Information and Software Technology xxx (xxxx) xxx–xxx

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Learning from the past: A process recommendation system for video game projects using postmortems experiences☆

Cristiano Politowski*,[a], Lisandra M. Fontoura[a], Fabio Petrillo[b], Yann-Gaël Guéhéneuc[b]

[a] Departamento de Computação Aplicada (DCOM), Universidade Federal de Santa Maria, Santa Maria, RS, Brazil
[b] Department of Computer Science & Software Engineering, Concordia University, Montréal Quebec H3G 1M8, Canada

ARTICLE INFO

Keywords:
Software development process
Video game development
Recommendation system

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.

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 [1].

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

Problems like "unrealistic scope", "feature creep"[1] and "cut of features" [3,4] are recurrent in game development. In addition, because of lack of maturity, development teams often do not adopt systematic processes [5] or they use traditional approaches, such as the waterfall process [6]. Other problems were identified in the IGDA annual report [7] in which, for example, 52% of the interviewees answered "yes" when asked if "crunch time"[2] is necessary during game development.

To help game developers minimize these problems, researchers proposed approaches to apply agile practices in game development [8–10]. However, recent work showed that at least 35% of the development teams continue using hybrid processes (mixing waterfall and agile) or *ad-hoc* processes [6]. Other researchers have tried, through a survey, understand the problems faced by game developers and the role

☆ All the material of this project, including the extracted processes, video game contexts, and validations data is available on-line at https://polako.github.io/rec-sys-proc-games/.

* Corresponding author.
E-mail addresses: cpolitowski@inf.ufsm.br (C. Politowski), lisandra@inf.ufsm.br (L.M. Fontoura), fabio@petrillo.com (F. Petrillo), yann-gael.gueheneuc@concordia.ca (Y.-G. Guéhéneuc).

[1] *Feature creep* occurs in any software project when new functionalities are added during the development phase, thus increasing the project size [4].

[2] *Crunch time* is the term used in the video game industry for periods of extreme work overload, typically in the last weeks before the validation phase and in the weeks that precede the final deadline for project delivery [4].

https://doi.org/10.1016/j.infsof.2018.04.003
Received 5 October 2017; Received in revised form 15 March 2018; Accepted 11 April 2018
0950-5849/ © 2018 Elsevier B.V. All rights reserved.

of software engineering in the game industry [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 "postmortems", 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. Fig. 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.

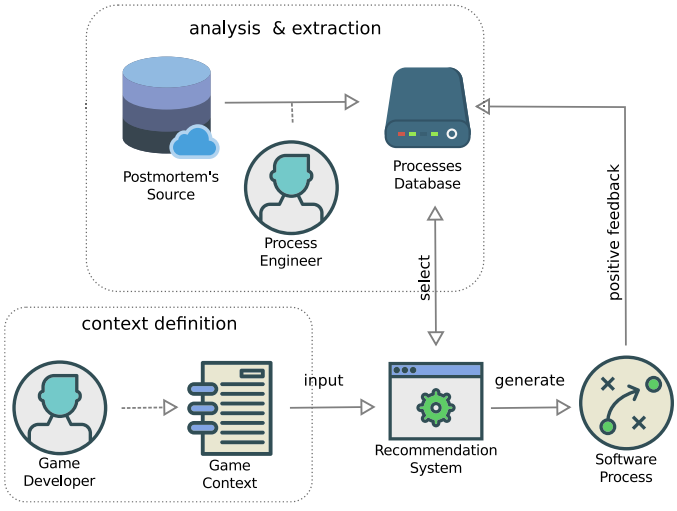

Fig. 1. Summary of the approach.

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 a 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 postmortems. 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

Download English Version:

https://daneshyari.com/en/article/6948020

Download Persian Version:

https://daneshyari.com/article/6948020

Daneshyari.com