Learning from the Past: a Process Recommendation System for Video Game Projects using Postmortems Experiences

Cristiano Politowski\textsuperscript{a,}\textsuperscript{*}, Lisandra M. Fontoura\textsuperscript{a}, Fabio Petrillo\textsuperscript{b}, Yann-Gaël Guéhéneuc\textsuperscript{b}

\textsuperscript{a}Departamento de Computação Aplicada (DCOM), Universidade Federal de Santa Maria, Santa Maria, RS, Brasil
E-mail: \{cpolitowski, lisandra\}@inf.ufsm.br

\textsuperscript{b}Department of Computer Science & Software Engineering, Concordia University, Montréal Quebec H3G 1M8, Canada
E-mail: fabio@petrillo.com, yann-gael.gueheneuc@concordia.ca

Abstract

Context: The video game industry is a billion dollar industry that faces problems in the way games are developed. One method to address these problems is using developer aid tools, such as Recommendation Systems. These tools assist developers by generating recommendations to help them perform their tasks.

Objective: This article describes a systematic approach to recommend development processes for video game projects, using postmortem knowledge extraction and a model of the context of the new project, in which “postmortems” are articles written by video game developers at the end of projects, summarizing the experience of their game development team. This approach aims to provide reflections about development processes used in the game industry as well as guidance to developers to choose the most adequate process according to the contexts they’re in.

Method: Our approach is divided in three separate phases: in the first phase, we manually extracted the processes from the postmortems analysis; in the second one, we created a video game context and algorithm rules for recommendation; and finally in the third phase, we evaluated the recommended processes by using quantitative and qualitative metrics, game developers feedback, and a case study by interviewing a video game development team.

Contributions: This article brings three main contributions. The first describes a database of developers’ experiences extracted from postmortems in the form of development processes. The second defines the main attributes that a video game project contain, which it uses to define the contexts of the project. The third describes and evaluates a recommendation system for video game projects, which uses the contexts of the projects to identify similar projects and suggest a set of activities in the form of a process.

Keywords: Software development process, Video game development, Recommendation system

1. Introduction

The video game industry is a multi-billionaire market, which revenues have been growing up through the years. According to the digital marketing company Newzoo, in 2016, the video game industry achieved a revenue of US$99.6 billions, 8.5\% more than in 2015, expecting US$118 billions in 2019\cite{1}.

Video game development is a highly complex activity\cite{2}. Studies about the video game industry concluded that video game projects do not suffer from technical problems but face management and processes problems\cite{3,4}.

Problems like “unrealistic scope”, “feature creep”\cite{2} and “cut of features”\cite{3,4} are recurrent in game development. In addition, because of lack of maturity, development teams often do not adopt systematic processes\cite{5} or they use traditional approaches, such as the waterfall process\cite{6}. Other problems were identified in the IGDA annual report\cite{7} in which, for example, 52\% of the interviewees answered “yes” when asked if “crunch time”\cite{2} is necessary during game development.

To help game developers minimize these problems, researchers proposed approaches to apply agile practices in game development\cite{8,9,10}. However, recent work showed that at least 35\% of the development teams continue using hybrid processes (mixing waterfall and agile) or \textit{ad-hoc} processes\cite{6}. Other researchers have tried, through a survey, understand the problems faced by game developers and the role of software engineering in the game industry\cite{11}. Their results showed that (1) the use of systematic...
processes, opposed to the development lacking any process, had a correlation to project success and (2) the most frequent problems were delays, non realistic scope, and lack of documentation. They also analyzed 20 postmortems and extracted their corresponding software processes and showed that the adoption of agile processes is increasing, corresponding to 65% of the identified processes [6].

Consequently, we claim that game development teams must improve their project management practices by adopting systematic processes. We also claim that an approach to improve processes is to learn from past project problems. Thus, we use as main source of information “post-mortems”, which are articles written by game developers at the end of their projects, summarizing the experiences of the game development team [12]. However, to be a useful source of information for developers, the non-structured data contained in the postmortems must be structured. The developers should also be warned of possible problems that may occur during development, given the lack of video game developers’ knowledge about software process [4]. Thus, we want to develop a recommendation system to support video game development teams in building their development processes, recommending a set of activities and characteristics from past projects with similar contexts.

We describe an approach to build a recommendation system to recommend software processes, using context and a similarity degree with previous game development projects. We want the recommendation system to support game developers in choosing a sequence of activities and practices that fit within their contexts. Figure 1 summarizes our approach: (1) the construction of the processes database with characteristics extracted from postmortems; (2) the definition of the video game context to characterize video game projects; and (3) the building of a recommendation system that suggests the software process using the defined context. Finally, we evaluate the recommendation system quantitatively, qualitatively and by performing a case study with a video game development team.

The remainder of this paper is structured as follows. Section 2 describes the background of our approach. Section 3 describes the method used to build and validate the recommendation system. Section 4 explains how we build and apply the recommendation system. Section 5 describes the validation of the recommended processes. Section 6 discusses our approach, the recommendation system, and its recommendations. Section 7 summarizes threats to the validity of our validation. Section 8 describes the related work. Section 9 summarizes our approach, its results, our contributions, and future work.

2. Background

This section briefly describes the concepts necessary to understand the rest of the article.

2.1. Recommendation Systems

Recommendation Systems (RS) are programs that use data to build models to help users in making decision. They recommend items based of explicit or implicit users’ preferences. They help in alleviating the problem of information overload, showing users the most interesting items according to some criteria, offering relevance and diversity [13].

For example, e-commerce Web sites use recommendation systems to suggest their customers items. They may also be used in technical domains as well to help developers in their tasks. They are then labeled Recommendations Systems for Software Engineering (RSSE). They offer informations regarding software development activities. RS are divided in four different groups [14]:

1. Collaborative filtering: A technique for generating recommendations in which the similarity of opinions of agents on a set of items is used to predict their similarity of opinions on other items.
2. Content-based: A technique for creating recommendations based on the contents of the items and the profiles of the users’ interests.
3. Knowledge-based: A technique for providing recommendations using knowledge models about the users and the items to reason about which items meet users’ requirements.
4. Hybrid: A technique to generate recommendations that combines two or more of the previous techniques to build its recommendations.

RSSE have been used to a wide range of purposes. For example, source code within a project, helping developers to navigate through project’s source code; reusable source code, providing new discoveries to programmers, like classes and functions; code examples, elucidating coders...
who does not know how implement a particular library / framework; issue reports, gathering and providing knowledge about past issues; recommending tools, commands and operations to solve certain problems in software projects with big scope; and recommending the best person to perform a task.

Our problem is similar to these previous scenarios, particularly with issue reports, recommending operations and persons, because we want to help developers by recommending processes based on previous successful and unsuccessful video game projects, described in form of post-mortems. Thus, we will use the collaborative filtering technique, however, instead of opinions of agents we will use game project details to compare the similarities between video game projects.

2.2. Software Process

Humphrey [15] defined a software process as a set of activities, methods, and practices used in the production and evolution of software systems. There are many process types available in the literature, both academic and professional. Fowler [16] divides software processes into predictive and adaptive processes, according to their characteristics. Predictive processes emphasize sequential flows of activities and requirements definitions before the beginning of software development. Activities are defined beforehand. Such processes require strong requirements definitions. Waterfall is an example of this type of processes [17]. Adaptive processes imply short development cycles performed continuously, delivering ready-to-use features at the end of each cycle [18]. They emphasize continuous improvement of the processes [16]. Examples of this type of processes are Scrum [19] and Extreme Programming [20].

Some authors claim that adaptive or agile development has been used since the beginnings of software development [21], meanwhile others see the adoption of agile methods as a “natural evolution” in the way software is developed [16]. In spite of that, the fast adoption of agile processes is notorious, principally after the Agile Manifesto in 2001³. According to the annual report report made by Version One [22], in 2016, 96% of respondents claimed making use of agile practices. A fact also evidenced in the video games area, where the agile methodologies are majority [6, 23].

2.3. Software Context

Knowing a project context—its characteristics—is crucial to the success of a game development project. However, the variety of situations in which software projects are used is expanding, making existing premises outdated [24].

However, to the best of our knowledge, there has been only few attempts to categorize projects contexts. Boehm [25] defined five characteristics that distinguish predictive processes from agile processes: size, criticality, personnel, dynamism and culture. Kruchten [26] defined an approach to contextualize software projects by defining characteristics of agile projects in a model called Octopus. This model defines eight characteristics for software projects: size, criticality, business model, architecture, team distribution, governance, rate of change and age of the system.

Both previous works tried to contextualize software projects. However, video game projects, although software as well, have greater multidisciplinarity [3, 4] and, consequently, we claim that these small sets of characteristics do not properly define video game software development projects.

2.4. Video Games Postmortems

Video games postmortems are documents that summarize the developers’ experiences of a game development project after (either successful or unsuccessful) completion. They emphasize positive and negative outcomes of the development [12]. They are usually written right after the ends of game development projects by managers or senior developers [27]. They are an important tool for knowledge management and sharing [28]. They help teams to learn from their own experiences and plan future projects and also learn from the successes and problems of other teams. They are pieces of knowledge that can be reused by any development team and they capture examples and real-life development experiences. They are so important that some authors argue that no project should finish without postmortem [28].

Usually, postmortems follow the structure proposed in the Open Letter Template [29] and divide in three sections. The first section summarizes the project and presents some important outcomes of the project. The next two sections discuss the two most interesting characteristics of any of game development:

- **What went right** discusses the best practices adopted by developers, solutions, improvements, and project management decisions that help the project. All these characteristics are critical in planning future projects.
- **What went wrong** discusses difficulties, pitfalls, and mistakes experienced by the development team in the project, both technical and managerial.

It is important to note that postmortems are not exclusively about process. As a matter of fact, this kind article is, most of the time, very informal and does not contain any useful information regarding the development process.

Given its free structure, authors used to write about a wide range of subjects including game design, game balancing, gameplay and many others details that do not add to development process details.

³http://agilemanifesto.org
2.5. Principal Component Analysis

Principal Component Analysis (PCA) reduces the dimensionality of a set of data constituted of many related variables, keeping the majority of the variance contained in the data [30].

PCA is useful when there is a large set of variables because the data can be plotted in two dimensions, allowing a straightforward visual representation, instead of comparing numbers. A plot of the first two sets of variables, that is, a biplot, provides the best possible data representation in two dimensions, useful to find patterns as well [30]. Figure 2 shows an example of biplot with four variables: V1 to V4. It is important to note that the more close a sample is from another, the more correlated they are, that is, more similar.

Figure 2: PCA biplot example.

3. Research Method

Our approach, illustrated by Figure 1, is inspired by the “design concerns” related to build recommendation systems described by Robillard et al. [14]. It consists in four challenges:

1. **Data preprocessing.** Transform raw data in a sufficiently interpreted format. For instance, source code has to be parsed or commits have to be aggregated. See Section 3.1.

2. **Capturing context.** Different from traditional RS which are dependents of user profiles, RSSE focus on tasks. In this regard, a task context is all information which the recommender has access to produce recommendations. This information is generally incomplete and/or noisy. It is described in Section 3.2.

3. **Producing recommendations.** It is the algorithms’ work on the data. Different strategies can be used, however, the most suitable approach for RSSE is **Collaborative Filtering**. See Section 3.3 for details.

4. **Presenting the recommendations.** Presents the items of interest, that is, activities in our case. Described in Section 3.4.

3.1. Data and Preprocessing

Because of constraints, in particular trade secret, to access real documentation and artifacts from video game development projects, we gathered data from public-domain source: postmortems from game development Web sites, like **Gamasutra**

4https://www.gamasutra.com/ and Wikipedia entries of game projects. We observed that only Gamasutra publishes postmortems about video games. Its dataset is large enough to fulfill our needs in this article. Moreover, all postmortems offered by Gamasutra follow the same format, thus a single source of data helps mitigating the lack of formal structure of postmortems.

We analyzed the data gathered from Gamasutra by reading every postmortem, collecting information about the development processes, activities, and team characteristics. We ignored details about game design or other information that does not pertain to the development processes. We organized data—process’ elements—in a database following the model described in Figure 3.

We defined a development process as a set of elements separated in different phases and sub-phases. The **Phase** categorizes the elements of the process, which can be:

- **Activities**, which represent tasks or steps describing work units that result in artifacts. For example, the activity of prototyping.
- **Team**, which pertain to the characteristics of teams. For example, the team’s characteristic of outsourcing.
- **Characteristics**, which describe meta-data about the development process. For example, the characteristic of project duration.

Based on previous work [31, 32], we define three main process sub-phases in which **Elements** in Activities can be contained:

- **Preproduction**, containing activities like brainstorming, prototyping, validation and tasks to find the “fun factor”;
- **Production**, where technical tasks occur;
- **Postproduction**, containing activities like testing and deployment.
Post-production, containing any activities executed after the game has been launched.

Each Element can have one of two possible status:
Normal, when the postmortem author does not report any problem regarding the activity execution, and Problematic, when developers encountered some problems related to the activity.

Each element has an index name, which we call Key, as well as a description, called Value. We put the post-mortem extracted data in a map data-structure in which the key is an element name and the value is a quotation gathered from postmortems. Listing 1 shows an example of data extracted from postmortems and stored in the map data-structure.

For ease of conveying information from the processes to developers, we defined a process structure template file with the elements of process described earlier to generate a graphical representation of the processes. Figure 4 shows the elements.

Finally, to improve the reliability of the extracted processes, we showed these processes to the postmortem authors and game developers, asking for feedback about these processes. We showed the extracted process and ask the following questions:

1. “Is this process similar of what you used when developing your game?”
2. “What are the elements that does not make sense?”

With their answers, we modified the extracted process according to their feedbacks. This step helps us to ensure that the extracted processes are as close as possible to the development processes that they used during the video game development.

3.2. Capturing Context

We encoded the data gathered from postmortems in JSON format and put it in a database to perform queries. The JSON structure consists of five key:value for each process element. A process is thus a collection of N elements. Listing 2 shows an example of the JSON format.

Although the data is structured, it is not normalized. Some elements have the same meaning and, therefore, must be merged to a common description. We create an abstraction dictionary that contains all the elements and their respective indexes. For example, the elements “local play testing” and “beta testing”, refer to the activity “testing”.

Lastly, we must define the variables that will be used by the algorithm as project context. A project context defines its characteristics, which are the variables used by the algorithm. There exist some attempts to contextualize non-video game software projects [26, 25]. However, the sets of variables are larger in video game projects than in other projects because of their multidisciplinarity.
We divide the video game context into six categories, totaling 61 variables: activities, team, management, technical, platform, and design. Table 1 shows all the variables of the video game context.

3.3. Producing Recommendations

The project contexts obtained from different postmortems are input for Principal Component Analysis (PCA), discussed in Section 2.5, to identify and quantify the relations in large sets of variables. Principal Component Analysis (PCA) was used in this research because it was well suited to the problem and its simplicity, enabling a quick prototyping. Others techniques were used, like Decision Trees and K-Means algorithms. However, given the labels restrictions, we changed the approach.

The result of applying PCA on the project context is an array of similar projects. With this list, we query the database for all elements related to the projects and perform a merge, respecting the elements’ phase. Because we stored the process’ elements without its relationship, that is, flow between the activities, after put all similar elements together, we had to allocate the elements manually.

3.4. Presenting the Recommendations

The last step is to present the recommended activities: the development process. We choose to display recommended processes graphically, using the same graphical representation used for the extracted processes. Consequently, we use the same visualization template (see Figure 4).

4. Recommendation System Development

We now present the recommendation system and how we built it. We followed the same four steps as in Section 3.

4.1. Data Preprocessing

We used a Web scrapping tool to gather postmortems from Gamasutra automatically. Gamasutra was our source of information of choice because of the size of its postmortems database and its reliability. We check and filtered data, extracting the details of the video game projects, like game title, author(s), release date, url, as well as data from Wikipedia, like game mode, genre, developers, engine, etc. The total amount of game projects was 234 from 1997 (which is the oldest postmortem we found) to 2016.

Starting from the most recent postmortem, we analyzed and extracted the data to populate our JSON database. To analyze the postmortems we made use of Mendeley and its feature to create notes. Every information regarding the development process was highlighted and tagged with an index. For example, Figure 5 shows a piece of a postmortem in which we highlighted some excerpts and associated an index with each one, like “initial prototyping”.

![Figure 5: Example of the postmortem analysis made to extract and index the relevant data about the video games.](image-url)

Some postmortems do not have useful data and, thus, among the 100 postmortems that we analyzed we kept...
Table 1: Video Game context variables.

| #  | Group       | Description                  |
|----|-------------|-----------------------------|
| v01| Activities  | Agile                       |
| v02| Prototyping |                             |
| v03| Performance Optimization |                 |
| v04| Tools Development |                  |
| v05| Outsourcing: Assets |                |
| v06| Outsourcing: Work |                 |
| v07| Pre-Production: Short |               |
| v08| Pre-Production: Long |               |
| v09| Post-Production: Normal |             |
| v10| Post-Production: Heavy |              |
| v11| Reuse: Code |                             |
| v12| Reuse: Assets |                             |
| v13| Testing: In-ouse Qa Team |             |
| v14| Testing: Closed Beta |                  |
| v15| Testing: Open Beta |                    |
| v16| Testing: Early Access |               |
| v17| Marketing/Pr: Self |                        |
| v18| Marketing/Pr: Outsourced |            |
| v19| Team Size   | <=5                        |
| v20| Team Size   | 5-25                       |
| v21| Team Size   | >25                        |
| v22| Type        | Single                     |
| v23| Type        | Collaborative              |
| v24| Distributed |                             |
| v25| Management  | First-Party                |
| v26| Development | Second-Party               |
| v27| Development | Third-Party                |
| v28| Indie       |                             |
| v29| Funding     | External                   |
| v30| Funding     | Self                       |
| v31| Funding     | Crowdfunding               |
| v32| Publisher   | External                   |
| v33| Publisher   | Self (same Developer)      |
| v34| Technical   | Intelectual Property: Port |
| v35| Technical   | Intelectual Property: Remaster / Reboot |
| v36| Technical   | Intelectual Property: Franchise Sequence |
| v37| Technical   | Intelectual Property: Expansion |
| v38| Technical   | Intelectual Property: Mod |
| v39| Technical   | Intelectual Property: NewIp |
| v40| Engine      | In-House (new)             |
| v41| Engine      | In-House Ready             |
| v42| Engine      | Off-The-Self               |
| v43| Platform    | Console: Microsoft         |
| v44| Console     | Song                       |
| v45| Console     | Nintendo                   |
| v46| Pc          | Windows                    |
| v47| Pc          | Mac                        |
| v48| Pc          | Linux                      |
| v49| Mobile      | Android                    |
| v50| Mobile      | Ios                        |
| v51| Design      | Genre: Action              |
| v52| Design      | Genre: Action-Adventure    |
| v53| Design      | Genre: Adventure           |
| v54| Design      | Genre: Role-Playing        |
| v55| Design      | Genre: Simulation          |
| v56| Design      | Genre: Strategy            |
| v57| Genre       | Puzzle                     |
| v58| Genre       | Sports                     |
| v59| Mode        | Single-Player              |
| v60| Mode        | Multi-Player (offline)     |
| v61| Mode        | Multi-Player Online        |

only 55 that had sufficient information to create a development process. The other postmortems mainly focused on game designs or gave too little details about team and the project itself.

We use GraphViz to display processes extracted from the notes graphically. GraphViz uses a special text format to generate graphical files, which can be saved as PDF and PNG files. We created a script that query the database, which contains the process elements, and generate a new file, using the process template (see Figure 4), in DOT syntax, which can be converted into an image.

We requested feedback from the developers, authors of the postmortems, about the extracted processes. We sent a on-line form asking about the completeness and viability of the processes. A total of 20 forms were sent with 7 answers (33% answer rate). We refactored few wrong or misplaced elements in some of the processes. All of the answers agreed with the extracted processes, for example in the citation of the postmortem author, developer of the game “Prune”:

“Yeah, that’s about right. I would maybe call the first 6 months of work on the game “pre-production”, which included things like prototyping, test prioritization, and play testing.”

Figure 6 shows an example of an extracted process, validated by the authors of the origin postmortem, of the game project Slow down, Bull. This process shows two phases: preproduction and production, without post-production. Each phase has a start and an end date, demarcated by the dark discs. In both phases, the elements in green are activities that occurred naturally during the development while the red ones encountered problems.

Considering that we extracted process elements and, normally, they do not had chronological information about its execution, we cannot, precisely describe the process flow. In these cases, we marked such elements with two question marks (“?”). In production, the element described as “gold” indicates the final version of the game, the complete game. Each activity, in the gray boxes, has a direct related to the postmortem. There are two frames containing the characteristics and general practices together with the ones related to the development team.

4.2. Capturing Context

We normalized all the elements keys using the abstraction dictionary and inserted the elements in the database. We chose MongoDB as storage engine because it was easy to import JSON files. We stored a total of 913 different elements in the database from 55 projects.

With the data ready, we now needed a project context (see Table 1). We analyzed the project data and, according to their characteristics, manually build contexts. The context values (see Table 2) are composed of 61 variables ([v01..vN]) from the 55 extracted process (game projects, [g1..gN]). Every time a new project is added to the database, it helps in future recommendations.

---

5To save space, we manually redesigned the process, respecting
Figure 6: Extracted development process from the bull game.

1. **Pre-production**
   - Concept development
   - Storyboarding
   - Scriptwriting

2. **Production**
   - Animation
   - Illustration
   - Sound design

3. **Post-production**
   - Editing
   - Sound mixing
   - Color grading

4. **Quality assurance**
   - Quality control
   - Testing

5. **Launch**
   - Promotion
   - Distribution

6. **Maintenance**
   - Bug fixing
   - User feedback

7. **Feedback loop**
   - Continuous improvement
   - Iteration
We converted the new recommended process in a textual DOT file, which can be compiled into an image file.

Figure 8 shows an example of a recommended process and its preproduction phase while Figure 9 shows the production phase using the game context of the game *Slow down, Bull* shown in Table 3.

Table 3: Project context of the game *Slow down, Bull*. The false values were omitted.

| #  | Context                        | g10 values |
|----|--------------------------------|------------|
| p01| Agile                          | 1          |
| p02| Prototyping                    | 1          |
| p06| Outsourcing: Work              | 1          |
| p08| Pre-Production: Long           | 1          |
| p09| Post-Production: Normal        | 1          |
| p13| Testing: In-Use Qa Team        | 1          |
| p19| Size: <=5                      | 1          |
| p22| Type: Single                   | 1          |
| p24| Distributed                    | 1          |
| p27| Developer Types: Third-Party   | 1          |
| p28| Indie                          | 1          |
| p29| Funding: External              | 1          |
| p33| Publisher: Self (same Developer)| 1          |
| p39| Intellectual Property: New Ip  | 1          |
| p42| Engine: Off-The-Shelf          | 1          |
| p46| Pc: Windows                    | 1          |
| p57| Genre: Puzzle                  | 1          |
| p59| Mode: Single-Player            | 1          |

Figure 8 shows the preproduction and has the activities: requirements and constraints, exploration phase, planning documentation, and milestones planning linked with quotations from the game *Vanishing Point*: prototyping is linked to the games *Catlateral Damage* and *Jetpack High*.

Figure 9 shows the production phase. The process flow starts with an activity to choose a feature or a bug, depending on the iteration type (component delivery or fixing a problem). As input, this activity receives a list of game properties from the preproduction phase. This list is similar to sprint backlog used in agile methods like Scrum [19].

The process flow continues with the development iterations loop. This element development iterations loop marks the beginning of the productions activities. Two quotes, gathered from the postmortems of games *Vanishing Point* and *Catlateral Damage* are highlighted. The premise is similar to sprints in agile methods in which the process occurs in short cycles with continuous deliveries of complete features. At the end of each iteration, a new, complete component must be aggregated to the existent game. The artifact feature delivery represents this component.

The technical activities happen inside each iteration. There are four described activities, all of them connected with at least one citation from a game: polish and refinement, meetings, refactoring the development, and design tasks.

According to the development flow of the generated process, after the delivery loop of components, the testing
activity will start, which is crucial to the success or failure of a game. This activity may happen in many ways, for example, the team can have a play testing laboratory with users, a specialized testing team, automated tests, users testing, beta testing, among others. Only one project, Vanishing Point mentions explicitly testing.

After the end of the iterations, with the game tested, the flow arrives on a condition to test whether all the iterations are completed. If positive, the next activity is quality assurance that suggests to have the game reviewed by an external entity, either a publisher or a platform (Sony, Steam, Microsoft) to identify possible nonconformities with standards required by such publisher or platform. Then, the final build is realized, fulfilling the last requirements, to obtain and publish a gold master.

5. Evaluation

To assess our recommendation system, we performed three types of evaluations:

A. Quantitative, by measuring Correctness and Coverage of the recommended elements in Sections 5.1;  
B. Qualitative, by asking feedback from the game developers on the recommended process in Sections 5.2;  
C. Case Study, by applying a recommended process in a real world project in Section 5.3.

5.1. Quantitative Evaluation

To quantitatively assess the recommendation system, we chose four projects which we extracted from postmortems and validated it with their respective developers. We extracted their contexts and generated processes using our recommendation system. We compared if each recommended element is present in the extracted (validated) process. We used four measures to assess our recommendation system quantitatively [33], shown in Table 4.

| Dimensions | Description |
|------------|-------------|
| Correctness | How close are the recommendations to a set of recommendations that are assumed to be correct? |
| Coverage | To what extent does the recommendation system cover a set of elements or user space? |
| Diversity | How diverse (dissimilar) are the recommended items in a list? |
| Confidence | How confident is the recommendation system in its recommendations? |

Recommendation-centric dimensions primarily assess the recommendations generated by the recommendation system itself: their coverage, correctness, diversity, and level of confidence [33]. We chose two metrics to evaluate our recommendation system: Correctness and Coverage. Diversity does not apply to our system because our dataset is composed by unique elements; each element stored in the database is unique. Also, confidence requires observable variables with users and we received feedback from developers, as in Section 5.2.

5.1.1. Correctness Metric

The objective of the correctness metric is to assess how useful are the results without being overwhelming with unwanted items [33]. With this metric, we measure how close the recommended process is compared to the expected, ideal process, and already validated process.

There are different means of evaluating the recommended processes, like Predicting User Ratings and Ranking Items. However, we chose Recommending Interesting Items,
Figure 9: Recommended process production phase for the game Slow down, Bull.
which tests if the recommendation system provide process elements that users may like to use.

We fed the recommender system with data from a project context which project we already validated, a process extracted from a postmortem verified by its authors and—or developers. The confusion matrix shown in Table 5 compares each process element of the recommended process test set with the elements from the extracted process (training set).

Table 5: Confusion matrix used to calculate correctness [33].

| Recommended | Not Recommended |
|-------------|-----------------|
| Used        | False Positives (FP) | True Negatives (TN) |
| Not Used    | True Positives (TP)  | False Negatives (FN) |

For example, consider a recommended process with an element team and an extracted process that contains this element, we have a true positive (TP) element. Otherwise, if the recommended element is not contained in the extracted process, then we have a false positive (FN) element. Then, we benchmarked our recommendation system by calculating some metrics\(^7\):

- **Precision**: the percentage of the elements predicted to be relevant that are actually relevant.
  \[
  \text{precision} = \frac{TP}{TP + FP}
  \]

- **Recall**: the percentage of the elements that are actually relevant that are predicted to be relevant.
  \[
  \text{recall} = \frac{TP}{TP + FN}
  \]

- **Accuracy**: the percentage of the elements available that are either correctly recommended or correctly not recommended.
  \[
  \text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN}
  \]

- **False positive rate**: the percentage of the irrelevant elements that are predicted to be relevant.
  \[
  \text{falsePositiveRate} = \frac{FP}{FP + TN}
  \]

- **False negative rate**: the percentage of the elements predicted to be irrelevant that are actually relevant.
  \[
  \text{falseNegativeRate} = \frac{FN}{FN + TN}
  \]

- **Specificity**: a heuristic measure [34] of the likelihood of the relevance of an element given its relation to other elements, some of which are known to be relevant.
  \[
  \text{specificity} = \frac{TN}{FP + TN}
  \]

- **F-measure**: a measure combining precision and recall.
  \[
  F - \text{measure} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}
  \]

5.1.2. Coverage Metric

Coverage refers to the proportion of available information for which recommendations can be made. It can be calculated as catalog coverage (about elements) or prediction coverage (about developers) [33]. We use the former because we want to assess the recommended processes. This metric can be calculated as the proportion of elements (process elements) that we can be recommended:

\[
\text{catalogCoverage} = \frac{|S_r|}{|S_a|}
\]

where:
- **Sr**: is set of process elements recommended;
- **Sa**: is the set of all available elements;
- **Ss**: is the set of elements useful to developers (validated elements).

We computed correctness and coverage of the recommendation system by comparing the similarity between the extracted processes, validated by their authors, and the recommended processes, generated by the recommendation system. We used four different project contexts, resulting in an array of similar projects. Figure 10 shows the resulting byplots.

1. The game project **Prune** is similar to **Red Skies, Out There, INK, Cave Dash, Caseys Contraptions**;

2. The game project **Offworld Trading Company** is similar to **Mini Metro, Sunless Sea, Race the Sun, I Cant Escape: Darkness, 80 Days, Natural Selection 2, Aaaaa! – A Reckless Disregard for Gravity, Zack Zero**;

---

\(^7\)Except Specificity, which is a heuristic measure, all other values are in [0..1].
3. The game project NFL Rush Heroes & rivals is similar to Ori and the Blind Forest, Rollers of the Realm;

4. The game project Slow Down, Bull is similar to Catlateral Damage, Jetpack High, Vanishing Point.

We created a script to compare each element from each recommended process, simplified in the pseudo-code listed in Algorithm 1. We compared if the recommended elements were in the extracted processes, which we considered as correct, because they were reviewed by the post-mortem authors. If the elements recommended by the recommendation system were in the extracted processes, then they received a True Positive (TP). Otherwise, if the recommended elements were not part of the extracted processes, then they received a False Positive (FP)\(^8\). If process elements were in the extracted processes but not recommended, they received a False Negative (FN). Finally, if process elements were not part of the recommended process nor in the extracted process, then they received True Negative (TN).

Table 6 shows the confusion matrix with aggregated data and Table 7 shows the metrics results. The high number of false negatives and low number of true positives can be improved by increasing the range of similarity: by adding more similar projects. Adding more projects would also increase recall, given that recall was low. However, allowing for a longer list of recommendations improves recall but is likely to reduce precision and improving precision may decrease recall [37].

Although accuracy and specificity rates were good, all others values were low. The recommendation system reached a precision above 40% in two projects. We used PCA to find similarity and will consider other algorithms in future work. We can also improve our recommendation system by improving the context variables and samples number. Table 7 shows the coverage metric results with the low values in Catalog and Weighted Catalog, which could be improved if more elements where recommended. Increasing coverage may increase correctness as well.

5.2. Qualitative Evaluation

We performed the qualitative validation using a set of questions in a on-line form that we sent to project authors and/or developers. For each one of the chosen project, that is, project #1 to #4 already used in the quantitative validation, we asked recipients to analyze both the extracted and the recommended processes. We also asked them to highlight the elements that were wrong or misplaced. Then, we asked them about the viability of the application.

![Figure 10: Biplots PCAs from analyzed projects showing their similarity. From left to right: #1, #2, #3, #4.](image_url)

**Algorithm 1** Pseudo-code to compare the process elements.

```python
for gameTested ∈ gamesTested do
    similarGames[] ← getSimilarGamesPCA(gameTested)
    elemGameTested[i] ← getAllElements(gameTested)
    elemNotRecommended[i] ← getAllElementsExcept(elemGameTested + elemRecommended)
    if elemRecommended[i] ∈ elemGameTested then
        truePositive[i] ← elemRecommended[i]
        end if
    else
        falsePositive[i] ← elemRecommended[i]
        end if
    if elemNotRecommended[i] ∈ elemGameTested then
        falseNegative[i] ← elemNotRecommended[i]
        end else
        trueNegative[i] ← elemNotRecommended[i]
        end if
end for
```

|          | T. Positive | F. Positive | F. Negative | T. Negative |
|----------|-------------|-------------|-------------|-------------|
| #1       | 32          | 44          | 285         | 540         |
| #2       | 23          | 105         | 153         | 620         |
| #3       | 10          | 32          | 171         | 692         |
| #4       | 14          | 19          | 358         | 505         |

\(^8\)Even if an element is not part of the extracted process, it still can make sense and be useful for the developers [36].
Table 7: Correctness and coverage results.

| Correctness | Coverage |
|-------------|----------|
|             |          |

were similar to the processes that they used during their
game development. One developer stated that the activities
flow is similar to what was actually done:

“Yes, this workflow looks similar to what we
did with [game title], in that we did a lot of
iteration with our feature development.”

In another example, the postmortem author and game
developer answered about the generated process viability,
stating that the most useful contribution may be during
the project planning, but not for creation of any produc-
tion pipeline.

“(…) this might be a useful thing to look at in
the beginning, but I would not use it to create
a production pipeline because the circumstances
have almost assuredly changed since.”

Still, the postmortem’s author pointed out that the rec-
commended process is interesting for analysis, but not some-
thing to be used practically to build a set of production
activities. The developers claimed that each project de-
mands a different process based on the game design choice.

“I think the tool that you describe has some re-
really interesting implications from an analytical
standpoint, but it is not something I would use
in a practical sense, like I wouldn’t use it to
develop processes or pipelines from. In prac-
tice, every game requires different processes to
make, and in fact the process used to make a
game is as much a part of its design as the ac-
tual game. (…) The reasons for variation in
process are vast and constantly changing. Even
in big studios you tend to end up with processes
that are perfect for making the game you were
working on the year before (…) However, I
still think it is valid work from an analytical
standpoint, like as something to consider when
having the conversation about dev process for
the next game.”

5.3. Case Study

We performed a case study using qualitative metrics
from user-centric dimensions to assess if the recommendation
system fulfills the needs of developers [33]. We use
the metrics in Table 8.

Table 8: Dimensions: User-centric.

| Dimensions     | Description                                      |
|----------------|-------------------------------------------------|
| Trustworthiness| How trustworthy are the recommendations?        |
| Novelty        | How successful is the recommendation system in
               | recommending items that are new or unknown to|
               | users?                                          |
| Serendipity    | To what extent has the system succeeded in       |
               | providing surprising yet beneficial recommenda-
               | tions?                                         |
| Utility        | What is the value gained from this recommenda-
               | tion for users?                                |
| Risk           | How much user risk is associated in accepting   |
               | each recommendation?                           |

We interviewed with an team that is currently develop-
ing a game. We asked questions to measure these user-
centric dimensions. Firstly, we gathered the project con-
text and generated a new process using the recommenda-
tions system. Then, we presented the recommended pro-
cesses to the team members and asked them to answer the
following statements using a five-point Likert scale:

1. “(Trustworthiness) The recommendation is similar
   compared to my project.”
2. “(Novelty) The recommendation is new to me (re-
gardless its usefulness).”
3. “(Serendipity) The recommendation is surprisingly
   good for my project.”
4. “(Utility) The recommendation is useful to my project.”
5. “(Risk) It would be risky to use the recommendation
   in my project (considering other practices already
   settled).”

We chose a simulation-like video game project that uses
Unity engine and has been in development for more than
three years. We asked the lead developers to fill the con-
text form according to the project context, shown in Ta-
ble 9.

We appended the context values in the context table,
executed the recommendation system, and obtained a list
of similar projects: Ashes of the Singularity, Baldurs Gate
Enhanced Edition, Natural Selection 2, Anomaly Warzone
Earth and Aaaaa! – A Reckless Disregard for Gravity.

A total of 31 elements were recommended and so,
for each one of them, we asked the development team to
Table 9: Context values from the project used in case study. The values were omitted.

| #  | Context                     | Values |
|----|-----------------------------|--------|
| v01| Agile                       | 1      |
| v02| Prototyping                 | 1      |
| v03| Performance Optimization    | 1      |
| v04| Tools Development           | 1      |
| v05| Outsourcing Assets          | 1      |
| v07| Pre-Production Short        | 1      |
| v11| Reuse Code                  | 1      |
| v12| Reuse Assets                | 1      |
| v13| Testing In-Ouse Qa Team     | 1      |
| v20| Size 5 - 25                 | 1      |
| v23| Type Collaborative          | 1      |
| v27| Developer Type Third-Party  | 1      |
| v30| Funding Self               | 1      |
| v53| Publisher Self (same Developer) | 1  |
| v39| Intelectual Property New Ip| 1      |
| v42| Engine Off-The-Shelf        | 1      |
| v46| Pc Windows                  | 1      |
| v55| Genre Simulation            | 1      |
| v56| Genre Strategy              | 1      |
| v59| Mode Single-Player          | 1      |

fill the statements, comparing the recommended elements with their actual activities. Table 10 shows the recommended elements and Figure 11 shows the results.

The Trustworthiness metric had 50% of elements considered similar to what was used or applied in the development process. 28.13% of the elements were not considered similar while 21.88% were neutral.

test team “We have a third party company who perform the tests before it reach the final client (...) the testers have access to the bug database.”

refactoring the development “It happens often as the development adopt an evolutionary approach (...)”

retrospective meeting “(...) retrospective meetings occur regularly with development team and the final client.”

general team details “These recommendations do not applied in our context: The team had not the proper knowledge in the beginning (...) nor had been worked together”

The Novelty metric had 40.63% of elements considered new by the development team. 34.38% were considered already known and 25.00% were neutral.

in-house tools development “Although these practices were not used (by us) today (...) we found it a good surprise”

The Serendipity metric had 37.50% of elements considered surprisingly good for the project. 25% were considered not good and 37.50% were neutral.

engine and tools “We do not have a build tool (...) most of the techniques applied are learned from scratch.”

The Utility metric had 59.38% of elements considered useful to be used in the actual process. 12.50% were not considered good and 28.13% were neutral.

The Risk metric had 28.13% considered risky if added to the project. 46.88% were considered not risky and 25% were neutral. However, among these elements, none received a “5” as a answer.

The team cited a interesting point, as the project had already more than three years of development, many of the activities used were the results of “trials and errors”. Using the recommendation system, some of these errors could have been avoided:

“This practices could be used since the beginning, for example, prototyping, test team, bug control, small team, building tools and do less but better.”

Generally speaking, the team emphasized the the following subset of activities does not applied to their project: user community, gathering resources, marketing, fun factor related practices and any activity multi-player. From the recommendation system, there were two details to consider. Firstly, some elements had details from more than one project, which were merged together. Therefore, a recommended element may contradict itself, showing a activity that makes another impossible. Secondly, the context has some particular elements not contained in the analyzed postmortems. For this reason, a few elements were not considered similar nor useful.
6. Discussion

The use of postmortems to analyze video game projects is not new but used with another goal in this article: to extract the processes used by developers. Even though the intent of postmortems is not to document processes, it is still possible to extract data relevant to the processes and, with many postmortems, assemble activities to create a flow of activities realized by the team.

For this article, we extracted 55 processes for post-mortem analysis. These processes document what occurred during the development of the games. Thus, the extracted processes, as well as the extraction system, may serve as a tool to analyze past projects. For example, a developer who had developed 10 games and wants to make a retrospective of all projects, can use our system to extract 10 processes and generate their graphical representations and make comparisons, group similar projects, and even define metrics and evaluate the processes, projects, and teams.

Something that may help to solve some issues regarding the postmortem analysis and help the developers community, would be a creation of an experiences repository where finalized projects reports would be stored using a more rigid structure, describing errors and hits during the project execution.

We did not find a source related to a definition of video game project context. We created a way to define video game project contexts, based on literature, books, and postmortems. We defined 61 video game variables, divided into 6 categories. This set of variables is not immutable but is a basis for future extensions. Moreover, we encourage aggregation or refactoring variables because such operations do not interfere with how the recommending system works.

The recommendation system is the main contribution of this article. The aim of the generated processes is to help the development team in planning a game project, even before the preproduction phase. The resulting processes help developers to make decisions regarding the development phases or serving as a high level process to be followed.

For example, before the development starts, with the game context defined, the context form is filled by game developers. The recommendation system creates a new process using similar projects as a basis. The development team or responsible developer may analyze all the problems that occurred in other, similar games. With this data, it is possible to draw plans that prevent such problems and apply activities that went right.

The system evaluation, although with relatively low measures, results in good acceptance by the game developers. Developers who wrote postmortems acknowledged the similarity of the extracted and recommended processes. The developers related that many of the activities presented by the recommendation system were already part of the processes used by their teams. It emphasizes that the premise of this article, “learning with the past”, can be a way to avoid video game development problems.

7. Threats to Validity

There are concerns about the generated processes, given that the postmortem main purpose is not describe processes’ elements. It took us to carry out the validation of both, the extracted and recommended processes.

As pointed in a feedback, the recommended process cannot define specifics tasks to be followed by developers. Again, it is because the information contained in postmortems does not have such specificity. Nonetheless, we argue that it is the best source of information about development experiences.

Most of the work of creating a process visualization had to be made manually, which cost much time. Although the process visualization is built mainly by scripts, its flow is entirely made by hand. It happened because we did not stored the meta-data about elements’ flow, which can be done in future extensions.

We performed tests using a dataset with 55 samples against 61 contextual properties. This might cause noise in the dataset, that is, Overfitting issues [14].

8. Related Work

Postmortems was used by researchers as a primary source of information about game development. Petrillo et al. [3, 4] extracted, with postmortem analysis, the most common problems faced by developers in game industry. They argued that these problems occur in the traditional software industry as well. Still, the authors stated that game developers suffer mainly from managerial problems, not technical ones.

In a complementary research, Petrillo et al. list the most common practices adopted in game projects. The results showed that, even if informally, game developers are adopting agile practices and, for this reason, implanting agile methods may occur naturally [9].

Another research, done by Washburn et al. [38], regarding development practices and postmortems, analyzed 155 articles extracting the characteristics and pitfalls from the game development. The result was a set of good practices for game developers.

Murphy-Hill et al. [5] realized 14 qualitative interviews and 364 quantitative, with professionals from game development area. The authors highlighted the dubious agile methods adoption, that is, the use of the word “agile” to justify the lack of process. Moreover, Pascarella et al. [39] extended this work assessing how developing games is different from developing non-game systems by performing analysis in both Open Sourced repositories types. They found that developing video games differs from developing traditional software, in many aspects, like: the way resources evolve during the project; the expertise required by the developers; the way the software is tested; how bugs
are fixed; how release planning is followed and regarding requirements handling.

O’Hagan et al. did a literature review of 404 articles, from industry and academy, showing a total of 356 software processes, grouped in 23 models, of which 47% were agile and 53% hybrid [40]. Still, O’Hagan et al. made a case study in game industry exploring the role of software process in game development. The results showed that there is no good practices model for game development and suggested to create such model, based on ISO/IEC 29110 [41].

Kanode et al. also explored the role of software engineering in game development. For the authors, game development has unique characteristics and problems, and software engineering practices may help to overcome these difficulties. For example, implanting agile methods in pre-production phase and managing the large amount of game assets [8].

Regarding Recommendation Systems, there are several attempts to provide better tools to developers. For example, recommendation of developments artifacts [42, 43, 44, 45], contextual information [46], libraries [47], methods [48], commands [49] and even pull-requests commenter [50].

These related works showed the game industry problems, suggested the use of good practices and provide recommendation of different development artifacts. However, there is no mention, as far as we know, about a implementation and validation of a system to aid developers, recommending a set of activities and practices in the form of a process, based on similar projects context.

9. Conclusions

Building a video game demands a large number of activities and choosing these activities is arduous and demands experience. This article presented an approach and a recommendation system, based on video game development postmortems, to suggest software processes using contexts and similarity degrees. The recommendation system helps video game developers to conduct new projects by suggesting a set of activities in a form of a process. It learns from previous game development experiences within similar contexts. Moreover, provides to video game developers an approach to reflect about the processes used in game industry.

This article thus presents three main contributions. It first describes the creation of a database of game development processes from the analysis of 55 postmortems. The second identifies video game project characteristics, like team attributes and technical details, to model projects contexts that work as input for the recommendation system. The third and final one describes and validates a recommendation system capable of generating processes based on previous projects with similar contexts.

The quantitative results of the validation of the recommendation system showed an average precision of 31.58%; good accuracy of 67.59%; and excellent specificity of 92.48%. However, other metrics had lower outcomes, like recall at only 8.11%. These results showed that a better technique must be provided, in order to filter the recommended processes’ elements: provide more elements with more relevance. The former is achievable by adding a larger number of postmortems in the database while the latter requires using better classification algorithms, improving the video game context categories, and enriching the elements of the processes.

The qualitative validation provided by video game developers showed that our approach does extract similar process from postmortems as well our recommendation system suggests useful new processes.

Finally, through a case study with a team of developers, we showed that video game development teams may, transparently, improve their software processes during the course of game development. Thus, our recommendation system can recommend video game development processes that help game developers to avoid possible problems.

As future work, we intend to improve our recommendation system by adding more postmortems and analyzing their activities. We will gather more resources and feedbacks from video game developers to improve our model. Moreover, we will improve the method used to generate processes visualization as well as the user interface for the recommendation system. Further, we will add more metadata in processes’ elements, like order and flow. With this we hope cut most of the hand working and provide a completely automated processes recommendation system.

Acknowledgements

We give our thanks to the authors of the postmortems for sharing their cases as well for their feedbacks. This work has been supported by the National Council for Scientific and Technological Development of Brazil.

References

[1] NEWZOO, The global games market reaches $99.6 billion in 2016, mobile generating 37%, https://goo.gl/hrDlhL (Apr. 2016).
[2] A. Gershenfeld, M. Loparco, C. Barajas, Game plan: the insider’s guide to breaking in and succeeding in the computer and video game business, St. Martin’s Griffin Press, New York, 2003.
[3] F. Petrillo, M. Pimenta, F. Trindade, C. Dietrich, Houston, we have a problem:: a survey of actual problems in computer games development, in: Proceedings of the 2008 ACM symposium on Applied computing - SAC 08, ACM Press, 2008. doi:10.1145/1363686.1363854. URL https://doi.org/10.1145%2F1363686.1363854
[4] F. Petrillo, M. Pimenta, F. Trindade, C. Dietrich, What went wrong? a survey of problems in game development, Computers in Entertainment 7 (1) (2009) 1. doi:10.1145/1486508.1486521. URL https://doi.org/10.1145%2F1486508.1486521
[5] E. Murphy-Hill, T. Zimmermann, N. Nagappan, Cowboys, ankle sprains, and keepers of quality: how is video game development different from software development?, in: Proceedings
of the 36th International Conference on Software Engineering - ICSE 2014, ACM Press, 2014. doi:10.1145/2568225.2568226.

[6] C. Poltowsi, L. Fontoura, F. Petriollo, Y.-G. Guchéneuc, Are the old days gone?: A survey on actual software engineering processes in video game industry, in: Proceedings of the 5th International Workshop on Games and Software Engineering, GAS ‘16, ACM, New York, NY, USA, 2016, pp. 22-28. doi:10.1145/2896956.2896960. URL https://doi.org/10.1145/2896956.2896960.

[7] J. Westmar, M. Andrei-Gedja, Developer Satisfaction Survey 2015 Industry Trends and Future Outlook Report, Tech. rep., International Game Developers Association (IGDA) (2016).

[8] C. M. Kanode, H. M. Haddad, Software engineering challenges in game development, in: 2009 Sixth International Conference on Information Technology: New Generations, IEEE, 2009. doi:10.1109/itng.2009.74. URL https://doi.org/10.1109/itng.2009.74.

[9] F. Petriollo, M. Pimenta, Is agility out there?, in: Proceedings of the 28th ACM International Conference on Design of Communication - SIGDOC 10, ACM Press, 2010. doi:10.1145/1878640.1878453. URL https://doi.org/10.1145/1878640.1878453.

[10] C. Keith, Agile Game Development with Scrum, 1st Edition, Addison-Wesley Professional, 2010.

[11] C. Poltowsi, D. de Vargas, L. M. Fontoura, A. A. Fuleto, Software engineering processes in game development: a survey about brazilian developers’ experiences, in: XV Simpósio Brasileiro de Jogos e Entretenimento Digital, SGBAMES ’16, 2016.

[12] W. Hamann, Goodbye postmortems, hello critical stage analysis, https://goo.gl/bbLVWA (jul 2003).

[13] M. P. Robillard, W. Maalej, R. J. Walker, T. Zimmermann, Recommendation systems for software engineering, IEEE Software 27 (4) (2010) 80–86. doi:10.1109/MS.2009.161.

[14] M. P. Robillard, W. Maalej, R. J. Walker, T. Zimmermann, Recommendation systems in software engineering, Springer Science & Business, 2014.

[15] W. S. Humphrey, The software engineering process: Definition and scope, in: Proceedings of the 4th International Software Process Workshop on Representing and Enacting the Software Process, ISPW ’88, ACM, New York, NY, USA, 1988, pp. 82–83. doi:10.1145/75110.75122. URL http://doi.acm.org/10.1145/75110.75122.

[16] M. Fowler, The new methodology, Wuhan University Journal of Natural Sciences 6 (1) (2003) 12–24. doi:10.1007/BF03160222. URL http://dx.doi.org/10.1007/BF03160222.

[17] W. W. Royce, Managing the development of large software systems: Concepts and techniques, in: Proceedings of the 9th International Conference on Software Engineering, ICSE ’87, IEEE Computer Society Press, Los Alamitos, CA, USA, 1987, pp. 328–338. URL http://dl.acm.org/citation.cfm?id=41765.41801.

[18] C. Larman, V. R. Basili, Iterative and incremental developments, a brief history, Computer 36 (6) (2003) 47–56. doi:10.1109/MC.2003.1204375.

[19] K. Schwaber, Agile Software Development with Scrum, Series in agile software development, Pearson Education International, 2002. URL https://books.google.com.br/books?id=103XLgA4CAAJ.

[20] K. Beck, Extreme Programming Explained: Embrace Change, An Alan R. Apt Book Series, Addison-Wesley, 2000. URL https://books.google.com.br/books?id=GEL8r4f7UC.

[21] C. Larman, V. Basili, Iterative and incremental developments, a brief history, Computer 36 (6) (2003) 47–56. doi:10.1109/mc.2003.1204375. URL https://doi.org/10.1109/MC.2003.1204375.

[22] J. Musil, A. Schweda, D. Winkler, S. Biff, A survey on the state of the practice in video game software development, Tech. rep., Technical report, QSE-IFS-10/04, TU Wien (2010).

[23] J. Musil, A. Schweda, D. Winkler, S. Biff, A survey on the state of the practice in video game software development, Tech. rep., Technical report, QSE-IFS-10/04, TU Wien (2010).

[24] A. Fuggetta, E. D. Nitto, Software process, in: Proceedings of the on Future of Software Engineering - POSE 2014, ACM Press, 2014. doi:10.1145/2593882.2593883. URL https://doi.org/10.11452593882.2593883. https://doi.org/10.11452593882.2593883.
[43] R. Holmes, A. Begel, Deep intellisense: A tool for rehydrating evaporated information, in: Proceedings of the 2008 International Working Conference on Mining Software Repositories, MSR ’08, ACM, New York, NY, USA, 2008, pp. 23–26. doi:10.1145/1370750.1370755.
URL http://doi.acm.org/10.1145/1370750.1370755

[44] R. Holmes, R. J. Walker, G. C. Murphy, Strathcona example recommendation tool, SIGSOFT Softw. Eng. Notes 30 (5) (2005) 237–240. doi:10.1145/1095430.1081744.
URL http://doi.acm.org/10.1145/1095430.1081744

[45] T. Zimmermann, P. Weisgerber, S. Diehl, A. Zeller, Mining version histories to guide software changes, in: Proceedings of the 26th International Conference on Software Engineering, ICSE ’04, IEEE Computer Society, Washington, DC, USA, 2004, pp. 563–572.
URL http://doi.acm.org/citation.cfm?id=998675.999460

[46] L. Ponzanelli, S. Scalabrino, G. Bavota, A. Mocci, R. Oliveto, M. Di Penta, M. Lanza, Supporting software developers with a holistic recommender system, in: Proceedings of the 39th International Conference on Software Engineering, ICSE ’17, IEEE Press, Piscataway, NJ, USA, 2017, pp. 94–105. doi:10.1109/ICSE.2017.17.
URL https://doi.org/10.1109/ICSE.2017.17

[47] A. Ouni, R. G. Kula, M. Kessentini, T. Ishio, D. M. German, K. Inoue, Search-based software library recommendation using multi-objective optimization, Information and Software Technology 83 (2017) 55–75.

[48] I. Velásquez, A. Caro, A. Rodríguez, Kontun: A framework for recommendation of authentication schemes and methods, Information and Software Technology

[49] M. Gasparic, A. Janes, F. Ricci, G. C. Murphy, T. Gurbanov, A graphical user interface for presenting integrated development environment command recommendations: Design, evaluation, and implementation, Information and Software Technology 92 (2017) 236–255.

[50] J. Jiang, Y. Yang, J. He, X. Blanc, L. Zhang, Who should comment on this pull request? analyzing attributes for more accurate commenter recommendation in pull-based development, Information and Software Technology 84 (2017) 48–62.
Table 10: Recommended elements for the context used in the case study project.

| Element                  | Description                                                                 |
|--------------------------|-----------------------------------------------------------------------------|
| test team                | Small team with focused on development, not marketing                       |
| small team               | Beta test group from community. Beta testers with access to bug database.    |
| general team details     | Lead designer no present. Lack of artists. Experienced team who already worked together. No need to research, team have know-hall about tasks to complete. Full focused with no business meetings. |
| outsourcing              | Outsourcing experienced people with technical expertise. Outsourcing PR / marketing. Outsourcing code and assets. |
| horizontal development   | Follow a higher principle. Everyone may vet new ideas                        |
| development problems     | Keep the pace and not “crunch” (work overtime) Lack of proper pipeline. Hard to solve things by yourself. Cutting or reworking features. |
| development process      | Lack of process or structure. Just code without planning. List of main features instead of a design document. Share the development with the audience. No schedule, nor milestones, nor meetings nor design document nor technical plan Development flow |
| engine and tools          | Use a tool for build distribution. Learn everything from scratch.             |
| infrastructure           | Self made engine and-or tools. Engine and-or legacy code limiting improvements. |
| project focus            | Focus on the team strength.                                                 |
| concept                  | Artists, level designers, programmers, and animators working together. Set of design principles. Brainstorming ideas based in a goal. Align thinking before prototyping phase. Research similar games. Stick with the game concept since the beginning. |
| brainstorming features    | Brainstorming ideas finding “fun factor”.                                    |
| business tasks           | Market study. Selling directly to customers. Pre-order program. Marketing strategy |
| design tasks             | Procedurally generated instead of hand craft levels. AI specialists.          |
| development iterations   | Digital distribution allowing testers give feedback quickly and often. Heavy tested by non designers. Schedule iterations with buffers, that is, double or triple the time required for a task. Heavily focused on iterations and constant improvements in the game. Legacy problems being solved as the development goes on. |
| exploration phase        | Prototyping a game without polishing, trying to find a good game play. Heavily prototyping and testing iterations. Avoid making experiments in production phase. |
| in-house tools development| Creating tools to aid developers with a new technology and avoid unknown bugs. Tools allowing non coders to change parameters and experiment with the game engine. The main goal is to create a new engine. Creating an IDE. Creating an Engine with experienced team. Creating tools to automatize tasks and processes. |
| milestones planning      | Pre-order program to raise money allowing the team continue the development. Early access feedback. |
| multi-player construction| Specialist in multi-player development.                                      |
| pitch                    | Pitch the game concept.                                                      |
| planning documentation   | Production plan with buffers (more time) after each milestone. Cutting features, making less but better. Improving performance |
| polish and refinements   |                                                                                           |
| professional feedback    | Getting technical feedback.                                                   |
| refactoring the development| Time expended with re-designs. Refactoring old code.                          |
| requirements and constraints | Game design documentation not updated. Define a list of features and tools. Port UI to other platform. |
| retrospective meeting    | Research target audience.                                                      |
| testing                  | Beta test group. Multi-platform testing. Delivery builds quickly to test team. Long time beta testing the game. Get feedback from QA with volunteers. Feedback from testers not used properly. |
| vertical slice           | Community working together creating assets.                                  |