FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Collaborative Feedback System

BACHELORARBEIT

zur Erlangung des akademischen Grades

Bachelor of Science

im Rahmen des Studiums

Medieninformatik und Visual Computing

eingereicht von

Sebastian Gilits
Matrikelnummer 00825197

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Pretitle Peter Purgathofer, Posttitle
Mitwirkung: Pretitle Forename Surname, Posttitle

Wien, 1. Februar 2018

Sebastian Gilits	Peter Purgathofer

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Collaborative feedback system

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Media Informatics and Visual Computing

by

Sebastian Gilits
Registration Number 00825197

to the Faculty of Informatics

at the TU Wien

Advisor: Pretitle Peter Purgathofer, Posttitle
Assistance: Pretitle Forename Surname, Posttitle

Vienna, 1st February, 2018

Sebastian Gilits

Peter Purgathofer

Erklärung zur Verfassung der Arbeit

Sebastian Gilits
Rosengasse ..., ...

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 1. Februar 2018

Sebastian Gilits

Kurzfassung

Ihr Text hier.

Abstract

Enter your text here.

Contents

CHAPTER 1

Introduction

Getting feedback for a software system is one of the most important tasks, since it gives insights if the implemented system works, if there are any bugs and also what other features could be implement in order to improve the software.

1.1 Motivation/Problem statement

For the last n years the learning platform Aurora has been used by hundreds(?) of students for several courses at the Vienna University of Technology. The system was originally implemented by other students for the master thesis or for courses(?).

A system created by so many different persons naturally introduces new bugs, which are hard to track, since often the not the original author, but a different student might have to fix it.

In the past an online notebook has been used to keep track of different issues and features request for the aurora system. Every year a new notebook was created and people could freely add bugs and feature requests to it. This method of getting feedback came at high cost though:

The notebook would get cluttered within the first weeks and keeping it organised, meant that someone had to constantly moderate the notebook. People would often add the same issue, since it was hard to find existing issues. As soon as an issue was fixed, the issue had to be found in the notebook and somehow marked as such.

1.2 Aim of work

The aim of this thesis was to find a solution to collect feedback efficiently, without the high moderation costs that normal issue queues introduce and with ability to give users

feedback about the state of their issue. Another requirement was that system should collaborative in order to encourage to the users to post, but also discuss feedback given.

The last requirement was to find a system that could be integrated directly into aurora. Using an external was not an option, since Austrian laws requires to keep the data within the infrastructure of the university, but also because requiring students to register at another platform would have added an additional barrier for the students to give feedback.

CHAPTER 2

State of art

Collecting feedback for software projects is a common task. There are dozens of different solutions and more open source and commercial services available then one can count.

This systems range from simple contact forms, which send out an email to the provider, to chat bots and sophisticated ticketing system, with in-depth dashboards, visualizing how good the feedback system itself works,...

In this chapter I'll present some of the most popular solutions and compare them to each other and also pointing out how good it would work with Aurora.

2.1 Existing solutions/Literature studies

I this chapter I will present some typical forms of collecting feedback, as well as some system that use this methods.

2.1.1 Contact form

2.1.2 Ticketing system

(describe solutions like redmine, jira, etc.)

2.1.3 Kanban

Kanban, which is Japanese and literally translates to signal board, is a system originally invented at the Japanese car manufacturer Toyota, in order to improve production speed and lower costs. In this environment cards are used to signal a supplier the depletion of a part or some material. The supplier would then restock the needed items to certain limit, thus implementing a consumption driven supply chain.

In 2007 David R. Anderson presented an adaption of this system for software engineering. Manufacturing products and software development are two very different tasks, and thus the Kanban system used by Toyota can't be used literallyfor IT projects. David Anderson combined the essence of the system with lean software approaches to create software development process.

In this new system each card represents work that is pulled through several steps. Exemplary steps would be "Backlog", "In Progress", "In testing" and "Deployed". Steps where actual work has to be done, like "In Progress" or "In testing" have to be limited, so that the work capacity of the development team is represented. If for example within team with two developers has two tickets for "Work in progress", it shouldn't accept a 3rd one, since most probably the two programmers can't work on it. Only after finishing one of the two existing tickets, and pushing them to the next lane, the next card can be pulled to the "Work in progress" step.

With this system it's easy to determine how much workload exists, what the throughput is and thus optimise workflow.

2.2 Comparison of existing solutions

CHAPTER 3

Methodology

3.1 Used concept/Kanban

3.2 Design methods

CHAPTER 4

Implementation

CHAPTER 5

Critical reflection

CHAPTER 6

Summary

List of Figures

List of Tables

List of Algorithms

Bibliography