Design Principles for Serious Games Authoring Tools

Maxence Laurent[1], Sandra Monnier[1], Audrey Huguenin[1], Pierre-Benjamin Monaco[1], Dominique Jaccard[1]
[1]*Media Engineering Institute, University of Applied Sciences Western Switzerland*
{maxence.laurent, sandra.monnier, audrey.huguenin, pierre-benjamin.monaco, dominique.jaccard}@heig-vd.ch

Abstract

Serious game development involves a multidisciplinary team of teachers and computer scientists. But the difference in computer competencies between the team members is a recurring difficulty in this collaboration. Authoring tools, which provide interfaces adapted to users' competencies, are promising solutions to overcome this difficulty. However, existing authoring tools are either limited in their functionalities (not powerful) or too complex for non-computer scientists (not usable). A comprehensive set of design principles to address this limitation does not yet exist.

The objective of this research was to define a set of design principles for the development of powerful and usable authoring tools.

To achieve this objective, we first defined a set of design principles. We then developed an authoring tool corresponding to these principles. Finally, we carried out test uses of that tool through the development of twelve serious games. Results show that this authoring tool enabled the development of a wide variety of serious games (powerful) by teams with heterogeneous computer skills (usable).

Design principles defined in this research integrate and extend previous works. They allow to overcome the dilemma between the power and usability of authoring tools. This could unlock new possibilities for collaborative approaches in serious games developments.

Keywords: *Authoring Tool; Authoring System; Game Editor; Design Principles; Educational game; Collaborative Development.*

1 Introduction

In this article, we use the term serious games in the sense of the definition by [1], thus encompassing any kind of digital learning games or simulation created for educational or training purposes.

1.1 Serious games development

Serious games development requires a multidisciplinary team, including teachers and game developers [2]–[5]. Involving teachers as from the beginning of the development is recognized as a key success factor of the relevance of the resulting serious game [2], [3], [5]. However, this multidisciplinary collaboration is known to be difficult. First of all, ensuring the coherence of the partial contribution of the different authors is hard [6].

Secondly, a large part of the serious game content has to be developed by teachers, or subject-matter specialists, who may have few computer competencies [7]. The lack of efficient tools enabling the contribution of those non computer specialists is perceived as one of the main obstacles to serious games development [8].

1.2 Authoring tools as a support to serious games development

Authoring tools are seen as a possible solution both for supporting the collaboration and involvement of non-computer scientists in the development team. Authoring tools are expected to offer two main advantages: (1) allow non-computer scientists to take part in the development of the serious game and its content, and (2) reduce development costs by allowing the reuse of developed elements and increasing authors productivity [7], [9].

1.3 Existing authoring tools

Game engines, such as Unity [10] or Unreal Engine [11], are widely used for the development of entertainment games, but do not take into account all the specificities of educational games. A first specificity is the integration in the development team of teachers with few computer skills. Other specificities are linked to the educational context that require specific features, such as teachers dashboards [12], [13] or learning analytics [14], [15].

During the last 10 years, several efforts have been made in order to develop specific authoring tools for serious games, such as [16]–[19]. However, current serious games authoring tools are not always adapted to non-computer scientists [20]. Current serious games authoring tools can be classified into two categories. In the first category, authoring tools offer a wide range of features, but are too complex for teachers. In the second category, they are simple to use, but offer limited features [20].

A first step toward an authoring tool that is both features-rich and usable has been made with the “JEM Inventor mobile learning game authoring tool” [21]. But JEM Inventor is limited to mobile learning games and Android devices. For example it doesn't seem suitable for the development of a serious game for medical anamnesis. It also does not include multiplayer functionality, and therefore does not allow for the development of the full range of competitive or collaborative game models.

None of the existing authoring tools integrate the following features: possibility of collaborative development by a multidisciplinary team, wide range of features and possible game models, ease of use for non-computer scientists, and specific educational features [8], [20], [21].

1.4 Challenges for serious games authoring tools development

In this section, we describe the main challenges that need to be addressed to develop the next generation of serious games authoring tools.

1.4.1 Power or usability

A recognized challenge for authoring tools design is the tradeoff between the power of the tool and its usability [9], [21]. The power of the tool is related to its flexibility and to its ability to allow the development of a wide range of games. The usability is related to the simplicity and efficiency of use. More powerful authoring tools that are more powerful are usually less usable [9], [20]. For example, an authoring tool in which the game design is done by choosing options among proposed checkboxes may be easy to grasp (usable), but will likely limit the range of possible games (not powerful). On the other hand, an authoring tool with a multitude of functionalities would enable the development of many kinds of serious games (powerful), but may be nearly as difficult to learn as a programming language (not usable) [7], [9]. This tradeoff between power and usability is linked with the abstraction level proposed by the authoring tool. A low abstraction level means that the game editing

system is close to the programming code. It is more powerful, but less usable by non computer scientists. A high abstraction level corresponds to a game editing system with interfaces allowing visual programming or check boxes to choose between predefined options. It is more usable for non computer scientists but offers less possibilities and is less powerful.

In order to develop a tool that is both powerful and usable, it is considered necessary to differentiate levels of abstraction according to users competencies and needs [7], [9], [20], [22]. At one end of the spectrum, computer scientists need an authoring tool that allows as much flexibility as possible. They feel confident and efficient with a low level of abstraction, close to the code, allowing the development of any desired features. At the other end of the spectrum, teachers with few computer skills do not need advanced features and prefer an authoring tool with a high level of abstraction.

1.4.2 Usability throughout the entire game life cycle

Serious game life cycles include game development and game usage. Collaborative and agile processes are considered the most appropriate for serious games development [5]–[7], [23]. Authoring tools should thus enable a heterogeneous team to simultaneously perform a wide range of tasks [24]. It should provide interfaces that allow different users to access the same game editor with different levels of abstraction, while having an up-to-date view of the work done by others. Once the game is developed, it is necessary to share it with teachers. And, when using the games in training sessions, several teachers must be able to collaborate and interact with the student-players.

1.4.3 Managing complex projects with limited budget

When designing and developing serious games authoring tools, the development team faces the challenge of creating a complex software application with limited resources. Developing a serious game is a complex activity, but developing an authoring tool that enables the development of a variety of serious games is moving at a higher level of complexity [9]. And the simpler the authoring tool appears to the end user, the greater its internal complexity may be. Meanwhile, the market for educational serious games authoring tools is limited [9] and therefore development budgets, outside research budgets, are also limited.

This budgetary constraint is found again when using the authoring tool for the development of serious games. While traditional games require the integration of playful aspects, serious educational games need to integrate both playful and educational aspects. However, due to their limited markets, educational games projects often have limited budgets compared to pure entertainment game projects. This confirms the need of reducing development costs by the reuse of previously developed elements, as proposed by [7], [9].

1.4.4 Authoring tools design principles

Design principles may serve as guidelines for the development of serious games authoring tools. Gicquel et al. [8] propose some design guidelines, which are mainly recommendations on the overall process, such as using an iterative approach, user-centered design, incremental design or tests in real world situations. Results of the European project “RAGE” [25] included recommendations on the necessary design conditions for facilitating the reuse of components through different development platforms [26], but does not include an overall set of design principles for developing serious games authoring tools. Research in scientific databases do not enable us to find a complete set of design principles, dedicated to serious games authoring tools, that can be used as a basic guideline by the authoring tool development team.

2 Objectives

To address the challenges described above, we formulated the following research question:

- Which design principles could enable the development of a serious games authoring tools that integrates the following Characteristics (C):
  - C1: Powerful. The tool enables the creation of a wide variety of game models for a wide variety of educational areas.
  - C2: Usable. The tool is usable by a wide variety of users
    - C2.1: with various computer competencies and needs.
    - C2.2: all along the project life cycle (development process and usage in educational context).
  - C3: Reusable. Components developed for one game are reusable in other games' developments.

Partial answers exist for several of these characteristics. However, there is currently no example of their integration into a single set of design principles.

3 Methods

3.1 Pragmatism as the main research paradigm

Serious games authoring tools design and development is a practical problem, in the sense that is about finding concrete solutions to real-world problems [27, p. 36]. Thus pragmatism, as mainly practice-driven [28], was considered as an appropriate research paradigm.

Pragmatism was furthermore found to be appropriate for the following reasons:

- In pragmatism, the research is problem-centered, experimental and utility oriented [29]. This was appropriate for our research, as an experimental approach should lead to the proposal of authoring tools design principles that may be useful for futures authoring tools development.
- Pragmatism is oriented toward finding solutions to practical problems, and results are to be judged by their practical utility [27, p. 36]. As shown in the background, there is currently a lack both in existing serious games authoring tools and in authoring tools design principles. Pragmatic research, based on design and development experimentations, should provide design principles that may have practical utility for the development of future serious games authoring tools.
- In pragmatism, research results are not to be considered as positivist immutable schemes for explaining the world, but as constructivist creation, useful if they help to solve practical problems [27, p. 36], [30], [31]. We hypothesize that at the current state of knowledge and technology, while authoring tools are still in their early stages, we will not be able to define definitive design principles. But it is rather possible to propose a set of design principles that can be used as a basis to help to solve the concrete problem of authoring tools development.

3.2 Research methodology

Pragmatism argues that the appropriate method is 'what works' to answer the research questions [31]. Methods can be any combination of qualitative and quantitative approaches, including case studies, experimentation, surveys or any other approach [27, p. 36], [31]. To answer our research question, we defined the following combination of methods.

3.2.1 Users' categorization

A categorization of users of serious games authoring tools was defined based on the project life cycle of an authoring tool, and on users' competencies.

3.2.2 Hypothesis of authoring tools design principles

A set of design principles was defined based on a combination of published theoretical design principles, results of users' categorization and the needed characteristics of authoring tools. We did the following hypothesis (H):

- H1: the implementation of these principles into a single authoring tool is possible
- H2: an authoring tool that integrates all those design principles would have the characteristics C1 (usable), C2 (powerful) and C3 (reusable components).

3.2.3 Tests of hypothesis

In pragmatism, theories have to be evaluated through their utility, their observable practical consequences, and their links with experience [27, p. 36]. Thus, we tested the practical consequences of the set of design principles through two experimentations.

H1 was tested through the development of an authoring tool that will integrate all design principles.

H2 was tested by using the authoring tool in different uses-cases to evaluate whether it integrates the desired characteristics (C1, C2 and C3).

Use-cases were selected to ensure that they met the following conditions:

- Variety of game models. In order to test the ability to develop different game models, use-cases should integrate different types of games (e.g. management games, medical simulations or mobile games).
- Multi-disciplinary development team. To test the ability to offer appropriate abstraction levels to different roles, the development team should include at least one person of the "computer science side" (e.g. game developer) and one person of the "non-computer science side" (e.g. teacher without specific computer education).
- Ecologically valid. Use-cases should encompass the entire process, from development to usage in real teaching conditions.

During use cases, we chose observation as the main strategy for data collection as it enabled us to directly observe how game developers and trainers used the authoring tool, providing more authentic data than with mediated methods [27, p. 561]. We made observations both on facts (such as how many scenarists were involved in game development) and of qualities (such as usage of the authoring tool by users with different computer competencies). Because observation of qualities may depend on researchers' interpretation, interpretations made by one author were then confirmed by at least two other authors.

4 Results

4.1 User categorization and needs

Users were categorized along two dimensions. The first dimension is linked to the project life cycle, which includes serious games development and usage [7]. The second dimension is linked to users' computer skills, which is linked to the required abstraction level that the authoring tool should provide [7], [9]. Those two dimensions enabled us to classify users into four main roles: Game Modelers, Scenarists, Trainers, and Players (Fig.1).

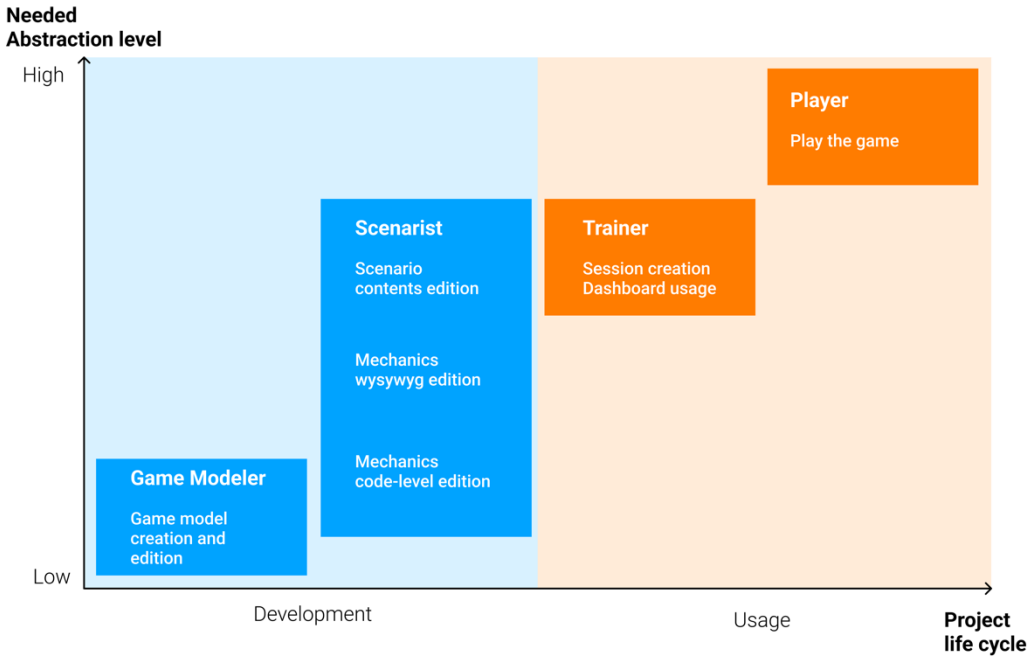

Figure 1. *The two dimensions "Project life cycle" and "Needed abstraction level" are used to classify the users of authoring tools into four main roles. Low abstraction levels correspond to interfaces near the programming code, while high abstraction levels correspond to simple user interfaces.*

4.1.1 Game modelers

The "Game Modeler" role represents users with high computer skills, responsible for game models development.

A game model corresponds to the skeleton of a game, including its basic structure, layout, and logic. Game models may be generic, such as interactive dialogues, point and click or treasure hunt. They may also be more specific, such as a combination of dialogues, questions, point and click and a simulated human body. Game model development includes tasks that require high computer skills, such as data model design or complex logic conception. Those tasks are mostly done by computer scientists and game designers who need a low level of abstraction, close to the code, allowing the development of any desired features. We observed that Game Modelers may spend days or weeks before mastering the authoring tool and are used to work both in high and low abstraction levels.

4.1.2 Scenarists

A complete and usable serious game is the result of the collaborative work of a multidisciplinary team. Part of the team is responsible for developing the game model. Others are responsible for creating the content and the narrative logic of the game. A game model filled with content is what we called a "scenario". We defined the role "Scenarist" to describe users responsible for creating the game narrative content and logic.

Once a game model has been defined, the scenarist comes into play. The scenarist's function is to flesh out the game model through the conception of the narratives and other game and subject-matter contents.

Multiple scenarios can be derived from a given game model. Each scenario will have its own content and narrative logic. For example, a "treasure hunt" game model can be used to create treasure hunt scenarios in the field of architectural education or for geological education. The Scenarist role includes contributors such as teachers, subject experts, or game designers. Different Scenarists will achieve tasks of different complexity levels. Some of them may have no computer skills and just need to adapt the narrative content of an existing scenario. Others, with basic computer competencies, may want to edit the

logical parts in a visual mode (for example conditions or networks such as finite state machines). Advanced Scenarists, with good computer competencies, may use nearly the same low-level tools as game-modelers.

In synthesis, the Scenarist role is divided into many sub-roles that are linked to users' computer competencies, and complexity of tasks to be done. The time needed to learn how to use the authoring tool increases with the targeted complexity level. It may vary from some hours to some days.

4.1.3 Trainers

The "Trainer" role includes all teachers, professors or professional instructors who use the developed serious games for educational purposes.

Trainers provide students with game access and then to monitor and manage the game session. Trainers are not expected to have any computer competencies. Based on our observations, they expect to spend no more than a few hours learning how to use the serious game trainers' interfaces.

4.1.4 Players

The "Players" role includes any final users that will experience the game, such as elementary school pupils, university students or participants to post-graduate training.

Players need to access the game and to play it on different types of devices such as laptops or mobile phones. Players are not expected to have any specific computer competencies. They should spend as little time as possible understanding how to join and begin to play a game.

4.2 *Authoring tool design principles*

In this section, we present a set of design principles (P) for serious games authoring tools. We defined these design principles based on the categorization of users, the desired characteristics of authoring tools, and previous research such as [7], [9], [13], [22], [32].

We hypothesize that altogether, these design principles should enable the development of an authoring tool that is powerful (C1), usable (C2), and with reusable components (C3).

4.2.1 P1: Ensure platform independence

Multiple devices may be used by developers, trainers, and players. Developers mainly use computers. Players, depending on the game model, may use computers, tablets, or mobile phones. Thus, the authoring tool should be usable with as many devices as possible and should allow authors to develop responsive interfaces for any kind of platform.

This principle is an extension of the proposed principle by Gicquel et al. [32] to use only web technologies. We consider web technologies as a means to achieve platform independence rather than as a principle in itself.

4.2.2 P2: Encompass the entire process

The authoring tool should support the entire process, from design and development to the distribution and usage of games (C2.2).

A same user may use the authoring tool with different roles, such as Scenarist when editing the game content, Player when testing it, and Trainer when using it in an educational context. Therefore, a single platform for development, tests, distribution, and usage brings among others the advantage of synchronicity between the different roles. A centralized rights' management system also offers the comfort of a single connection for all phases.

4.2.3 P3: Enhance collaborative works and interaction

The need for collaborative development implies that the tool enables all authors to simultaneously contribute to the development of logical, design or content elements of the game. The authoring tool should provide all contributors with an up-to-date preview of the current state of the game under development. This is confirmed by [7], [9], [22].
The tool should offer functionalities to share access with other game modelers or scenarists. Once the game is developed, it has to be shareable with trainers.
While using games in an educational context, the tool should support collaboration between trainers and players with features such as interactive trainer dashboards that let the trainer monitor students' work and interact with them [13]. The tool should also allow for collaboration or competition between students or teams of students.

4.2.4 P4: Provide basic components to create a wide range of game models

To fulfill their goal of creating any type of game model, Game modelers need low abstraction level components that should allow to create the game data structure, visual layout (data display, organize navigation & interactions), and logic (scripts for game mechanism, rules, formula, simulation model) [7], [9]. This principle is needed for enabling both the creation of a wide variety of serious games (C1) and the reusability of components (C3).

4.2.5 P5: Enable the development of composite components

Low-level components theoretically allow modelers to create almost any kind of game models. But, due to their low-abstraction level, such components are not always suitable for scenarists without programming skills. Thus, the authoring tool should enable the development of high abstraction level components, more easily usable by non computer specialists.

Composite components are a combination of basic components (data structure, display and logic). They should enable the development of components that are easier to deal with for non computer scientists. For example, a specific game may need game characters with their pictures, names, descriptions, and skills. Those game characters can be created as a composite component, made of images, text area and numerical variables. This principle is needed for enabling the creation of a wide variety of serious games (C1), developing specific users' interfaces (C2.1) and facilitating reusability of components (C3).

4.2.6 P6: Provide specific views for all roles of users

The authoring tool should enable the development of specific views that are adapted to users competencies and tasks [7, p. 16], [9], [9]. It should be possible to define as many views as needed [21]. As the authoring tool integrates both game development and game usage, it should be possible to create specific views both for Scenarists, such as game content editors, and Trainers, such as trainer dashboards. This principle is necessary for usability (C2).

4.2.7 P7: Provide authoring support

Authoring support should increase both the productivity and the quality of the developed games [7]. An interactive preview should make it possible to constantly monitor and test the game under development [7]. It should also enable to “debug” the game by offering functionalities such as monitoring the current state of the game. A search engine should help to highlight links between game elements and to find errors. This principle favors usability (C2).

4.2.8 P8: Make it possible to reuse developed components

In order to diminish serious games development costs, the concept of reusability is central [7], [9]. The authoring tool should provide the fundamental low-level components in the

most generic and flexible way, so they can be reused in various game models with ease. The authoring tool should also enable the creation of generic and parameterizable high-level components. It should finally be possible to share those components between game modelers. This principle is directly linked with the characteristic of reusability (C3).

4.2.9 P9: Infer generic components based on usage

To allow a continuous improvement of the range of components proposed by the core system, authoring tool administrators should be able to monitor the components created by game modelers, identify recurrent patterns and infer generic components that may be proposed to all users. This principle is necessary for facilitating reusability (C3).

4.2.10 P10: Be production compliant

The authoring tool should be designed, developed, and implemented with the constraint of being used in production. The infrastructure must be stable, with an adapted backup policy and a monitoring system that will alert the support team if there is an infrastructure problem (such as a server breakdown). From the software perspective, the authoring tool should include features enabling incremental game development with versioning possibilities. And when games are used in a training context, there should be a "usage isolation" that ensures that changes in game scenarios may not affect the ongoing usage of the game. This principle is confirmed by [32] and linked to C2.2.

4.3 Test implementation of design principles

In this section, we present the result of a test implementation of the design principles defined above into a single authoring tool, the Web Game Authoring System (WEGAS). WEGAS is an open-source serious games authoring tool, developed by the authors of this article.

4.3.1 P1: Ensure platform independence

WEGAS has been developed as a full web platform, thus ensuring that both game development and usage are platform independent. Moreover, web technologies do not require users to perform any software installation, and can be updated without users intervention [32]. They also facilitate game usage in distant-learning modes.

Fig. 2 presents the integration of all design principles into the general architecture of WEGAS.

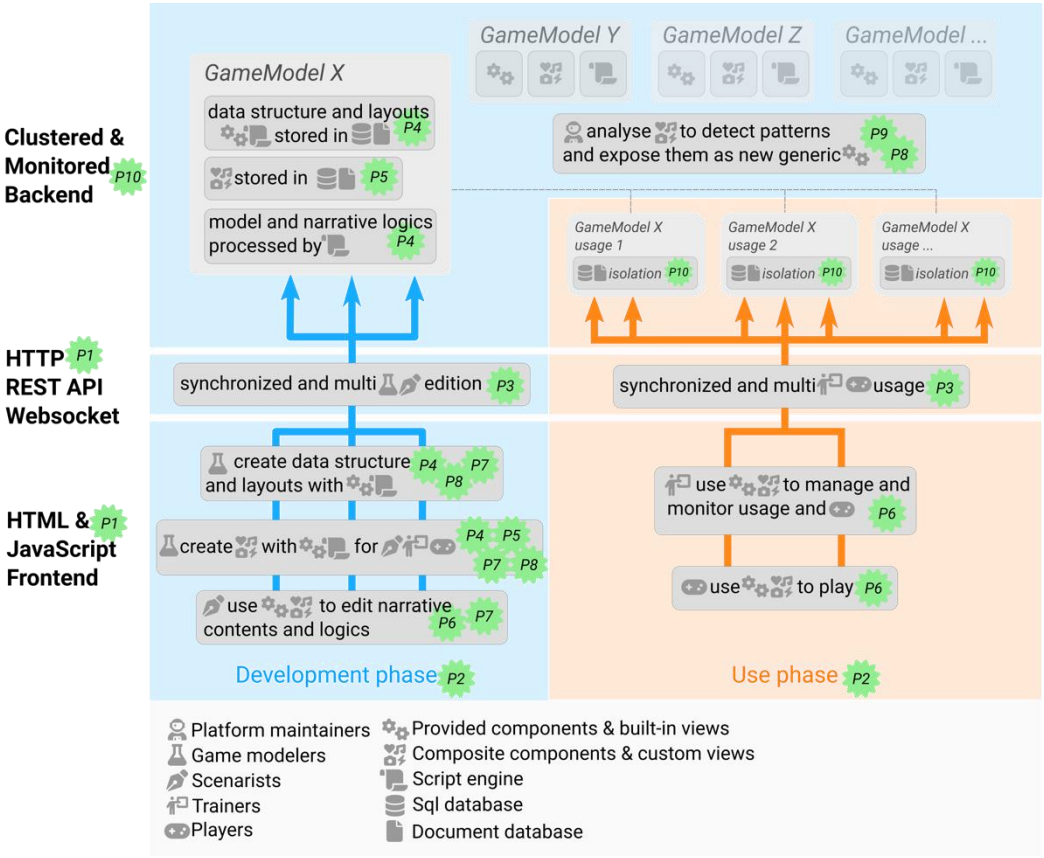

Figure 2. *WEGAS architecture shows how design principles (Pn) are supported by multiple elements of the overall technical architecture.*

4.3.2 P2: Encompass the entire process

To support the entire process, the WEGAS architecture was based on a role-based access, for the four roles Game modeler, Scenarist, Trainer, Player. An "Administrator" role was added to allow both for the system and users management.

Users can switch from one role to the other. For each role, the user accesses a lobby with the list of associated elements: game models, scenarios, training sessions or played games (Fig. 3).

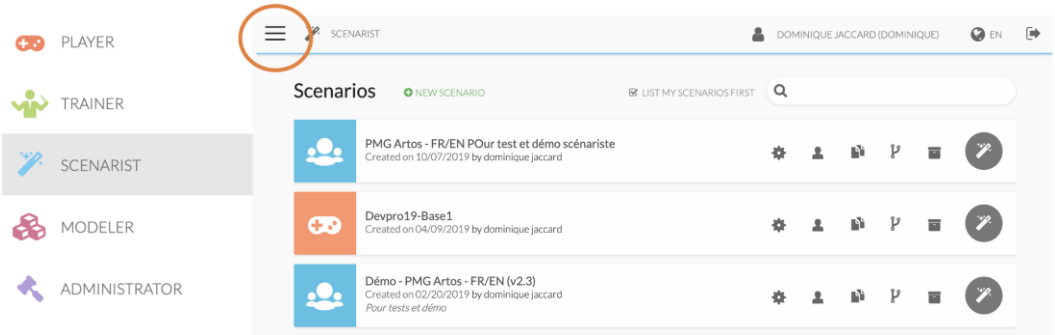

Figure 3. *Role-based access. The left side shows how users can switch between the 4 user's roles (only administrators of the system will see the administrator role). The right side shows an example of the Scenarist lobby.*

4.3.3 P3: Enhance collaborative works and interaction

The game model and scenario editors allow all team members to simultaneously develop game components, logic, design, and content. The preview system provides all authors with a constant up-to-date view of the developed game. Chat systems have been implemented to support communication between authors that are simultaneously developing the game.

Sharing features let Game modelers share their game models with other Game modelers and make them available to Scenarists. Scenarists can share their scenarios with other Scenarists and make them available to Trainers. Trainers can share their training sessions with other trainers or teaching assistants and make them available to Players.

Finally, when games are used in an educational context, interactions between teachers and students are possible through the trainer dashboard (Fig. 4).

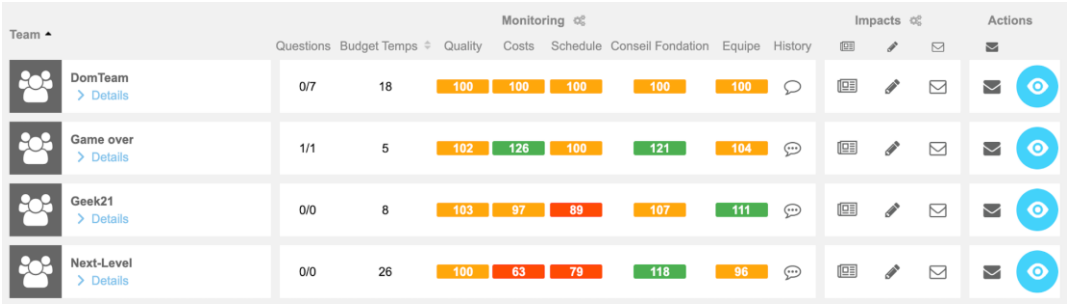

Figure 4. *The trainer dashboard enables to monitor students' work, to interact with them, and to directly interact with students' games.*

4.3.4 P4: Provide basic components to create a wide range of game models

WEGAS provides low abstraction level basic components that enable the creation of the game data structure, visual layout, and logic.

Basic components linked to the data structure are for example numerical variables, text variables, or folders. For the visual layout, basic components include the display of data (such as gauge for displaying numeric variables, text display or images display). Other layout components are linked with navigation and interactions (such as tabs or buttons). Finally, some basic components are linked with game logic. Those components enable the creation of specific scripts that may interact with the data or the layout.

For each data component, a three-level scope has been implemented: player, team, or game scope. This makes it possible to develop games that are played alone, in collaboration or in competition. This allows for example the development of a business game where each player takes a specific role in a company, each team plays a company, and all companies are in competition for the same market. In that example, a player scope component may be the player's avatar, a team scope component may be the company's reputation, and a game scope component may be the potential market to be shared between all companies.

4.3.5 P5: Enable the development of composite components

WEGAS enable the creation of composite components as an aggregation of basic components. Game modelers may define any composite component by grouping basic data, layout, and logical components. Each composite component may contain both its data structure, logic, and display interfaces.

For example, in a medical serious game, thumbnails of patients have been created as composite components (Fig. 5).

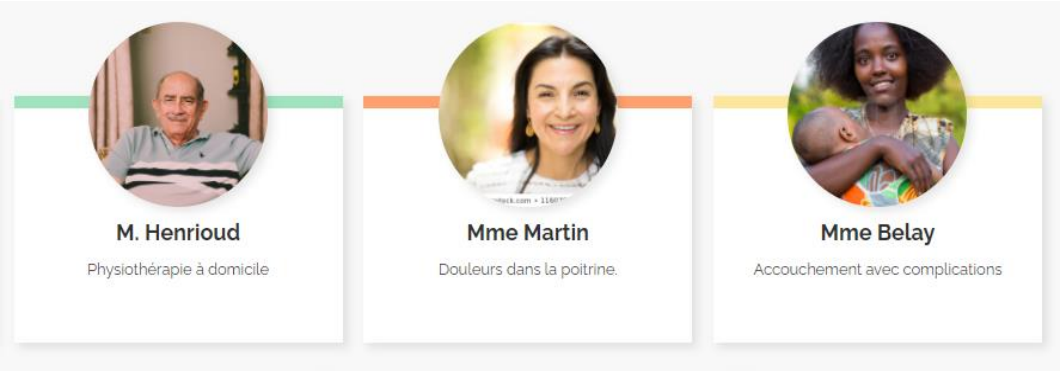

Figure 5. *The composite component "Patient thumbnail" is an aggregation of basic components such as texts and pictures.*

4.3.6 P6: Provide specific views for all roles of users

Specific views for each user role were defined and implemented.

Game Modeler views provide the complete set of features of WEGAS (Fig. 6). This includes a mix of low and high abstraction level views. Those views enable the development of game models, composite components or specific Scenarist or Trainers interfaces.

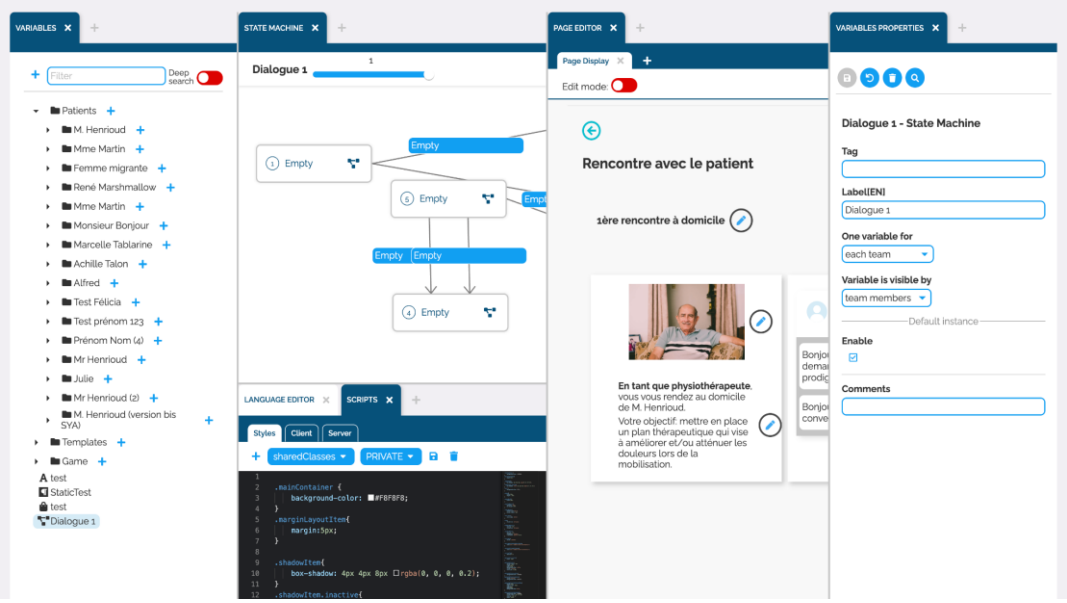

Figure 6. *Game Modeler views: the mix of low and high abstraction levels enables the edition of the game as from the code level.*

Scenarists' views should enable game content edition by non computer scientists. Those views may be a "reduction" of the game modeler view, showing less information. But specific views may also be created from scratch and dedicated to a specific task. A specific view is created with the same basic and composite components that are used for game models development. Fig. 7 illustrates an example of a high abstraction level view dedicated to content edition of patients' thumbnails.

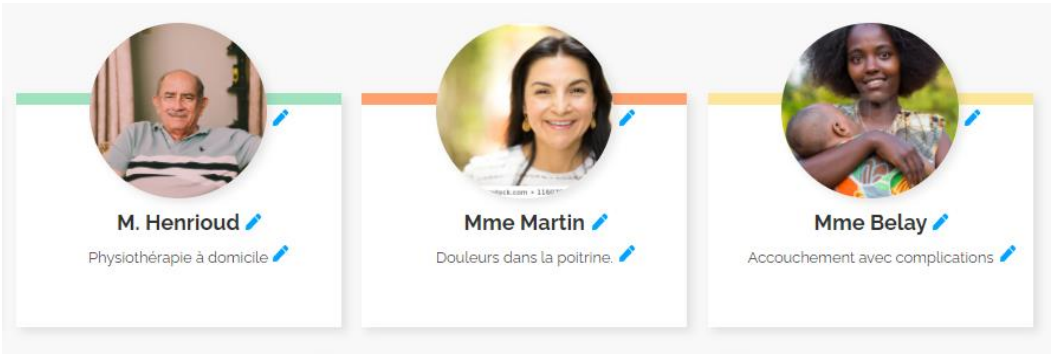

Figure 7. *Scenarist view: example of a restricted view enabling teachers with few computer competencies to only edit the content of the game.*

When creating a game model, it is also possible to create specific views for trainers, linked to game usage in training contexts. For example, it is possible to develop a specific trainer dashboard for each game model.

4.3.7 P7: Provide authoring support

WEGAS provides a preview system that allows the constant monitoring and testing of games under development. While testing, current states of all game elements are directly accessible and editable.

A "search" system allows to search for all occurrences of each variable in the entire game. For example, by searching on a specific variable, such as "life points", it is possible to highlight all game elements (dialogues, choices, clicks on a button, etc.) that impact this variable.

4.3.8 P8: Make it possible to reuse developed components

Once a basic or composite component is developed, WEGAS enables its reuse in other game models. To improve reusability, components can be developed with customizable parameters, such as display options.

4.3.9 P9: Infer generic components based on usage

WEGAS administrators have the possibility to continuously develop and share new components. New components will appear in the component library accessible by all Game modelers (Fig. 8). The component library includes basic components needed for the game layout (e.g. lists, lines or grids), inputs (e.g. numbers, text, button, selection list), or output (e.g. gauges, numbers, texts). Advanced components include composite components, such as questions lists or finite-state machines. It is also possible to define programmatic components, which consist of reusable low-level scripted sequences.

Figure 8. *Components developed for one game can be added to the component library, and thus made available for reuse in other games development.*

One limitation is that there is no automatic system to monitor the components created by the Game Modelers. Game modelers are not able to share developed components between themselves without the assistance of WEGAS administrators.

4.3.10 P10: Be production compliant

The WEGAS platform includes an automated backup system. For game developers, specific features allow the creation of intermediate versions of the developed games, as well as the possibility to export the whole game and its content in standardized file format such as Portable Document Format (PDF) or open data interchange format such as JavaScript Object Notation (JSON).

To ensure safety and stability when using games in teaching conditions, when a game session is created, a copy of the scenario is automatically created. The game session is then based on this copy and cannot be impacted by any modification that would be made in the original scenario.

4.4 Use cases

To test the characteristics of WEGAS, several use-cases have been conducted. A synthesis of use cases facts is presented in Table 1. For each use case, Table 1 presents the approximative number of users for each role (game modelers, scenarist, trainers, and players).

Table 1. *Synthesis of uses cases*

Game Models	Start Date	Game Modelers	Scenarists	Trainers	Students
Project Management [33], [34]	2012	5	10	50	15’000
Crime Scene Investigation [35]	2012	2	2	1	300
Computational thinking [36]	2012	3	3	10	1’500
Psychological tests [37]	2014	1	10	10	1’200
Clinical assessment	2015	3	2	2	1’600
Corporate law	2015	2	4	5	1’100
Oncology care	2016	2	2	8	850
Energy management [38]	2017	2	10	10	400
Mobile treasure hunt [39]	2018	2	4	2	100
Emergency triage [40], [41]	2018	3	3	1	110
Media engineering	2018	3	4	3	250
Patients' Rights [42]	2020	4	3	2	270

Table 1 shows that all use-cases integrated both the development phase and usage in educational contexts. It also shows that in each use-case, all users roles were represented. In the following sections, we present how use-cases enabled us to assess WEGAS characteristics.

4.4.1 C1. Enable the creation of wide variety of game models

The column “Game Model” illustrates that the WEGAS authoring tool enabled the development of serious games in various fields of knowledge. Fig. 9 presents some screenshots of those different game models. Those game models included genres such as simulation games, narratives and branching stories, puzzles, role plays or treasure hunts. Most of those game models included a combination of those genres. Some games were designed to be played alone, others to be played in teams. Some were designed to be played on computers (for in class activities), others for mobile phones (to be played outside of the classroom). Some of them used the responsive features of WEGAS to target both computers and phones

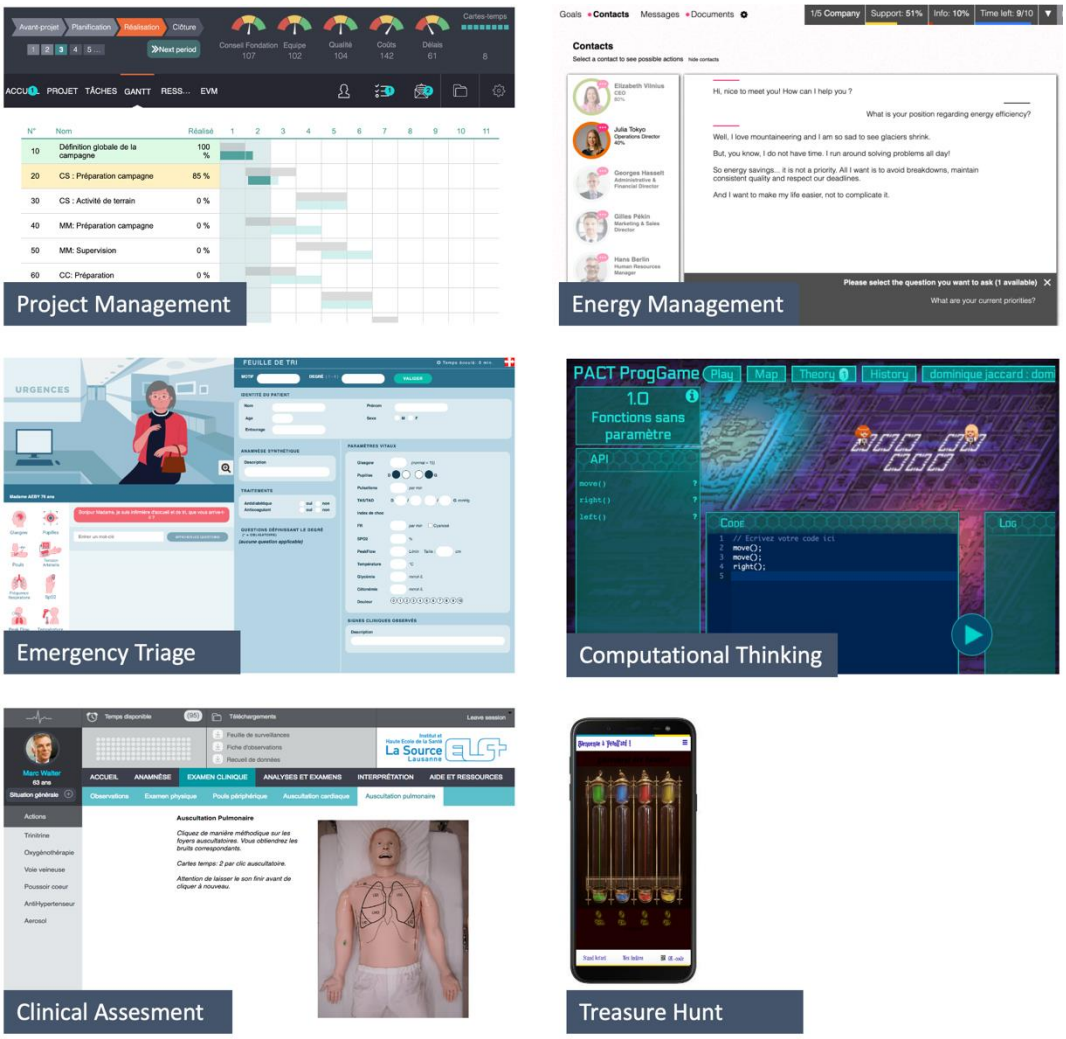

Figure 9. *Example of game models developed with WEGAS Game Model Editor.*

All use cases began with the development of a unique scenario. But some of those game models were later reused for the development of new scenarios. For example, the "Project Management" game model was used to develop scenarios such as new product development, events management or public management.

4.4.2 C2.1. Usable by users with various computer competencies and needs

Columns "Game modelers" and "Scenarists" highlight how the development of all serious games included both game modelers and scenarists. We observed that Game modelers included mostly highly computer skilled teams, such as front-end and back-end developers, or user experience experts. Depending on use cases, Scenarists ranged from lawyers, nurses, economists, criminologists, engineers, medical doctors, or educational scientists. Most of them had average computer competencies.

While Game modelers had access to the complete set of features, from low to high abstraction levels, specific views were created to consider Scenarists' competencies and tasks to be done. High abstraction views were created for non computer skilled scenarists. In all use-cases, those specific views enabled Scenarists without specific computer skills (such as lawyers, nurses, or managers) to create the narrative content of the scenarios. Fig. 10 is an example of a specific view developed for non computer skilled nurses who had to edit content for the serious game "Patient's Rights".

Figure 10. *A specific view developed for enabling non computer skilled nurses to edit the game content.*

In the use case "Energy Management", the game had to be translated into eight different languages. Translators were engaged for that specific purpose and a specific view has been developed for them (Fig. 11).

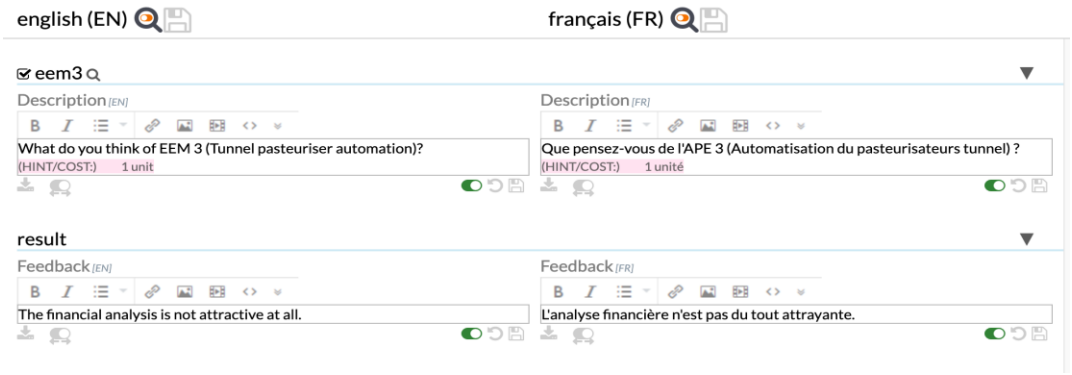

Figure 11. *A specific view allowed the translators to focus on the only task they had to do: translate.*

4.4.3 C2.2. Usable all along the project life cycle

In all use cases, the authoring tool made it possible to have all authors accessing and contributing simultaneously to game development. We observed that computer scientists developed the game model while graphic designers were implementing the front end, and scenarist (such as lawyers or medical doctors) were implementing the narrative content. Each role had its specific view for accessing the game in development. The preview of the game was simultaneously accessible by all authors, providing the development team with an up-to-date vision of the game in development.

We observed that in all use cases, all roles were used, from game modelers to players. The sharing system enabled the sharing of games between game developers and scenarists. It was then used for sharing developed games with trainers. In many cases, during the use of games in an educational context, trainers shared their training session with other trainers or teaching assistants. Players were able to play developed games both in individual and collaborative modes.

4.4.4 C3. Reusable components

The reusability of the developed components has often been observed. For example, composite components such as multiple-choice questions, mailboxes or dialogue simulators have been reused in different game models. Table 2 presents examples of components that were reused in different games. Lines of the table show how each game model was built on a combination of provided components. Columns show how each component was reused in different game models. Generic components, such as Button or Questions, were used in nearly all game models. More specific components, such as Mailbox, were used in fewer game models.

Table 2. *Example of component reuse in different Game Models*

Game Models	**Components**							
	Question	Gauge	Button	Text input	Mailbox	Point and click	Dialogue	State machine
Project Management	x	x	x	x	x			x
Crime Scene	x		x		x			x
Computational thinking			x	x	x			
Psychological tests			x			x		
Clinical assessment	x	x	x			x	x	x
Corporate law	x	x	x	x	x			x
Oncology care	x		x			x		x
Energy management	x	x	x	x	x		x	x
Mobile treasure hunt	x	x	x					
Emergency triage	x		x					
Media engineering	x	x	x	x	x			x
Patients' Rights	x		x				x	

For example, the reusability of the “dialogue display” component is illustrated in Fig. 12. The dialogue component made it possible to customize its implementation by choosing among display parameters, and by adapting its CSS style.

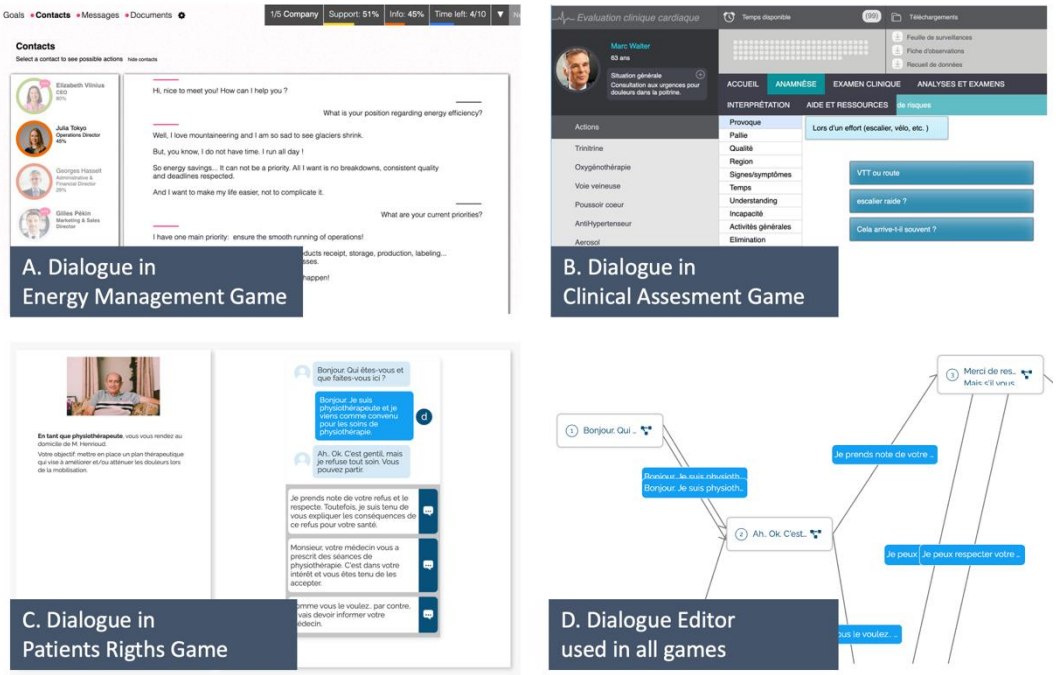

Figure 12. *The same dialogue component is implemented into different game models (A, B, C). All game models reused the same Dialogue editor (D).*

Reusability was also possible outside of the game itself. For example, all use cases reused some basic functionalities provided by the authoring system, such as trainer dashboards or players and trainers accounts management systems.

5 Discussion

In this research, we defined a set of ten design principles for serious games authoring tools development. We found that:

- It was possible to implement all those design principles into a single authoring tool.
- The authoring tool developed based on those design principles enabled the creation of a wide range of serious games, was usable by users with various computer competencies and needs, all along the project life cycle and allowed the reuse of developed components.

5.1 Principal findings

5.1.1 Authoring tool design principles

Based on desired characteristics of authoring tools, users categorization, and previous research, we defined the following set of design principles.

1. Ensure Platform Independence.
2. Encompass the Entire Process.
3. Enhance Collaborative Works and Interaction.
4. Provide Basic Components to Create a Wide Range of Game Models.
5. Enable the Development of Composite Components.
6. Provide Specific Views for All Roles of Users.
7. Provide Authoring Support.
8. Make It Possible to Reuse Developed Components.

9. Infer Generic Components Based on Usage.
10. Be Production Compliant.

5.1.2 Characteristics of the authoring tool developed based on those principles

We developed WEGAS based on those ten design principles. Use-cases showed that WEGAS seems to allow the achievement of the wanted characteristics: powerful (C1), usable (C2) and reusable components (C3).

C1: Powerful. Uses-cases illustrated that the authoring tool made it possible to develop different types of game models in different areas of knowledge. Game models included virtual patient simulations for medical education, 2D movement games, branching stories, or treasure hunt games played on mobile phones.

C2.1: Usable by a wide variety of users, with various computer competencies and needs. Use-cases showed that it was possible to provide computer scientists with a low abstraction level view and to simultaneously provide high-abstraction level views for non-computer scientists. Experimentation showed that those views enabled computer scientists to develop the game, and to non-computer scientists to create the game content. We observed that non-computer scientists have been able to develop fully new scenarios without any help from computer scientists. For example, after having been briefly instructed for the creation of a first scenario, fully new scenarios were created by nurses (in fields such as respiratory or stomach diseases), managers (in fields of project management) or lawyers (in fields of corporate law).

C2.2 Usable by users all along the project life cycle. In all use cases, WEGAS was used both for the game development (by game modelers and Scenarists) and for game usage in educational context (by Trainers and Students). In all use-cases, when offering the team the possibility to collaboratively develop the game, this possibility has been used. Instead of a sequential development with the development of the game model, followed by the graphic design and then the content editing, we observed the simultaneous development of all these aspects. While computer scientists were developing the game model, teachers were simultaneously editing the game content. This has led to what can be considered a co-creation of serious games, with the integration of teachers from the beginning of the development.

C3: Reusable components. We observed that many generic components developed for a specific game model have been reused in further game models. Reuse occurrences depend on the genericity of the component. We believe that offering a component library, with disposable components, may impact game design. This may favor creativity by proposing components that the design team would not have thought about (for example, as they were a mailbox component, some design teams decided to include a mailbox in their games). On the other hand, it may limit creativity in game design if design teams are anchored in the only proposed components.

5.2 Unexpected findings

Our targeted goal of integrating the characteristics of power, usability and reusability in a single authoring tool was initially seen as antagonistic or as involving tradeoffs. But we discovered that it wasn't always the case.

Synergies were possible. We discovered that some design principles supported more than one characteristic. For example, the development of an authoring tool providing basic components that could be combined into composite components was necessary to develop a wide range of game models. But the same components were reused to develop high abstraction level views for game edition by Scenarists. Furthermore, as "supporting the entire serious game life cycle" was another design principle, those same components were once again reused to create trainer dashboards. Thus those basic and composite components principles that where initially dedicated to power, supported simultaneously usability and reusability.

5.3 Limitations

We do not know if our set of design principles is the only one that enables the design of powerful and usable authoring tools, nor that the respect of those principles will guarantee power and usability. Rather, we believe that these 10 design principles can be used as a basic guideline for the design of serious games authoring tools.

In all use cases, at least one member of the WEGAS development team was part of game modelers teams. It has not been tested to what extent the development of game models could be realized without the help of the WEGAS team.

Although many use cases have been carried out, it was not possible to test the development of all types of game models. While WEGAS makes it possible to develop many kinds of games, some kinds of game models are not possible to develop. For example, since we have not developed interfaces for virtual reality headsets, virtual reality-based game models cannot be developed with WEGAS.

No quantitative evaluation was conducted for assessing the usability of the content editing system. This would have been linked to other research questions, and out of the scope of this research.

We supposed that when developers were able to reuse an existing component (such as a questions list, mailbox, or dialogue display) instead of developing that component, this reduced development costs. But the cost impact of this reuse has not been measured.

5.4 Future work

We believe that further research should focus on how people with different computer competencies use and perceive the different views provided by WEGAS. To quantitatively validate the usability of the different abstraction levels, it would be worthwhile to perform comparative evaluations, for example with the system usability scale.

5.5 Conclusion

We propose a set of ten design principles for serious games authoring tools. Those design principles integrate and extend previous work in the field. The development of an authoring tool based on those design principles showed that it was possible to integrate power and usability, which were previously often presented as antagonistic.

Such authoring tools seem to facilitate the collaboration within multidisciplinary teams of teachers and computer scientists. This could unlock new possibilities for collaborative approaches in serious games development.

5.6 Acknowledgments

This research was carried out in the framework of the co.LAB project, supported by the Swiss National Science Foundation (NRP 77).

5.7 Conflicts of interest

None declared

6 References

[1] C. S. Loh, Y. Sheng, and D. Ifenthaler, “Serious Games Analytics: Theoretical Framework,” in *Serious Games Analytics*, C. S. Loh, Y. Sheng, and D. Ifenthaler, Eds. Cham: Springer International Publishing, 2015, pp. 3–29. doi: 10.1007/978-3-319-05834-4_1.

[2] J. Bourgonjon, F. De Grove, C. De Smet, J. Van Looy, R. Soetaert, and M. Valcke, “Acceptance of game-based learning by secondary school teachers,” *Computers and Education*, vol. 67, pp. 21–35, 2013, doi: 10.1016/j.compedu.2013.02.010.

[3] Y. Chaudy, T. M. Connolly, and T. Hainey, “EngAGe: A Link between Educational Games Developers and Educators,” in *2014 6th International Conference on Games and Virtual Worlds for Serious Applications (VS-GAMES)*, Valletta, Malta, 2014, pp. 1–7. doi: 10.1109/VS-Games.2014.7012156.

[4] D. Djaouti, “Serious Games pour l’éducation : utiliser, créer, faire créer ?,” *Tréma*, vol. 4, pp. 51–64, 2016, doi: 10.4000/trema.3386.

[5] S. Verschueren, C. Buffel, and G. Vander Stichele, “Developing Theory-Driven, Evidence-Based Serious Games for Health: Framework Based on Research Community Insights,” *JMIR Serious Games*, vol. 7, no. 2, p. e11565, May 2019, doi: 10.2196/11565.

[6] D. Jaccard, L. Suppan, E. Sanchez, A. Huguenin, and M. Laurent, “The co.LAB Generic Framework for Collaborative Design of Serious Games: Development Study,” *JMIR Serious Games*, vol. 9, no. 3, Jul. 2021, doi: 10.2196/28674.

[7] F. Mehm, R. Dörner, and M. Masuch, “Authoring Processes and Tools,” in *Serious Games*, Springer, 2016, pp. 83–106. doi: 10.1007/978-3-319-40612-1_4.

[8] P.-Y. Gicquel, S. George, P. Laforcade, and I. Marfisi-Schottman, “Design of a component-based mobile learning game authoring tool,” in *International Conference on Games and Learning Alliance*, 2017, pp. 208–217. doi: 10.1007/978-3-319-71940-5_19.

[9] T. Murray, “Coordinating the complexity of tools, tasks, and users: On theory-based approaches to authoring tool usability,” *International Journal of Artificial Intelligence in Education*, vol. 26, no. 1, pp. 37–71, 2016, doi: 10.1007/s40593-015.

[10] “Unity Real,” 2021. https://unity.com/ (accessed Nov. 04, 2021).

[11] “Unreal Engine,” *Unreal Engine*, 2021. https://www.unrealengine.com/en-US/ (accessed Nov. 04, 2021).

[12] M. Plumettaz-Sieber, D. Jaccard, J. Hulaas, and E. Sanchez, “Évaluation de l’acceptabilité, de l’utilité et de l’utilisabilité du tableau de bord du jeu ‘Programming Game,’” in *Atelier “Apprentissage de la pensée informatique de la maternelle à l’Université : retours d’expériences et passage à l’échelle,”* Paris, 2019.

[13] K. Verbert *et al.*, “Learning dashboards: an overview and future research opportunities,” *Personal and Ubiquitous Computing*, vol. 18, no. 6, pp. 1499–1514, 2014, doi: 10.1007/s00779-013-0751-2.

[14] D. Jaccard, J. Hulaas, and A. Dumont, “Using Comparative Behavior Analysis to Improve the Impact of Serious Games on Students’ Learning Experience,” in *International Conference on Games and Learning Alliance*, Utrecht, 2016, pp. 199–210. doi: 10.1007/978-3-319-50182-6_18.

[15] K. Verbert, E. Duval, J. Klerkx, S. Govaerts, and J. L. Santos, “Learning Analytics Dashboard Applications,” *American Behavioral Scientist*, vol. 57, no. 10, pp.

1500–1509, 2013, doi: 10.1177/0002764213479363.

[16] F. Mehm, J. Konert, S. Göbel, and R. Steinmetz, "An authoring tool for adaptive digital educational games," in *European Conference on Technology Enhanced Learning*, 2012, pp. 236–249. doi: 10.1007/978-3-642-33263-0_19.

[17] A.-F. Lai and H.-D. Gu, "Developing an educational game authoring system: Edu-game maker," in *2017 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW)*, 2017, pp. 389–390. doi: 10.1109/ICCE-China.2017.7991159.

[18] Z. Khan, K. Maddeaux, and B. Kapralos, "Fydlyty: A Low-Fidelity Serious Game for Medical-Based Cultural Competence Education," in *Proceedings of the 7th International Conference on Intelligent Technologies for Interactive Entertainment*, Torino, Italy, 2015. doi: 10.4108/icst.intetain.2015.259567.

[19] A. Yessad, J.-M. Labat, and F. Kermorvant, "Segae: A serious game authoring environment," in *2010 10th IEEE International Conference on Advanced Learning Technologies*, 2010, pp. 538–540. doi: 10.1109/ICALT.2010.153.

[20] A. Karoui, I. Marfisi-Schottman, and S. George, "Mobile learning game authoring tools: assessment, synthesis and proposals," in *International Conference on Games and Learning Alliance*, 2016, pp. 281–291. doi: 10.1007/978-3-319-50182-6_25.

[21] A. Karoui, I. Marfisi-Schottman, and S. George, "JEM Inventor: a mobile learning game authoring tool based on a nested design approach," *Interactive Learning Environments*, pp. 1–28, 2020, doi: 10.1080/10494820.2020.1753214.

[22] T. Murray, "Design tradeoffs in usability and power for advanced educational software authoring tools," *Educational Technology*, vol. 44, no. 5, pp. 10–16, 2004.

[23] J. L. Plass, R. E. Mayer, and B. D. Homer, *Handbook of Game-Based Learning*. Mit Press, 2020.

[24] O. De Troyer, "Towards effective serious games," in *2017 9th International Conference on Virtual Worlds and Games for Serious Applications (VS-Games)*, 2017, pp. 284–289. doi: 10.1109/VS-GAMES.2017.8056615.

[25] "RAGE - Realising an Applied Gaming Eco-system," *H2020 European Commission*, 2015. https://cordis.europa.eu/project/id/644187 (accessed May 31, 2022).

[26] W. van der Vegt, W. Westera, E. Nyamsuren, A. Georgiev, and I. M. Ortiz, "RAGE Architecture for Reusable Serious Gaming Technology Components," *International Journal of Computer Games Technology*, vol. 2016, p. e5680526, Mar. 2016, doi: 10.1155/2016/5680526.

[27] L. Cohen, L. Manion, and K. Morrison, *Research Methods in Education*. Routledge, 2018.

[28] M. Denscombe, "Communities of practice: A research paradigm for the mixed methods approach," *Journal of mixed methods research*, vol. 2, no. 3, pp. 270–283, 2008, doi: 10.1177/1558689808316807.

[29] A. Lukenchuk, *Paradigms of research for the 21st century: Perspectives and examples from practice*. New York: Peter Lang, 2013.

[30] M. Y. Feilzer, "Doing mixed methods research pragmatically: Implications for the rediscovery of pragmatism as a research paradigm," *Journal of mixed methods research*, vol. 4, no. 1, pp. 6–16, 2010, doi: 10.1177/1558689809349691.

[31] L. E. Suter, "Multiple methods: research methods in education projects at NSF," *International Journal of Research & Method in Education*, vol. 28, no. 2, pp. 171–181, 2005, doi: 10.1080/01406720500256244.

[32] P.-Y. Gicquel, I. Marfisi-Schottman, and S. George, "Lessons Learned from the Development of a Mobile Learning Game Authoring Tool," in *International Conference on Games and Learning Alliance*, 2019, pp. 201–210. doi: 10.1007/978-3-030-34350-7_20.

[33] D. Jaccard, K. E. Bonnier, and M. Hellström, "How might serious games trigger a transformation in project management education ? Lessons learned from 10 Years of experimentations," *Project Leadership and Society*, vol. 3, p. 100047, Dec. 2022, doi: 10.1016/j.plas.2022.100047.

[34] R. Bonazzi, S. Missonier, D. Jaccard, P. Bienz, B. Fritscher, and E. Fernandes, "Analysis of Serious Games Implementation for Project Management Courses," Oct. 2011. doi: 10.1007/978-3-7908-2789-7_53.

[35] O. Delémont, R. Voisard, D. Jaccard, and M. Meyer, "From sg for crime scene coordination to a reflexion about the roles of images in police activity," presented at the GSGS'16-Gamification & Serious Game Symposium, 2016.

[36] M. Plumettaz-Sieber, J. Hulaas, E. Sanchez, and D. Jaccard, "Co-design of a serious game for computing education," presented at the Gamification & Serious Games Symposium (GSGS), Neuchâtel, Switzerland, 2019. [Online]. Available: https://ludovia.ch/2019/wp-content/uploads/2019/04/Actes-2019_V4.pdf

[37] D. Preissmann, E. Sylvestre, D. Jaccard, C. Junod, and C. El Bez, "Flexitests," in *Actes du VIIIe colloque: questions de pédagogie dans l'enseignement supérieur*, Brest, France, 2015.

[38] C. Cooremans, D. Jaccard, J. Hulaas, and C. Rohde, *Play the game: learning about energy efficiency can be fun – seriously!* France, 2019.

[39] S. Morard, E. Paukovics, E. Sanchez, J. Hulaas, and D. Jaccard, "Conception collaborative du jeu Péroll'ard : Outils, méthodes et processus," presented at the Ludovia#CH, Switzerland, 2019.

[40] J. Hulaas *et al.*, "A serious game for studying decision making by triage nurses under stress," in *International Conference on Games and Learning Alliance*, 2020, pp. 253–262. doi: doi.org/10.1007/978-3-030-63464-3_24.

[41] P. Delmas *et al.*, "Effects of environmental distractors on nurse emergency triage accuracy: a pilot study protocol," *Pilot and Feasibility Studies*, vol. 6, no. 1, p. 171, 2020,

doi: 10.1186/s40814-020-00717-8.

[42] D. C. De Oliveira, F. Bielser, D. Bonnard, Y. Songuel, and D. Jaccard, “A Serious Game for Patient’ Rights Education,” in *2021 IEEE 9th International Conference on Serious Games and Applications for Health(SeGAH)*, Aug. 2021, pp. 1–6. doi: 10.1109/SEGAH52098.2021.9551898.