthetics comprehends the desired emotional responses of players that emerge when they are playing the game. These are usually obtained from tests and experimentations with the prototypes or games. This three layered scheme allow us to observe three distinct parts of each game concept: how we plan it, how it works inside the game and which is the outcome of it to the player.

LeBlanc, Hunicke and Zubek [15] highlighted the importance of considering both the perspective of the designer and the player when designing games. In this sense, the designer "produces" the mechanics of the game and the player "consumes" them when playing the game, which is the exact moment where aesthetics emerges. This process highlights the focus on gameplay experiences when designing the game rules, and with it, the importance of building software prototypes for testing the outcome of the designed concepts.

The attention to the aesthetics of gameplay ratifies a very common practice of designers: the investigation of the emotional result of mechanics implemented in the existing games. As part of their work, designers constantly experience a vast amount of games in an analytical way, which is quite costly. In his sense, the importance of building a database of design concepts drawn from existing games becomes evident, though neither of the existing implementations has been successful on structuring it and putting it to real use.

Similar approaches to FADT [11] and GDP [14] were discussed in the work of the "Library of Game Mechanics" [16] and the definition of game mechanics inspired by the object-orientated paradigm [17]. There is also a collection of game concepts documented in the GiantBomb[3] site, which although presents an informal approach and a focus clearly aimed at end users, without the apparent intention of establishing a tool for designers, features a simple and functional solution based on collaborative construction. On the site's database, the understanding of the concepts is facilitated through the use of illustrations. Moreover, it has a considerable range over the library of available games, relating them to the concepts used. Although devoid of formalism, this collection of game concepts presents functional characteristics that may inspire future works in the area.

D. Game Design Guidelines and Dictionaries

Still aimed at defining a collection of commonalities in game design, but not specifically focused on design concepts, Falstein and Barwood [18] initiated the "The 400 Rules Project" in 2002. It aimed to identify, to record and to share a list of practical experiences of designers, thus indicating directions to be taken or avoided into a game project. Although the project title mentions 400 rules, only 112 were documented on the project's website[4]. Once more, the collaboration of various designers was a key component in order to define and document the practices, which also demonstrated an intention to make it an artifact of practical use (like the GDP project). Fabricatore, Nussbaum and Roses [19] also worked towards a list of design principles from an analysis of the influence of gameplay mechanisms in player's motivations. Overall, both projects aims to aid in the knowledge transfer between designers. However, guidelines alone don't constitute a tool that actually supports the idiosyncratic process of games creation, but help to avoid pits.

Other works addressed specifically the subject of defining a game design dictionary. Two of these projects were documented in publications, being the Videogame Lexicon [20] and the Games Ontology [21]. These projects aimed to create shared design dictionaries to provide unambiguous definitions of terms commonly used in game design. As a tool, these dictionaries would be digitally available to searching and to be embedded in design documents via XML or other markup languages. The Games Ontology project was published as a wiki[5], as an attempt to allow collaborative construction of the dictionary. There were a total of 179 definitions of design and videogames terms. Like the other works already discussed in this section, there were a strong focus on theoretical study and few results of practical use. Kreimeier [1] observed that although dictionaries alone cannot constitute design tools, they are necessary foundations to any method or conceptual tool. In fact, all the approaches addressed in this section aimed, at some extent, on establishing a collection of definitions for game design commonalities, which leads to the conclusion that this really is desired and important for designers.

Related works with vocabulary comprised studies about videogame taxonomies. Age, target audience, purpose and genre are examples of criteria used to classify games. The most popular is the genres classification, which organizes games with similar interaction characteristics under the same group. As a common language practiced by industry, academia, specialized media and end users, the hierarchical system of genres and subgenres are the most concrete example of use and value of a games vocabulary. Lindley [22] proposed a non-hierarchical model of games taxonomy called "Orthogonal Taxonomy". It defines a spatial system of characteristics, where games and genres are positioned according to their proximity to these characteristics, such as simulation, ludology and narratology.

The terminology used in games classification helps to define what to expect from a specific game type. It also it helps to identify the typical elements of each game genre and how the mixing between genres occurs. This is specially valuable as further studies could relate the taxonomies in games and the collections of design concepts in order to investigate which quantitative and qualitative relationships can be drawn between elements, market and user preferences. The outcome of these

3 http://www.giantbomb.com/concepts/ (visited on 2013/03/13)

4 http://www.finitearts.com/Pages/400page.html (visited on 2013/03/13)

5 http://www.gameontology.com/index.php/Main_Page (visited on 2013/03/13)

studies could suggest which elements are desirable for a particular game project that fit into a classification.

V. VISUAL LANGUAGES FOR GAME DESIGN MODELING

Aside from the discussed approaches that addressed the elaboration of a standard vocabulary for game design, and specially trying to define games as a composition of design concepts, other authors have focused on attempts to set up a visual language for game design. Although the currently main design documentation model, the GDD, has predominantly textual content, designers often embed visual artifacts into the document as auxiliary communication tools [9]. In such cases, game graphics are previewed with the use of conceptual illustrations and, diagrams and story boards are often used to describe game events and characters behaviors inside design documents.

Some designers have found in the visual modeling a strong ally in communicating their game vision to the development team. However, both this practice and the visual language used are not standardized. These designers often create their own visual notations to express a portion of the design [8]. The Librande's [6] "One Page Design" graphical schemes are an example of this practice. It comprises a sort of "game design map" freely created with textual and visual artifacts that represents an overview of the design for a game. He hangs the design map in a place where all development staff can easily access in their everyday work. Librande [6] emphasizes that visual models are more synthetic, naturally communicative and scale better. He used his schemes in various projects with success, as he states that they facilitated the understanding and update of the game's design.

Kuittinen [23] tried to create visual associations of Björk's Game Design Patterns through a software called CAGE – Computer-Aided Game Design. As one of the few initiatives to build concrete tools that can be integrated into the current design standards, his goal was to allow designers to select the patterns that will constitute the game concept and visualize their inter-relationships in a diagram. The descriptions of the selected patterns are then integrated into the design document. Although simple and academic, this work represents an interesting attempt to apply a conceptual model and to integrate it to the methods currently used in industry. This integration with current design tools should be a must-have feature of any attempt to bring new tools or methods to real usage, as it is very unlikely that someone will dispose all of their working tools, which has its value, to riskily try something totally different.

As another example of an attempt to bring visual artifacts to game design through computational support, the software Sketch-It-Up! [24] provides a visualization of the design narrative via *animatics* (animated sequences of game play), created during a brainstorming session by the participants, which simultaneously interact in the software. Although not related to any formal approach of design modeling, the Sketch-It-Up! is another proof of the need for computational support on game design.

Librande [6] observed that the diagramming practice forces designers to extract the essence of the gameplay in few visual elements, driven to specify the relationships between the components of gameplay through a top-down approach that breaks larger problems into smaller concepts. In fact, visual languages for systems designing are not a novelty. Software development has been using visual diagramming in the entire production process for decades. The Unified Modeling Language (UML) [25], a comprehensive vocabulary of visual languages for various diagrams has been a proven approach long applied in software engineering. The fact that designers tried to use informal approaches to document the game concepts leaded them to pure textual language, far from structured visual languages. However, designers are gradually seeking for structured languages. Many authors have suggested the use of formal approaches to build game design diagrams, such as Petri Nets and UML. They are attempting to bring standardization to design documentation through these languages.

The use of visual languages for representing the design resembles the requirements and design modeling on the software development. This similarity has been noted by some authors, who discussed the use of UML diagrams to create design maps. Sicart [17] presented a definition for game design based on the concept of object orientation and suggest the use of UML for modeling. While discussing the appliance of agile methods of software development into the production of games, Demachy [5] also pointed to UML as a game design diagramming tool, especially for describing the elements of gameplay.

Some publications have reported study cases about the appliance of UML in the designing of games. As an example, Blumenthal [26] designed the Space Invaders game as a pedagogical demonstration, using UML to capture the requirements and design of the game. However, as can be seen in these publications, raw application of UML into game design will not work effectively. Further studies of UML application must be performed to make it suitably adapted to the needs of game design.

Finally, in contrast to design modeling, some academic approaches had focused on a lower level of game description. They are trying to put together a language of logical constructions to allow the representation of game mechanics, often described as the game's rules. Brom and Abonyi [27], Araujo and Roque [28] and Natkin and Vega [29] have addressed applications of Petri Nets for this mechanics representation. Koster [30] and Bura [31] presented adaptations of Petri Nets with several own customizations to the same purpose. In the practical field, Dorman [2] and Smith, Nelson and Mateas [32] built software tools for mechanics modeling and simulation in real-time. Dorman's Machinations employ a Petri Nets based visual language to specify the mechanics of the game. Once completed, the rules in the diagrams can be executed, allowing a "game simulation". Similarly, Smith, Nelson and Mateas tool called Ludocore allows the same, but uses a high level textual programming language to describe the game mechanics models. Although promising, both languages are mostly unintuitive and the "game simulation" of these tools focuses solely on the "producer" perspective, not allowing experimentation of the "consumer" point of view through the game aesthetics.

VI. REQUIREMENTS GATHERING

After reviewing the current design tools and the major attempts to bring contributions do the game design process, we will analyze them and, in the light of their previous attempts, we will propose a list of features as requirements for future design tools and methodologies.

A. Gathering requirements from current design tools

Kreimeier [1] surveyed the state of the art of game design tools and methods of its time and suggested that more agile methods were needed. In a similar discussion, Neil [9] pointed that while designers are searching for more formal and structured tools, these cannot impose restrictiveness and work load.

The GDD has been the standard documentation artifact since the earlier steps of the games creation profession, probably a legacy from the old software specification documents, later replaced by more synthetic approaches in that area. As already pointed, the GDD lacks standardization, both in textual structure and visual aids. As narrated by some designers, visual languages are well suited for the documentation and communication of the designer's vision. Moreover, a more agile approach for game design is desired once changes often occur in game projects and preventing them is not a practicable solution.

Regarding the prototyping process, designers need a tool that would allow them to build experimental prototypes directly from the definition of a set of game characteristics. This could provide a way to perform instant proof of concepts while conceiving the game, through an iterative process of design and testing of gameplay.

We can gather the following requirements based on what was pointed by designers and researchers as necessities:

- The design process must allow changes in the game definitions. Changes are necessary once the gameplay evolves along the game conception.
- The design documentation must be lightweight and minimalist: it cannot adds workload to the designer;
- The design documents must be easy to create and maintain by using standard languages or tools;
- The design documentation must allow the use of visual languages in order to express design concepts through visual models.
- The design tools must provide a way to build some "design knowledge base" in order to allow the reuse of already proven past concepts and ideas.
- The prototype tools must provide a Game Design IDE (Integrated Development Environment) in order to allow designers to visually model and "playtest" games instantly.

In fact, some of these gathered requirements appear to be rather vague, not indicating a straight direction to follow in possible future design tools. However, they will serve as foundation to the analysis of the new approaches of design tools and methods, proposed by various authors since 1994, especially because these approaches mostly focused on fulfilling these requirements.

B. Gathering requirements from the proposed approaches

The works towards improved design instruments previously presented mainly focused on two approaches: the conceiving of a shared vocabulary of design concepts and the set up of a standardized visual language for design. Their aim can be mostly synthesized as to improve communication and knowledge transfer through more standardized and structured documentation. Thus, they expected to bring contributions to the whole game design process, as the design documents are its guidance.

Four main approaches of conceiving a shared design vocabulary were discussed throughout the present paper: collections of design concepts, design guidelines, dictionaries of terms and taxonomies in games. Although all of them are described by their authors as design tools proposals, their use with such objective is not accomplished. Overall, all approaches show an evident lack of maturity and computational support for adoption and experimentation in real world scenarios. On the other hand, the approaches based on dictionaries, taxonomies and guidelines do have an intrinsically practical nature and although cannot surface as tools, they are significant contributions to the conception and adoption of a common vocabulary.

The approaches based on collections of design concepts has promising applications as supporting tools for game analysis and design, as they may allow designers to see games as a collection of small, structured and interrelated parts that can be assembled in a different combination to create new games. Furthermore, the concept that games are made upon groupings of smaller parts is very intuitive, as we can observe a lot of commonalities between elements of the existing games, whether they belong to the same game genre or not. The problem is not the concept itself, which is fairly accepted, but how to structure the collection of components and what exactly they are. This is where the real challenge lies: the building of a database of design knowledge that can be easily accessed, that allows to be evolved and can be mapped to the existing games in a productive way. If accomplished, this tool may help to improve the entire process of game design. However, the existing implementations still do not accomplish this.

The GDP is the most significant contribution in trying to set up a database of design concepts. The use of a collaborative construction tool, such as a wiki, shows a clear intention of turning the GDP into a practical tool of game design. Church [11] tried the same with the FADT project by allowing designers to collaboratively propose and discuss the FADTs through a forum hosted in the Gamasutra website. Therefore, any attempt to bring a database of design concepts for real use needs strong support by software tools. The tool must provide guidance to designers in order to the correct organization of each concept. As the collection of concepts grow, it is needed a place to store and management them. Furthermore, advanced search and application mechanisms are necessary to allow the use of the collection. Both FADT and GDP fail in this

requirement. A forum is a tool made for discussion, not a system for content management. The same occurs in GDP, once he wiki documentation and navigation model hinders both the understanding and the use of the patterns. Aside software support, both approaches tried to define a formal structure of the collection, most notably the GDP, with the use of the design patterns structure of description and interrelation. Also, both tried to allow collaboratively construction of the collection of games concepts, as they realized that is impracticable to one designer to build and maintain the whole database, of possible hundreds or thousands of design concepts.

Regarding the attemps to bring visual languages to game design, we have found two main approaches: game design modeling and mechanics composition. Mostly aided by software tools, these languages have one single purpose: to facilitate communication. In order to not "reinvent the wheel", their authors have drawn inspiration over proven languages from other areas, especially software engineering, which is a clever decision, as these known languages have a strong user base, built along many years of proven usage. These known languages are not strictly related to programming only, but to a wider focus of analysis and design of systems, whether software or not. Games can be approached as object-oriented systems, and thus, adaptations of UML or other standards can be accordingly applied.

A key element that has not been explored in the previous works discussed, with the exception of the work of Kuittinen [23], is the integration between the collection of design concepts and the visual language for design modeling. Diagrams of the design of a game could be made through the gathering of the concepts that the game implements. It would make easer to see the interrelations between these concepts and, the games and genres that employ them. Beyond that, it would take the visual documentation to a more concrete level, once the concepts of a game wouldn't be more just a collection of scattered game parts. Although designers who heavily base their work on narrative techniques may show some resistance to the use such formal visual languages, these languages represent a trend and a necessity already highlighted by researchers and practitioners.

By bringing together the features discussed and reviewed in both design vocabularies and visual languages approaches, we may provide the following requirements:

- It must provide software tools to aid game design, not only at a conceptual level;
- It must provide integration with the tools currently used in game industry, mostly the design document and the prototyping approach.
- It must define a formal structure for the concepts documentation, with well defined fields and relations;
- It must provide guidance on how to build and use the collection;
- It must relate concepts, games and genres;
- It must allow designers to use and extend the concepts in order to compose the design of new games;
- It must considerer both the designer's perspective (game rules) and the player's perspective (aesthetics) when describing each concept;
- It must provide analysis of concepts, games and genres related to market, critics and player data.
- It must provide a formal visual language to model games through assembling of smaller concepts and to allow visualization of relations between them.
- The visual modeling must address different aspects of the design, both high (design overview) and low level of abstractions (concepts details).
- The visual language should be based on a proven existing language from other area, but must be specifically tailored to game design.
- It must provide a database tool in order to manage the concepts collections, its maintenance and usage.
- It must allow a collaborative environment in order to allow designers to work and evolve the collection.
- It must have a moderation mechanism to ensure that erroneous concepts are not added to the database.

VII. FINAL THOUGHTS

Game design lacks a shared tool box that contains both solutions of broad application and specific to certain genres of games. Researchers, renowned designers and independent developers search for better design tools. While game programmers, graphical designers and musicians are well served by sophisticated tools powered by constantly improving software and hardware technologies, designers still work with text tools, which seems a primitive solution when consider the matter of their work: the conception of complex interaction mechanisms. Beyond that, there is even no standardization on the current design tools, though most authors agree that the use of standardized tools can bring industry and academia closer, contributing to build a universal knowledge base of game design.

In general, designers crave for productive and standardized tools and techniques that do not sacrifice the freedom and creativity inherent to their craft. Several attempts to set up new tools have been made and some overall characteristics are clearly identifiable across them: more agility, less workload and more standardization. However, none of the previous approach had succeeded as tools of practical use. In these implementations, the use of unfamiliar and counter intuitive languages, the high learning curve required to use them and the uncertainty of practical gains, keep designers away. Moreover, these tools have been evaluated in academic environments, applied to restricted work groups, usually composed by academics and beginners, which does not reflect the area. Even its authors are uncertain about the gains that such tools will bring to other designers and highlight the need to evaluate them on the field. Neil [9] proposed to carry out such evaluation.

Beyond than recognizing the designer's wishes, we needed to know their particular methods, those not widely

documented. The commercial success of the industry since its beginning cannot be overlooked, a fact that certainly contributes to certify these “secret” methods. Valuable tools may have already been developed to a project and thrown away. The work of Librande [6] is an example of such “secret weapons” that need to be known. Becoming aware of such tools, crafted for specific design situations, may contribute with the approaches discussed in this paper towards a productive design toolbox. The fact that industry and academia agree about the needs of game designers indicates that both know what must be done, but the rejection of using the conceptual and physical implementations available, makes it clear they yet do not know how to do it.

REFERENCES

[1] B. Kreimeier, Game Design Methods: A 2003 Survey. Gamasutra, Mar. 2003. http://www.gamasutra.com/view/feature/2892/game_design_methods_a_2003_survey.php, 2003.

[2] J. Dormans, Engineering Emergence: Applied Theory for Game Design. Doctoral Dissertation. Amsterdam University of Applied Sciences, 2012.

[3] C. Keith, Agile Game Development with Scrum, Addison-Wesley Professional, 2010.

[4] G. Costikyan, I Have No Words & I Must Design. Interactive Fantasy, Eng., n2, 1994.

[5] T. Demachy, Extreme Game Development: Right on Time, Every Time. Gamasutra, Jul. 2003, http://www.gamasutra.com/view/feature/2827/extreme_game_development_right_on_.php, 2003.

[6] S. Librande, One-Page Designs. Presentation at GDC 2010, San Francisco CA, Mar. 2010. http://stonetronix.com/gdc-2010/, 2010.

[7] M. Cerny. The Road to the PlayStation 4. Presentation at GameLab Conference 2013. http://www.ign.com/videos/2013/06/29/the-road-to-the-playstation-4, 2013.

[8] K. Salen, E. Zimmerman, Rules of Play: Game Design Fundamentals. MIT Press. MIT Press, 2003.

[9] K. Neil, Game Design Tools: Time to Evaluate. Proceedings of DiGRA Nordic 2012 Con-ference: Local and Global – Games in Culture and Society, 2012.

[10] T. Sigman, The Siren Song of the Paper Cutter: Tips and Tricks from the Trenches of Paper Prototyping. Gamasutra. http://www.gamasutra.com/view/feature/2403/the_siren_song_of_the_paper_.php, 2005.

[11] D. Church, Formal Abstract Design Tools. Gamasutra, Jul. 1999. http://www.gamasutra.com/view/feature/3357/formal_abstract_design_tools.php, 1999.

[12] T. Fullerton, C. Swain, S. Hoffman, Game Design Workshop: Designing, Prototyping, and Playtesting Games. CMP Books, San Francisco, 2004.

[13] E. Gamma, R. Helm, R. Johnson and J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1994.

[14] S. Björk, S. Lundgren, J. Holopainen, Game Design Patterns. Level Up - Proceedings of Digital Games Research Conference (DiGRA). Utrecht University, 2003.

[15] M. LeBlanc, R. Hunicke, R. Zubek, MDA: A formal approach to game design and game research. Proceedings of the AAAI-04 Workshop on Challenges, 2004.

[16] A. Järvinen, Games Without Frontiers: Theories and Methods for Game Studies and De-sign. Doctoral Dissertation. University of Tampere, 2008.

[17] M. Sicart, Defining Game Mechanics. Game Studies: The International Journal of Computer Game Research, v. 8, n. 2, dez. 2008. http://gamestudies.org/0802/articles/sicart, 2008.

[18] N. Falstein, H. Barwood, More of the 400: Discovering Design Rules. Presentation at GDC 2002. http://www.gdconf.com/archives/2002/hal_barwood.ppt, 2002.

[19] C, Fabricatore, M. Nussbaum, R. Rosas, Playability in Action Videogames: A Qualitative Design Model. Proceedings of Human-Computer Interaction 17:4, 311-368, 2002.

[20] P. Burkart, Discovering a lexicon for video games: New research on structured vocabularies. International Digital Media and Arts Association Journal, 2(1), 18-24, 2005.

[21] J. P. Zagal, M. Mateas, C. Fernadez-Vara, B. Hochhalter, N. Lichti, Towards an Onto-logical Language for Game Analysis. Proceedings of the DiGRA, Canada, Jun. 2005.

[22] C. Lindley, Game Taxonomies: A High Level Framework for Game Analysis and Design. 2003, www.gamasutra.com/view/feature/2796/game_taxonomies_a_high_level_.php., 2003.

[23] J. M. Kuittinen, Computer-Aided Game Design. Master’s Thesis in Information Technology, University of Jyväskylä, Jan. 19, 2008.

[24] B. Karakaya, C. Garcia, D. Rodriguez, M. Nityanandam, N. Labeikovsky, and T. Al Tamimi. Sketch-It-Up! Demo. Entertainment Computing–ICEC: 313–314, 2009.

[25] M. Fowler, UML Distilled: A Brief Guide to the Standard Object Modeling Language . Boston, MA: Addison-Wesley, 3a ed, 2004.

[26] R. Blumenthal, Space Invaders: A UML Case Study. Regis University, class notes, 2005.

[27] C. Brom, A. Abonyi, Petri-Nets for Game Plot. Proceedings of AISB, Vol. 3, 2006.

[28] M. Araújo, L. Roque, Modeling Games with Petri Nets. Proceedings of 2009 Digital Games Research Association Conference (DiGRA). Brunel University, 2009.

[29] S. Natkin, L. Vega. A petri net model for computer games analysis. International Jour-nal of Intelligent Games Simulation 3 (1): 37-44, 2004.

[30] R. Koster, A Grammar of Gameplay. Presentation at GDC 2005, San Francisco CA, Mar. 2005. http://www.raphkoster.com/gaming/atof/grammarofgameplay.pdf, 2005.

[31] S. Bura, A Game Grammar, http://www.stephanebura.com/diagrams/ 2006.

[32] A. M. Smith, M. J. Nelson, M. Mateas, LUDOCORE: A logical game engine for model-ing videogames. CIG 2010 IEEE Symposium, 91–98, 2010.