FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

WebProcessPAIR: Recommendation System of Improvement Actions

Luis Amaro

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: João Carlos Pascoal Faria

June 27, 2016

WebProcessPAIR: Recommendation System of Improvement Actions

Luis Amaro

Mestrado Integrado em Engenharia Informática e Computação

June 27, 2016

Abstract

To help in the development of high-integrity software it is recommended the use of proper processes, such as PSP (Personal Software Process) or TSP (Team Software Process). PSP/TSP are software development processes of high maturity used to improve the ability of estimating and planning projects, manage their quality as well as reduce defects.

Processes like PSP/TSP generate large amounts of data about the performance of a user which can be periodically analysed to identify performance problems, identify the causes of the problems and devise improvement actions.

However, the manual analysis of the data generated by these processes is time-consuming and tiring, due to its large quantity and the time and expertise required to perform the analysis. To solve this problem, it was developed in previous work a tool in java, called ProcessPAIR, that automatically analyses performance data, identifies performance problems, and identifies and ranks their root causes.

The main objective of this dissertation is to extend the ProcessPAIR tool to the web, WebProcessPAIR, with a new feature, recommend improvement actions. This functionality is based on building a catalogue of possible improvement actions to address the causes of performance problems. For this purpose, it was used not only the existing literature but also used methods such as crowdsourcing, i.e., appeal to the knowledge of a community of users and experts. Thus, WebProcessPAIR aims to help software developers analyse their performance data with less effort, and to identify potential performance problems, causes and improvement actions to address those causes.

The dissertation follows an organization that presents clearly the topics covered. Initially it is presented the context and the expected objectives in this dissertation, followed by the background and state of the art about software process improvement, crowdsourcing, and recommendation systems. Regarding the development of WebProcessPAIR, it is discussed the implementation, design and testing, as well as the WebProcessPAIR usage and validations. Finally, it is presented the conclusions and future work, followed by the references.

Resumo

Para ajudar no desenvolvimento de *software* de elevada integridade é recomendado o uso de processos apropriados, como por exemplo, o PSP (*personal software process*) ou o TSP (*team software process*). O PSP/TSP é um processo de desenvolvimento de *software* de elevada maturidade usado para melhorar a capacidade de estimativa e planeamento de projetos, gerir a sua qualidade, assim como reduzir defeitos. Processos como PSP e TSP geram grandes quantidades de dados sobre o desempenho do utilizador que periodicamente podem ser analisados para identificar problemas de desempenho, determinar as causas dos problemas, e desenvolver ações de melhoria. No entanto, a análise manual dos dados gerados por estes processos é uma tarefa demorada e cansativa, devido à quantidade de dados a analisar e ao tempo e conhecimento especializado necessários para realizar a análise. Para resolver este problema, foi desenvolvida uma ferramenta em java, denominada ProcessPAIR, que analisa automaticamente os dados de desempenho, identifica possíveis problemas de desempenho e determina e prioritiza as causas desses problemas.

O principal objetivo desta dissertação é estender a abordagem e ferramenta ProcessPAIR para a *web*, WebProcessPAIR, com uma nova funcionalidade, recomendar ações de melhoria. Esta funcionalidade baseia-se na construção prévia de um catálogo de possíveis ações de melhoria para abordar as causas dos problemas de desempenho. Para esse efeito recorreu-se não só à literatura existente, mas também ao uso de métodos como *crowdsourcing*, ou seja, ao conhecimento de uma comunidade de utilizadores e especialistas. Sendo assim, o WebProcessPAIR permite ajudar os desenvolvedores de *software* a analisar os seus dados de desempenho com menos esforço, e a obter indicações sobre possíveis problemas de desempenho, causas e ações de melhoria para as mesmas.

A dissertação obedece a uma organização própria que pretende apresentar de forma clara os temas abordados. Assim, foram inicialmente apresentados o contexto e os objetivos esperados nesta dissertação, seguido do estado da arte sobre processos de melhoria de *software*, *crowdsourcing* e sistemas de recomendação. Sobre o desenvolvimento do WebProcessPAIR, foi abordada a implementação, design e testes, assim como explicadas as instruções de uso e validações. Por último foram apresentadas as conclusões e trabalho futuro, seguido das referências.

Contents

List of figures

List of tables

Abbreviations

AIMS2	Accelerated Improvement Method Strongstep 2
CRUD	Create, Read, Update, Delete
IA	Improvement action
MVC	Model, Views, Controllers
navbar	Navigation bar
PI	Performance indicator
PSP	Personal Software Process
SEI	Software Engineering Institute
TSP	Team Software Process
UI	User interface

Chapter 1

Introduction

In this chapter is introduced the dissertation work, presenting the context and motivation for its development and explaining the objectives to be achieved. Finally, it is presented the structure of the dissertation.

1.1 Context and motivation

To help software engineers in the development of high-integrity software, it is recommended the use of processes, such as the PSP (Personal Software Process)/ TSP (Team Software Process). The Personal Software Process (PSP) [1] is an example of a process framework that was specifically designed by the Software Engineering Institute (SEI) to help individual software engineers improve their performance and become effective team members following the additional team building and team management practices of the Team Software Process (TSP).

High-maturity processes like the PSP generate large amounts of data about the developer performance. This data can be analysed to identify performance problems (for example, a high density of defects in delivered products), identify the causes of the problems (for example, high number of defects that reach the testing phase of the system) and develop improvement actions (for example, introducing unit testing and code reviews).

However, the manual analysis of the data generated by these processes is a time-consuming and tiring task, due to the large quantity of data to analyse, and the time and expertise required to perform the analysis. To solve this problem, it was developed in a PhD thesis, a tool in java, called ProcessPAIR [1], which identifies and ranks performance problems and their root causes.

This dissertation will be held in partnership with the project AIMS2, promoted by Strongstep, FEUP and Match Profiler, with the objective of developing a tool (Scraim) to help organizations increase their productivity and reduce their expenses, through the provision of a service of integrated project management, supported by turnkey development processes.

1.2 Objectives

Like is shown in Figure 1, so far the ProcessPAIR tool identifies performance problems, identifies root causes and ranks root causes. The main objective of this dissertation is the development of a web application, named WebProcessPAIR, that extends the previous approach and tool with a new feature, the recommendation of improvement actions. Incorporating the ProcessPAIR tool in the website, WebProcessPAIR, allows users to analyse automatically their performance data generated by PSP and Scraim (future work), presenting them performance problems, their causes and possible improvement actions.

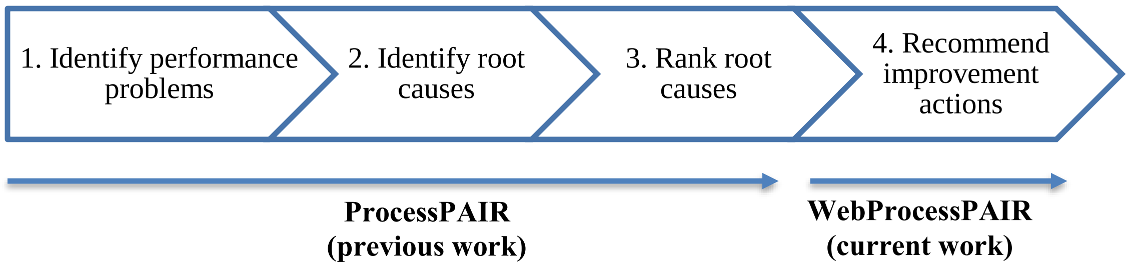

1. Identify performance problems
2. Identify root causes
3. Rank root causes
4. Recommend improvement actions
ProcessPAIR (previous work)
WebProcessPAIR (current work)

Figure 1 - Steps of the WebProcessPAIR approach

To better illustrate the approach of ProcessPAIR, in Figure 2 is shown an example of a common identified problem in project management, "Project behind schedule", that is identified through metrics such as the Schedule Performance Index (SPI), which calculates the deviation of the deadline (reference value: deviation of 20%). For this problem were presented three possible causes: effort under-estimation, under-allocation of resources and blocked tasks. Based on identified causes and expert knowledge we can recommend improvement actions (e.g.

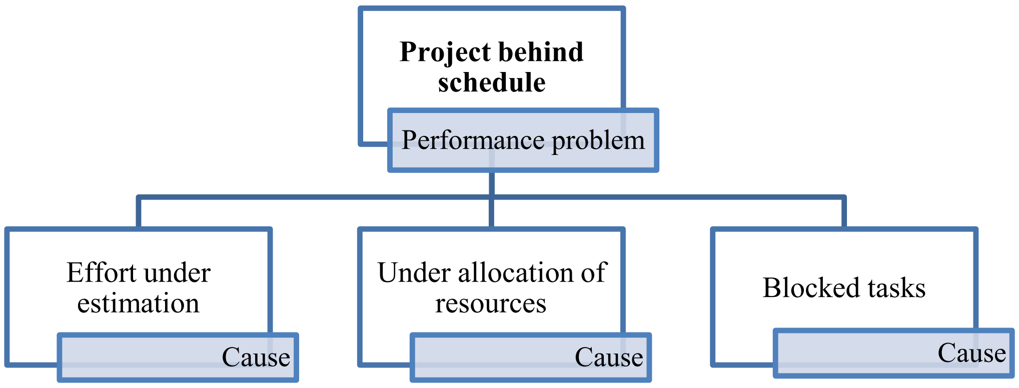


Figure 2 - Example of performance problem and causes

recommend the usage of reliable estimation techniques). These improvement actions should prevent or reduce subsequent occurrences of the same type of problem.

WebProcessPAIR allows to do this in a practical and quick way, helping users to improve their performance in software development. For a better understanding of the approach, in Figure 3 is shown a scheme of the functionality of WebProcessPAIR. This functionality is divided into two parts, building a catalogue of improvement actions, and recommending improvement actions to the problems encountered.

The construction of the catalogue refers to the highest-ranked causes of performance problems found by the tool, and to this end, the objective is to appeal not only to existing literature but also to the use of methods such as crowdsourcing, i.e., appeal to the knowledge of a community of users and experts to propose solutions to these causes. In the website is presented a list of performance problems from many developers, where the community will contribute to the solution of the identified problems through comments or votes. As new recommendations are made, the system will update the catalogue with the recommendations with the most positive votes given by users, thus maintaining a catalogue updated with the best solutions for various types of problems.

Later, this catalogue will be used for the recommendation of improvement actions. For each performance problem and respective cause presented in the platform, are recommended improvement actions from the catalogue, suggested by the community of WebProcessPAIR. For each improvement action suggested to a user, the user can leave his feedback, accepting the proposal or not.

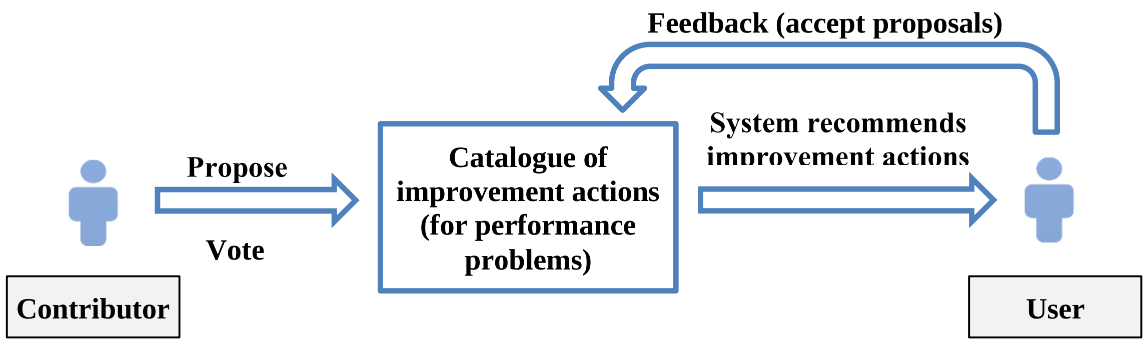

Feedback (accept proposals)
Propose
Vote
Catalogue of improvement actions (for performance problems)
System recommends improvement actions
Contributor
User

Figure 3 - Functionality scheme of WebProcessPAIR

1.3 Dissertation's structure

The rest of this dissertation is organized as follows. Chapter 2 describes the background and state of art regarding software process improvement, crowdsourcing and recommendation systems based on crowdsourcing. Chapter 3 presents the WebProcessPAIR design, implementation and testing such as use cases, conceptual model, architecture and technologies, database structure, MVC components, navigation map, integration with ProcessPAIR and acceptance tests. Chapter 4 describes the WebProcessPAIR usage, such as how to access WebProcessPAIR, user registration, definition of performance indicators, recommend actions and performance analysis. Chapter 5, last chapter, describes the conclusions and future work. Finally, Appendix A presents the current status of WebProcessPAIR regarding the improvement actions created by WebProcessPAIR users.

Chapter 2

Background and State of the Art

This chapter describes the state of the art in order to contextualize the topic of work. This way it will be discussed the following topics: software process improvement, crowdsourcing, and recommendation systems based on crowdsourcing.

2.1 Software process improvement with ProcessPAIR and PSP PAIR

In order to better understand what the ProcessPAIR tool is and how it works, in this section will be presented a detailed analysis of ProcessPAIR, as well as its previous version, PSP PAIR.

2.1.1 ProcessPAIR

ProcessPAIR is a novel tool for automated model-based analysis of performance data produced in the context of high-maturity software processes, such as the Personal Software Process (PSP) and the Team Software Process (TSP). The tool is currently able to analyse the performance data produced by PSP users, to identify performance problems and rank their root causes. The analysis is based on a performance model calibrated based on a large dataset from the Software Engineering Institute (SEI), referring to more than 30,000 projects. The tool is currently freely available at https://blogs.fe.up.pt/processpair/ [2].

Overall approach

For enabling the automated analysis of performance data of PSP developers, i.e., the identification of performance problems and root causes, it was conceived a performance model. The performance model comprises a set of performance indicators (PI) and cause-effect relationships, for identifying root causes of performance problems.

An overview of the artefacts and steps involved in the approach for automated performance analysis is shown in Figure 4. In order to enable the automated identification of performance problems (B1), one has to first decide on the relevant performance indicators (A1) and ranges (A3). In order to enable the automated identification of root causes of performance problems (B2), one has to first decide on the relevant cause-effect relationships (A2). When multiple root causes are identified for a performance problem, it is important to rank them according to a cost-benefit estimate of improvement efforts (B3). In our approach, the cost of improving an affecting PI (root cause) is estimated based on its approximate statistical distribution (A4), and the benefit on the affected PI is estimated based on a sensitivity coefficient (A5). The proposed work for this dissertation is to develop a recommendation system which will recommend improvement actions for the highest ranked causes identified in the previous work (B4). [3]

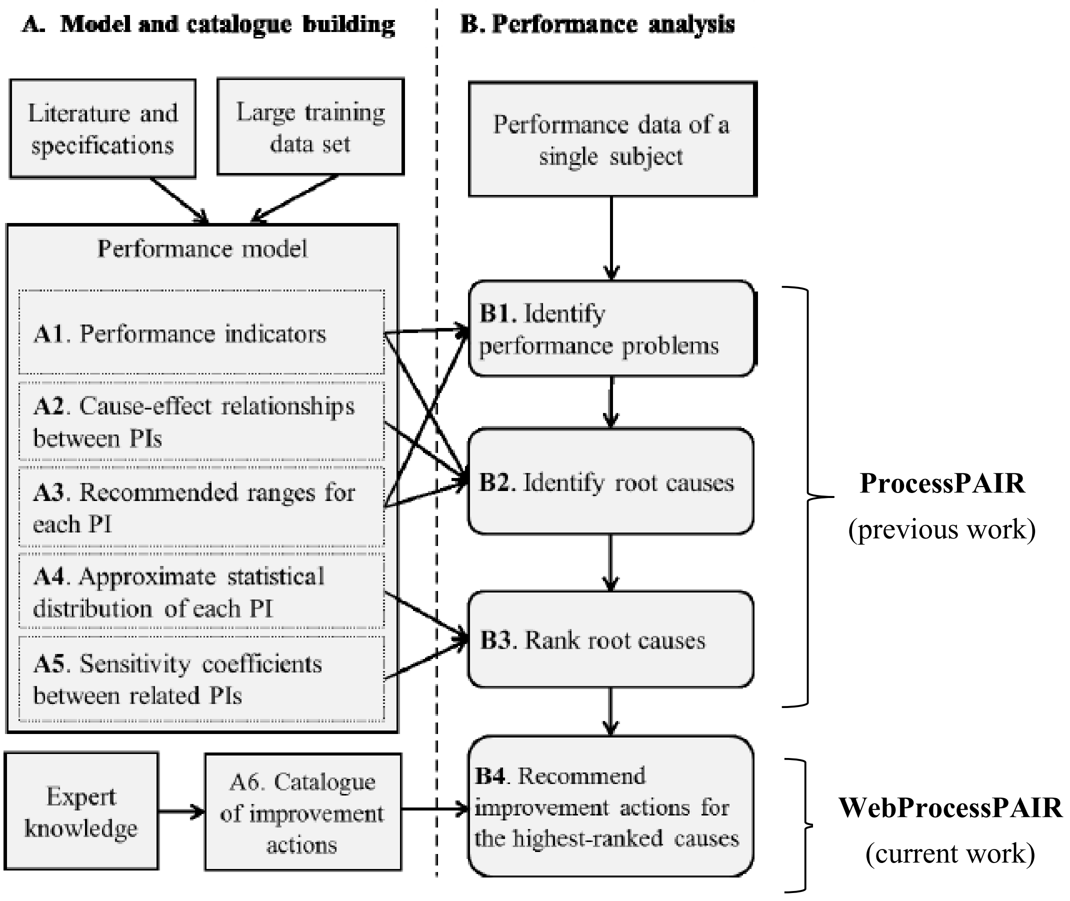
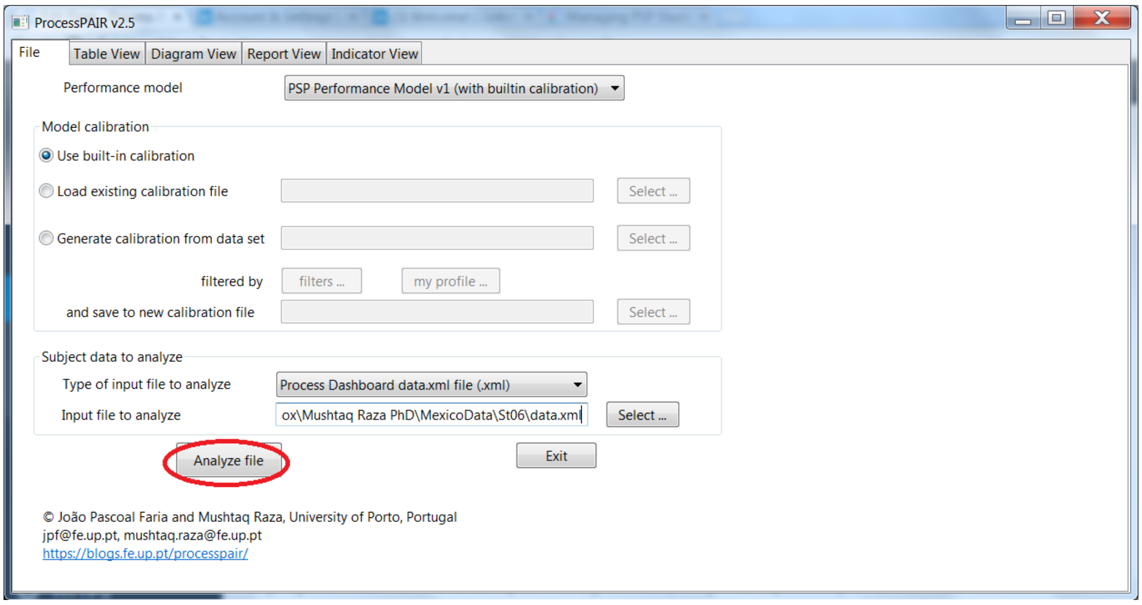

A. Model and catalogue building
B. Performance analysis
Literature and specifications
Large training data set
Performance data of a single subject
Performance model
A1. Performance indicators
A2. Cause-effect relationships between PIs
A3. Recommended ranges for each PI
A4. Approximate statistical distribution of each PI
A5. Sensitivity coefficients between related PIs
B1. Identify performance problems
B2. Identify root causes
B3. Rank root causes
ProcessPAIR
(previous work)
Expert knowledge
A6. Catalogue of improvement actions
B4. Recommend improvement actions for the highest-ranked causes
WebProcessPAIR
(current work)

Figure 4 - UML activity diagram depicting the main activities and artefacts in the approach

Tool overview

Launch and file view

By launching ProcessPAIR tool (Figure 5) [2] the user can choose the type of input file to analyse. Currently, the tool is able to analyse the export files, with ".mdb" extension, generated from the SEI's PSP Student Workbook, or the "data.xml" files exported from Process Dashboard[1]. By selecting the input file and pressing the "Analyze file" button the user will get the detailed analysis of the performance data.



ProcessPAIR v2.5

File | Table View | Diagram View | Report View | Indicator View

Performance model: PSP Performance Model v1 (with builtin calibration)

Model calibration
- [✓ Use built-in calibration]
- Load existing calibration file — Select ...
- Generate calibration from data set — Select ...
  - filtered by: filters ... | my profile ...
  - and save to new calibration file — Select ...

Subject data to analyze
- Type of input file to analyze: Process Dashboard data.xml file (.xml)
- Input file to analyze: ox\Mushtaq Raza PhD\MexicoData\St06\data.xml — Select ...

Analyze file | Exit

© João Pascoal Faria and Mushtaq Raza, University of Porto, Portugal
jpf@fe.up.pt, mushtaq.raza@fe.up.pt
https://blogs.fe.up.pt/processpair/

Figure 5 - ProcessPAIR main menu

Performance indicators

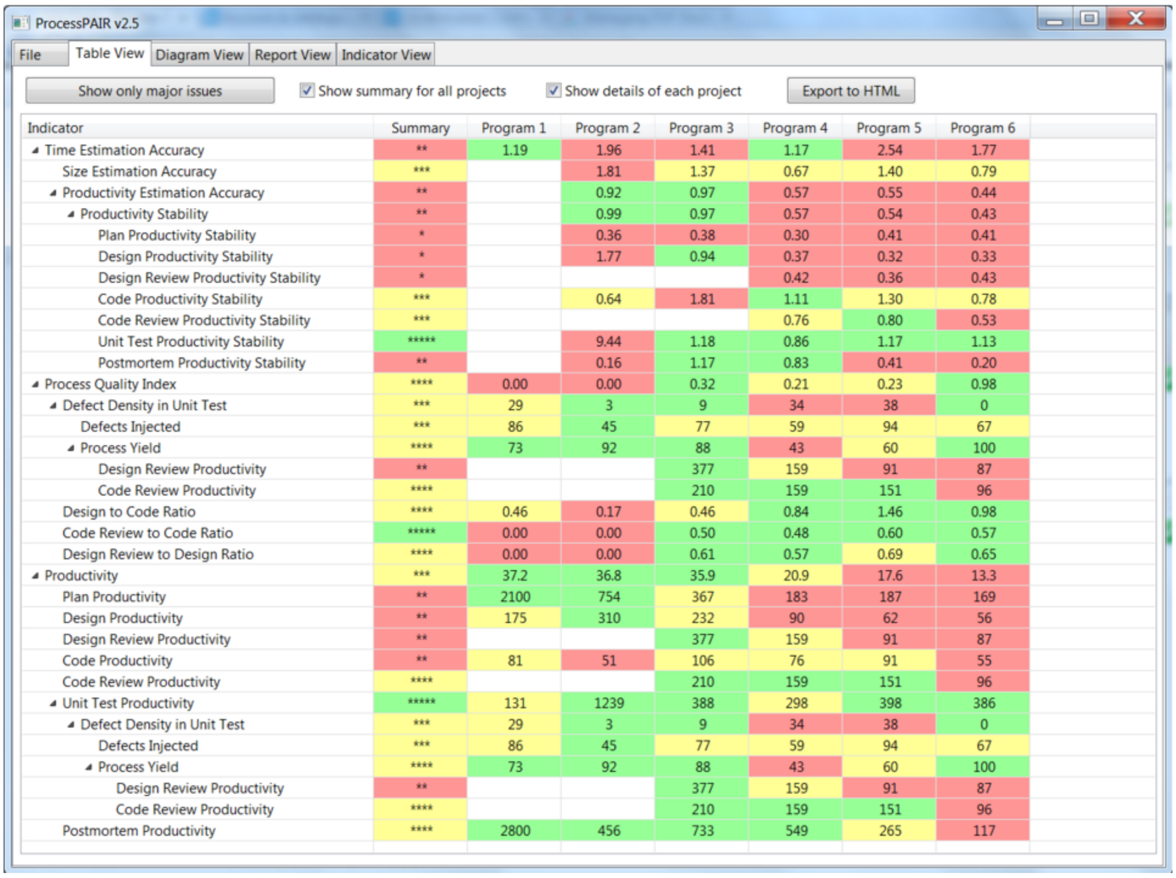
In the ProcessPAIR tool the user can check in the "Table view" (Figure 6) the values of the performance indicators for the data analysed, as well as summarized performance information.

Each cell is coloured green, yellow or red, in case its value suggests no performance problem, a potential performance problem, or a clear performance problem, respectively. This way, the Table View helps quickly identifying performance problems. The exact ranges considered can be consulted in the "Indicator View". To skip to a specific indicator in the "Indicator View", the user can click on the indicator name in the first column.

The "Summary" column shows an overall rating between 1 and 5 stars for each performance indicator, computed from the per project values (with higher importance for the last projects), and coloured according to the number of stars.

[1] www.processdash.com/

The performance indicators are organized hierarchically, starting from three top-level indicators (Time Estimation Accuracy, Process Quality Index, and Productivity), and descending to lower level indicators (child indicators) that affect the higher level ones according to a formula or statistical evidence [1]. This way, by drilling down from the top-level indicators to the lower level ones, focusing on the red (or yellow) coloured cells, the user can easily identify potential root causes of performance problems.

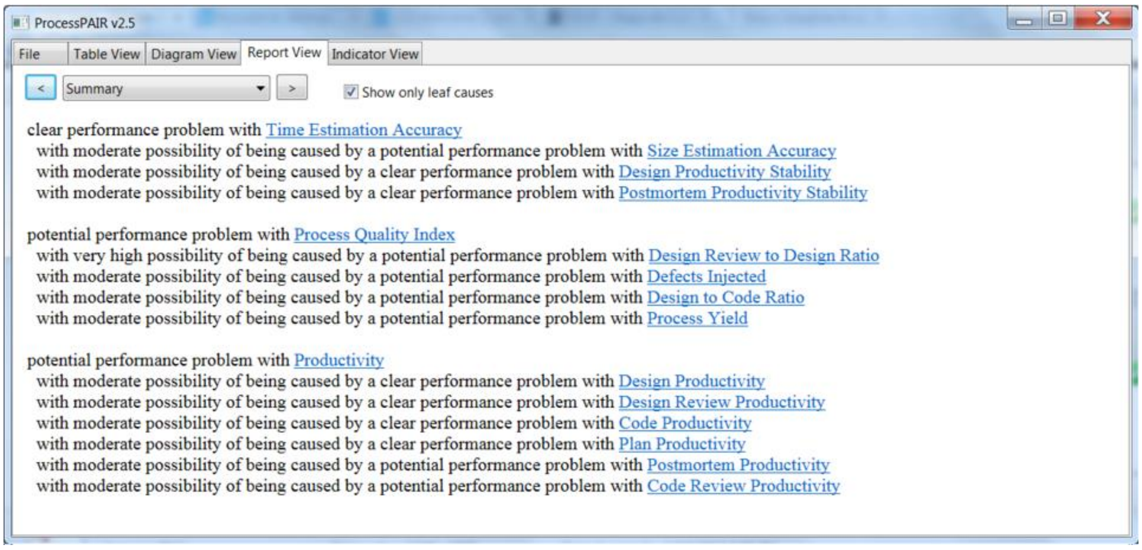


Indicator	Summary	Program 1	Program 2	Program 3	Program 4	Program 5	Program 6
◢ Time Estimation Accuracy	**	1.19	1.96	1.41	1.17	2.54	1.77
Size Estimation Accuracy	***		1.81	1.37	0.67	1.40	0.79
◢ Productivity Estimation Accuracy	**		0.92	0.97	0.57	0.55	0.44
◢ Productivity Stability	**		0.99	0.97	0.57	0.54	0.43
Plan Productivity Stability	*		0.36	0.38	0.30	0.41	0.41
Design Productivity Stability	*		1.77	0.94	0.37	0.32	0.33
Design Review Productivity Stability	*				0.42	0.36	0.43
Code Productivity Stability	***		0.64	1.81	1.11	1.30	0.78
Code Review Productivity Stability	***				0.76	0.80	0.53
Unit Test Productivity Stability	*****		9.44	1.18	0.86	1.17	1.13
Postmortem Productivity Stability	**		0.16	1.17	0.83	0.41	0.20
◢ Process Quality Index	****	0.00	0.00	0.32	0.21	0.23	0.98
◢ Defect Density in Unit Test	***	29	3	9	34	38	0
Defects Injected	***	86	45	77	59	94	67
◢ Process Yield	****	73	92	88	43	60	100
Design Review Productivity	**			377	159	91	87
Code Review Productivity	****			210	159	151	96
Design to Code Ratio	****	0.46	0.17	0.46	0.84	1.46	0.98
Code Review to Code Ratio	*****	0.00	0.00	0.50	0.48	0.60	0.57
Design Review to Design Ratio	****	0.00	0.00	0.61	0.57	0.69	0.65
◢ Productivity	***	37.2	36.8	35.9	20.9	17.6	13.3
Plan Productivity	**	2100	754	367	183	187	169
Design Productivity	**	175	310	232	90	62	56
Design Review Productivity	**			377	159	91	87
Code Productivity	**	81	51	106	76	91	55
Code Review Productivity	****			210	159	151	96
◢ Unit Test Productivity	*****	131	1239	388	298	398	386
◢ Defect Density in Unit Test	***	29	3	9	34	38	0
Defects Injected	***	86	45	77	59	94	67
◢ Process Yield	****	73	92	88	43	60	100
Design Review Productivity	**			377	159	91	87
Code Review Productivity	****			210	159	151	96
Postmortem Productivity	****	2800	456	733	549	265	117

Figure 6 - ProcessPAIR table view with list of performance indicators

Report view

The objective of Report View (Figure 7) is to indicate in a simple way, overall or project by project, the most relevant top-level performance problems (coloured red or yellow in the previous views) and potential root causes. To see all the intermediate causes, the user needs to uncheck the “Show only leaf causes” checkbox; to see information related to a single project, the user needs to select the project on the top left combobox, and to get detailed information about a performance indicator, need to press the link.



clear performance problem with Time Estimation Accuracy
with moderate possibility of being caused by a potential performance problem with Size Estimation Accuracy
with moderate possibility of being caused by a clear performance problem with Design Productivity Stability
with moderate possibility of being caused by a clear performance problem with Postmortem Productivity Stability

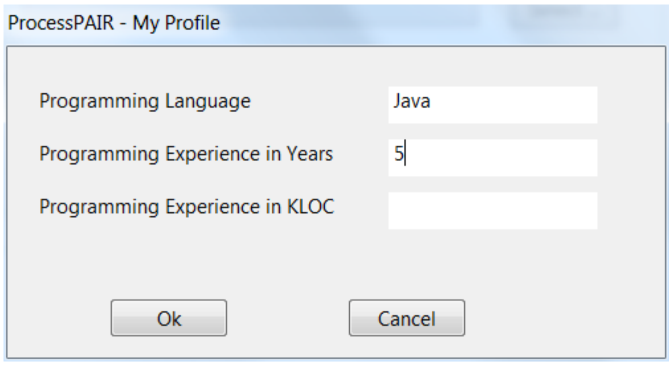
potential performance problem with Process Quality Index
with very high possibility of being caused by a potential performance problem with Design Review to Design Ratio
with moderate possibility of being caused by a potential performance problem with Defects Injected
with moderate possibility of being caused by a potential performance problem with Design to Code Ratio
with moderate possibility of being caused by a potential performance problem with Process Yield

potential performance problem with Productivity
with moderate possibility of being caused by a clear performance problem with Design Productivity
with moderate possibility of being caused by a clear performance problem with Design Review Productivity
with moderate possibility of being caused by a clear performance problem with Code Productivity
with moderate possibility of being caused by a clear performance problem with Plan Productivity
with moderate possibility of being caused by a potential performance problem with Postmortem Productivity
with moderate possibility of being caused by a potential performance problem with Code Review Productivity

Figure 7 - ProcessPAIR report view with a list of performance problems and causes

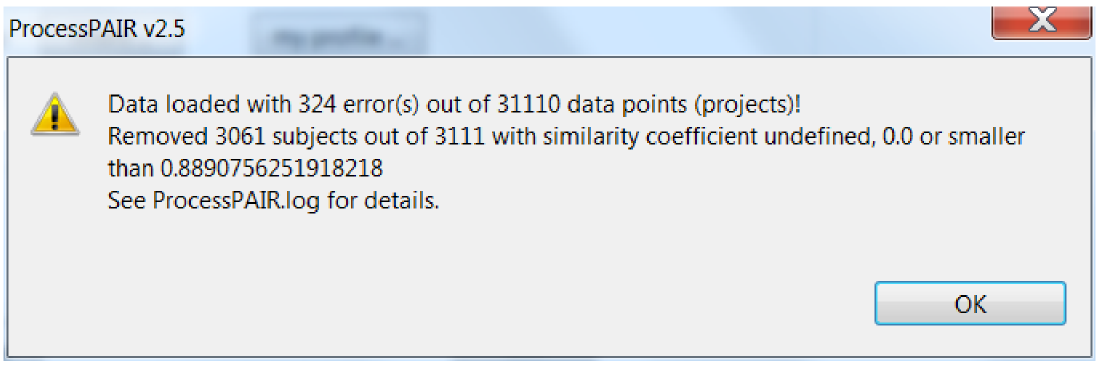
Profile filtering

It is also possible for the user to filter the data points used for calibration based on his profile (Figure 8). This way, only the projects from subjects with most similar profiles to the one indicated will be selected. After the calibration process, the user will get an information message (Figure 9). In this example, only 50 most similar subjects were selected (the minimum number required by the tool for statistical significance), with a similarity coefficient greater than 0.889 (the similarity coefficient ranges between 0 and 1).



Programming Language	Java
Programming Experience in Years	5
Programming Experience in KLOC	

Figure 8 - ProcessPAIR my profile filter



Data loaded with 324 error(s) out of 31110 data points (projects)!
Removed 3061 subjects out of 3111 with similarity coefficient undefined, 0.0 or smaller than 0.8890756251918218
See ProcessPAIR.log for details.

Figure 9 - Result of the profile filtering

2.1.2 PSP PAIR

PSP PAIR [4], an earlier version of the current ProcessPAIR tool, calibrated based only on student data from FEUP, has an initial attempt for implementing the recommendation of improvement actions for the time estimation performance problem, shown in Table 1.

The initial attempt towards a recommendation system was based on hardcoded suggestions from a single process expert.

Our current approach for a recommendation system for the ProcessPAIR tool (a complete model and tool, calibrated based on a large data set from the SEI, having not only time estimation but also quality and productivity) is based on crowdsourcing.

Table 1 – Example of identified problems and recommended improvement actions in PSP PAIR [4]

Indicator	**Problems identified and suggested improvement actions**
Missing parts	The problem with the identification of the parts. Recommended actions: • A more careful Conceptual Design is recommended, with more detail or time spent in design.
Expurious Parts	The problem with the identification of the parts. Recommended actions: • A more careful Design is recommended, with more detail or time spent in design.
Part Estimation Error	The problem in the relative size table and/or problem with the identification of the parts. Recommended actions: • Review the types of the parts, their relative size and related topics.
Productivity Stability – Plan and Postmortem (same for other phases)	The problem with changes in the way of working. Recommended actions: • Try to keep a stable productivity and check what has been modified lately. • If the productivity has increased try to keep it stable at that new value in future projects.
Process Stability	Recommended actions: • Try to keep the process stable, changing the process usually leads to a variation in productivity.
Total Defect Density	Recommended actions: • Improve the process quality. • Do an analysis of the more frequent and more expensive errors in order to not repeat them, define preventive actions.
Review Rate	Recommended actions: • If the value is too high, it is necessary to reduce it by doing the review more carefully and slowly. • If it is too slow (low value), do the review in a more focused and efficient way. • Check if the artefacts are understandable.
Review to Development Ratio	Recommended actions: • If the value is too high, it is necessary to do the review more carefully and slowly or code faster and more efficiently. • If the value is too low, do the review in a more focused and efficient way or take your time coding. • Check if the artefacts are understandable.

2.2 Crowdsourcing

In this section, an overview of the concept of crowdsourcing and its different types applied by different authors over the years is discussed in sub-sections 2.2.1 and 2.2.2, how to motivate and incentive people in a crowdsourcing platform is discussed in sub-section 2.2.3, the benefits and challenges are discussed in sub-section 2.2.4, some examples of services that use crowdsourcing are discussed in sub-section 2.2.5, and how crowdsourcing approach will work for developing a recommendation system for ProcessPAIR is discussed in sub-section 2.2.6.

2.2.1 Definition of crowdsourcing

The word “crowdsourcing” comes from a portmanteau of "crowd" and "outsourcing”; by joining ideas from a group of people (crowd) and the practice of outsourcing tasks, we can introduce new and more developed skill sets or a larger workforce to achieve a specific goal. [5]

Crowdsourcing can take place on many different levels, due to the growth of digital innovation, it is now easier than ever for people to contribute, whether with ideas, improvements, learnings, funds or anything useful, to a project or a cause. Crowdsourcing works for a common goal, to share knowledge between communities.

The first introduction to Crowdsourcing was in 2006 by the journalist Jeff Howe from the Wired magazine, where he published an article that sets the definition of crowdsourcing as: “*Simply defined, crowdsourcing represents the act of a company or institution taking a function once performed by employees and outsourcing it to an undefined (and generally large) network of people in the form of an open call”.* [5]

2.2.2 Types of crowdsourcing

Since the introduction of the concept of crowdsourcing by Howe, it has evolved over the years, through several different practices. In this topic will be presented several types of crowdsourcing approached by different authors over the years.

2.2.2.1 Jeff Howe typology

In 2008 [6], Howe distinguished four types of crowdsourcing based on how various applications function:

- **Crowd Creation**

Crowd creation happens when the company turns to its users to actually create, or co-create, a product or service. For example, activities such as asking the crowd to film TV commercials, perform language translation or solve challenging scientific problems. This strategy is combined with crowd voting to create marketing campaigns.

- **Crowd Voting**

Crowd voting is the most popular form of crowdsourcing, which generates the highest levels of participation. Crowd voting leverages the community's judgment to organize, filter and stack-rank content such as brainstorming ideas, newspaper articles, music and movies. It allows the crowd to express their opinion through voting or rating.

- **Crowd Wisdom**

Howe defined the "Wisdom of Crowds" principle as the attempt to gather many people's knowledge in order to solve problems, predict future outcomes or help direct corporate strategy. Howe understands that a larger group of diverse people can make better decisions, and display more intelligence than any smaller collection of experts.

- **Crowd Funding**

Crowdfunding differs from other types of crowdsourcing since it does not depend on the skills, knowledge, creativity or feelings of the crowd but on the funding of them, establishing a connection between people that have money with those that need it. This way it allows people to fund projects in which they have an interest. Crowd funding is a great example of a process innovation.

Based on the typology approached by Jeff Howe in 2008, were created other types that synthesize crowdsourcing initiatives. Below are described the typologies of Daren Brabham [7], David Geiger [8] and Estellés-Arolas[9].

2.2.2.2 Daren Brabham typology

In 2011 Brabham [7] defined crowdsourcing as: "*Crowdsourcing is an online, distributed problem-solving and production model that leverages the collective intelligence of online communities to serve specific organizational goals*."

According to Daren Brabham, the problems that crowdsourcing is best suited to solve are:

- **Knowledge Discovery and Management**

An organization tasks a crowd with finding and collecting information into a common format. Ideal for information gathering, organization and reporting problems.

- **Broadcast Search**

Organizations tasks a crowd with solving empirical problems. Ideal for ideation problems with empirically provable solutions such as scientific challenges.

- **Peer-Vetted Creative Production**

Organizations tasks a crowd with creating and selecting creative ideas.

- **Distributed Human Intelligence Tasking**

Large data problems are decomposed into small tasks requiring human intelligence, and individuals in the crowd are compensated for processing bits of data. Monetary compensation is a common motivator for participation.

2.2.2.3 David Geiger typology

In 2011, David Geiger [8] with the combination of two different crowdsourcing systems, one that treat the external elements as homogenous, and the other that explicitly seeks the participation of heterogeneous elements, created four different types of crowdsourcing (Figure 10). Every type represents a prototypical system that provides a distinct service to a crowdsourcing organisation.

- **Crowd rating**

Crowd rating makes use of large quantities of homogeneous contributions, seeking the value of an emerging property. These contributions are made in a collective way since they are aggregated to a collective response to the given problem. In crowd rating, every contribution represents a specific vote, so there is no prior or wrong result. These systems are often used to collect reviews, like for example, TripAdviser or eBay reputation system.

- **Crowd creation**

Crowd creation makes use of a variety of heterogeneous external contributions. In crowd creation, the contributions need to be put in relation to each other since the objective of these systems is to produce satisfying outcome for the tasks, where the variety of contributions increases with size and diversity of the crowd. A known service of a user-generated content system that uses crowd creation is Wikipedia.

- **Crowd processing**

Crowd processing, like crowd rating, makes use of large quantities of homogeneous external contributions, without aiming for an emerging property. These contributions are independent of each other and can be evaluated individually. The objective of crowd processing is to combine individual contributions to deliver the best solution for the given problem.

- **Crowd solving**

Crowd solving refers to a qualitative approach using heterogeneous stimuli that represent the solution to a specific problem. The external elements in crowd solving systems are evaluated individually on the basis of objective and well-defined criteria. The objective in crowd solving is to get as close as possible to the best solution of a problem, where the process can terminate when the best solution is found and every additional contribution potentially increases the quality of the results.

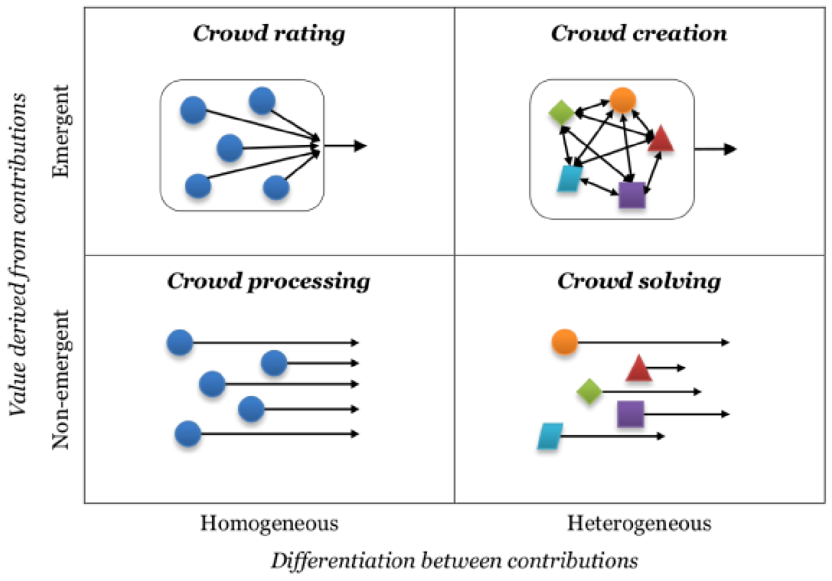


Figure 10 - Four types of crowdsourcing by David Geiger [8]

2.2.2.4 Estellés-Arolas typology

In 2012, Estellés [9] analysed the different typologies made by different authors (Howe, Brabham, Kleeman et al., Greets, Reichwald & Piller and Burger-Helmchen & Penin) to develop a new integrated typology.

- **Crowdcasting**

In this type of crowdsourcing, a problem or a task is presented to the crowd and who settle first or provide the best solution is rewarded.

- **Crowdcollaboration**

In this type the communication occurs between individuals of the crowd, while the company which initiates the process stays out. In these tasks, individuals contribute through their knowledge to solve problems or raise ideas collaboratively and usually there is no financial reward. This type of crowdsourcing is divided into two subtypes that differ in the final goal:

  - **Crowdstorming**: Consists in brainstorming online sessions, in which the crowd can propose ideas, or comment or vote on other ideas.
  - **Crowdsupport**: In this case, the customers or users themselves are the ones who solve the questions or problems of others, so that they don't need to contact the official support of the company.

- **Crowdcontent**

In these tasks, the crowd use their labour and knowledge to create or find the content of various kinds. In Crowdcontent, each individual works individually and at the end, the outcomes of all the participants comes together. This type of crowdsourcing is divided into three subtypes:

  - **Crowdproduction**. The crowd is responsible for creating content. For example, the translation of short extracts of text or identification of images.
  - **Crowdsearching**. The crowd is responsible for looking for content on the internet according to a particular task.
  - **Crowdanalyzing**. This case is similar to crowdsearching, with the difference that the search is not done with Internet text documents, but with multimedia documents like images or videos

- **Crowdfunding**

In crowdfunding, an individual, organization or company seeks to fund from the crowd in exchange for a reward. It allows the financing of projects through small contributions made by a broad group of people.

- **Crowdopinion**

This type of crowdsourcing aims to meet users' opinion about a particular problem or question, or on products through, for example, comments or votes. In other words, the crowd gives his opinion or judgment to make assessments.

2.2.2.5 Overview of crowdsourcing typologies

To conclude this subsection, an overview of crowdsourcing typologies approached by different authors and grouped by the most common practices in the use of crowdsourcing, such as, problem-solving, creative input, opinion, outsourcing tasks and search information is shown in Table 2.

Table 2 – Overview of crowdsourcing typologies

Author / **Aim**	**Jeff Howe**	**Daren Brabham**	**David Geiger**	**Estellés & Guvara**
Problem solving	Crowd wisdom	Broadcast Search	Crowd solving	Crowdcollaboration Crowdcasting
Creative input	Crowd creation	Peer-Vetted Creative Production	Crowd creation	Crowdcontent
Opinion	Crowd voting		Crowd rating	Crowdopinion
Outsourcing tasks		Distributed Human Intelligence Tasking	Crowd processing	
Raising money	Crowd funding			Crowdfunding
Search information		Knowledge Discovery and Management		Crowdsearching (crowdcontent)

2.2.3 Motivations and incentives

2.2.3.1 Self-determination theory

Crowdsourcing has many benefits and advantages, however for crowdsourcing to happen and be successful, there needs to be a group of people involved and with willingness to contribute and participate, as they are the basis of the success of crowdsourcing. For this, it is essential to know what motivates and encourages people who constitute the crowd.

Ryan & Deci [10] state that "to be motivated means to be moved to do something". Self-Determination Theory (SDT), distinguishes different types of motivation (Figure 11). Motivation can be defined as the force or set of interior forces that drive the behaviour of the subject to achieve a particular goal. Represents the needs, interests and expectations that make us feel motivated for a particular action. We can then distinguish three types of motivation: intrinsic, extrinsic and amotivation.

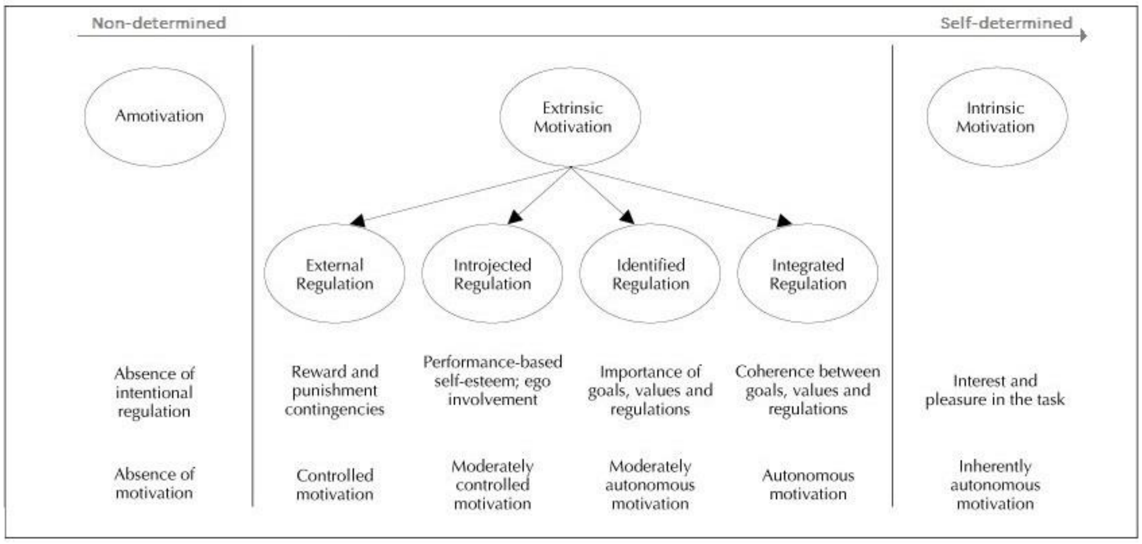

Non-determined
Self-determined
Amotivation
Extrinsic Motivation
Intrinsic Motivation
External Regulation
Introjected Regulation
Identified Regulation
Integrated Regulation
Absence of intentional regulation
Reward and punishment contingencies
Performance-based self-esteem; ego involvement
Importance of goals, values and regulations
Coherence between goals, values and regulations
Interest and pleasure in the task
Absence of motivation
Controlled motivation
Moderately controlled motivation
Moderately autonomous motivation
Autonomous motivation
Inherently autonomous motivation

Figure 11 - Self-determination continuum [10]

The self-determination continuum indicates also the levels of internalisation increasing from left to right, which is the process of adopting a value or regulation, and the degree to which this has been integrated to be a part of the self.

Amotivation is the least self-determined and refers to the lack of intention to perform a task, as a result of not valuing an activity, not feeling competent to do it, or not believing it would amount to anything.

Extrinsic motivation refers to behaviour that is driven by external sources. These rewards provide satisfaction and pleasure that the task itself may not provide.

Ryan and Deci developed the Organismic Integration Theory as a sub-theory of SDT, to explain the different ways that extrinsically motivated behaviour is regulated.

Organismic Integration Theory describes that extrinsic motivation should be divided into four types of behaviour regulation that often differ in terms of their relative autonomy: external, introjected, identified and integrated.

External regulation is the least autonomous, it is performed because of external demand or possible reward.

Introjected regulation introjection is associated with ego involvement, in which a person performs an act in order to enhance or maintain self-esteem and the feeling of worth.

Identified regulation is a more autonomously driven form of extrinsic motivation. It is related to the personal importance of a behaviour.

Integrated regulation is the most autonomous extrinsic motivation. Integration occurs when regulations are fully assimilated with self so they are included in a person's self-evaluations and beliefs on personal needs. It is the most related to intrinsic motivation but still classified as extrinsic because the motivation in performing the task is extrinsic to the self, rather than the inherent enjoyment or interest in the task.

Hossain [11] states that in crowdsourcing platforms, the significant extrinsic motivational factors are reputation, status, peer pressure, fame, community identification and fun.

Intrinsic motivation is the most self-determined and refers to behaviour that is driven by internal rewards, which comes from the pleasure that someone gets by the task itself, the resulting satisfaction in completing or even working on a task.

In the crowdsourcing context, Hossain understands that intrinsic motivation mainly depends on members and their motivation to participate. The motivation is considered as a process to control, release and maintain physical and mental activities

Being that, it is very important to know what motivate people to participate, since the quality and the quantity of contributions can be determined by knowing insight about incentives required to increase motivation.

In Table 3 is shown the variation of different factors that motivate people in the communities of crowdsourcing [12], according to the types of motivation in the Self-determination theory.

In order to increase motivation, it is important to incentive the users of the crowdsourcing platform. While the motivational factors correspond to factors that induce a certain behaviour, incentives correspond to rewards, which are offered to the participants of these platforms, in order to stimulate their participation in carrying out the tasks [13].

Table 3 - Motivations in the crowdsourcing community [12]

	Motivation type	**Motivations in crowdsourcing community**
Extrinsic motivation	**External regulation**	To make money by contributing creative ideas or solutions. To share profit by sell creative ideas to firms. To improve job prospect.
	Introjected regulation	To burnish reputation in specific field. To demonstrate own ability. To alleviate peer pressure.
	Identified regulation	Identify the value of sponsored firms. Identify the value of crowdsourcing community. Desire to increase the welfare of other people.
	Integrated regulation	Believe that he belongs in the crowdsourcing community. Past success makes participants sense of belonging to the sponsored company. The challenge of solving difficult problems.
	Intrinsic motivation	Feel happy to express creative ideas freely. Be interested in the firms' new product development process. Addiction to the tasks proposed and love to the community.

Leimeister [14] in his research summarizes this interplay between motives and incentives that imply requirements for incentive-supporting components. In Table 4 are presented some examples of incentives that can be provided according to the motive associated. The first column contains the motives as mentioned before. The second column contains the incentives assigned to the motives in the first column. Each motive can be activated by one or more of these incentives.

Table 4 - Motives and incentives in crowdsourcing platforms [14]

Motives	**Incentives**
Learning	Access to the knowledge of experts Access to the knowledge of mentors Access to the knowledge of peers
Direct compensation	Prizes Career options
Self-marketing	Profiling options
Social motives	Appreciation by the organizer Appreciation by peers

2.2.4 Benefits and challenges of crowdsourcing

Analysing the concept of crowdsourcing and its existing literature, it can be concluded that the advantages and disadvantages of their use will be very dependent on the type of crowdsourcing used since this concept is heterogeneous. Thus, the benefits and challenges presented in this section are in compliance with the WebProcessPAIR platform.

2.2.4.1 Benefits

Quality

Quality refers to the characteristics and originality of a solution proposed by contributors and the way it matches the asker tastes and expectations. Addressing a mass of skilled individuals through an open call is a relatively proven approach for problem solving.

Reduce costs

One of the major benefits of crowdsourcing is its relatively low cost, although the amounts involved differ according to the type of crowdsourcing [15].

Since WebProcessPAIR participants are contributors (instructors, coaches, teachers, etc.) and users (students, developers, etc.), the remunerations can be relatively low, however, voluntary work is not a rule in Crowdsourcing.

Network externalities

Positive network externalities occur when the value of a system increases dependent on the number of other people using it. Crowdsourcing is a way to foster network externalities and the adoption of new technologies [15].

Access to a diverse set of people

Through crowdsourcing, companies have access to a large number of people who have a diversity of skills and abilities, and that can contribute to the multiplicity and innovation of solutions [6].

2.2.4.2 Challenges

Quality control

By creating an online environment in which people can engage with each other directly, trust and safety, as well as quality control, can be highly challenging and given that this is a base expectation for most users, needs to be managed well from the outset.

How to ensure the quality of the contributions?

Information-activism [16] understands that there is no right solution for this problem, however, there are ways in which it's possible to alleviate the possibility of false or unusable information. The most reliable contributions are those which are confirmed by multiple sources and collected through different types of media, or validated by a trusted source on the ground.

Another way is to identify different tiers of contributors, and attribute them a level of trust. Trusted community members with good reputation and which gave credible information over the time would rate higher on this scale, than other unknown members.

The best way to ensure verification of the contributions, is to designate one or more trusted members of the community as moderators. For example, in cases where the contribution can defame other members or simply be irrelevant, the moderator can report this contribution as well as report the user who made it. A mutual incentive system ensures more reliable reports.

Lack of contributors

While crowdsourcing is likely to benefit from network externalities, it's also possible that the platform fails to attract sufficient contributors, and since crowdsourcing is about having a crowd, that could be a serious problem. This concept relies on voluntary participation of people so reaching a good number of contributors is not guaranteed. One possible solution is to rely on financial rewards in order to increase incentives for individual's participation, however it will reduce the cost advantage of crowdsourcing [10].

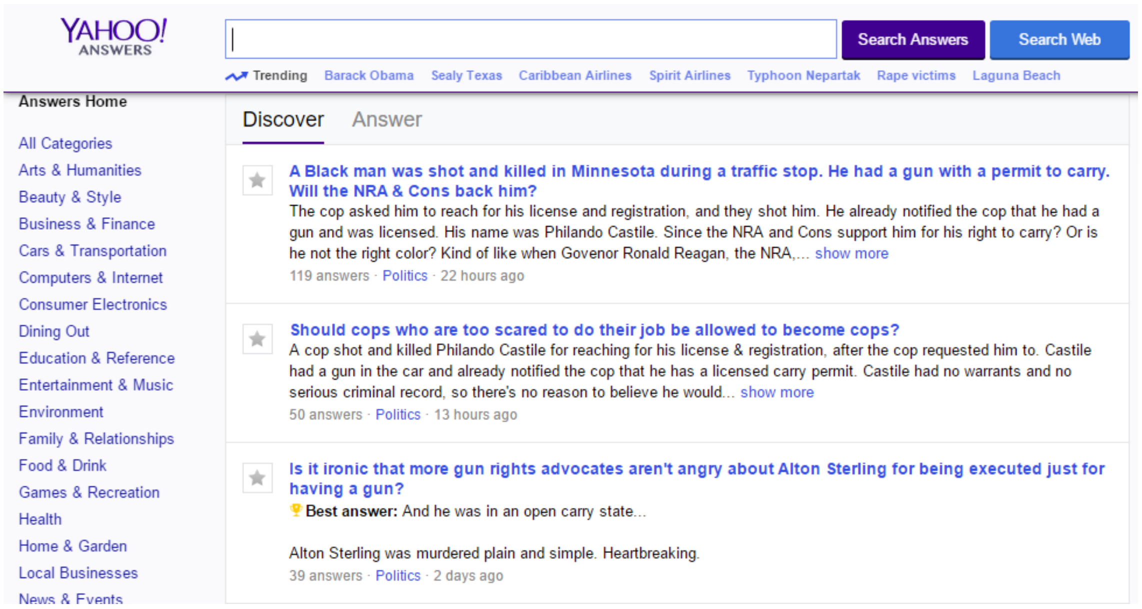
2.2.5 Examples of crowdsourcing services

In this section will be presented two examples of solution finding platforms that use the concept of crowdsourcing for the same purpose of WebProcessPAIR, to offer solutions to proposed problems, through the use of knowledge of people willing to help.

Yahoo answers

Yahoo Answers is an online question-and-answer community launched by Yahoo!, where members can share their questions and seek answers from the community, like is shown in Figure 12. Yahoo Answers differentiate the questions into several categories such as Art & Humanities, Beauty & Style, Business & Finance and Consumer Electronics.

Questions are initially open to answers for four days. However, the asker can choose to pick a best answer for the question after a minimum of one hour. However, comments and answers can still be posted after this time. To ask a question, the person must have a Yahoo! account with a positive score balance of five points or more. Users also receive ten points for contributing the "Best Answer" which is selected by the question's asker. [17].



A Black man was shot and killed in Minnesota during a traffic stop. He had a gun with a permit to carry. Will the NRA & Cons back him?
The cop asked him to reach for his license and registration, and they shot him. He already notified the cop that he had a gun and was licensed. His name was Philando Castile. Since the NRA and Cons support him for his right to carry? Or is he not the right color? Kind of like when Govenor Ronald Reagan, the NRA,... show more
119 answers · Politics · 22 hours ago

Should cops who are too scared to do their job be allowed to become cops?
A cop shot and killed Philando Castile for reaching for his license & registration, after the cop requested him to. Castile had a gun in the car and already notified the cop that he has a licensed carry permit. Castile had no warrants and no serious criminal record, so there's no reason to believe he would... show more
50 answers · Politics · 13 hours ago

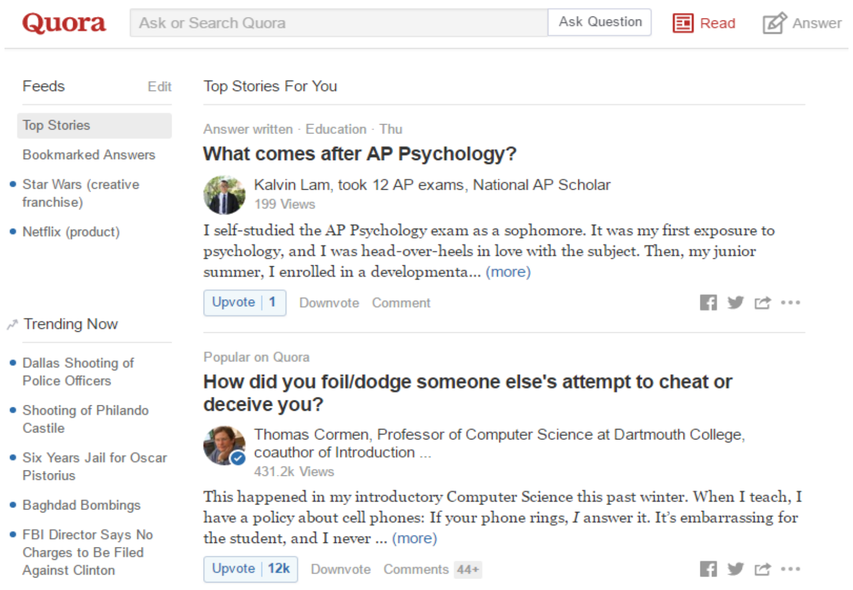
Is it ironic that more gun rights advocates aren't angry about Alton Sterling for being executed just for having a gun?
Best answer: And he was in an open carry state...

Alton Sterling was murdered plain and simple. Heartbreaking.
39 answers · Politics · 2 days ago

Figure 12 – Homepage of Yahoo answers

Quora

Quora is another solution finding platform where people seek help for any questions from people who have proper background or have the first-hand experience, like is shown in Figure 13. Besides this, people can also post short blogs on Quora to share knowledge or experience on any topic. Unlike InnoCentive or Yahoo Answers, Quora requires its users to register with their real names and backgrounds in order to ensure the qualities and trustworthiness of answers and shares [17].


Quora
Ask or Search Quora
Ask Question
Read
Answer
Feeds
Edit
Top Stories
Bookmarked Answers
Star Wars (creative franchise)
Netflix (product)
Trending Now
Dallas Shooting of Police Officers
Shooting of Philando Castile
Six Years Jail for Oscar Pistorius
Baghdad Bombings
FBI Director Says No Charges to Be Filed Against Clinton
Top Stories For You

Answer written · Education · Thu
What comes after AP Psychology?
Kalvin Lam, took 12 AP exams, National AP Scholar
199 Views
I self-studied the AP Psychology exam as a sophomore. It was my first exposure to psychology, and I was head-over-heels in love with the subject. Then, my junior summer, I enrolled in a developmenta... (more)
Upvote | 1 Downvote Comment

Popular on Quora
How did you foil/dodge someone else's attempt to cheat or deceive you?
Thomas Cormen, Professor of Computer Science at Dartmouth College, coauthor of Introduction ...
431.2k Views
This happened in my introductory Computer Science this past winter. When I teach, I have a policy about cell phones: If your phone rings, *I* answer it. It's embarrassing for the student, and I never ... (more)
Upvote | 12k Downvote Comments 44+

Figure 13 - Homepage of Quora

2.2.6 Type of crowdsourcing in WebProcessPAIR

As mentioned previously the WebProcessPAIR platform uses the concept of crowdsourcing to appeal to the knowledge of a community of users and experts. The community participates indirectly in the construction of a catalogue of possible improvement actions for the performance problems described in the application.

For each performance problem, the contributors are able to suggest improvement actions in order to address the problem.

From the analysis of various types of crowdsourcing, taking into account the purpose of using it in WebProcessPAIR, the most appropriate typology from each author is given in Table 5.

Table 5 - Crowdsourcing typologies used in WebProcessPAIR

WebProcessPAIR	
Author	**Typologies**
Howe	Crowd wisdom
Bradham	Broadcast search
Geiger	Crowd solving
Estélles-Arolas	Crowdcollaboration

Regarding the typologies of Howe, the most appropriated is crowd wisdom, because the goal is to use the crowd community to contribute with solutions to problems proposed by other users.

As regards the types addressed by Bradham, the closest would be the Broadcast search because it is related to "problem-solving" through ideas provided by the crowd.

The typology “crowd solving” from Geiger is quite in accordance with the aim pursued in WebProcessPAIR, since the objective is to get as close as possible to the best solution of a problem and every additional contribution potentially increases the quality of the results

At last, from Estélles-Arolas typologies, WebProcessPAIR use the crowdcollaboration and its subtype crowdstorming; like in the other typologies chosen, the goal is to let users contribute with their knowledge to solve problems, through brainstorming online sessions, where the crowd can propose ideas, or comment or vote on other ideas.

2.3 Recommendation systems

In this section it will be presented the analysis of recommendation systems and their different approaches. In context with the previous section, will also be analysed recommendation systems for problem-solving based on crowdsourcing and explained what kind of recommendation system is best suited to be used in WebProcessPAIR. To better illustrate the use of these systems, will be presented two examples of websites that use problem-solving recommendation systems, StackOverflow and WebMD-symptom checker, having been made a detailed analysis of the functioning of StackOverflow, since it has many characteristics similar to WebProcessPAIR.

Recommendation systems (or recommender systems) are a subclass of information filtering systems which aim to predict the "rating" or "preference" that a user would give to an item. Over the past years, vast advancements in recommendation systems have been done, becoming this an extremely common feature on the internet since it can be applied in a variety of applications.

Many websites use recommendation techniques to increase users' experience or to help improving the information overload problem. The techniques are based on similar ones found in information retrieval and, until recently, recommendation systems were commonly classified together with information retrieval systems.

Recommendation systems are normally used in two ways, for e-commerce and for problem solving. In e-commerce, recommendation systems typically produce a list of recommendations, through collaborative filtering, content-based filtering, or a combination of both, hybrid recommendation systems. In problem solving, the recommendations are made based on expert knowledge or are based on crowdsourcing.

2.3.1 Recommendation systems for e-commerce:

Collaborative filtering recommendation systems

Collaborative filtering recommendation attempts to identify similarities between entities (users or items) based on the interaction history of these entities. Being that, it recommends data that were previously ranked by a number of users who presumably have similar interests as the user who requested the recommendation. The great advantage of the collaborative filtering approach is that it is capable of making recommendations without requiring previous knowledge about the content of the data recommended, for example movies or music.

For measuring user similarity or item similarity, many algorithms have been used, for example, the k-nearest neighbourhood (k-NN) approach and the Pearson Correlation [18].

One of the most famous examples of collaborative filtering is item-to-item collaborative filtering (people who buy x also buy y), an algorithm used by Amazon.com's recommender system. Other examples are Facebook, MySpace, LinkedIn, and other social networks that use collaborative filtering to recommend new friends, groups, and other social connections (by examining the network of connections between a user and his friends).

Content-based filtering recommendation systems

In content-based filtering the recommendations are based on the content of the items, expressing the content of each data in a form that can be objectively evaluated, and filtering out data whose content doesn't match the user's preferences. In a content-based recommender system, keywords are used to describe the items and a user profile is built to indicate the type of item the user likes [18]. To build the user profile the system mostly focuses on two types of information, a model of the user's preference, and a history of the user's interaction with the recommendation system. Direct feedback from a user, usually in the form of a like or dislike button, can be used to assign higher or lower weights on the importance of certain attributes.

A common example where it is used content-based filtering is the recommendation of movies, used for example in known websites like IMDb (Internet Movie Database) or Rotten Tomatoes.

Hybrid recommendation systems

Recent researches have demonstrated that a hybrid approach, combining collaborative filtering and content-based filtering could be more effective in some cases. Hybrid approaches can be implemented in several ways. Several studies empirically compare the performance of the hybrid with the pure collaborative and content-based methods and demonstrate that the hybrid methods can gain better system optimization and fewer weaknesses than pure approaches [18].

A good example of a company that uses a hybrid recommendation system is Netflix. They make recommendations by comparing the watching and searching habits of similar users (collaborative filtering) as well as they suggest movies that share characteristics with films that a user has rated highly (content-based filtering).

2.3.2 Recommendation systems for problem solving:

Knowledge-based recommendation system

Knowledge-based recommendation systems provide the user with advice about a decision to make or an action to take. These systems rely on knowledge provided by human experts, encoded in the system and applied to input data, in order to generate recommendations.

Most knowledge-based systems use a case-based approach as their recommendation technique. Normally in these systems, the user enters his problem description. The cases in the case base are then ranked accordingly, and the most appropriate case is retrieved as a solution to the user problem.

In the case-based recommendation, the knowledge acquisition approach amounts to store the known cases into a knowledge base (or case base). A step of knowledge evolution is usually present. The human experts provide explanations on the application of a particular case as the solution to the user problem. These explanations constitute knowledge added to the case base. In parallel, some sort of user profile needs to be created or the user behaviour needs to be modelled. Information about the user has to be collected: his/her preferences, interests, or any other elements characterizing his situation requiring advice [19].

Recommendation systems based on crowdsourcing

Like knowledge-based recommendation system, recommendation systems based on crowdsourcing provide the user with advice about the decision to make or an action to take. However, this refers to the knowledge of a group of people who are willing to contribute with solutions to the problems.

In 2014, Geiger [20] explains that almost all of the contributor profiles that are built in existing task recommendation approaches based on crowdsourcing, are limited to information that is collected internally. By perceiving a contributor not only as an isolated component of one particular system but as a connected individual that is part of many systems, a recommendation can incorporate a rich variety of digital information already present on the Web, thus mitigating sparse data and creating more exact profiles. External information may come, for example, from other crowdsourcing systems, knowledge communities, social networks, blogs, expert directories, and research publications.

Most recommendation systems research provides an extensive body of knowledge on all stages of recommendation development. Based on the concept of crowdsourcing, the design of personalized task recommendation systems could benefit from a more substantial application of foundational knowledge. By adapting this knowledge to a crowdsourcing context, researchers can draw on insights regarding, for instance, the selection and implementation of suitable techniques. Recommendation system approaches based on crowdsourcing have at least two exclusive characteristics that differentiate them from most traditional recommendation scenarios. The first is that they implicitly or explicitly model their users' capabilities in addition to their interests. And the second is that they have the ability to draw on their users' contributions as an additional source of knowledge.

2.3.3 Examples of recommendation systems for problem solving

In this section will be presented two examples of systems that have characteristics that go in accordance to the objectives of WebProcessPAIR: Stack Overflow and WebMD-Symptom Checker.

Stack Overflow

Stack Overflow [21] is a good example to be analysed in detail, since its structure and use of the concept of crowdsourcing in a problem-solving recommendation system is very similar to WebProcessPAIR. In order to better understand how it works, let's look at the following analysis.

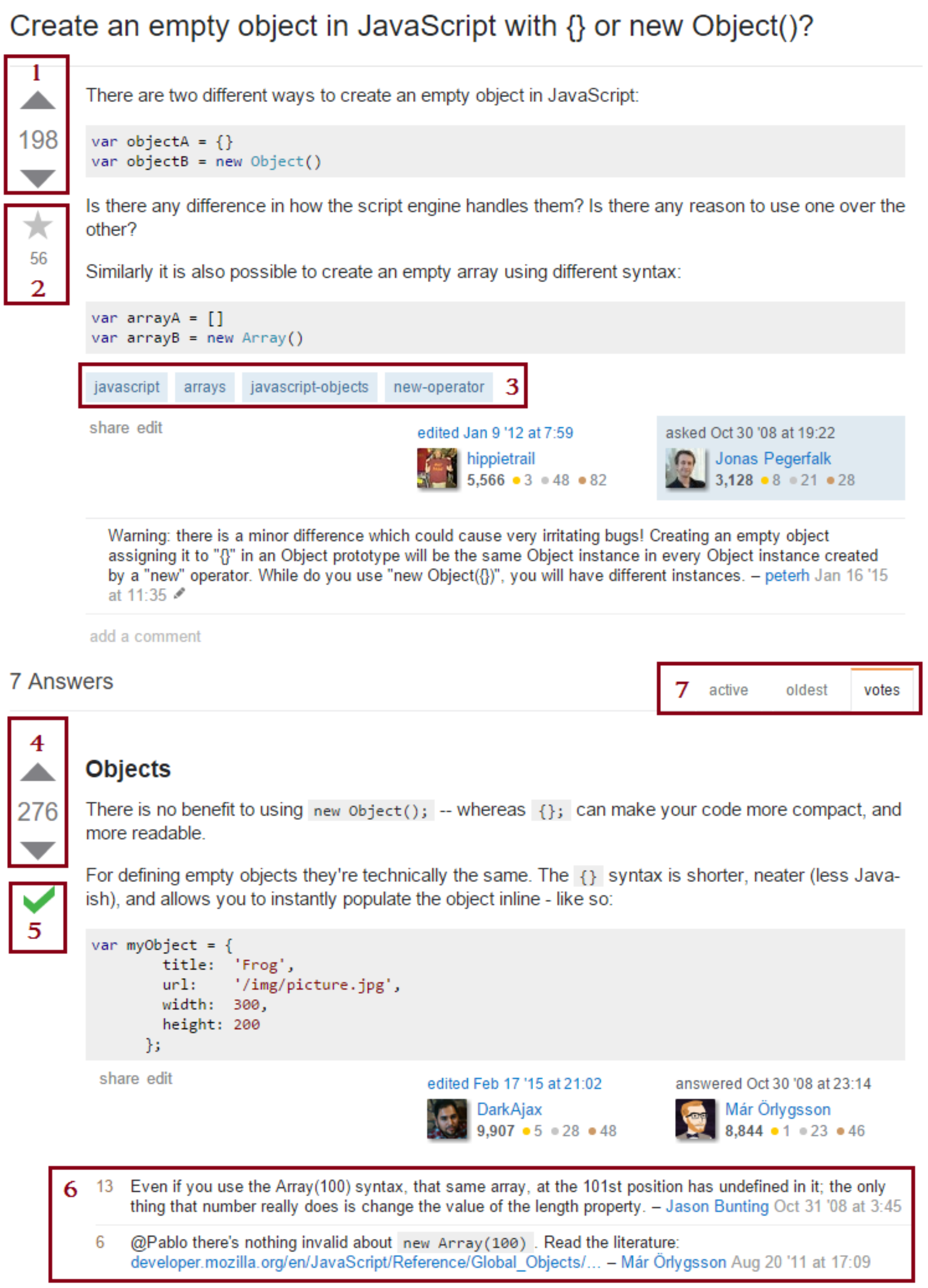
Stack Overflow is a privately held Q&A (questions & answers) website, the flagship site of the Stack Exchange Network, created in 2008 by Jeff Atwood and Joel Spolsky.

It features questions and answers on a wide range of topics in computer programming that are tightly focused on a specific problem. Questions that are of a broader nature or invite answers that are inherently a matter of opinion are usually closed by a process carried out by the site's participants.

The four most common forms of participation are question asking, question answering, commenting, and voting/scoring. Experts are motivated to answer questions because they enjoy helping, and because good answers increase their prominently advertised reputation score.

Each question, answer, and comment someone makes can be voted up or down by anyone with a certain minimum reputation score.

To better understand how this system works, it is shown in Figure 14 an example of a question and their answers.



Create an empty object in JavaScript with {} or new Object()?

There are two different ways to create an empty object in JavaScript:

```
var objectA = {}
var objectB = new Object()
```

Is there any difference in how the script engine handles them? Is there any reason to use one over the other?

Similarly it is also possible to create an empty array using different syntax:

```
var arrayA = []
var arrayB = new Array()
```

javascript arrays javascript-objects new-operator

share edit edited Jan 9 '12 at 7:59 hippietrail 5,566 ●3 ●48 ●82 asked Oct 30 '08 at 19:22 Jonas Pegerfalk 3,128 ●8 ●21 ●28

Warning: there is a minor difference which could cause very irritating bugs! Creating an empty object assigning it to "{}" in an Object prototype will be the same Object instance in every Object instance created by a "new" operator. While do you use "new Object({})", you will have different instances. – peterh Jan 16 '15 at 11:35

add a comment

7 Answers

Objects

There is no benefit to using `new Object();` -- whereas `{};` can make your code more compact, and more readable.

For defining empty objects they're technically the same. The `{}` syntax is shorter, neater (less Java-ish), and allows you to instantly populate the object inline - like so:

```
var myObject = {
        title:  'Frog',
        url:    '/img/picture.jpg',
        width:  300,
        height: 200
      };
```

share edit edited Feb 17 '15 at 21:02 DarkAjax 9,907 ●5 ●28 ●48 answered Oct 30 '08 at 23:14 Már Örlygsson 8,844 ●1 ●23 ●46

13 Even if you use the Array(100) syntax, that same array, at the 101st position has undefined in it; the only thing that number really does is change the value of the length property. – Jason Bunting Oct 31 '08 at 3:45

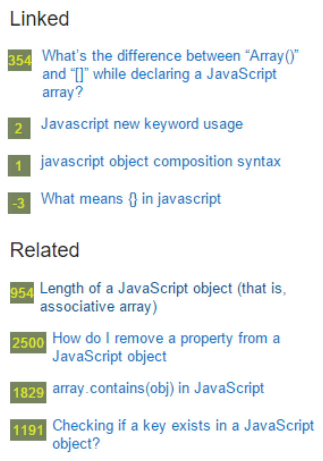
6 @Pablo there's nothing invalid about `new Array(100)`. Read the literature: developer.mozilla.org/en/JavaScript/Reference/Global_Objects/... – Már Örlygsson Aug 20 '11 at 17:09

Figure 14 - Structure of question and answer in StackOverflow

In (1) users can give their feedback on the question, through a system of upvote/downvote, in case they find it interesting or otherwise. In (2), it is possible to mark the question as "favourite". The user can check all his favourite questions in his profile section. In (3) are the tags chosen by the user that made the question.

With respect to answers, in (4), users can leave their vote to an answer via the up/down vote system. In (5) the user who asked the question, pointed that answer as the best one (only the user who asked the question can indicate the best answer). In (6) are the comments made to an answer. To each questions and answers it is possible to comment and also to vote (up/down) on other comments. Finally, in (7) it is possible to sort all the answers by the best rating (votes), or by the data that was submitted (oldest) or that have been discussed recently (active).

For each question (Figure 15) it is possible to view related questions (related) with similar tags for example, or questions that were provided through answers or comments (linked).


Linked

354 What's the difference between "Array()" and "[]" while declaring a JavaScript array?

2 Javascript new keyword usage

1 javascript object composition syntax

-3 What means {} in javascript

Related

954 Length of a JavaScript object (that is, associative array)

2500 How do I remove a property from a JavaScript object

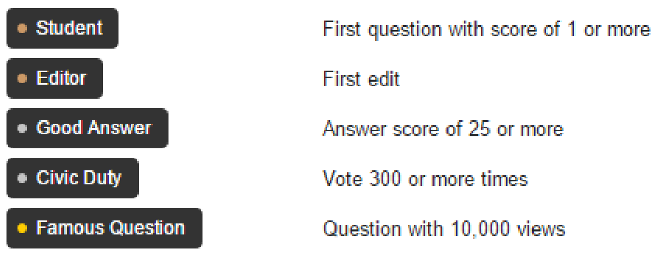
1829 array.contains(obj) in JavaScript

1191 Checking if a key exists in a JavaScript object?

Figure 15 - Example of linked and related questions

Users have an overall reputation score. Answers earn their author 10 points per up-vote, questions earn 5, and comments earn 2. As users gain reputation, they earn administrative privileges, and more importantly, respect in the community. Administrative privileges include the ability to edit, tag, or even delete other people's responses. These and other administrative contributions also earn reputation, but most reputation is earned through questions and answers.

Users also earn badges (Figure 16), which focuses attention on the different types of contributions.



● Student	First question with score of 1 or more
● Editor	First edit
● Good Answer	Answer score of 25 or more
● Civic Duty	Vote 300 or more times
● Famous Question	Question with 10,000 views

Figure 16 - Example of badges in StackOverflow

Crowdsourcing is based on the idea that knowledge is diffuse, but web technology makes it much easier to harvest distributed knowledge. A voting and reputation system isn't necessary for all forms of crowdsourcing, but as the web matures, we're seeing voting and reputation systems being applied in more and more places with amazing results [22].

Even having the recommendation system for problem solving based on crowdsourcing and other similar characteristics to WebProcessPAIR, StackOverflow lacks a very important feature, the automatic recommendation of solutions. While at StackOverflow recommended solutions have to be made through a question proposed by a user, in WebProcessPAIR, the recommended improvement actions are proposed automatically to users that run their performance data in the application. WebProcessPAIR recommend the best improvement actions for each problem, based on a catalogue of improvement actions. The construction of the catalogue is similar to StackOverflow, where users can check the list of problems encountered, and write recommendations for each problem.

WebMD - Symptom Checker

Another example that has some similar characteristics with the intended objectives of WebProcessPAIR is the WebMD - symptom checker [23]. In this case the system does not use the concept of crowdsourcing, but like WebProcessPAIR, from causes and their problems, presents possible solutions. In order to better understand how it works, let's look at the following analysis.

Symptom checker is a tool from WebMD intended for informational purposes on the symptoms provided by the user.

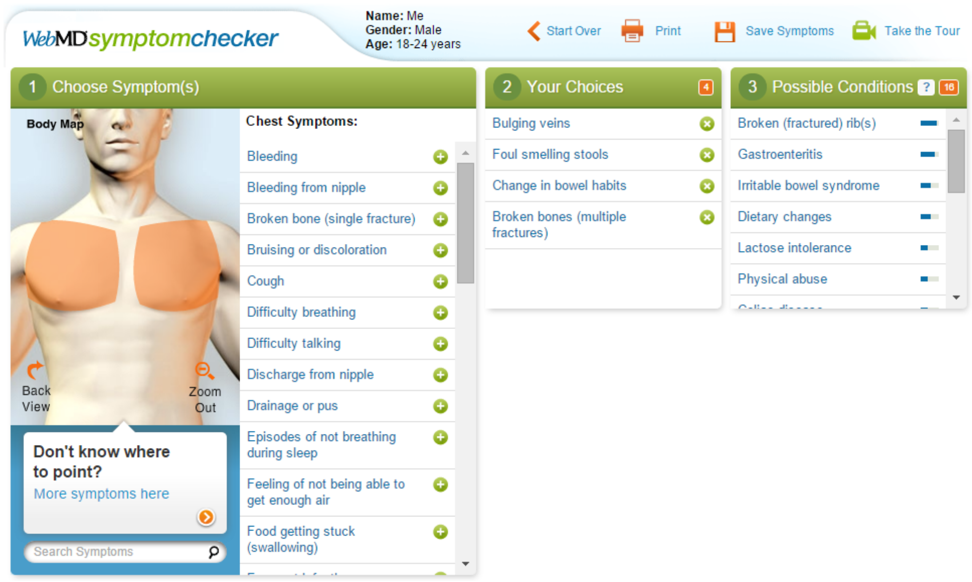
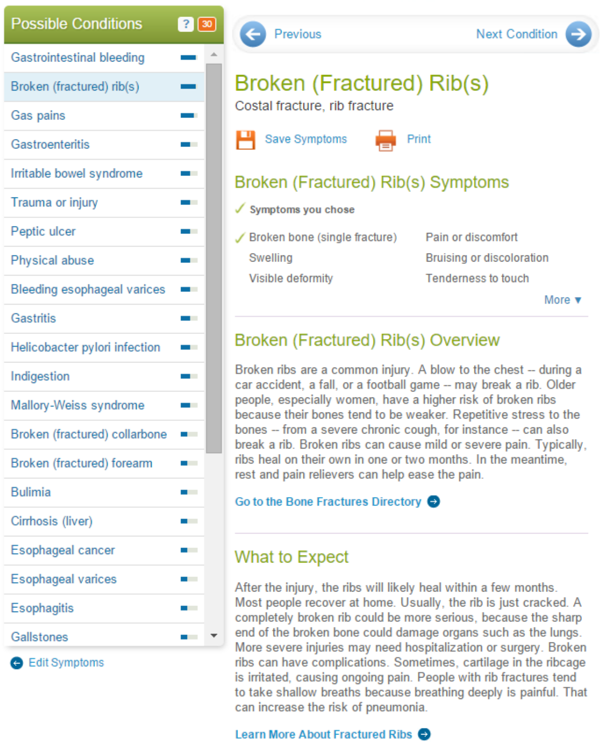


Figure 17 - Structure of WebMD symptom checker

In Figure 17 it is presented the structure of the system, including the choice of symptoms in section 1, selecting first the part of the body with the problem (one or more parts), and then choosing the symptoms from the list. In section 2 we can see the list of symptoms chosen. In point 3 are presented the solutions to the problem, in this case, the possible conditions, listed in order of how closely the symptoms match those conditions. Finally, by clicking in a condition it is presented a detailed analysis of the condition (Figure 18).


Possible Conditions

Gastrointestinal bleeding
Broken (fractured) rib(s)
Gas pains
Gastroenteritis
Irritable bowel syndrome
Trauma or injury
Peptic ulcer
Physical abuse
Bleeding esophageal varices
Gastritis
Helicobacter pylori infection
Indigestion
Mallory-Weiss syndrome
Broken (fractured) collarbone
Broken (fractured) forearm
Bulimia
Cirrhosis (liver)
Esophageal cancer
Esophageal varices
Esophagitis
Gallstones

Edit Symptoms

Previous | Next Condition

Broken (Fractured) Rib(s)
Costal fracture, rib fracture

Save Symptoms | Print

Broken (Fractured) Rib(s) Symptoms

✓ Symptoms you chose

✓ Broken bone (single fracture)	Pain or discomfort
Swelling	Bruising or discoloration
Visible deformity	Tenderness to touch

More ▾

Broken (Fractured) Rib(s) Overview

Broken ribs are a common injury. A blow to the chest -- during a car accident, a fall, or a football game -- may break a rib. Older people, especially women, have a higher risk of broken ribs because their bones tend to be weaker. Repetitive stress to the bones -- from a severe chronic cough, for instance -- can also break a rib. Broken ribs can cause mild or severe pain. Typically, ribs heal on their own in one or two months. In the meantime, rest and pain relievers can help ease the pain.

Go to the Bone Fractures Directory

What to Expect

After the injury, the ribs will likely heal within a few months. Most people recover at home. Usually, the rib is just cracked. A completely broken rib could be more serious, because the sharp end of the broken bone could damage organs such as the lungs. More severe injuries may need hospitalization or surgery. Broken ribs can have complications. Sometimes, cartilage in the ribcage is irritated, causing ongoing pain. People with rib fractures tend to take shallow breaths because breathing deeply is painful. That can increase the risk of pneumonia.

Learn More About Fractured Ribs

Figure 18 - Analysis of a possible condition

2.3.4 Type of recommendation system in WebProcessPAIR

Based on the types of recommendation systems analysed, WebProcessPAIR uses recommendation system based on crowdsourcing, for building the catalogue, on the side of contributors, and recommend improvement actions, on the side of users with identified problems.

For building the catalogue of improvement actions, this type of recommendation system permits the contributors to comment with solutions, for the problems exposed by the platform, which can be voted on (up/down voting system) by other users. The best proposed solutions, i.e. those which have the highest number of votes, are used as improvement actions for similar problems in the future.

For recommending improvement actions, WebProcessPAIR selects the best recommendations on the catalogue (those with the most votes), for each problem encountered in the personal performance data of the user, and presents a list with the most appropriate improvement actions for those problems. For each improvement action recommended, the user can leave his feedback (accept answer, vote or comment).

Chapter 3

WebProcessPAIR Design, Implementation and Testing

This chapter describes the design and implementation of WebProcessPAIR, presenting the use case model, conceptual model, architecture and technologies, database structure, MVC components, gems installed, navigation map and integration with ProcessPAIR.

3.1 Use cases

Like is shown in the use case diagram in Figure 19, WebProcessPAIR counts with 3 types of user: contributors, developers and administrators.

The contributors may contribute to the catalogue, proposing improvement actions for the problems exposed in the website. In addition to proposing, the contributors can also vote in other contributions. Examples of contributors are instructors, coaches, and members of PSP/TSP community (Tsp-psP Heroes group of LinkedIn).

The developers, through the submission of their performance data, generated by PSP or Scraim (future work), can automatically analyse their performance problems and causes. For the identified problems are presented the best three improvement actions (from the catalogue of improvement actions). For the improvement actions recommended, the user can give his feedback, accepting the proposal or not. Examples of developers are students in training courses and software developers.

The aim of the administrator is to validate the improvement actions, checking if the proposal is in accordance with the rules of the website. The administrator can also create performance indicators and manage all users.

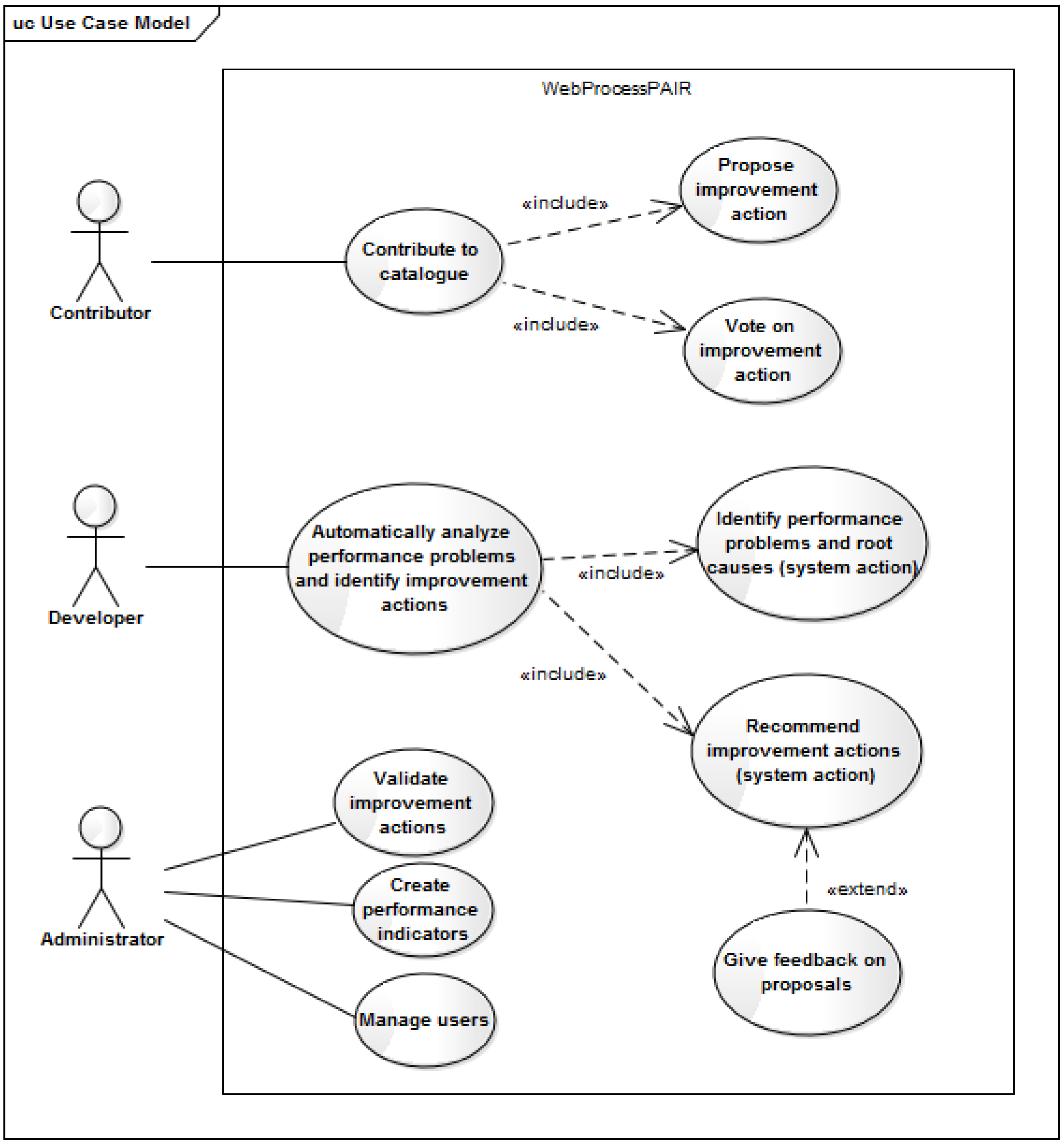

uc Use Case Model
WebProcessPAIR
Contributor
Developer
Administrator
Contribute to catalogue
«include»
Propose improvement action
«include»
Vote on improvement action
Automatically analyze performance problems and identify improvement actions
«include»
Identify performance problems and root causes (system action)
«include»
Recommend improvement actions (system action)
«extend»
Give feedback on proposals
Validate improvement actions
Create performance indicators
Manage users

Figure 19 – UML use case diagram for WebProcessPAIR

3.2 Conceptual model

The conceptual structure of the classes involved in WebProcessPAIR approach is depicted in Figure 20. For a better understating of the presented conceptual model, it is shown in Table 7 a description of all classes involved.

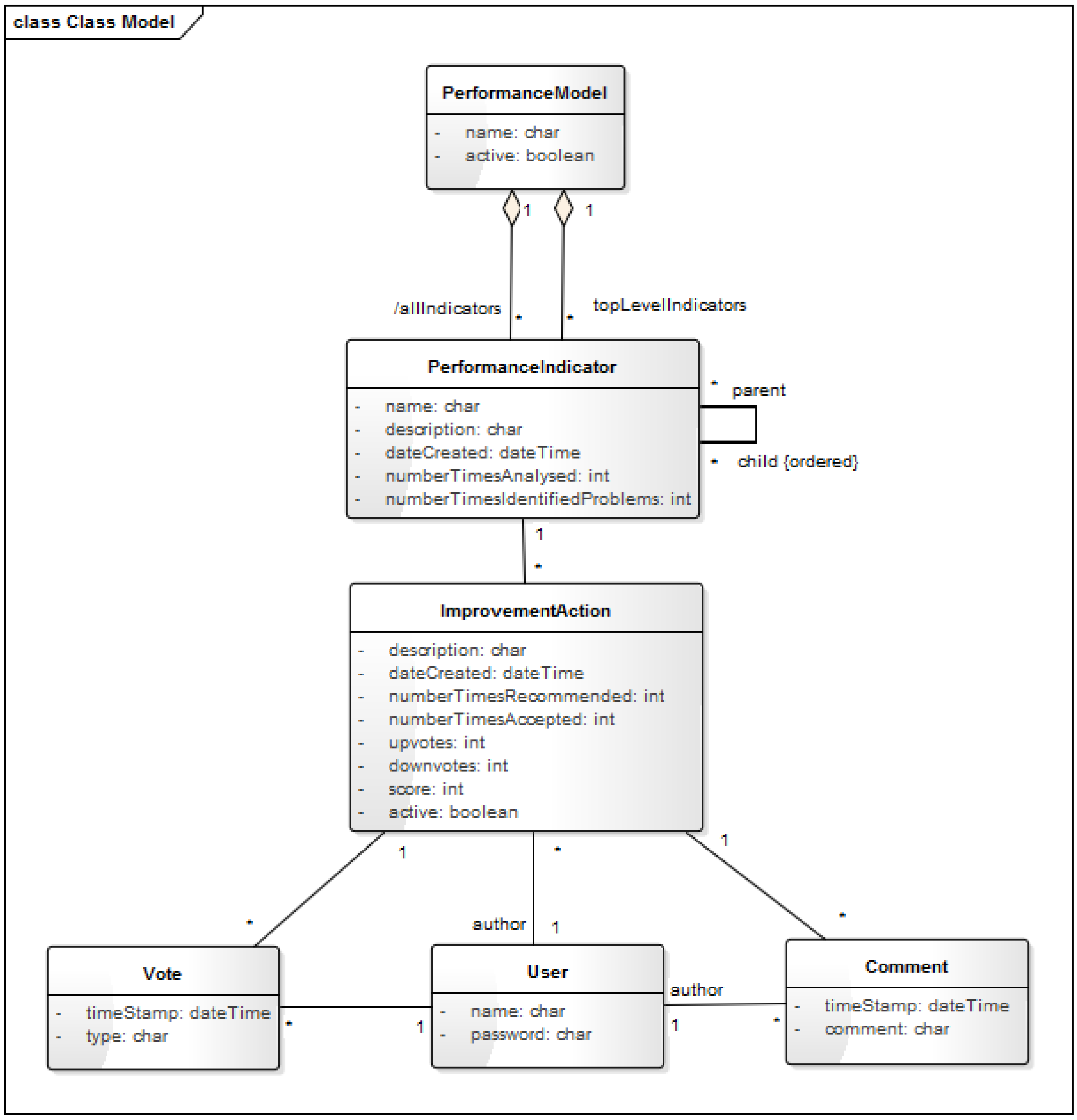


Figure 20 - UML class diagram depicting the main concepts of WebProcessPAIR

Table 6 – Description of classes in the conceptual model

Class	Description
PerformanceModel	A performance model comprises a set of performance indicators (top level or child) allowing that the performance data of individual developers can be automatically analysed by ProcessPAIR, to identify performance problems and identify and rank potential causes for those problems.
PerformanceIndicator	List of performance indicators considered in the performance model to analyse the data. Affecting indicators can be filtered and sorted when drilling down from higher-level (parent) to lower-level (child). To control which are the most frequent problems, it is counted how many times the PI was analysed and how many times were identified problems in the PI. Performance indicators are related by parent-child relationships, meaning the child (lower level) PI's affect the parent (higher level) PI's.
ImprovementAction	Improvement actions (IA) for the performance indicators, proposed by the users. All IA have a voting system (upvote/downvote), resulting in a voting score. For controlling the best IAs, it is counted the number of times an IA is recommended and accepted. The moderator can delete an IA if it's inappropriate.
User	The user can recommend, vote or comment improvement actions.
Comment	Comments are made by users with respect to proposed improvement actions. The user can also respond to a comment.
Vote	Votes are made by users with respect to proposed improvement actions. The type of vote is: upvote or downvote.

3.3 Architecture and technologies

In Figure 21 it is shown the deployment diagram of architecture and technologies of WebProcessPAIR. The diagram is composed by two nodes, client and server. The client needs a web browser to run the application. The server hosts the WebProcessPAIR application, that is developed in Ruby on Rails, which implements the MVC architecture (models, views, controllers). The application is connected to the WebProcessPAIR database, developed in PostgreSQL and to the ProcessPAIR java tool, which needs the Java Runtime Environment. The connection to the tool is done by running the business logic of the tool for identification of problems and causes implemented in a jar library. A more detailed information about the integration with ProcessPAIR is done in section 3.7.

The web deployment is done in a virtual machine provided by FEUP and accessed via SSH. On the virtual machine are installed all the necessary tools for running a Ruby on Rails[2] application, as well as all the project code. The web server is deployed with Thin[3] gem and Nginx[4], that are running on the VM. Nginx receives every request, then serves static files (css, js, images, cached files) directly. If the request requires processing, then it hands the request off to a rails process (Thin).

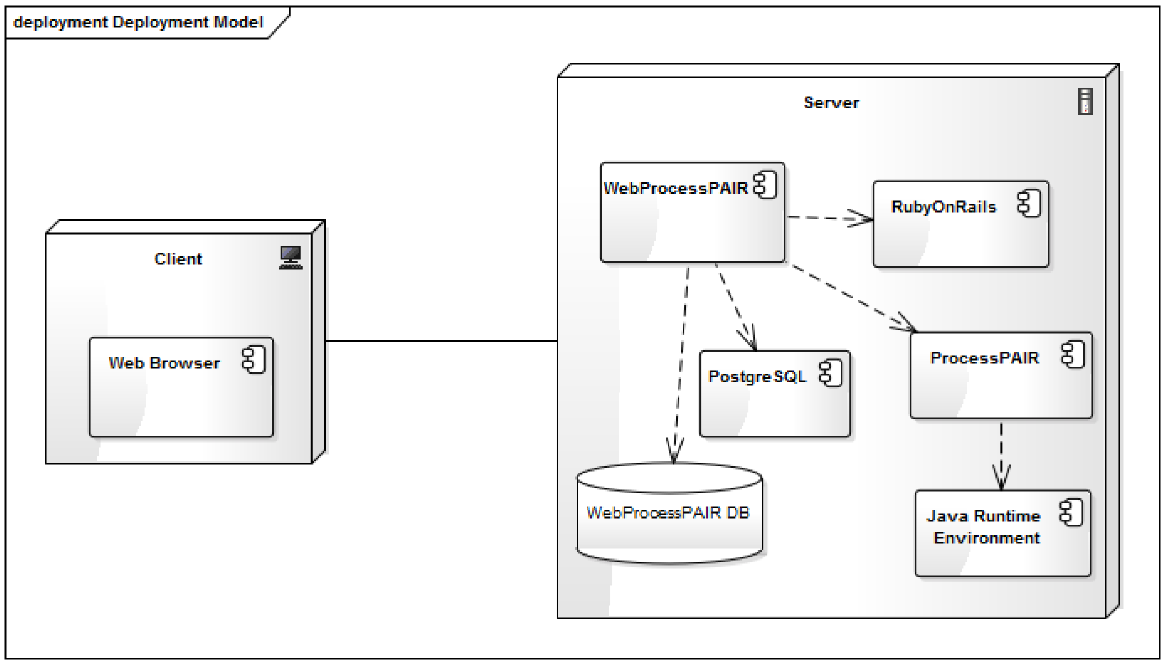


Figure 21 – Deployment diagram of WebProcessPAIR

[2] http://rubyonrails.org/
[3] https://rubygems.org/gems/thin/versions/1.7.0
[4] https://nginx.org/

3.4 Database structure

The database implementation was made from the Ruby on Rails application. For configuring the database it's needed the PostgreSQL gem and the following configuration in config/database.yml.

```
production:
  adapter:                                                   postgresql
  host:                                                       localhost
  encoding:                                                     unicode
  database:                                  WebProcessPAIR_production
  pool:                                                               5
  username:                                                       *****
  password:                                                       *****
  template: template0
```

After configuring the connection with Ruby on Rails and PostgreSQL, the following command creates the database.

```
rake db:create
```

To create a table, the following commands are used:

```
rails   generate   scaffold   User   name:string   email:string   image:string
personal_file:string admin:boolean
```

```
rake db:migrate
```

The above commands, create a model, view and controller for user interaction as well as the table in the database with the inputted fields.

Database diagram

In Figure 22 is presented the database diagram followed by an explanation of all tables in Table 8.

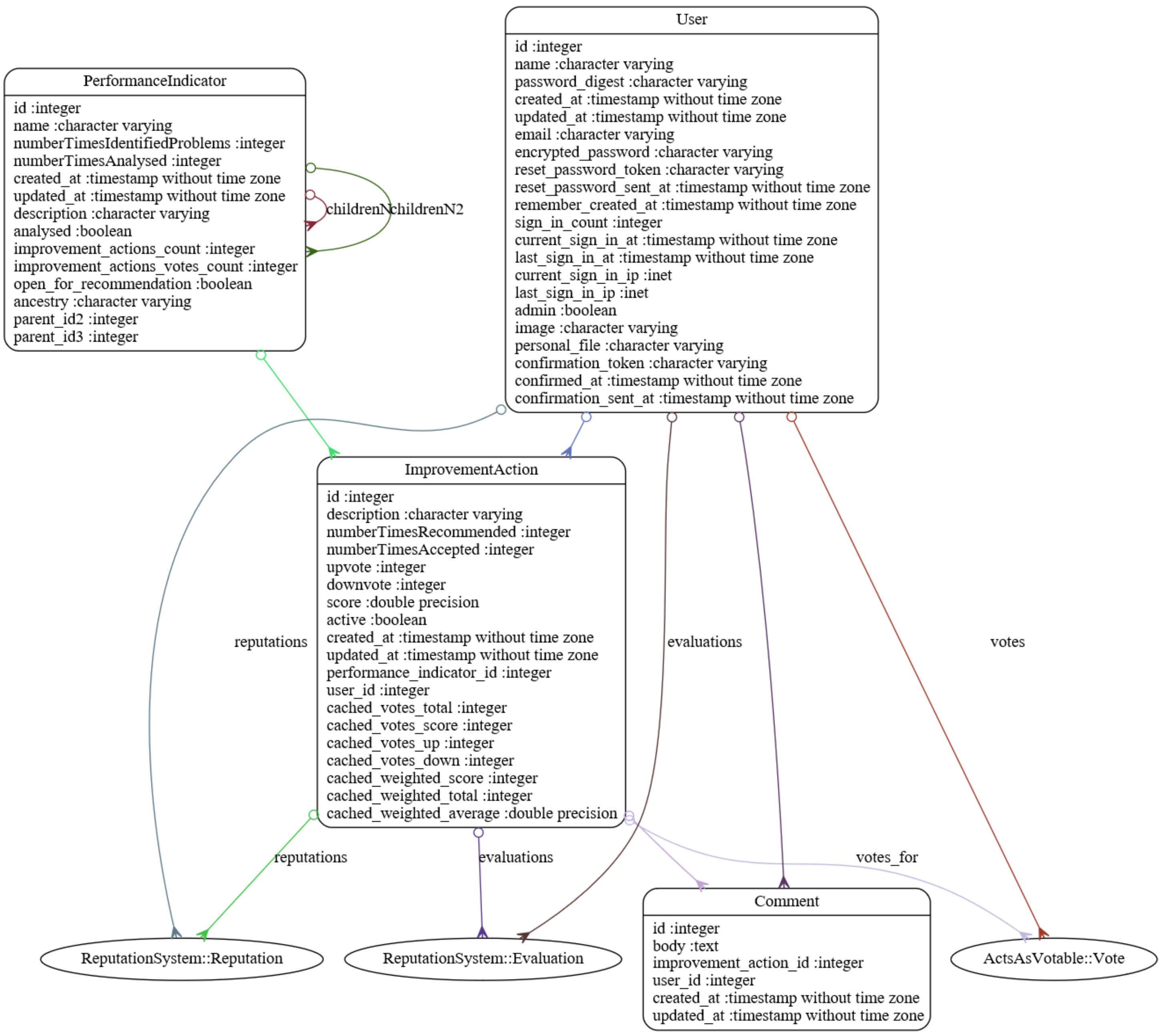

PerformanceIndicator
id :integer
name :character varying
numberTimesIdentifiedProblems :integer
numberTimesAnalysed :integer
created_at :timestamp without time zone
updated_at :timestamp without time zone
description :character varying
analysed :boolean
improvement_actions_count :integer
improvement_actions_votes_count :integer
open_for_recommendation :boolean
ancestry :character varying
parent_id2 :integer
parent_id3 :integer
childrenNchildrenN2
User
id :integer
name :character varying
password_digest :character varying
created_at :timestamp without time zone
updated_at :timestamp without time zone
email :character varying
encrypted_password :character varying
reset_password_token :character varying
reset_password_sent_at :timestamp without time zone
remember_created_at :timestamp without time zone
sign_in_count :integer
current_sign_in_at :timestamp without time zone
last_sign_in_at :timestamp without time zone
current_sign_in_ip :inet
last_sign_in_ip :inet
admin :boolean
image :character varying
personal_file :character varying
confirmation_token :character varying
confirmed_at :timestamp without time zone
confirmation_sent_at :timestamp without time zone
ImprovementAction
id :integer
description :character varying
numberTimesRecommended :integer
numberTimesAccepted :integer
upvote :integer
downvote :integer
score :double precision
active :boolean
created_at :timestamp without time zone
updated_at :timestamp without time zone
performance_indicator_id :integer
user_id :integer
cached_votes_total :integer
cached_votes_score :integer
cached_votes_up :integer
cached_votes_down :integer
cached_weighted_score :integer
cached_weighted_total :integer
cached_weighted_average :double precision
reputations
evaluations
votes
reputations
evaluations
votes_for
Comment
id :integer
body :text
improvement_action_id :integer
user_id :integer
created_at :timestamp without time zone
updated_at :timestamp without time zone
ReputationSystem::Reputation
ReputationSystem::Evaluation
ActsAsVotable::Vote

Figure 22 – Diagram of relational database model of WebProcessPAIR

Table 7 – Description of classes of the database diagram

Table	Description
User	The original fields created for the user are: name, password, email, image, personal file and admin. The image field permits the user to use a profile picture. The personal file field permits the user to upload his performance data. The admin boolean field is set to true if the user has moderator previleges and the remaining fields are generated by devise gem, used to perform various functionalities related to the user, such as sending email with the confirmation of registration or to reset the password.
PerformanceIndicator	To control which are the most frequent problems, it is counted how many times the PI was analysed in numberTimesAnalysed, and how many times were identified problems in the PI in numberTimesIdentifiedProblems. For listing only analysed PI's, the table has an "analysed" boolean that changes to true when the PI his analysed the first time. For ordering the PI by number of improvement actions and by votes, the table has the columns improvement_actions_count and improvement_actions_votes_ count. For opening or closing PI recommendations, the table has an open_for_recommendation boolean. To represent the performance indicators in the hierarchical tree, the parent nodes are saved in ancestry columns and, for multiple parents, in parent_id2/3.
ImprovementAction	To control wich are the best recommendations, it is counted how many times the (Improvement Action) IA was recommended in numberTimesRecommended, and how many times the IA was accepted in numberTimesAccepted. For listing the best recommendations, the table has a column "score" that is updated if the IA gets an upvote/downvote or accepted. The IA table has also vote columns generated by the acts_as_votable gem, which permits a user to vote on an improvement action.
Comment	The table comment consists of a body that represents the comment, associated with a user_id and improvement_action_id.
Vote	The vote table is generated by acts_as_votable gem.
Reputation/Evaluation	The reputation and evaluation tables are generated by activerecord-reputation-system gem, that permits the user to accept an improvement action that was recommended to him.

3.5 MVC components

In this section will be presented all MVC components for each class. For each component it is presented if it was reused, modified or new.

Table 8 - MVC components for each class

	Model	**View**	**Controller**
User	• devise • acts_as_voter • has_reputation: accepts • has_many: improvement_actions • has_many: comments • mount_uploader: image • mount_uploader: personal_file	• Devise views • Show • Edit • Index	• Devise controller • CRUD • update_user • bat_file • copy_file
Performance Indicator	• has_many: improvement_actions • has_ancestry • belongs_to:parent, class_name: "PerformanceIndicator", :foreign_key => 'parent_id2' • has_many:childrenN, class_name: "PerformanceIndicator", :foreign_key => 'parent_id2' • has_many:childrenN2, class_name: "PerformanceIndicator", :foreign_key => 'parent_id3'	• CRUD • tab_tree	• CRUD
Improvement Action	• belongs_to: performance_indicator, :touch => true • belongs_to: user • has_many: comments • acts_as_votable • belongs_to: user • has_reputation: accepts	• CRUD	• CRUD • accept • upvote • downvote • recommendations_update_score

Comments	• belongs_to: improvement_action, :touch => true • belongs_to: user	• CRUD	• CRUD
Analyse		• personal_performance • upload_file	• upload_file • personal_performance • destroy • parse_xml

CRUD (create, read, update, destroy): Ruby on rails actions - Index, show, create, new, edit, update, destroy.

- Reused 🟦
- Modified 🟥
- New ⬛

Components description

In the following list it's presented the description of the actions of each class in the MVC components.

Models

- User:
  - Devise - devise gem functionalities for the user
  - acts_as_voter - acts_as_votable gem that defines the user as the voter
  - has_reputation: accepts - activerecord-reputation-system gem that permits the user to "accept" some object
  - has_many: improvement_actions – each user can have many improvement actions
  - has_many: comments – each user can have many comments
  - mount_uploader: image – each user can upload an image to his profile picture
  - mount_uploader: personal_file – each user can upload his personal performance data to be analysed. The upload file will be stored on the database
- Performance indicator:
  - has_many: improvement_actions – each PI can have many improvement actions
  - has_ancestry – ancestry gem for the parent/child association between performance indicators, used for representing the tree of PI's.
  - belongs_to:parent, class_name: "PerformanceIndicator", :foreign_key => 'parent_id2' – extension for the parent/child association for children with multiple parents.

 - has_many:childrenN, class_name: "PerformanceIndicator", :foreign_key => 'parent_id2' – extension for the parent/child association for children with multiple parents.
 - has_many:childrenN2, class_name: "PerformanceIndicator", :foreign_key => 'parent_id3' – extension for the parent/child association for children with multiple parents.
- Improvement actions:
  - belongs_to: performance_indicator, :touch => true – each improvement action (IA) belongs to a performance indicator. Touch saves the last IA creation date, and is used for ordering the PI's by recent activity.
  - belongs_to: user – each IA belongs to a user
  - has_many: comments – each IA can have many comments
  - acts_as_votable – acts_as_votable gem that defines each IA as votable
  - belongs_to: user – each IA belong to some user
  - has_reputation: accepts – activerecord-reputation-system gem that permits an IA to be accepted
- Comments:
  - belongs_to: improvement_action, :touch => true – each comment belongs to an IA. Touch saves the last comment creation date, and is used for ordering the PI's by recent activity.
  - belongs_to: user – each comment belongs to a user

Views:

- User:
  - Devise views – login page, edit user, registration
  - Index – List all users (only for admin)
  - Show – Show user profile page (admin can also check other profiles)
  - Edit – Edit profile information (admin can also edit other users)

- Performance indicators:
  - CRUD – index, form, show, edit pages
  - Tab_tree – Partial view for the tree representation of performance indicators in the tab tree.

- Improvements actions:
  - CRUD – the CRUD pages from improvement actions are all partials from PI. Inside PI show are the IA's index

- Comments:
  - CRUD – the CRUD pages from comments are all partials from IA. Inside IA partial are the comments index
- Analyse:
  - Upload_file – Upload file page where the user can see his uploaded files, upload a file or delete a file.
  - Personal_performance – Report view of the upload file, where the user can check his performance problems and the best recommendations for each problem.

Controllers:

- User:
    - Devise controller – hardcoded devise controller that gives access to many functionalities related to the user
    - CRUD – index, show, edit, create, new, update, destroy actions
    - update_user – this action is for admins only, allowing them to update the data of other users
    - copy_file – this action is called after uploading a file, and copies the uploaded file to the folder where processpair is.
    - bat_file – after copying the file, this action runs a batch file which runs processpair.jar with the copied file as the data input. This generates the final report (xml), with the performance problems found in the input file.

- Performance indicators:
    - CRUD – index, show, edit, create, new, update, destroy actions

- Improvement actions:
    - CRUD – index, show, edit, create, new, update, destroy actions
    - accept – permits the user to accept a recommended IA.
    - upvote - permits the user to upvote an IA.
    - downvote – permits the user to downvote an IA
    - recommendations_update_score – after each upvote/downvote/accept the score of the IA is updated, depending on the user action. This is used for ordering the best recommendations in the user performance report

- Comments:
    - CRUD – index, show, edit, create, new, update, destroy actions

- Analyse:
    - upload_file – permits the user to upload his performance data file (xml)
    - parse_xml – Uses the nokogiri gem to read the report (xml) generated by the batch file of processpair.jar and to parse it to html
    - personal_performance – after the user uploading a file redirects the user to the personal performance view, running the parse_xml action to parse the uploaded xml file to html
    - destroy – the action permits the user to remove his uploaded data file

3.6 Gems installed

In this section will be presented the list of gems installed in WebProcessPAIR, followed by the description of each gem (the Ruby on Rails default gems are not listed).

- bootstrap-sass – Sass-powered version of Bootstrap 3, for using Bootstrap front-end framework.
- pg – PostgreSQL gem for running the database in the application.
- devise – gem for authentication system.
- acts_as_votable – gem for implementing the vote system in the application.
- carrierwave – gem for implementing the upload files and profile picture system for users.
- thin – gem for running the web server in the virtual machine.
- activerecord-reputation-system – gem for implementing the accept recommendations system.
- will_paginate – gem for paginate the list of performance indicators.
- trix – gem for using WYSIWYG editor (e.g. bold, italic) for writing recommendations.
- ancestry – gem for the parent-child relation of performance indicators for building the tree.
- Nokogiri – gem for parsing ProcessPAIR output xml file to html.

3.7 Navigation map

In this section will be presented UML state diagrams for all the available navigation options in WebprocessPAIR, for an unregistered user, registered user and administrator.

Unregistered user

In Figure 23 is shown the UI state diagram for an unregistered user. When an unregistered user enters WebProcessPAIR, he can access from anywhere on the website, the registration page, login page, about page, and list performance indicators page by using the search form or by clicking on the link to access the page. If the user accesses the performance indicators page, and selects one performance indicator, he is redirected to that performance indicator page, with its description, improvement actions and improvement actions comments.

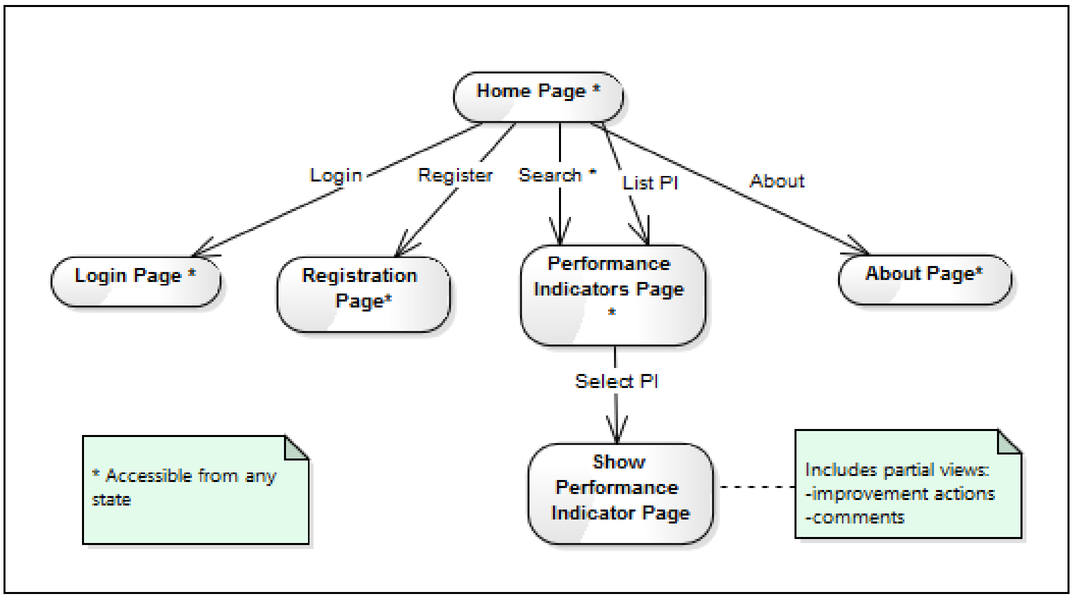


Figure 23 - UI state diagram for unregistered users

Registered user

In Figure 24 is shown the UI state diagram extension for registered users. A registered user, in addition to all the features an unregistered user has access, can also access his profile page as well as the edit profile page. In the performance indicators page, a registered user can make an improvement action (or edit/remove his IA's), comment improvement actions (or edit/remove his comments) and vote/unvote IA's. The registered user also has access from anywhere in the website to the analyse performance page. If the user uploads his personal data, he’s redirected to the personal performance page (report view of his data). In the personal performance page, the user can accept the recommendations to his problems or select a PI and see the PI page.

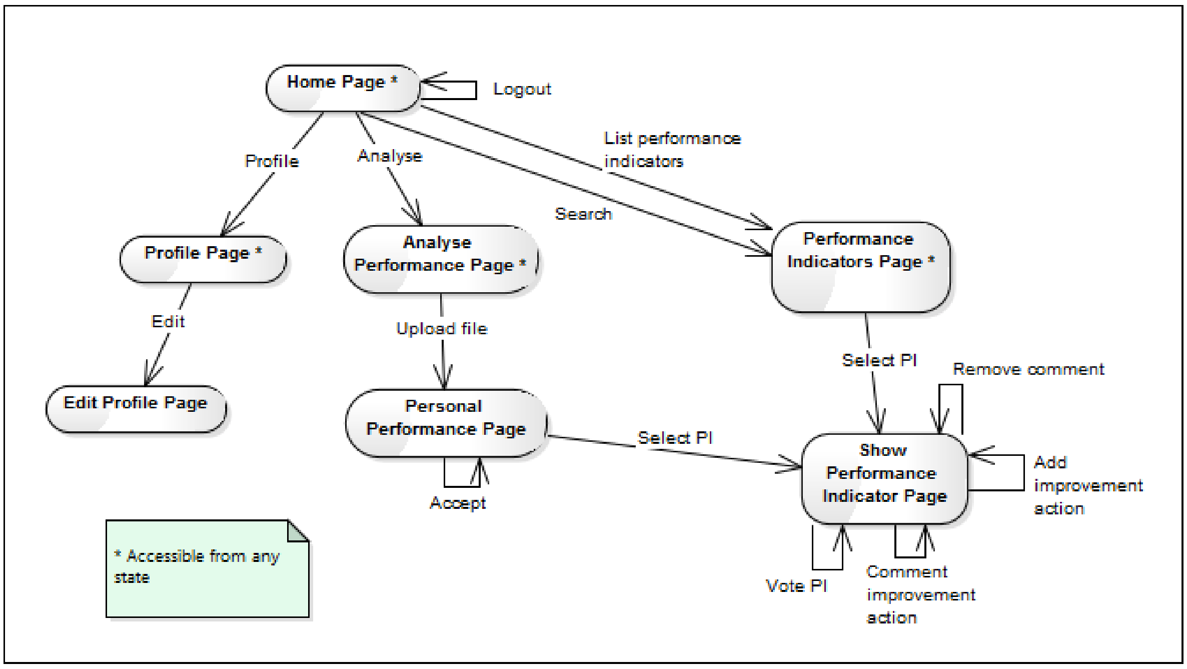

Home Page *
Logout
Profile
Analyse
List performance indicators
Search
Profile Page *
Analyse Performance Page *
Performance Indicators Page *
Edit
Upload file
Select PI
Remove comment
Edit Profile Page
Personal Performance Page
Select PI
Show Performance Indicator Page
Add improvement action
Accept
* Accessible from any state
Vote PI
Comment improvement action

Figure 24 – UI state diagram extention for registered users

Administrator

In Figure 25 is shown the UI state diagram extension for administrators. An administrator, in addition to all the features a registered user has access, can also access from anywhere in the website, the list users page, where he can view or edit a user. In the performance indicators page,

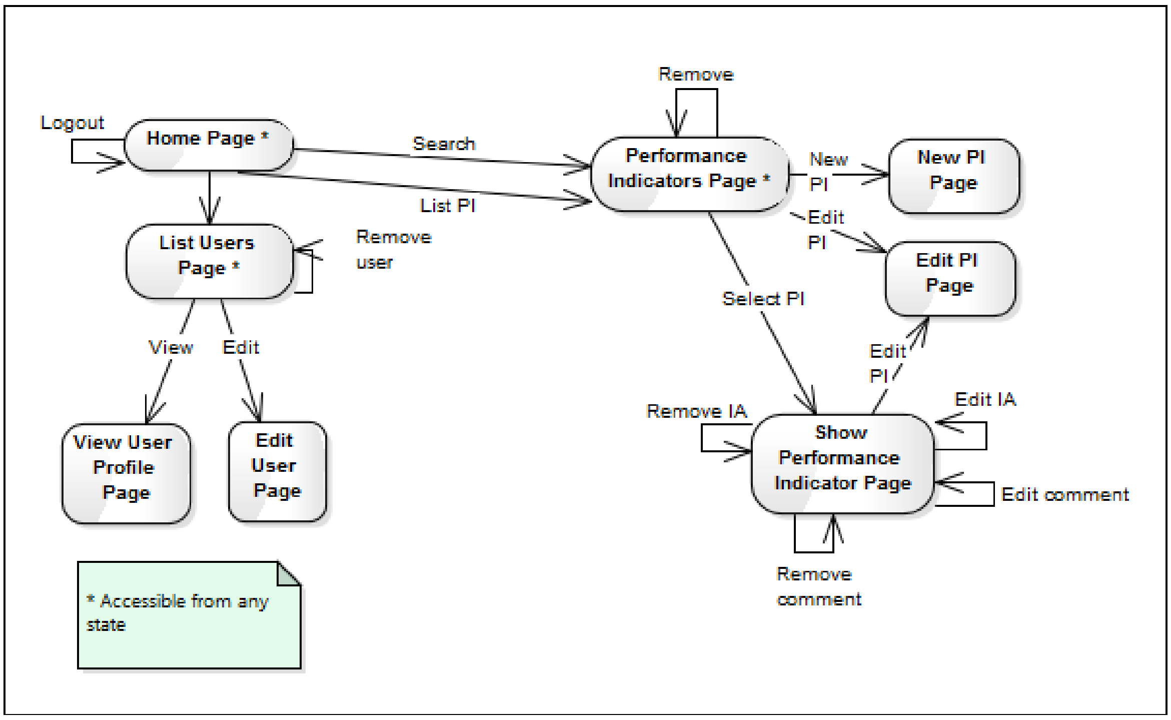


Figure 25 – UI state diagram extention for administrator

the admin can create, edit or remove a PI. Inside the PI page, he can edit that PI, edit or remove improvement actions and edit or remove comments.

3.8 Integration with ProcessPAIR

In Figure 26 it is presented a scheme of the integration between WebprocessPAIR and ProcessPAIR.

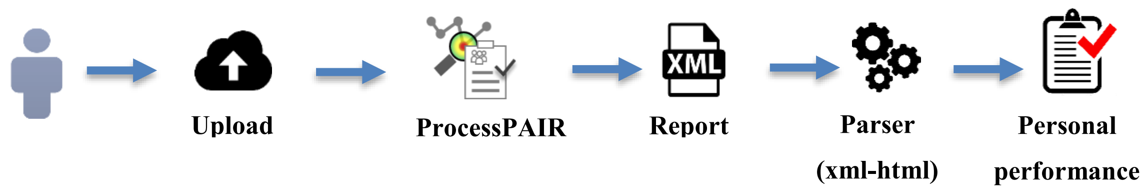


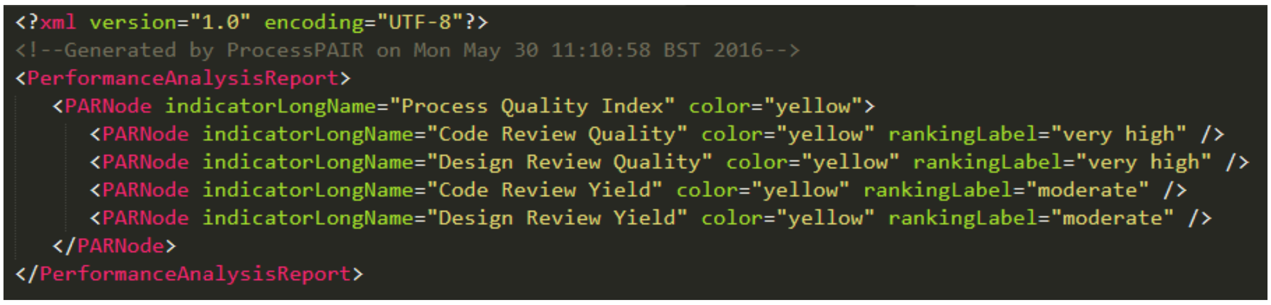
Figure 26 - Scheme of integration with ProcessPAIR

The process starts when a user uploads his performance data (XML format). After the file submission, the ProcessPAIR application runs via a batch file, which receives the following parameters:

```
ProcessPAIR.jar ProcessPairCalibration.xml data.xml report.xml
```

- ProcessPAIR.jar: Business logic of the application for identification of problems and causes implemented in a jar library.
- ProcessPairCalibration.xml: Required calibration file used to calibrate the performance model. For this calibration, the recommended performance ranges and statistical distribution were derived from a large PSP data set from the SEI referring to more than 3,000 PSP developers and 30,000 projects.
- Data.xml: User’s uploaded file with his performance data.
- Report.xml: After processpair.jar runs, it exports the personal data report to a xml file. In Figure 27 it’s presented an example of a report.xml.

After generating the report.xml, it is made the conversion from xml to html via the Nokogiri gem. Then the user is redirected to his performance analysis report, being presented the performance problems and causes and recommended improvement actions.



```
<?xml version="1.0" encoding="UTF-8"?>
<!--Generated by ProcessPAIR on Mon May 30 11:10:58 BST 2016-->
<PerformanceAnalysisReport>
   <PARNode indicatorLongName="Process Quality Index" color="yellow">
      <PARNode indicatorLongName="Code Review Quality" color="yellow" rankingLabel="very high" />
      <PARNode indicatorLongName="Design Review Quality" color="yellow" rankingLabel="very high" />
      <PARNode indicatorLongName="Code Review Yield" color="yellow" rankingLabel="moderate" />
      <PARNode indicatorLongName="Design Review Yield" color="yellow" rankingLabel="moderate" />
   </PARNode>
</PerformanceAnalysisReport>
```

Figure 27 - Example of report.xml generated by ProcessPAIR

3.9 Acceptance tests

In this section will be presented the acceptance tests made for WebProcessPAIR, in order to verify if the final product meets the requirements and specification of the original proposal.

The tests we're made by the following types of users:

- Developer tests – Tests performed by the author to check if the final product is in accordance to the proposal.
- Customer tests – Tests performed by others (supervisor, co-supervisor), with scenarios and defined case tests, in order to provide feedback and to verify compliance with the requirements.
- Usability tests – Test performed by real users (e.g. PSP community). These tests are conditioned by the delivery date of the dissertation, because to users start using the product it may be required more time, however, a request was made for the group in Linkedin, PSP Heroes, and PSP teachers at Tecnológico de Monterrey, Mexico to test the final product.

List of acceptance tests:

- Register user
- Login after registration
- Edit profile
- List performance indicators (orber by recent, recommendations, votes, tree)
- Search performance indicators
- List improvement actions (order by votes, recent, oldest)

- Create improvement action
- Comment on improvement action
- Edit/delete improvement actions/comments
- Vote on improvement actions
- Submit personal performance file
- Check performance problems
- Accept recommended actions

Chapter 4

WebProcessPAIR Usage

This chapter describes how to use WebProcessPAIR, presenting the following topics: how to access WebProcessPAIR, user registration, definition of performance indicators, recommend actions and performance analysis with usage instructions and validation case studies.

4.1 How to access WebProcessPAIR

In order to access WebProcessPAIR, the user must do the following steps:

1. Visit: webprocesspair.fe.up.pt (Figure 28).
2. To have full access to the website, the user must enter his email and password in the login form in the homepage or by clicking in “Log in” in the navigation bar.

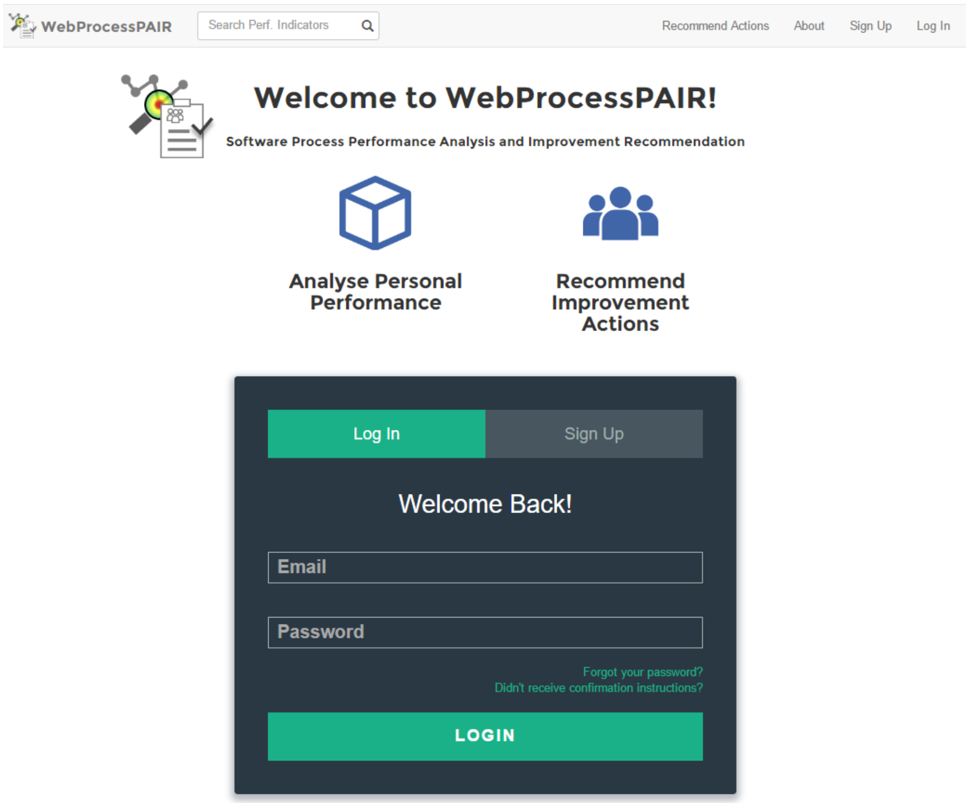

WebProcessPAIR
Search Perf. Indicators
Recommend Actions
About
Sign Up
Log In
Welcome to WebProcessPAIR!
Software Process Performance Analysis and Improvement Recommendation
Analyse Personal Performance
Recommend Improvement Actions
Log In
Sign Up
Welcome Back!
Email
Password
Forgot your password?
Didn't receive confirmation instructions?
LOGIN

Figure 28 - WebprocessPAIR homepage

4.2 User registration

To create an account in WebProcessPAIR, the user must do the following steps:

1. Access the registration form by clicking on "Sign up" on the navbar, or in sign up tab in homepage (Figure 29).
2. Complete the registration form and click on Sign Up. After completing the registration, an email with a link token, is send to the email address provided, in order to confirm the registration.
3. Click on the token link in the received email. After confirming the link in the email, the user is registered in WebProcessPAIR (Figure 30). After registration, the user can see and edit his profile by clicking on his name in the navbar followed by "profile".

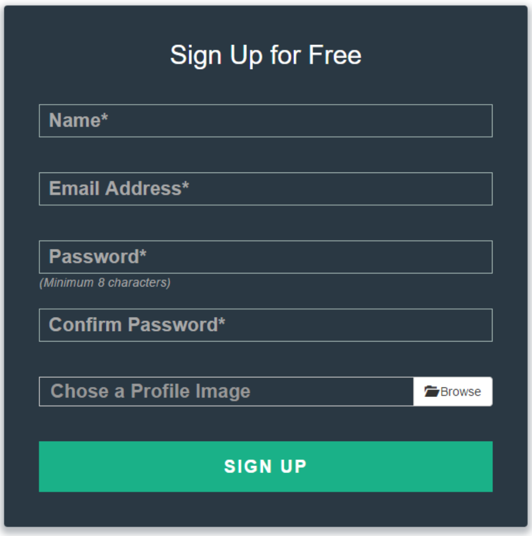


Figure 29 - Registration form

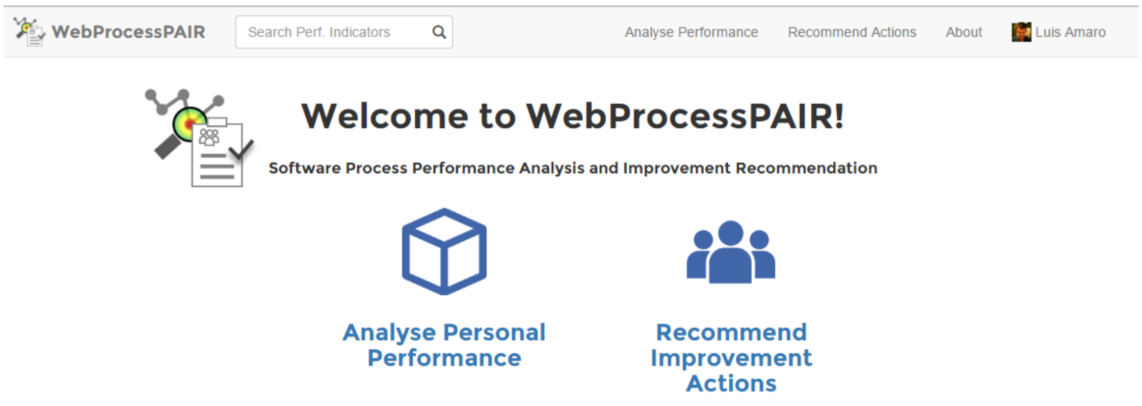

WebProcessPAIR
Search Perf. Indicators
Analyse Performance
Recommend Actions
About
Luis Amaro
Welcome to WebProcessPAIR!
Software Process Performance Analysis and Improvement Recommendation
Analyse Personal Performance
Recommend Improvement Actions

Figure 30 - Registered user homepage

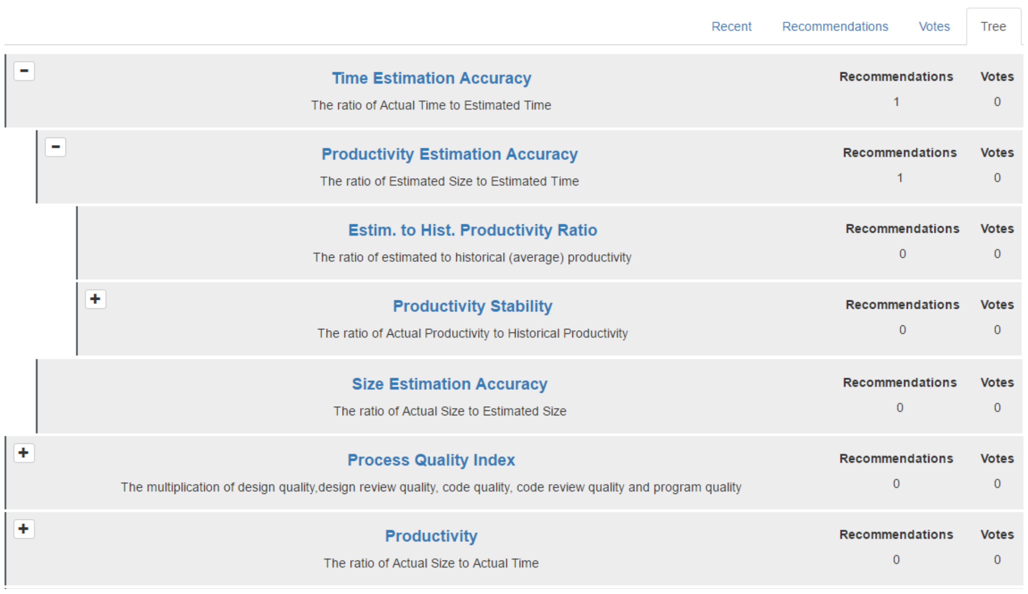
4.3 Definition of performance indicators

4.3.1 Usage instructions

To access the performance indicators list, the user must do the following steps:

1. Click "Recommend Improvement Actions" on the homepage or "Recommend Actions" on the navbar.

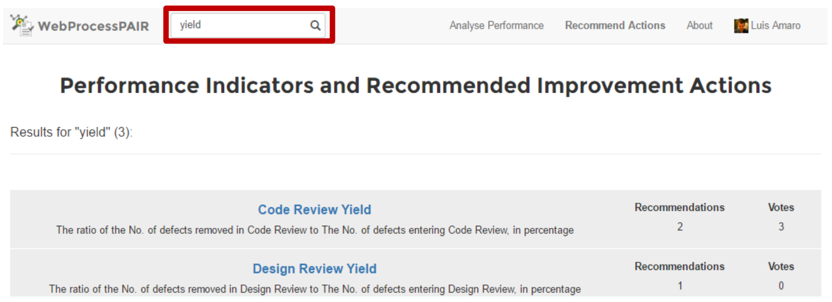
2. The user can order the list of PI in different ways by clicking on the tabs (Figure 31):
   a. Recent: PI's are ordered by the ones with the most recent improvement action or comment.
   b. Recommendations: PI's are ordered by number of recommendations.
   c. Votes: PI's are ordered by number of votes.
   d. Tree: Like is shown in Figure 31, the tab tree shows all PI's in a hierarchical order, showing the parent-child relation of performance indicators. In recent, recommendations and votes, are shown only the PI's that have already been analysed in user's personal performance.



Performance indicator	Recommendations	Votes
Time Estimation Accuracy – The ratio of Actual Time to Estimated Time	1	0
Productivity Estimation Accuracy – The ratio of Estimated Size to Estimated Time	1	0
Estim. to Hist. Productivity Ratio – The ratio of estimated to historical (average) productivity	0	0
Productivity Stability – The ratio of Actual Productivity to Historical Productivity	0	0
Size Estimation Accuracy – The ratio of Actual Size to Estimated Size	0	0
Process Quality Index – The multiplication of design quality,design review quality, code quality, code review quality and program quality	0	0
Productivity – The ratio of Actual Size to Actual Time	0	0

Figure 31 - Performance indicators page (tab "tree")

The user can also search for performance indicators, by submitting the desired text on the search form, like is shown in Figure 32. After submitting, the performance indicators found will appear (the search form looks for performance indicators name or description equal to the submitted text).



Performance Indicators and Recommended Improvement Actions

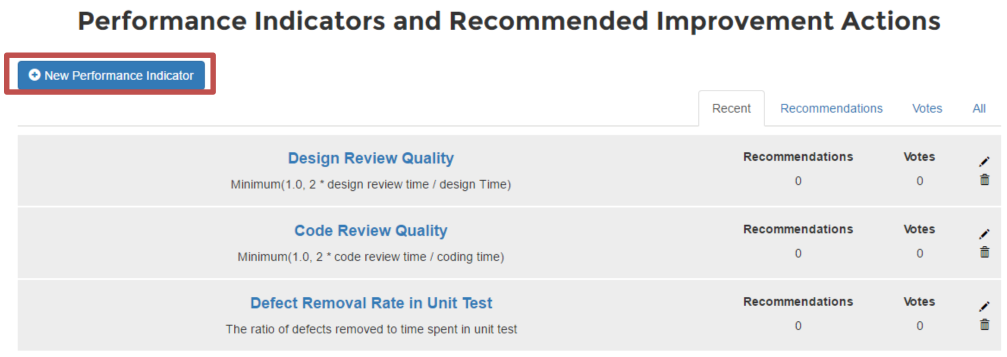
Results for "yield" (3):

Performance indicator	Recommendations	Votes
Code Review Yield – The ratio of the No. of defects removed in Code Review to The No. of defects entering Code Review, in percentage	2	3
Design Review Yield – The ratio of the No. of defects removed in Design Review to The No. of defects entering Design Review, in percentage	1	0

Figure 32 - Performance indicators found using the search form

To create, edit or remove a performance indicator, the user must be an administrator. If the user has administrator privileges, he can create a new PI in the PI's page, by doing the following steps:

1. Access the performance indicators page.
2. Click on "New Performance Indicator" (Figure 33).
3. Enter name, description, open or closed for recommendation and select the indicator parent (nil if it's a top level indicator; if the indicator has multiple parents, the administrator must select parent 2 or 3) for the performance indicator (Figure 34).
4. Click on "Create Performance Indicator".

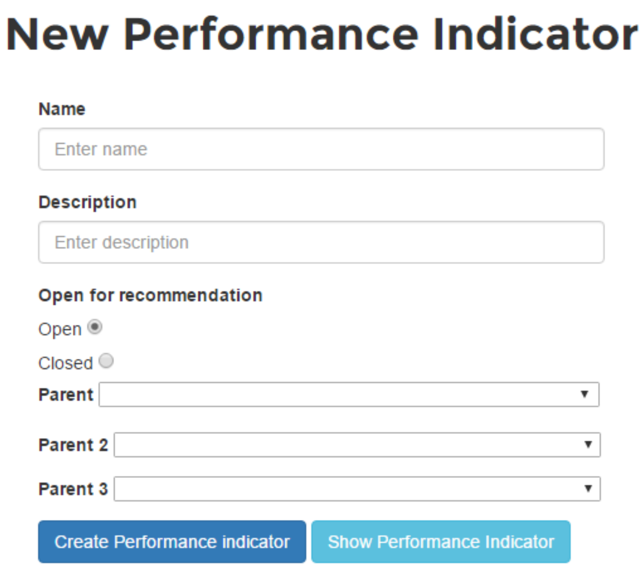

Performance Indicators and Recommended Improvement Actions

New Performance Indicator

Recent | Recommendations | Votes | All

Design Review Quality Minimum(1.0, 2 * design review time / design Time)	Recommendations 0	Votes 0	
Code Review Quality Minimum(1.0, 2 * code review time / coding time)	Recommendations 0	Votes 0	
Defect Removal Rate in Unit Test The ratio of defects removed to time spent in unit test	Recommendations 0	Votes 0	

Figure 33 - Performance indicators page (admin)


New Performance Indicator

Name
Enter name

Description
Enter description

Open for recommendation
Open [✓]
Closed

Parent

Parent 2

Parent 3

Create Performance indicator | Show Performance Indicator

Figure 34 - New performance indicator page (admin)

4.3.2 Performance indicators loaded to the system

With an administrator account were added to the system the following performance indicators (hierarchy list, showing 58 parent/children performance indicators):

- Time estimation accuracy
    - Productivity Estimation Accuracy
        - Estim. to Hist. Productivity Ratio
        - Productivity Stability
            - Plan Productivity Stability
            - Design Productivity Stability
            - Design Review Productivity Stability
            - Code Productivity Stability
            - Code Review Productivity Stability
            - Compile Productivity Stability
            - Unit Test Productivity Stability
            - Postmortem Productivity Stability
    - Size Estimation Accuracy
- Process Quality Index
    - Code Quality
        - Defect Density in Compile
            - Defects Injected
                - Defects Injected in Plan
                - Defects Injected in Design
                - Defects Injected in Design Review
                - Defects Injected in Code
                - Defects Injected in Code Review
                - Defects Injected in Compile
                - Defects Injected in Unit Test
                - Defects Injected in Postmortem
            - ProcessYield

            - Code Review Yield
                - Code Review Productivity
            - Design Review Yield
                - Design Review Productivity
    - Code Review Quality
    - Design Quality
    - Design Review Quality
    - Program Quality
        - Defect Density in Unit Test
            - Defects Injected*
                - Defects Injected in Plan*
                - Defects Injected in Design*
                - Defects Injected in Design Review*
                - Defects Injected in Code*
                - Defects Injected in Code Review*
                - Defects Injected in Compile*
                - Defects Injected in Unit Test*
                - Defects Injected in Postmortem*
            - ProcessYield
                - Code Review Yield*
                    - Code Review Productivity*
                - Design Review Yield*
                    - Design Review Productivity*
- Productivity
    - Plan Productivity
    - Design Productivity
    - Design Review Productivity
    - Code Productivity
    - Code Review Productivity

- Compile Productivity
  - Defect Density in Compile*
    - Defects Injected*
      - Defects Injected in Plan*
      - Defects Injected in Design*
      - Defects Injected in Design Review*
      - Defects Injected in Code*
      - Defects Injected in Code Review*
      - Defects Injected in Compile*
      - Defects Injected in Unit Test*
      - Defects Injected in Postmortem*
    - ProcessYield*
      - Code Review Yield*
        - Code Review Productivity*
      - Design Review Yield*
        - Design Review Productivity*
- Unit Test Productivity
  - Defect Density in Unit Test*
    - Defects Injected*
      - Defects Injected in Plan*
      - Defects Injected in Design*
      - Defects Injected in Design Review*
      - Defects Injected in Code*
      - Defects Injected in Code Review*
      - Defects Injected in Compile*
      - Defects Injected in Unit Test*
      - Defects Injected in Postmortem*
    - ProcessYield*
      - Code Review Yield*

            - Code Review Productivity*
        - Design Review Yield*
            - Design Review Productivity*
    - Postmortem Productivity
- DRL of Reviews or Compile versus Unit Testing
    - DRL of Design Review versus Unit Test
    - DRL of Code Review versus Unit Test
    - DRL of Compile versus Unit Test
- Defects Removed
    - Defect Density in Design Review
    - Defect Density in Code Review
    - Defect Density in Compile*
    - Defect Density in Unit Test*
- Defect Removal Rate
    - Defect Removal Rate in Design Review
    - Defect Removal Rate in Code Review
    - Defect Removal Rate in Compile
    - Defect Removal Rate in Unit Test
- Cost of Quality
    - Appraisal Cost of Quality
    - Failure Cost of Quality
- Appraisal to Failure Ratio

*Repeated performance indicators (with multiple parents) were inserted in the system only once.

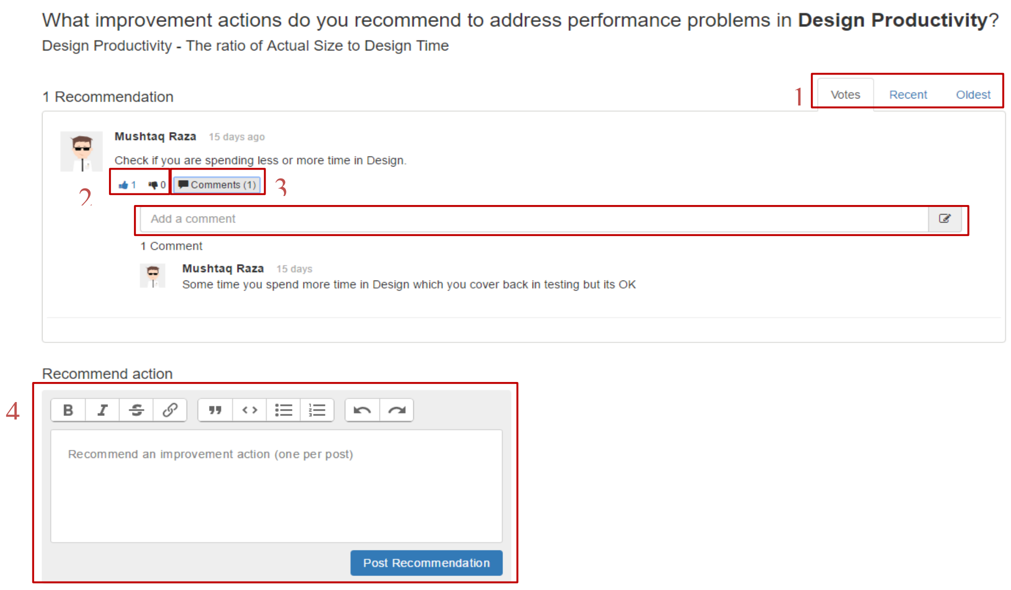
4.4 Recommend actions

4.4.1 Usage instructions

If the user clicks on a performance indicator from the list, the description of the PI, followed by the recommended actions will appear. The recommendations with the most votes and accepted answers are used to recommend in the personal performance (section 4.5).

In the PI page the user can:

1. Order the improvement actions by clicking on the tabs (1 on Figure 35):
    a. Votes: IA's are ordered by number of votes.
    b. Recent: IA's are ordered by creation date.
    c. Oldest: IA's are ordered by oldest creation date.
2. Vote, downvote or unvote recommendations, by clicking on the thumbs up/down buttons to vote, or by clicking again on the previous vote to unvote (cancel vote) (2).
3. Comment a recommendation, by clicking on the "Comments" button to expand comments area and submit his comment. The user can edit or remove his comments (3).
4. Recommend an improvement action for that performance indicator. The user can edit or remove his recommendations (4).


What improvement actions do you recommend to address performance problems in **Design Productivity**?
Design Productivity - The ratio of Actual Size to Design Time

1 Recommendation

Mushtaq Raza 15 days ago
Check if you are spending less or more time in Design.

1 Comment
Mushtaq Raza 15 days
Some time you spend more time in Design which you cover back in testing but its OK

Recommend action

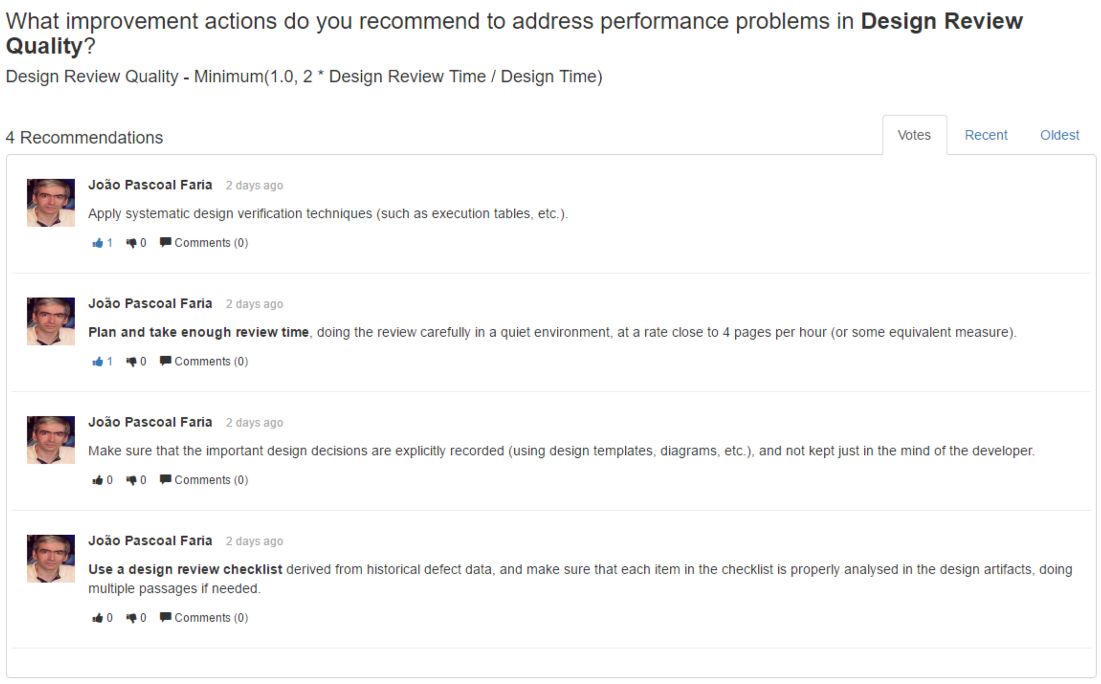
Figure 35 - Performance indicator with improvement actions page

4.4.2 Recommendations loaded to the system

In this section will be presented an example of the current status of WebProcessPAIR regarding the improvement actions created by real users. For that, it was requested to the group of Linkedin, “PSP Heroes”, and to professors at Tecnológico de Monterrey, Mexico, to start using the website and to write improvement actions for the performance indicators. However, that could take time, so at the moment, the current improvement actions are from supervisor João Pascoal Faria and co-supervisor Mushtaq Raza.

In Figure 36 is shown the current improvement actions created for performance indicator “Design review quality”. The full current status is presented in Appendix A, with all the performance indicators that have improvement actions.

Design review quality:



What improvement actions do you recommend to address performance problems in **Design Review Quality**?

Design Review Quality - Minimum(1.0, 2 * Design Review Time / Design Time)

4 Recommendations

João Pascoal Faria 2 days ago

Apply systematic design verification techniques (such as execution tables, etc.).

1 0 Comments (0)

João Pascoal Faria 2 days ago

Plan and take enough review time, doing the review carefully in a quiet environment, at a rate close to 4 pages per hour (or some equivalent measure).

1 0 Comments (0)

João Pascoal Faria 2 days ago

Make sure that the important design decisions are explicitly recorded (using design templates, diagrams, etc.), and not kept just in the mind of the developer.

0 0 Comments (0)

João Pascoal Faria 2 days ago

Use a design review checklist derived from historical defect data, and make sure that each item in the checklist is properly analysed in the design artifacts, doing multiple passages if needed.

0 0 Comments (0)

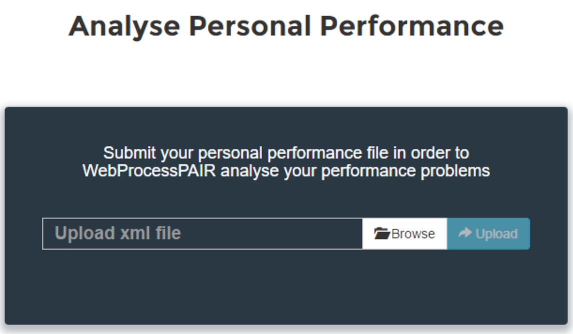
Figure 36 - Improvement actions for performance indicator "Design Review Quality"

4.5 Performance analysis

4.5.1 Usage instructions

To analyse the performance data, the user must do the following steps:

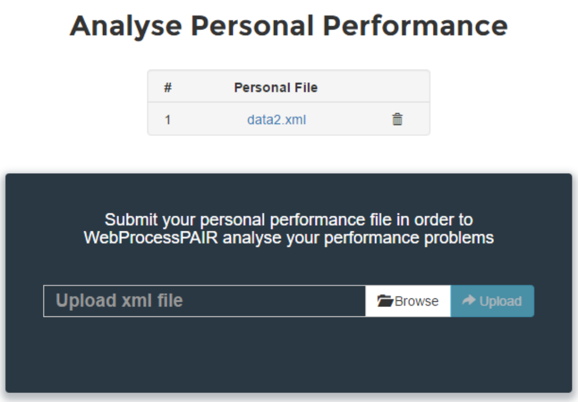
1. Click on "Analyse personal performance" on the homepage, or in "analyse performance" on the navbar.
2. Click on browse and select his personal data file (xml) (Figure 37).
3. Click on upload and be redirected to the performance problems analysis of his uploaded file (Figure 39).


Analyse Personal Performance

Submit your personal performance file in order to WebProcessPAIR analyse your performance problems

Figure 37 - Analyse personal performance page

If the user has already uploaded a personal data file, the analyse personal performance page should look like Figure 38. In this case to check his performance problems the user must click on his file name. The user can also upload other file or remove an uploaded file.

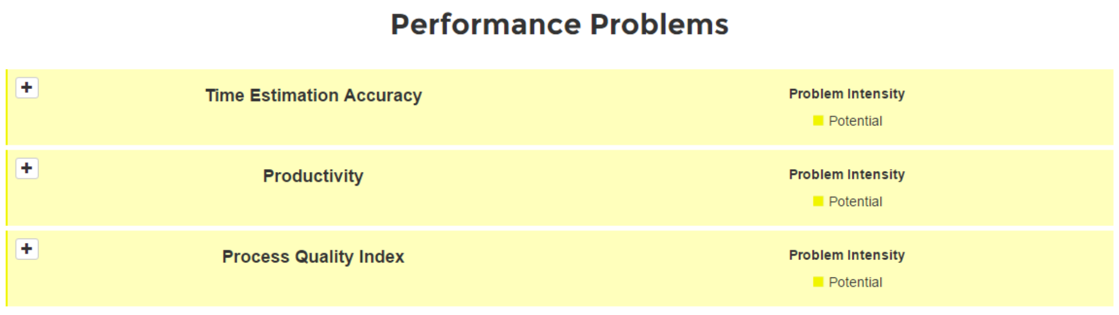

Analyse Personal Performance

#	Personal File
1	data2.xml

Submit your personal performance file in order to WebProcessPAIR analyse your performance problems

Figure 38 - Analyse personal performance page with an uploaded file

The goal of the Performance Problems page is to indicate the most relevant top-level performance problems (collapsed top level problems, Figure 39) and potential root causes (expanded top level problems, Figure 40). The top-level performance problems are ordered by problem intensity (potential - red, moderate - yellow), and the root causes are ordered by relation intensity (very high - red, high - orange, moderate - yellow).


Performance Problems

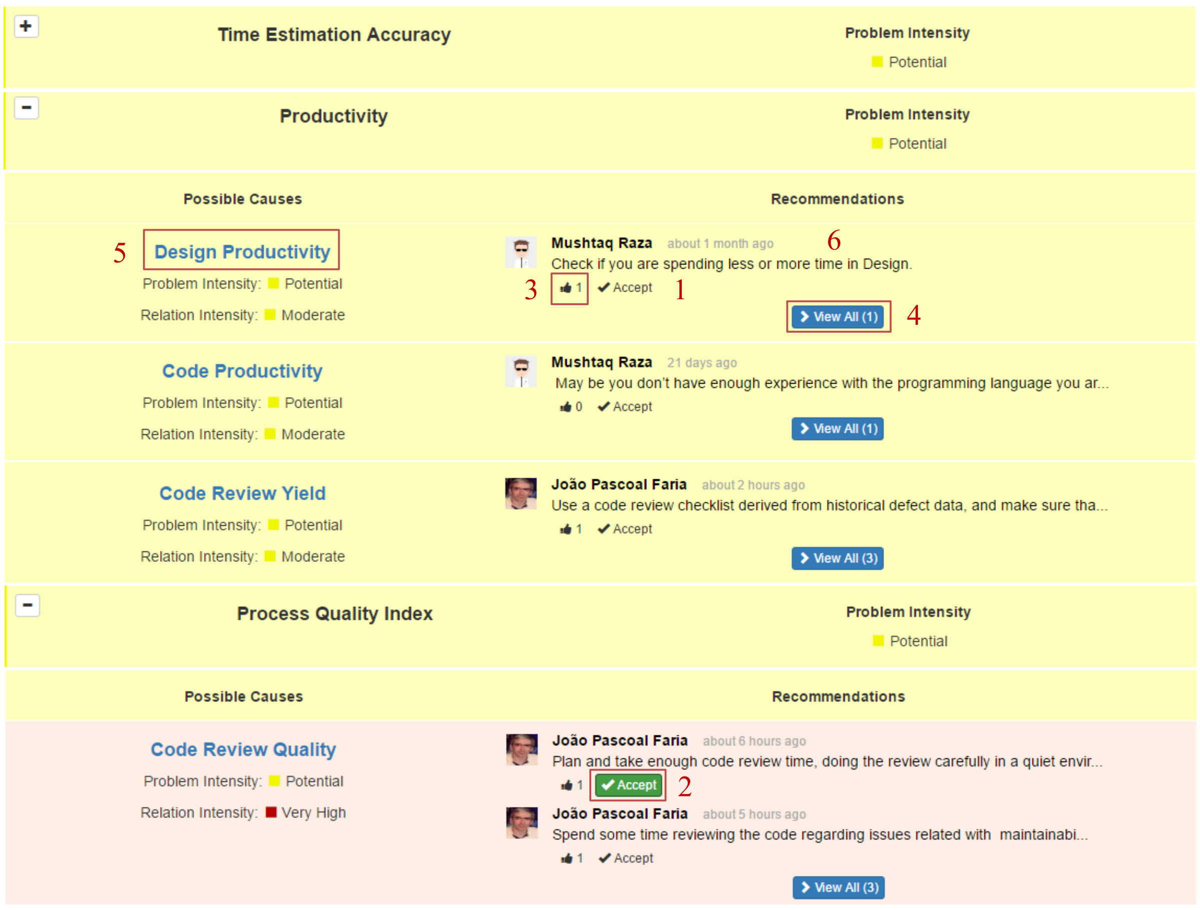
+	Time Estimation Accuracy	Problem Intensity 🟨 Potential
+	Productivity	Problem Intensity 🟨 Potential
+	Process Quality Index	Problem Intensity 🟨 Potential

Figure 39 - Performance problems page (top level indicators)

In performance problems page the user can:

1. Check the top level problems and its problem intensity (Figure 39). If the user hovers the mouse in the indicator name, he can see the indicator description.
2. Check the possible root causes for each problem and its problem and relation intensity. By clicking on the top-level problem, the list will expand like as shown in Figure 40. If the user hovers the mouse in the indicator name, he can see the indicator description.
    a. For each root cause, it is presented the best 3 recommendations (or less if it has less than 3 recommendations), made by users to that performance indicator. The recommendations are ordered by the following formula: "*score = 0.6*accepts + 0.4*votes*", prioritizing the number of accepts over the number of votes. Every time a user accepts or vote/downvote (in PI page) an action, the score to that recommended action is updated. If the top-level problem has no root causes, the recommendations will be made to that indicator.
3. Accept each recommended action (1 in Figure 40). Accepted answers appear in a green button (2 in Figure 40). If the user likes the recommendation, he should click on accept button to "thank" the user that recommended and to ensure that the best recommendations are presented. Only users that got recommended can accept answers.
4. Check the number of votes for each recommendation (3 in Figure 40).

5. To go to the indicator page, the user can click on (Figure 40):
   a. View all (4).
   b. Indicator name (5).
   c. On the recommendation (6) will redirect directly to that specific recommendation, like is shown in Figure 41. The clicked recommendation will appear highlighted for a few seconds.



Time Estimation Accuracy — Problem Intensity: 🟨 Potential

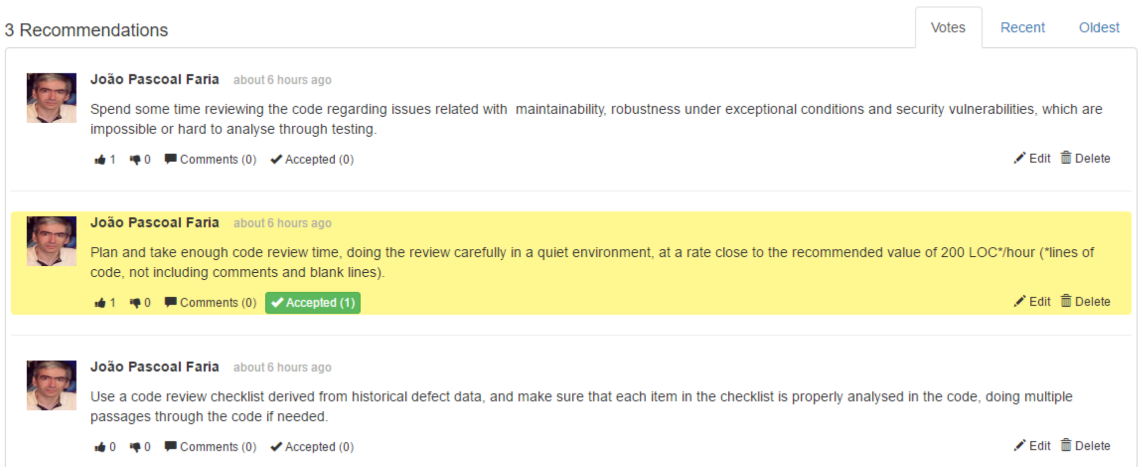
Productivity — Problem Intensity: 🟨 Potential

Possible Causes	Recommendations
Design Productivity Problem Intensity: 🟨 Potential Relation Intensity: 🟨 Moderate	Mushtaq Raza about 1 month ago Check if you are spending less or more time in Design. 1 ✓ Accept View All (1)
Code Productivity Problem Intensity: 🟨 Potential Relation Intensity: 🟨 Moderate	Mushtaq Raza 21 days ago May be you don't have enough experience with the programming language you ar... 0 ✓ Accept View All (1)
Code Review Yield Problem Intensity: 🟨 Potential Relation Intensity: 🟨 Moderate	João Pascoal Faria about 2 hours ago Use a code review checklist derived from historical defect data, and make sure tha... 1 ✓ Accept View All (3)

Process Quality Index — Problem Intensity: 🟨 Potential

Possible Causes	Recommendations
Code Review Quality Problem Intensity: 🟨 Potential Relation Intensity: 🟥 Very High	João Pascoal Faria about 6 hours ago Plan and take enough code review time, doing the review carefully in a quiet envir... 1 ✓ Accept João Pascoal Faria about 5 hours ago Spend some time reviewing the code regarding issues related with maintainabi... 1 ✓ Accept View All (3)

Figure 40 - Performance problems page (expanded with potential root causes)


3 Recommendations

Votes Recent Oldest

João Pascoal Faria about 6 hours ago

Spend some time reviewing the code regarding issues related with maintainability, robustness under exceptional conditions and security vulnerabilities, which are impossible or hard to analyse through testing.

1 0 Comments (0) Accepted (0) Edit Delete

João Pascoal Faria about 6 hours ago

Plan and take enough code review time, doing the review carefully in a quiet environment, at a rate close to the recommended value of 200 LOC*/hour (*lines of code, not including comments and blank lines).

1 0 Comments (0) ✓ Accepted (1) Edit Delete

João Pascoal Faria about 6 hours ago

Use a code review checklist derived from historical defect data, and make sure that each item in the checklist is properly analysed in the code, doing multiple passages through the code if needed.

0 0 Comments (0) Accepted (0) Edit Delete

Figure 41 - Highlighted recommendation accessed through performance problems page

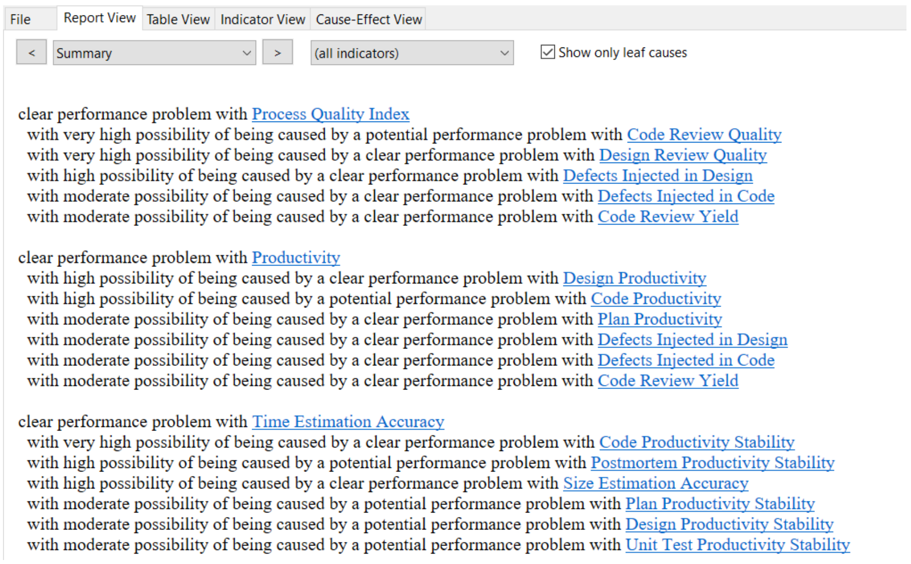
4.5.2 Case study

In this section will be presented a case study from a personal performance data file of a PSP trainee at Tecnológico de Monterrey, Mexico, in order to compare the results obtained in the ProcessPAIR tool and the WebProcessPAIR, as well as present the recommendations made by WebProcessPAIR users for the problems encountered.

Case study – St01

ProcessPAIR performance problems results:

In Figure 42 is shown the report view of ProcessPAIR tool for the data file of case study St01.



clear performance problem with Process Quality Index
  with very high possibility of being caused by a potential performance problem with Code Review Quality
  with very high possibility of being caused by a clear performance problem with Design Review Quality
  with high possibility of being caused by a clear performance problem with Defects Injected in Design
  with moderate possibility of being caused by a clear performance problem with Defects Injected in Code
  with moderate possibility of being caused by a clear performance problem with Code Review Yield

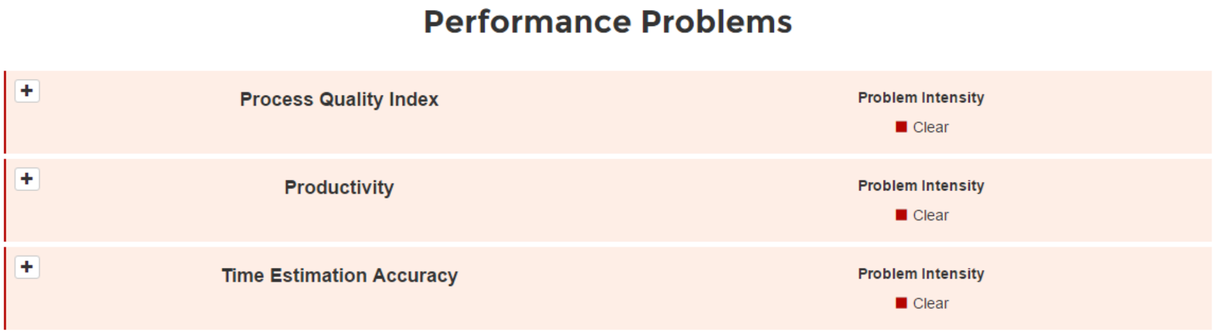
clear performance problem with Productivity
  with high possibility of being caused by a clear performance problem with Design Productivity
  with high possibility of being caused by a potential performance problem with Code Productivity
  with moderate possibility of being caused by a clear performance problem with Plan Productivity
  with moderate possibility of being caused by a clear performance problem with Defects Injected in Design
  with moderate possibility of being caused by a clear performance problem with Defects Injected in Code
  with moderate possibility of being caused by a clear performance problem with Code Review Yield

clear performance problem with Time Estimation Accuracy
  with very high possibility of being caused by a clear performance problem with Code Productivity Stability
  with high possibility of being caused by a potential performance problem with Postmortem Productivity Stability
  with high possibility of being caused by a clear performance problem with Size Estimation Accuracy
  with moderate possibility of being caused by a potential performance problem with Plan Productivity Stability
  with moderate possibility of being caused by a potential performance problem with Design Productivity Stability
  with moderate possibility of being caused by a potential performance problem with Unit Test Productivity Stability

Figure 42 – Case study: ProcessPAIR personal problems results (report view)

WebProcessPAIR performance problems results:

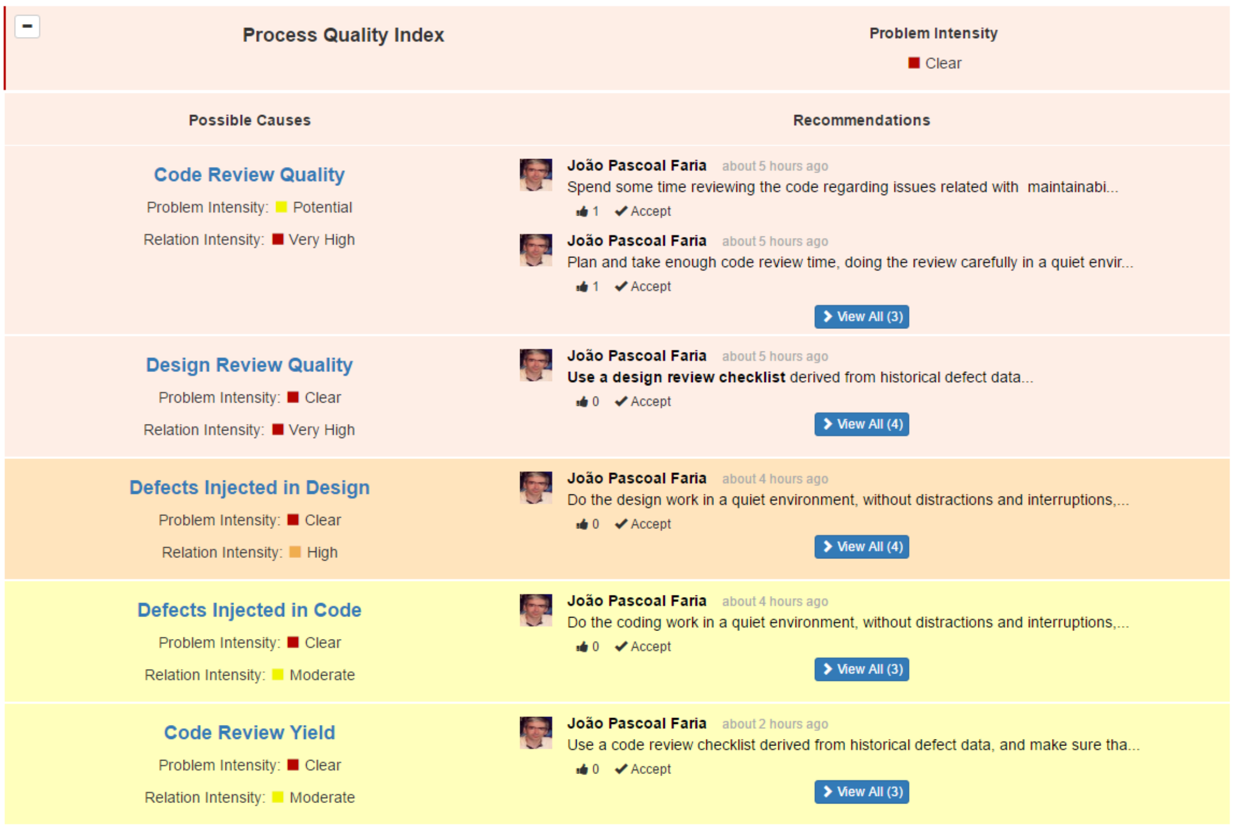
In Figure 43 is shown the personal problems page in WebProcessPAIR, presenting the results for the same personal file as above. A more detailed analysis is shown in Figures 44, 45 and 46, presenting all the top level indicators expanded, with the recommended improvement actions for each performance indicator.



Performance Problems

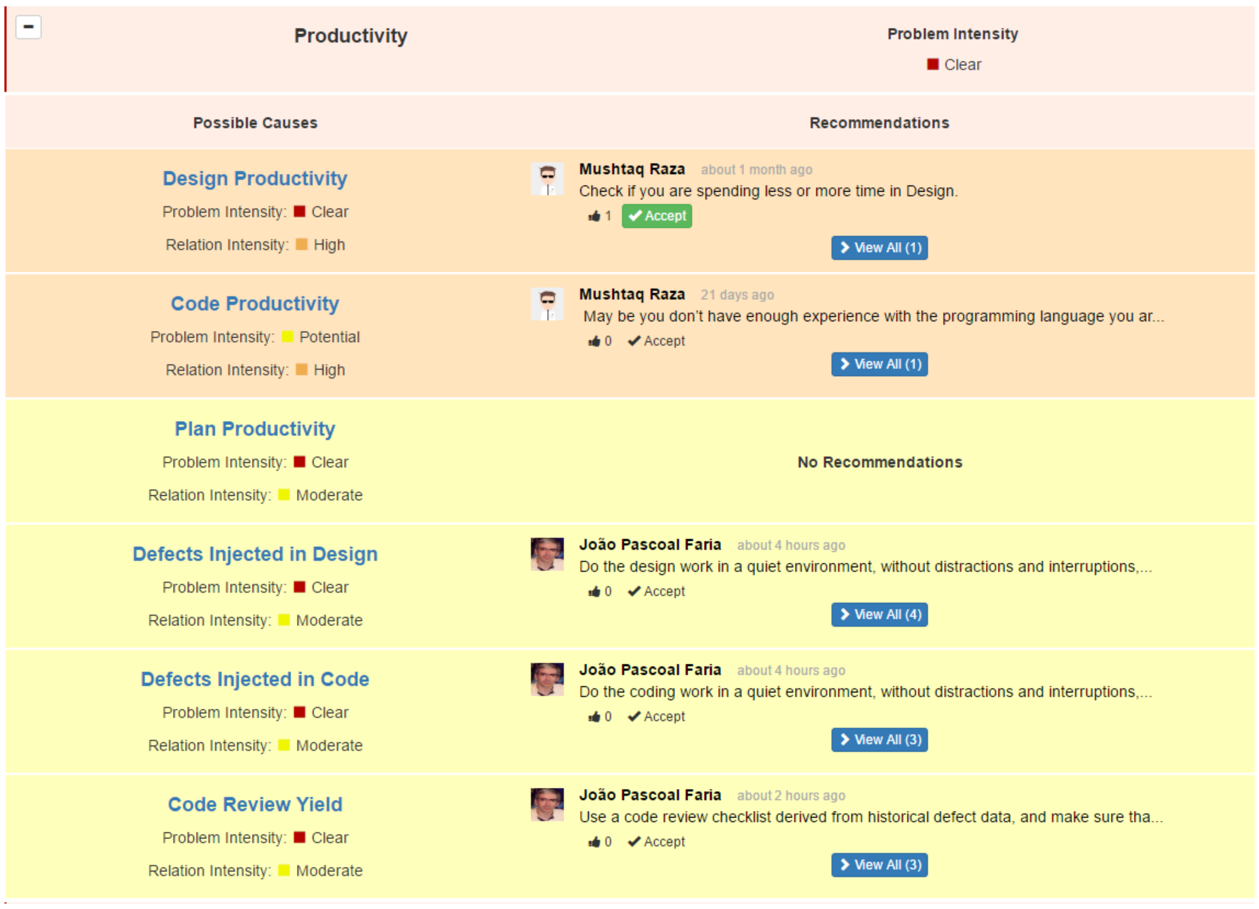
+ Process Quality Index	Problem Intensity 🟥 Clear
+ Productivity	Problem Intensity 🟥 Clear
+ Time Estimation Accuracy	Problem Intensity 🟥 Clear

Figure 43 - Case study: WebProcessPAIR personal problems results (top-level indicators)



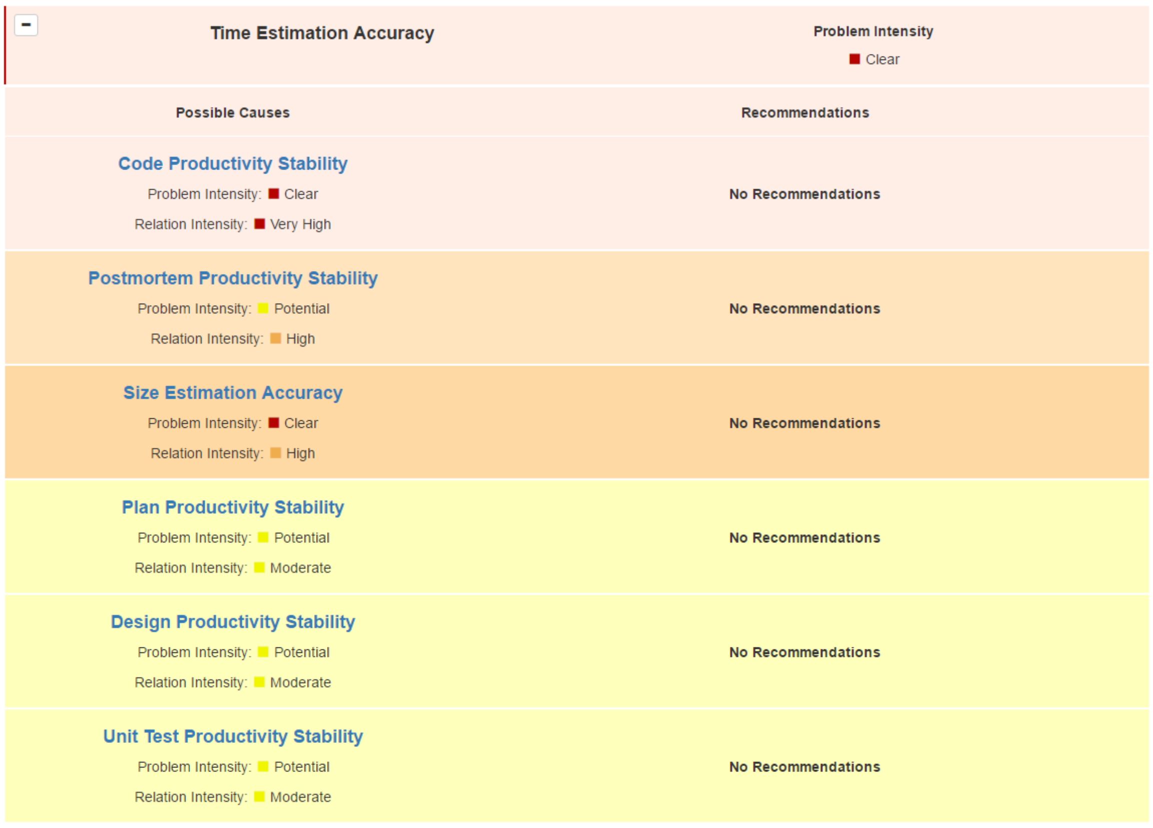
− Process Quality Index	Problem Intensity 🟥 Clear
Possible Causes	**Recommendations**
Code Review Quality Problem Intensity: 🟨 Potential Relation Intensity: 🟥 Very High	**João Pascoal Faria** about 5 hours ago Spend some time reviewing the code regarding issues related with maintainabi... 1 ✓ Accept **João Pascoal Faria** about 5 hours ago Plan and take enough code review time, doing the review carefully in a quiet envir... 1 ✓ Accept View All (3)
Design Review Quality Problem Intensity: 🟥 Clear Relation Intensity: 🟥 Very High	**João Pascoal Faria** about 5 hours ago **Use a design review checklist** derived from historical defect data... 0 ✓ Accept View All (4)
Defects Injected in Design Problem Intensity: 🟥 Clear Relation Intensity: 🟧 High	**João Pascoal Faria** about 4 hours ago Do the design work in a quiet environment, without distractions and interruptions,... 0 ✓ Accept View All (4)
Defects Injected in Code Problem Intensity: 🟥 Clear Relation Intensity: 🟨 Moderate	**João Pascoal Faria** about 4 hours ago Do the coding work in a quiet environment, without distractions and interruptions,... 0 ✓ Accept View All (3)
Code Review Yield Problem Intensity: 🟥 Clear Relation Intensity: 🟨 Moderate	**João Pascoal Faria** about 2 hours ago Use a code review checklist derived from historical defect data, and make sure tha... 0 ✓ Accept View All (3)

Figure 44 - Case study: WebProcessPAIR personal problems results (expanded first top-level indicator)



Productivity	Problem Intensity 🟥 Clear
Possible Causes	**Recommendations**
Design Productivity Problem Intensity: 🟥 Clear Relation Intensity: 🟧 High	**Mushtaq Raza** about 1 month ago Check if you are spending less or more time in Design. 1 ✓ Accept · View All (1)
Code Productivity Problem Intensity: 🟨 Potential Relation Intensity: 🟧 High	**Mushtaq Raza** 21 days ago May be you don't have enough experience with the programming language you ar... 0 ✓ Accept · View All (1)
Plan Productivity Problem Intensity: 🟥 Clear Relation Intensity: 🟨 Moderate	No Recommendations
Defects Injected in Design Problem Intensity: 🟥 Clear Relation Intensity: 🟨 Moderate	**João Pascoal Faria** about 4 hours ago Do the design work in a quiet environment, without distractions and interruptions,... 0 ✓ Accept · View All (4)
Defects Injected in Code Problem Intensity: 🟥 Clear Relation Intensity: 🟨 Moderate	**João Pascoal Faria** about 4 hours ago Do the coding work in a quiet environment, without distractions and interruptions,... 0 ✓ Accept · View All (3)
Code Review Yield Problem Intensity: 🟥 Clear Relation Intensity: 🟨 Moderate	**João Pascoal Faria** about 2 hours ago Use a code review checklist derived from historical defect data, and make sure tha... 0 ✓ Accept · View All (3)

Figure 45 - Case study: WebProcessPAIR personal problems results (expanded second top-level indicator)



Time Estimation Accuracy	Problem Intensity 🟥 Clear
Possible Causes	**Recommendations**
Code Productivity Stability Problem Intensity: 🟥 Clear Relation Intensity: 🟥 Very High	No Recommendations
Postmortem Productivity Stability Problem Intensity: 🟨 Potential Relation Intensity: 🟧 High	No Recommendations
Size Estimation Accuracy Problem Intensity: 🟥 Clear Relation Intensity: 🟧 High	No Recommendations
Plan Productivity Stability Problem Intensity: 🟨 Potential Relation Intensity: 🟨 Moderate	No Recommendations
Design Productivity Stability Problem Intensity: 🟨 Potential Relation Intensity: 🟨 Moderate	No Recommendations
Unit Test Productivity Stability Problem Intensity: 🟨 Potential Relation Intensity: 🟨 Moderate	No Recommendations

Figure 46 - Case study: WebProcessPAIR personal problems results (expanded third top-level indicator)

In this case study it's possible to check that the same problem list was presented in ProcessPAIR (Figure 42) and WebProcessPAIR (Figures 43/44/45/46) and also the same problem intensity and relation intensity ("problem intensity" in WebProcessPAIR = "performance problem with" in ProcessPAIR; "relation intensity" in WebProcessPAIR = "with … possibility of being caused by" in ProcessPAIR). In addition, it's presented a list of improvement actions, made by WebProcessPAIR users, for the encountered problems. At the current moment, the performance indicator "Time Estimation Accuracy" and his children have no recommendations.

Chapter 5

Conclusion and Future Work

5.1 Conclusions

This dissertation had as main objective the development of the website WebProcessPAIR, incorporating the ProcessPAIR tool, along with a recommendation system of improvement actions for exposed problems. In this report, firstly it was presented an approach about the contextualization of the theme of the dissertation and about the expected objectives. Afterwards, it was discussed the background and state of the art on the topics of software process improvement, crowdsourcing and recommendation systems, which represented the essential bases for implementing the proposed objectives.

The topic about software process improvement was important to analyse in detail the previous ProcessPAIR tool. In addition, the topic of crowdsourcing shows us that this is a concept that brings many benefits to the development of systems such as the WebProcessPAIR, which together with recommendation systems allows the recommendation of improvement actions through the expert knowledge of a community, with the aim of solving problems. On recommendation systems, was also presented what type of recommendation was used in WebProcessPAIR.

For a more specific approach on the design and implementation, were presented the conceptual model, database structure, MVC components and the gems installed, which show more clearly what is the structure of WebProcessPAIR, as well as what is the architecture and what technologies are used. For explaining the types of users in WebProcessPAIR and what role they take in it, was presented a diagram of use cases and a website navigation map. Finally, was explained the integration process with ProcessPAIR tool, and presented the acceptance tests for validating the proposed objectives.

For a better understanding of the usage of WebProcessPAIR, were presented a detailed guide with usage instructions of the application, as well as the current performance indicators and improvement actions loaded on the website. In the conclusion of the section WebProcessPAIR usage, was made an analysis of two case studies, where was used two different data files from two students, in ProcessPAIR and WebProcessPAIR, and compared the results.

The objectives proposed in this dissertation were fully completed with positive results. The results obtained in WebProcessPAIR are in accordance with the ProcessPAIR tool, as well as the recommendation system is working flawless. This way, it’s now possible for users to automatically analyse their performance data, identify performance problems, and get recommended improvement actions for their problems. The next objective in WebProcessPAIR, is to get more users and improvement actions, in order to help other software developers improve their project quality.

5.2 Future Work

In the future work the objective is to extend the WebProcessPAIR application with Scraim, allowing Scraim users to analyse their personal performance and get recommended actions on possible problems.

References

[1] Raza, M., Faria, J. A Model for Analyzing Performance Problems and Root Causes in the Personal Software Process. *Journal of Software: Evolution and Process*, John Wiley & Sons, (2015), 1-4.

[2] Process PAIR. (2016, 1 February). Available from http://blogs.fe.up.pt/processpair/ (2006); accessed 1 February 2016.

[3] Raza, M., Faria, J., Salazar, R. Experimental Assessment of the ProcessPAIR Tool for Automated Performance Analysis. *Conference'10, Month 1–2*, (2016), 1-2.

[4] Duarte, C., Faria, J. and Raza, M. PSP PAIR: Automated Personal Software Process Performance Analysis and Improvement Recommendation. *8th International Conference on the Quality of Information and Communication Technologies* (2012).

[5] Howe, J. Crowdsourcing: A Definition. Available from: http://crowdsourcing.typepad.com/cs/2006/06/crowdsourcing_a.html (2015). Accessed on 27 January 2016.

[6] Howe, J. Crowdsourcing: Why the Power of the Crowd Is Driving the Future of Business. *Crown Publishing Group*, 1^{a} ed (2008).

[7] Brabham, D. Crowdsourcing: A Model for Leveraging Online Communities. Retrieved 20 January 2016 from: https://dbrabham.files.wordpress.com/2011/03/brabham_handbook_crowdsourcing.pdf.

[8] Geiger, D. Crowdsourcing Information Systems – A Systems Theory Perspective. *Proceedings of the 22nd Australasian Conference on Information Systems,* (2011), 5-6.

[9] Estélles, E. Types of crowdsourcing iniciatives. Crowdsourcing Blog Web site. Retrieved 20 January 2016 from: http://www.crowdsourcing-blog.org/tipos-de-iniciativas-de-crowdsourcing/?lang=en.

[10] Deci, E. and Ryan, M. Intrinsic and Extrinsic Motivations: Classic Definitions and New Directions. *Contemporary Educational Psychology*, 25 (2000).

[11] Hossain, M. Crowdsourcing: Activities, incentives and users' motivations to participate. *International Conference on Innovation, Management and Technology Research,* (2012), 21-22.

[12] Zou, L., Ke, W., Zhang, J. and Kee, K. User creativity in crowdsourcing community: from extrinsic motivation perspective. *PACIS 2014 Proceedings.* Paper 45 (2014).

[13] Barbosa, A. Recomendação de Boas Práticas no Crowdsourcing – um Caso de Estudo (Master's dissertation). Faculdade de Engenharia da Universidade do Porto, Porto (2015).

[14] Leimeister, J., Huber, M., Ulrich, B, and Krcmar, H. Leveraging Crowdsourcing: Activation-Supporting Components for IT-Based Ideas Competition. *Journal of Management Information Systems*, 26 (1) (2009), 197-224.

[15] Guittard, C. and Schenk, E. Towards a characterization of crowdsourcing practices. *Journal of innovation economics*, 1 (7) (2011), 93-107.

[16] Engage Comunities in participative projects: Crowdsourcing. Available from https://howto.informationactivism.org/content/engage-communities-participative-projects-crowdsourcing; accessed on 20 January 2016

[17] Yuan, F., Liang, J. & Xue, Z. *Crowdsourcing: Today and Tomorrow* (Degree of Bachelor of Science). Worcester Polytechnic Institute. Massachusetts (n.d).

[18] Chen, H. A Fashion Recommendation System Based on The Wisdom of Crowds. (Master's thesis). Faculty of Science and Engineering, Waseda University (2013).

[19] Bugara, S., Jureta, I., Faulkner, S. and Herssens, C. Knowledge-Based Recommendation Systems: A Survey. *International Journal of Intelligent Information Technologies*, Volume 10, Issue 2 (2014).

[20] Geiger, D. Personalized Task Recommendation in Crowdsourcing Systems. *Springer*, 65 (2014).

[21] Stack Overflow. Available from http://stackoverflow.com/ (2016); accessed 28 January 2016.

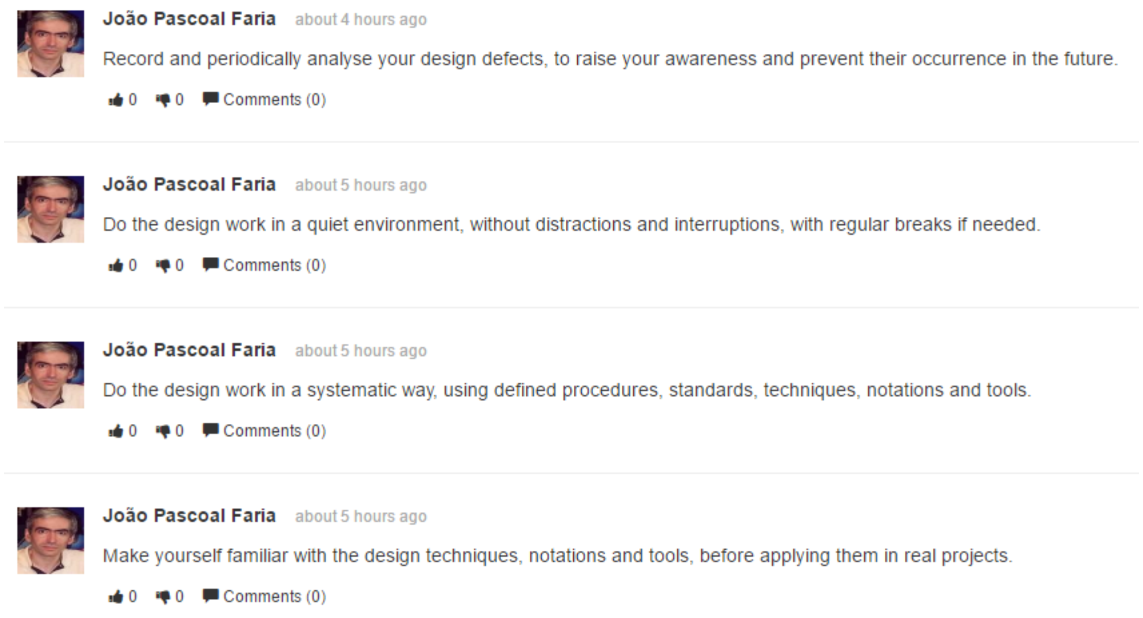
[22] The Success of Stack Exchange: Crowdsourcing + Reputation Systems. Available from https://permut.wordpress.com/2012/05/03/the-success-of-stack-exchange-crowdsourcing-reputation-systems/; accessed on 28 January 2016.

[23] WebMD. Available from http://symptoms.webmd.com/#introView (2016); accessed 28 January 2016.

Appendix A

In appendix A will be presented all the current improvement actions created in WebProcessPAIR. Each of the following figures represent a performance indicator with the improvement actions.

Defects injected in design:



João Pascoal Faria about 4 hours ago

Record and periodically analyse your design defects, to raise your awareness and prevent their occurrence in the future.

0 0 Comments (0)

João Pascoal Faria about 5 hours ago

Do the design work in a quiet environment, without distractions and interruptions, with regular breaks if needed.

0 0 Comments (0)

João Pascoal Faria about 5 hours ago

Do the design work in a systematic way, using defined procedures, standards, techniques, notations and tools.

0 0 Comments (0)

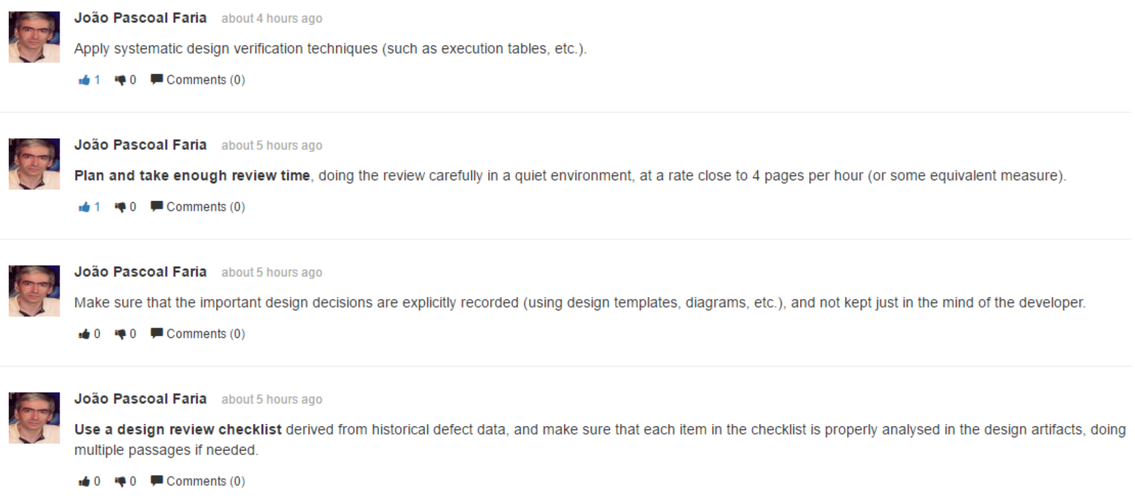
João Pascoal Faria about 5 hours ago

Make yourself familiar with the design techniques, notations and tools, before applying them in real projects.

0 0 Comments (0)

Figure 47 - Current improvement actions for PI "Defects injected in design"

Design review quality:



João Pascoal Faria about 4 hours ago

Apply systematic design verification techniques (such as execution tables, etc.).

1 0 Comments (0)

João Pascoal Faria about 5 hours ago

Plan and take enough review time, doing the review carefully in a quiet environment, at a rate close to 4 pages per hour (or some equivalent measure).

1 0 Comments (0)

João Pascoal Faria about 5 hours ago

Make sure that the important design decisions are explicitly recorded (using design templates, diagrams, etc.), and not kept just in the mind of the developer.

0 0 Comments (0)

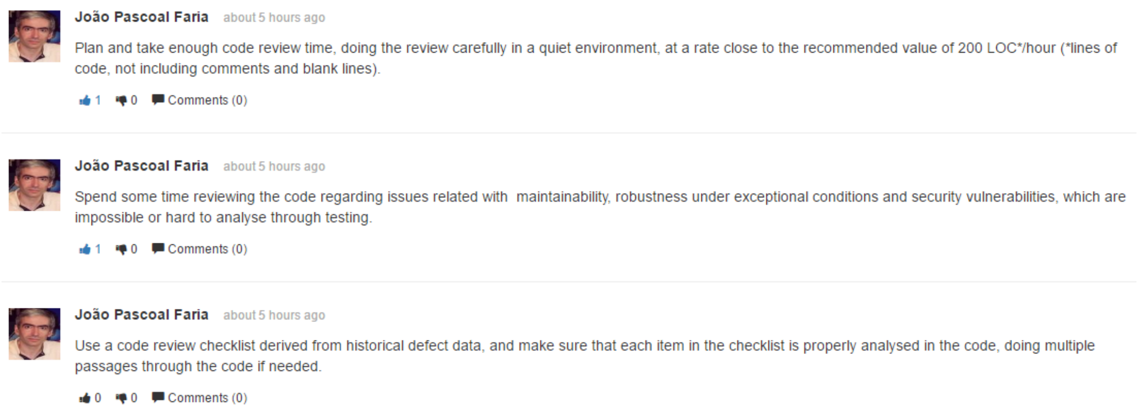
João Pascoal Faria about 5 hours ago

Use a design review checklist derived from historical defect data, and make sure that each item in the checklist is properly analysed in the design artifacts, doing multiple passages if needed.

0 0 Comments (0)

Figure 48 - Current improvement actions for PI "Design review quality"

Code review quality:



João Pascoal Faria about 5 hours ago

Plan and take enough code review time, doing the review carefully in a quiet environment, at a rate close to the recommended value of 200 LOC*/hour (*lines of code, not including comments and blank lines).

1 0 Comments (0)

João Pascoal Faria about 5 hours ago

Spend some time reviewing the code regarding issues related with maintainability, robustness under exceptional conditions and security vulnerabilities, which are impossible or hard to analyse through testing.

1 0 Comments (0)

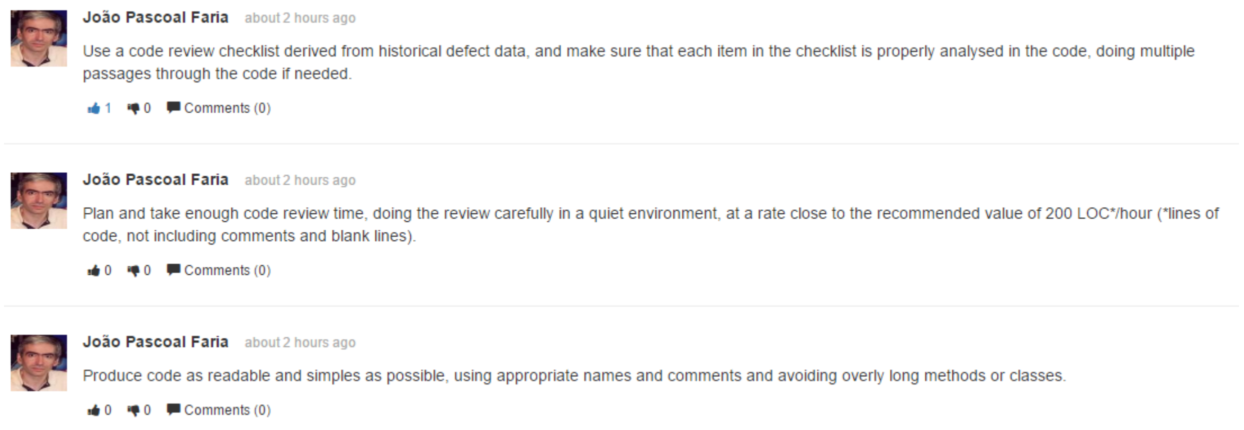
João Pascoal Faria about 5 hours ago

Use a code review checklist derived from historical defect data, and make sure that each item in the checklist is properly analysed in the code, doing multiple passages through the code if needed.

0 0 Comments (0)

Figure 49 - Current improvement actions for PI "Code review quality"

Code review yield:



João Pascoal Faria about 2 hours ago

Use a code review checklist derived from historical defect data, and make sure that each item in the checklist is properly analysed in the code, doing multiple passages through the code if needed.

👍 1 👎 0 Comments (0)

João Pascoal Faria about 2 hours ago

Plan and take enough code review time, doing the review carefully in a quiet environment, at a rate close to the recommended value of 200 LOC*/hour (*lines of code, not including comments and blank lines).

👍 0 👎 0 Comments (0)

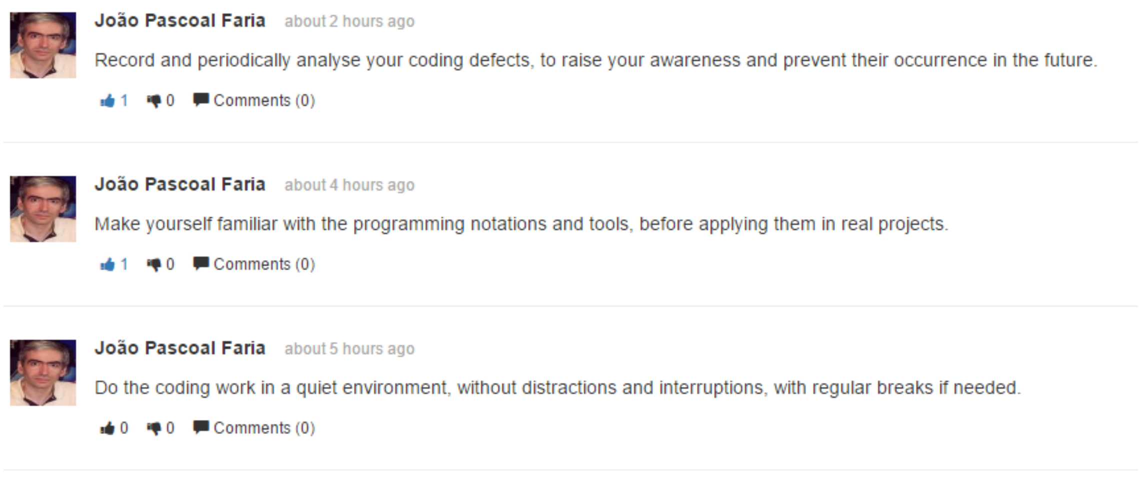
João Pascoal Faria about 2 hours ago

Produce code as readable and simples as possible, using appropriate names and comments and avoiding overly long methods or classes.

👍 0 👎 0 Comments (0)

Figure 50 - Current improvement actions for PI "Code review yield"

Defects injected in code:



João Pascoal Faria about 2 hours ago

Record and periodically analyse your coding defects, to raise your awareness and prevent their occurrence in the future.

👍 1 👎 0 Comments (0)

João Pascoal Faria about 4 hours ago

Make yourself familiar with the programming notations and tools, before applying them in real projects.

👍 1 👎 0 Comments (0)

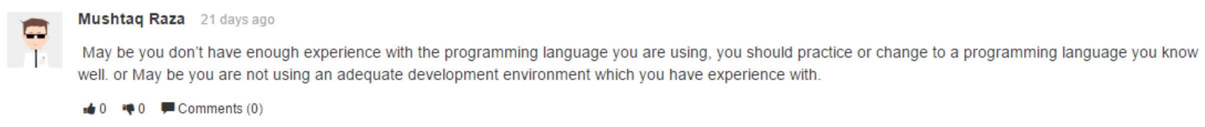
João Pascoal Faria about 5 hours ago

Do the coding work in a quiet environment, without distractions and interruptions, with regular breaks if needed.

👍 0 👎 0 Comments (0)

Figure 51 – Current improvement actions for PI “Defects injected in code”

Code productivity:



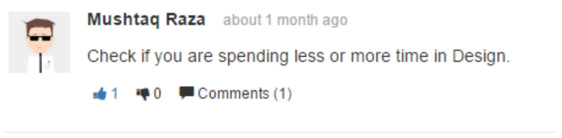
Mushtaq Raza 21 days ago

May be you don't have enough experience with the programming language you are using, you should practice or change to a programming language you know well. or May be you are not using an adequate development environment which you have experience with.

0 0 Comments (0)

Figure 52 - Current improvement actions for PI "Code productivity"

Design productivity:



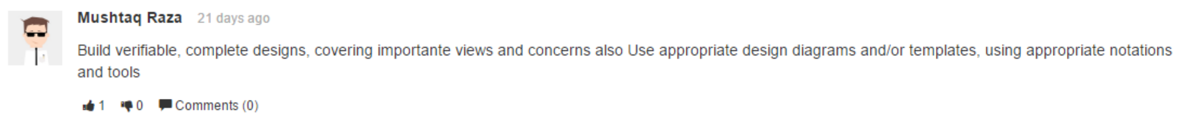
Mushtaq Raza about 1 month ago

Check if you are spending less or more time in Design.

1 0 Comments (1)

Figure 53 - Current improvement actions for PI "Design productivity"

Design quality:



Mushtaq Raza 21 days ago

Build verifiable, complete designs, covering importante views and concerns also Use appropriate design diagrams and/or templates, using appropriate notations and tools

1 0 Comments (0)

Figure 54 - Current improvement actions for PI "Design quality"