FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Tools to Support Practical Teaching of Software Engineering

João Tiago Barbosa Pinto

Dissertation Report
Integrated Master in Informatics and Computing Engineering
Advisor: João Carlos Pascoal de Faria (Aux. Professor)
Advisor: Katalin Balla (Budapesti Műszaki és Gazdaságtudományi Egyetem)

June 2010

Abstract

Software applications are becoming more and more important in today's society. That is why currently in software development; failures are not allowed because each error implies increased resources and costs. In this context the need for methodologies and practices that serve as a helper tool for software development arises.

Nowadays the job of a software engineer is to deliver high-quality software products according to agreed-upon resources and schedule. Since the global market is growing really fast, it is important to do an effective work.

Through the years only few organizations have met their commitments in relation to cost and schedule causing in this way serious business problems.

One sentence started to be used:"So what to do??". One of the answers came from Watts Humphrey. This software engineer became known as the father of software quality, being one of the creators of the Capability Maturity Model (CMM), Team Software Process (TSP) and Personal Software Process (PSP). The answer consisted in the creation of PSP that was designed to help software engineers do a good work and effective one.

PSP provides detailed estimating and planning methods, showing how engineers should track their performance against plans and explains how defined processes can guide their work.

This methodology started to be taught in a fifteen lecture course where the students had to complete ten programming exercises and five analysis exercises. After the solution of each exercise the instructor should run and verify a series of items from a defined checklist.

Through the years some universities started to apply this methodology in their own courses, being the teaching customized to their own needs. In this document this subject will be approached.

Since in the Faculty of Engineering of the University of Porto this teaching started in a customized way, now it is important to show the students all processes of PSP the same way as the original course.

PSP instructors usually have to spend thirty minutes for each student and programming exercise, because the checking of the several items takes a long time. A close analysis of these items shows that most of them can be automatically verified. So this document shows and explains the creation of a supporting tool called PSPChecker that allows us to save time and increase accuracy when evaluating the work of PSP students.

Users with PSPChecker can create customized processes or use the ones already defined.

Until this moment there weren't any available tools that could reproduce the same result.

In conclusion, PSPChecker offers a lot of functionalities like exporting result for several formats such as Excel and PDF and allows the creation of charts.

Resumo

As aplicações de software são cada vez mais importantes na sociedade e na rotina diária. É por isso que actualmente em desenvolvimento de software, as falhas não são permitidas porque cada erro implica o aumento dos recursos e custos. Neste contexto, surge a necessidade de criação de metodologias e práticas que sirvam como ferramenta auxiliar para o desenvolvimento de software.

Hoje em dia o trabalho de um engenheiro de software é desenvolver e entregar produtos de alta qualidade tendo em conta os recursos que possui e a sua calendarização. Isto é importante pois actualmente o mercado global está a crescer de forma rápida, logo é importante fazer um trabalho eficaz.

Apenas algumas organizações ao longo dos anos conseguiram cumprir os seus compromissos em relação ao orçamento e ao cronograma causando desta forma problemas de negócios sérios.

Uma frase passou a ser utilizada: "Então o que fazer?". Uma das respostas veio de Watts Humphrey. Este engenheiro de software tornou-se conhecido como o pai da qualidade de software, sendo um dos criadores do Capability Maturity Model (CMM), Team Software Process (TSP) e Personal Software Process (PSP). A resposta consistia na criação do PSP que foi projectado para ajudar engenheiros de software no desenvolvimento de bons projectos e de forma eficaz.

PSP fornece informações detalhadas sobre a estimativa e métodos de planeamento, mostrando desta forma como os engenheiros devem acompanhar o seu desempenho face a esses planos e explica como definir os processos de modo a orientarem os seus trabalhos.

Esta metodologia começou a ser ensinada num curso de quinze aulas onde os alunos tinham de completar dez exercícios de programação e cinco exercícios de análise. Após a resolução do exercício o instrutor deve verificar uma série de itens de uma checklist definida.

Ao longo dos anos, muitas universidades começaram a aplicar esta metodologia nos seus próprios cursos, sendo o ensino personalizado às suas próprias necessidades. Neste documento, o assunto será abordado pois é um dos componentes de estudo para o desenvolvimento da ferramenta.

Uma vez que na Faculdade de Engenharia da Universidade do Porto, este ensinamento começou de forma personalizada mas agora é importante ensinar aos alunos todos os processos do PSP da mesma forma que o seu ensino no curso original.

Desta forma, cada professor teria que despender trinta minutos para cada aluno, pois a verificação dos diversos itens é demorada, quando essa verificação pode ser automática para a maioria dos itens. Portanto, este documento mostra e explica a criação de uma ferramenta de apoio chamado PSPChecker que permite poupar tempo e aumentar precisão.

Utilizadores com PSPChecker podem criar processos personalizados ou usar os já definidos.

Até ao momento não houve quaisquer ferramentas disponíveis que permitissem reproduzir o mesmo resultado.

Para conclusão PSPChecker oferece uma série de funcionalidades como exportação para diversos formatos, como Excel e PDF e permite a criação de gráficos.

Acknowledgements

I would like to thank my supervisor, Professor João Carlos Pascoal de Faria, from Faculty of Engineering of the University of Porto, for his guidance and support.

A special thank to my co-supervisor Professor Katalin Balla, from Budapest University of Technology and Economics, for her inputs, enthusiasm and her invaluable perceptiveness in the discussions we had.

João Tiago Barbosa Pinto

Contents

List of Figures

List of Tables

Abbreviations

CMM - Capability Maturity Model

CMU – Carnegie Mellon University

IEC - International Electrotechnical Commission

IEEE - Institute of Electrical and Electronics Engineers

ISO - International Organization for Standardization

LOC - Lines Of Code

PDF - Portable Document Format

PIP - Process Improvement Proposal

RUP – Rational Unified Process

SEI – Software Engineering Institute

TSP – Team Software Process

PSP – Personal Software Process

UML - Unified Modeling Language

Chapter 1 - Introduction

Currently in development of software, failures are not allowed because each error implies increased resources and costs. In this context there is the need for the creation of methodologies and practices that serve as a helper tool for software development. In this field, one of the driving forces for good practices was Watts Humphrey [1] a software engineer that in the 1960s stood at IBM for having been the first team manager to launch a licensed version of software. This software engineer became known as the father of software quality, being one of the creators of the CMM [2], Team Software Process (TSP) [3] and Personal Software Process (PSP) [4] and Carnegie Mellon University Lecturer in the Department of Computer Science.

This work discusses the last component indicated in the preceding subparagraph (PSP): its definition and important points to retain, PSP education in an academic level, creating helping tools to help software engineering education (automating checklists and information) and finally a possible application of this tool in enterprise-level.

1.1 Context and motivation

The initial study was carried out in the context of the curricular unit "Preparation for dissertation" at FEUP (Faculty of Engineering of the University of Porto) of the Integrated Master in Informatics and Computing Engineering. One part of the project continued at FEUP and the other part in BME (Budapesti Műszaki és Gazdaságtudományi Egyetem) under an Erasmus Protocol.

PSP is a software development process and is also a method of teaching best practices.

Both of these previous points highlighted the main reason for PSP being added as a curricular unit of software engineering.

The design and concept of a tool (PSPChecker) allows teachers to reduce time for evaluation and feedback. Another important point is the exchange of current PSP files produced locally by the ones used and available at CMU/SEI in order to improve the skills of several students.

One of the interesting points of PSPChecker is the possibility of obtaining information that understands the improvement in the development of a project.

As an overcome of each objective, personal capacity related with PSP will improve for a consistent application of these capacities in several companies.

Thus, one of the objectives of this work is to develop personal skills in the area of software quality and process improvement.

1.2 Objectives

The main objectives of this dissertation work are:

1. To capture and systematize the implicit knowledge that is held by experts when accessing processes already defined (despite verification checklists, the criteria used to check each item are not explicit);
2. To facilitate software engineering education using PSP locally;
3. To conceive and develop an automated verification tool of PSP adherence (in its various variants), based on verification checklists so often used by instructors, that integrates with existing tools and formats;
4. To demonstrate that the verification of software development processes can be significantly automated;
5. If appropriate, to propose amendments to the verification checklists that can also determine process adherence, but are easier to make automatic;
6. The tool should be able to create graphics, export to different formats and be able to create customized processes.

An intermediate goal, but no less important, is the study of PSP and all practices involved.

Finally the approach on this work should take into account various contexts, so the tool architecture should be as generic as possible.

To better understand the entire process described above there is a figure that illustrates in summary what is intended.

Here the users described are teachers and students who access SEI / CMU.

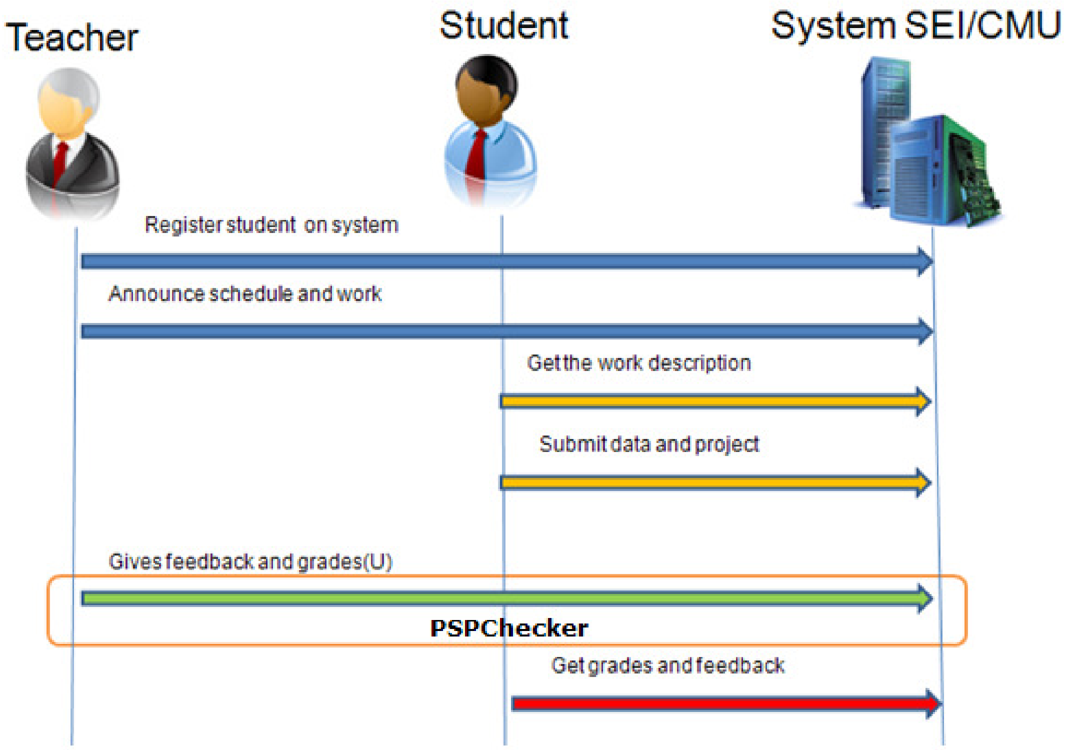


Figure 1 – Information flow

1.3 Methodology and review of the literature

The first step done was the analysis of the state of the art including literature review, methodology and interviews.

To better understand the whole project there should be a focus analysis of the entire search around the state of the art.

PSPChecker will have an essential advantage to the proper functioning of the PSP practices, which incorporates a novelty for the remaining existing tools in the market.

The literature review was, without doubt, one of the vital issues of research work as brings to who is running the research project the necessary knowledge to understand issues that will be studied. Moreover, it allows the beginning of the draft and will allow a discussion on the topic, since the bibliographic review is based in the information universe in question.

The second stage of the project consisted in defining the main requirements, architecture and list of accepting tests. After all these definitions the realization of the tool started regarding all previous points achieved.

1.3 Dissertation’s Structure

The structure used for the dissertation is the following:

Chapter 2 describes the study of the state of the art on this subject. This includes initial study and understanding of PSP. It is an important chapter because it allows an improved notion of PSP and how it has been used in an academic level.

Chapter 3 refers mainly the process of automatic verification of items. There will be a description of the method of verification for each item and finally there will be a presentation of the statistics related with the final results of verification.

Then in **Chapter 4** the tool will be described. All the requirements, architecture as well all the tests used in the development of PSPChecker will be described.

.**Chapter 5** presents the experimentation where PSPChecker working for different scenarios is showed.

Chapter 6 presents the conclusions and future work.

Appendix A describes the personal experience with the PSP.

Chapter 2 - Understanding PSP

This chapter describes the study that was made to understand PSP and its components. Some of the points discussed here will be: introduction to PSP, teaching of PSP in an academic level and finally some of the tools used to make the processes possible.

2.1 Introduction to PSP

PSP has come up with the need to improve the performance of processes in small organizations and reduced-size projects, because improving the skills of members of a team allows better and more efficient projects and teams. [5]

The main strategy in PSP is motivating each engineer to approach effective development. This adoption is made progressively and calculated. In each step in this progression a new method is introduced along with adapted exercises.

PSP can be considered as a discipline that provides a structured framework for improving and developing personal skills. PSP will accelerate the learning of these skills.

One of the main goals of PSP is being an improvement process, which means the way an engineer has to change in order to improve his work. The next figure shows how to improve the quality of a work. This figure belongs was defined in *"A Discipline for Software Engineering"* from Watts Humphrey.

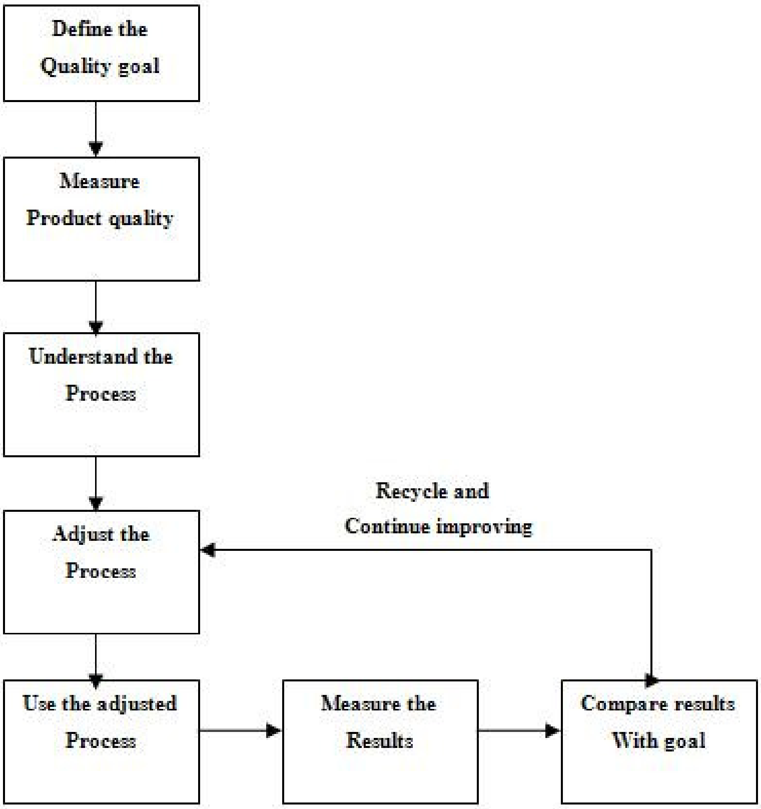


Figure 2- Improvement process

To better understand all the steps to improve the quality of a work, there are some examples that explain how an engineer should behave to obtain better results and improve skills. [1]

- **Define the quality goal** – Student becomes a better software engineer completing a process or product
- **Measure Product quality** – Instructor and student see what went bad in a product or process.
- **Understand the process** – Instructor observed what had been done and suggests what should change
- **Adjust the process** – Instructor suggests some changes
- **Use the adjusted process**
- **Measure the results** – Student checks if changes improved the work
- **Compare results with goal** – Student checks if results are consistent with the main goal
- **Recycle and continue improving** –It is important for a student since it is a continuous process improvement.

2.2 PSP Strategy and components

The PSP strategy is based on the following assumptions:

- We should define, measure and track a project, to better understand its performance.
- Using the steps above allows us to choose more easily the best suited methods to the project.
- Choosing the correct methods increases the performance of teams and projects.

PSP like other incremental methodologies also has a Framework, which is divided in levels, from a lower maturity level (level 0) to a higher level or optimization (level 3). To better understand these levels there is a diagram and a consistent description of each of these levels.

The next figure was created by PSP experts to simplify and to show the different levels of PSP.

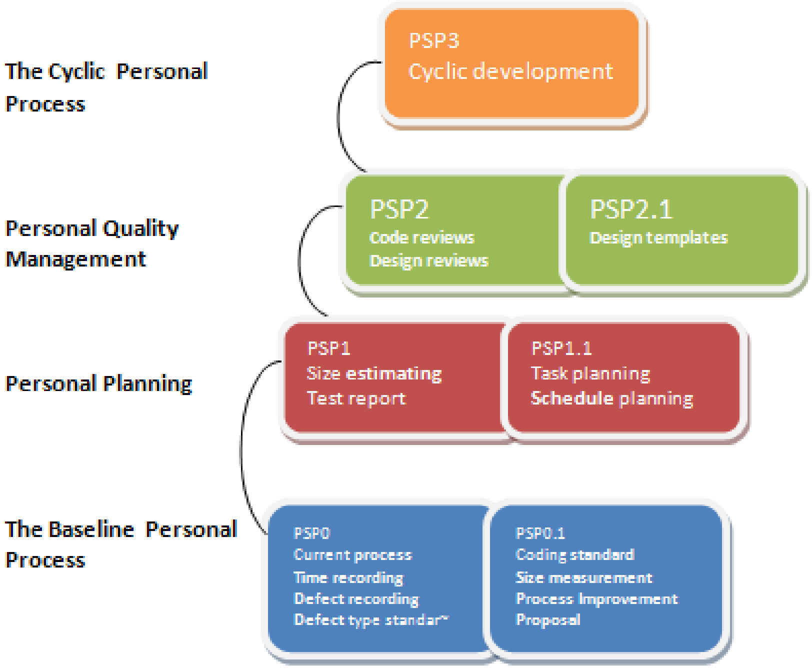


Figure 3– Evolution of PSP phases

2.2.1 PSP 0 – Baseline Personal Process

This initial phase establishes basic measures that must be followed by an engineer as well as a description of a possible report model. These initial data enable a consistent basis for measuring progress and improvements. PSP level 0 is basically a description of the essential points in the drafting of a software project.

In conclusion, the main objective is the incorporation of habits and team incentive to register their performance data.

As the image shows in level 0 there are several points that are important to a better understanding of the current point of a project.

Time Recording

In PSP, recording time is very important because it allows to manage time effectively. In the beginning of a project it is always necessary to try and make realistic plans, but for this to happen it is important to track the way time is spent.

It is important to document the initial plans and compare them with the actual ones. All this information should be classified into major categories, allowing to record in a standard way the time spent doing each activity.

This information is crucial to improve the quality of projects. But for an accurate time recording and planning it is important to have some aspects in mind:

- Always use as time measure minutes and not hours, because students rarely work for a full hour.
- Measure interruption time accurately or else a random number will be added in all data, making it difficult to plan and manage time. Interruptions may waste time and break train of thought leading to inefficiency and error.
- For each project it is important to have an independent time recording log.
- Use a standard Time Recording Log which contains some details like: date of an activity, start and stop time of that activity, interruption time during that activity, description of the activity and comments.
- Finally one forgets to record time it is important to make an estimative as fast as possible, being in this way close to the actual time of a project.

Defect Recording

Defects usually cause problems to programs users and can be very expensive to find and fix. A software defect indicates that something in the product is wrong. Since defects are caused by human errors it is important that engineers promptly find and repair defects that they inject.

The best way of having a good management of defects is to understand them. For this to happen it is essential to gather defect data, analyze and determine how to better prevent, find and repair these defects. The best way of identifying and removing defects is by personal code reviews.

For a better organization of these data PSP provides a Defect Recording Log that helps gathering defect data. This log contains several fields that help understand defects like: type of defect, its description, in which phase of project it was injected and in which phase it was removed and the time that took to fix the defect. It is important in the fulfilling of this document that the description is detailed enough to later understand it. In the end of each project it is important to analyze this data and see in which phases and which kind of defect types are more common.

Finally the use of a log allows the improvement of an engineer's programming skills, the reduction of the number of defects in programs, the save of time, the save of costs and finally a responsible way to do a job.

Defect Type Standard

In the book “Introduction to PSP”[1] the author writes that when analyzing defects it is important to divide them into main categories. This allows the engineer to quickly understand which categories cause more trouble and focus on their removal and prevention.

These categories were described with a type number, name and description like as it is possible to see in the next table.

Defect Types

Type Number	Name	Description
10	Documentation	Comments, messages
20	Syntax	Spelling, punctuation, typos, instruction formats
30	Build, Package	Change management, library, version control
40	Assignment	Declaration, duplicate names, scope and limits. Includes missing template parameters and erroneous method signatures
50	Interface	Procedure calls and references, I/O, user formats + incorrect error messages (syntax, semantics) + call violates precondition
60	Checking	Error messages, inadequate checks + decisions
70	Data	Structure, content
80	Function	Logic, pointers, loops, recursion, computation, function defects. Includes missing features in DLD (methods and fields) and using the wrong function,
90	System	Configuration, timing, memory
100	Environment	Design, compile, test, or other support system problems

Table 1 - Defect Type Standard

PSP 0.1 adds to PSP0 encoding standards, size measures and process improvement proposals (PIP). The main objective of this sub level is to provide a better understanding of the principles for measuring size.

Coding Standard

An accepted set of coding practices should be defined, which can serve as model to a project. Usually this standard is used as a guide when writing the source code. In PSP, like in other standards specify the way source code is formatted, what statements go on separate text lines, the use of code comments and other practices.

Size Measurement

PSP uses as size measurement lines of code (LOC) because it is generally applicable to most of programming languages. Size measurement is important to estimate time. By estimating how many LOC the program will likely require and calculating the number of minutes per LOC development in past projects, it is possible to estimate the total development time.

PIP Form

The PIP allows registering problems, experiences and suggestions for process improvement.

2.2.2 PSP 1- Personal Planning

PSP 1 uses the planning steps from level 0. The information here is size, resource estimation and reporting tests.

Test Report

One of the forms incorporated in PSP is a test report where students can specify the tests that will be performed to deliver a product correctly functioning. All the time spent in this phase should be counted and recorded in the Time Recording Log.

2.2.3 PSP 2 – Personal Quality Management

As the name suggests, the main objective of this level is the ability to develop software with high quality. This is possible, showing how to use the defects that are constantly made to reduce the number of stages of compilation and tests. Because an error can usually be constant in its appearance, having a record of these errors will let fix them more quickly.

This previous point is important because it helps reduce defects rates and understand its causes and consequences.

According to the PSP manual [1] for experienced programmers the number of defects in the compilation and tests must be in a range between 50 and 150 per thousand lines of code (KLOC). These values are presented to software that has not been reviewed or inspected.

As the project is directed to students, the emergence of high rates of defects will be frequent.

At this level there is the incorporation of design and code reviews. These code reviews allow to detect defects as soon as possible, being a most effective resolution.

The design process is inserted at the sub level 2.1 and indicates what a final project must illustrate, taking into account the requirements, documentation, development and testing.

2.2.4 PSP 3 – Cyclic Personal Process

The great advantage of this level is its use in large projects. This level is for high size projects that must be subdivided into several "components". I.e. after being developed these components will be integrated into a large component that is the project. Here the components are integrated iteratively with testing and compilation. Level 3 is preferred when a project has thousands of lines of code.

Finally the main objective of this level is the introduction of scaling principles. That is ensuring the quality of a successive draft form taking into account their cycles.

2.2.5 PSP Process Flow

Processes are typically composed of scripts, templates, standards and forms. On the next figure we can see the process flow of PSP with all the respective components mentioned before. [6]

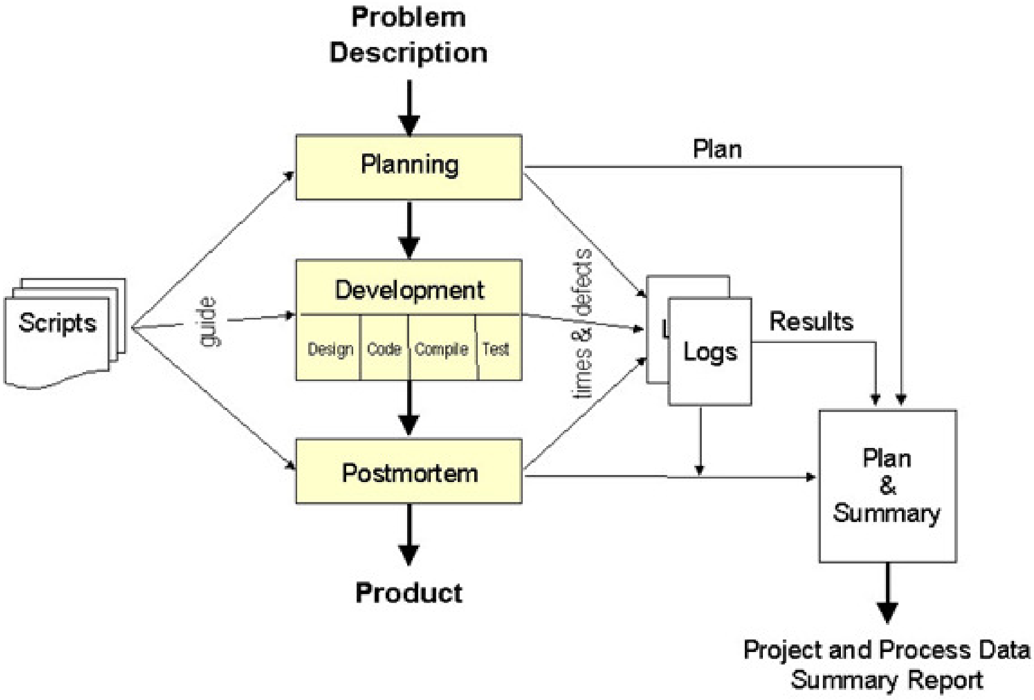


Figure 4- PSP Process Flow

2.2.6 PSP Process Script

A process script is a set of steps the process users should follow. The phases of the PSP are described bellow and summarized in the next table (In this case for PSP level 1).

Purpose:	**To guide you in developing small programs.**
Inputs Required	- The problem description - PSP Project Plan Summary form - A copy of the Code Review Checklist - Actual size and time data for previous programs - Time Recording Log - Defect Recording Log
Planning	- Obtain a description of the program functions. - Estimate the Max., Min., and total LOC required. - Determine the Minutes/LOC. - Calculate the Max., Min., and total development times. - Enter the plan data in the Project Plan Summary form. - Record the planning time in the Time Recording Log.
Design	- Design the program. - Record the design in the specified format. - Record design time in the Time Recording Log.
Code	- Implement the design. - Use a standard format for entering the code. - Record coding time in the Time Recording Log.
Code review	- Completely review the source code. - Follow the code review script and checklist. - Fix and record every defect found. - Record review time in the Time Recording Log.
Compile	- Compile the program. - Fix and record all defects found. - Record compiles time in the Time Recording Log.
Test	- Test the program. - Fix and record all defects found. - Record testing time in the Time Recording Log.
Postmortem	- Complete the Project Plan Summary form with actual time, size, and defect data. - Review the defect data and update the code review checklist. - Record postmortem time in the Time Recording Log.
Exit Criteria	- A thoroughly tested program - A properly documented design - A completed Code Review Checklist - A complete program listing - A completed Project Plan Summary - Completed time and defect logs

Table 2 – PSP Process Script

2.3 Teaching of PSP in Universities

After the creation of PSP Watts Humphrey said that the "headline goal was to help a graduated to become a better software engineer". The great advantage of PSP is the constant improvement of technical capabilities that enable greater productivity and increase the quality of software.

Humphrey designed the PSP so that it can be taught in a discipline of a half semester being taught to students that had already graduated. During this discipline each student will improve the ability to estimate and measure task times in project. Each level attained in PSP students gained new knowledge, skills and techniques to improve the quality of the software produced. To a greater understanding of the point at which student was, checklists, scripts and templates have been created, which help students achieve the final objectives. The great advantage of the above is that material could be changed by each student case these changes could explain weaknesses and strengths of the student.

It was previously described in this context that there is a need to investigate the PSP in FEUP education as well as browse other examples of success in world universities to truly understand whether there are advantages in teaching this process.

2.3.1 FEUP - Portugal

Regarding PSP education in Software engineering only basic notions are focused. The tool that students use was developed by local teachers and consists of an Excel file that writes times records for each phase of the project among other options.

The major objective is the use of FEUP tools provided by SEI/CMU which are better to understand and more complex. Hence, the creation of support tools can reduce teaching time and guarantee a better support to the students' work.

2.3.2 UMEA University – Sweden

This University **[7]** introduces concepts of PSP in a programming discipline (C++) and in an optional subject in the software engineering course. In the programming subject a simple version of PSP was developed, which allowed to plan, monitor and review everything that had been done by a student. The teaching of PSP process held over 2 lessons of 45 minutes each. After teaching this process, other students were asked to launch a project where the choice of PSP as a work methodology was optional. Only 6 students were chosen, from these 6 students none of them had actually notable improvements in software development. After this experience, teachers decided to abandon the teaching of this process.

The second case had a different outcome, since during the theoretical lessons concepts of PSP were presented and in practical lessons there was a development of tools that allowed the analysis of data and the collection of information to be processed by the PSP process. The final results of this method enabled students to understand the various problems that PSP can help solve.

2.3.3 UTAH University – USA

PSP in university **[7]** education stems to several years already. In this University teachers have to teach a light version of PSP processes to first year students and this education takes place over 2 semesters. During the first half basic notions concerning resource estimation and tracking of projects are taught, while the second half is focused on solution (reduction, prevention and design-code reviews). Along the lessons teachers provide students with statistics and select exams issues related to the process studied, as well.

As a result of this methodology of work, students feel that they are better prepared as software engineers, enabling them to obtain research fellowships more easily.

2.3.4 Montana Tech University – USA

Like the previous University **[7]**, PSP takes place over 2 semesters. Before being taught any theory, all students have a high level of competence for programming, because they were delivered some disciplines of data structure and algorithmic before. Something that teachers noticed was that all pupils started showing some resistance to learning something new, but in the end all these understood the importance of this process. The tool used here was developed locally and consisted of docs.

2.3.5 University Of Zagreb – Croatia

The implementation of PSP methodologies was incorporated, in the software engineering course of the Department of telecommunications. The aim of embedding these practices was the need to demonstrate to students that a project can have great chances of success if it follows a methodology. **[8]**

The various students have just some basic knowledge projects, so here was an adaptation of PSP concepts. These changes have led to remove various points proposed by PSP and restructure the definition of design and review aspects. Like most of the universities the tool was developed locally, making use of C++ and Excel.

Results of PSP in 5 universities

Universities	Teaching environment	PSP Coverage	Support tool	Comments
FEUP - Portugal	• 3rd year – Software engineering curricular unit	Simplified version of PSP (individual and pairs). Use of 4 projects and use of simplified checklists	Locally developed	• Great difficulty to apply PSP (time required of teachers), leading to a PSP only partial. • Missing space in curriculum
UMEA University – Sweden	• 2nd year – Programming Discipline C++	Use of modified PSP level 1	Locally developed	• Development of PSP Tool advantageous to obtain knowledge

	• Optional subject of SE			• Optional use of PSP inefficient
UTAH University – USA	• 1st year	simplified version of PSP	Locally developed	• implementation of PSP classes without problems • Students who worked in pairs had best results
Montana Tech University – USA	• 2nd year	simplified version of PSP	Locally developed	• Students resistance in the beginning, but happy with the final result
University of Zagreb - Croatia	• Without references	simplified version of PSP	Locally developed	• Reduce defects and time ratio of drafting projects • Incorporating concepts positively

Table 3– Results of academic study of PSP

The results of this conclusion are the following:

- There is a greater implementation of PSP in American universities, unlike the rest of the world.
- In the beginning there is always a rejection on the part of students, because they have to change habits in relation to planning and developing software.
- In almost all cases the educational support tool for PSP is developed locally.
- Generally, this process is delivered at the same time in disciplines of programming or optional subjects.
- The PSP education structure is often changed to simplify some components present in the process, and thus teaching becomes more accessible.
- A medium-term result that arises from the PSP is positive in education, allowing students to obtain competences and skills.

2.4 PSP Support Tools

When a project is started it is important to identify the various existing applications in terms of academic and business. Since the beginning of this project it was decided to use the tool developed by SEI/CMU, due to the possibility of integrating the current tool with the support tool to gain value in the recognition of projects and thus provide a greater recognition to FEUP.

Although the tool to be used in classes has already been chosen, it is essential to analyze the various existing applications on the market, because it is possible to identify strengths used by several applications that may not be used by the chosen tool.

Common sense is that various tools that support teaching of PSP help students improve capacities and skills but do not manage to solve problems, specifically tools that help spotlighting "what to do" but not "how to".

Existing tools are currently based in Word, Excel, Access or even simple applications in various programming languages. These tools are simple and essential to begin PSP projects.

After reading several articles all authors agree that all existing tools only assist in implementing certain parts of the PSP, not deepening the PSP study.

To create a viable tool to PSP these tools must possess the following functional and non functional requirements:

- Customization
- Selection of current phase of the project (PSP 0, 1, 2, 2.1)
- Follow-up times and stages
- Support for planning and estimating projects
- Identification of errors
- Privacy data

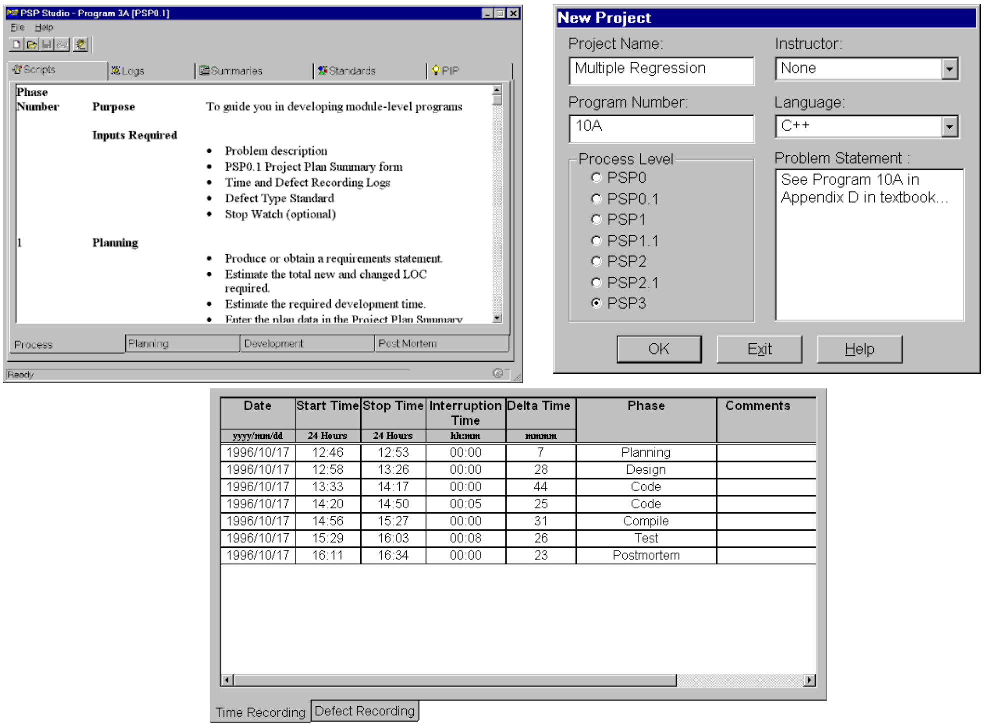
Another interesting point in the analysis tools is that there are manuals and automated tools in the calculation of various points of PSP processes. Unfortunately, as Humphrey says in his book it is impossible to automate completely a PSP tool, because there are certain requirements that software cannot judge and assess.

After this introduction, a series of applications and their characteristics are presented.

2.4.1 PSPStudio

This tool was designed by the University of Tennessee **[9]**. It supports all levels of PSP and contains an associated data base (application to run in parallel). Other interesting features are:

- Software has templates for quality and test plans.
- Simplified usage tool, where usability is a key point.
- Implementation of customizable Design Review Checklist (DRC) and code Review Checklist (CRC).
- Customization of test documentation
- Manual creation of State diagrams
- Illustration of important points in each PSP level



Phase Number | Purpose | To guide you in developing module-level programs

Inputs Required

- Problem description
- PSP0.1 Project Plan Summary form
- Time and Defect Recording Logs
- Defect Type Standard
- Stop Watch (optional)

1 Planning

- Produce or obtain a requirements statement.
- Estimate the total new and changed LOC required.
- Estimate the required development time.
- Enter the plan data in the Project Plan Summary

Project Name: Multiple Regression

Instructor: None

Program Number: 10A

Language: C++

Process Level
○ PSP0
○ PSP0.1
○ PSP1
○ PSP1.1
○ PSP2
○ PSP2.1
● PSP3

Problem Statement : See Program 10A in Appendix D in textbook...

Date	Start Time	Stop Time	Interruption Time	Delta Time	Phase	Comments
yyyy/mm/dd	24 Hours	24 Hours	hh:mm	mmmm		
1996/10/17	12:46	12:53	00:00	7	Planning	
1996/10/17	12:58	13:26	00:00	28	Design	
1996/10/17	13:33	14:17	00:00	44	Code	
1996/10/17	14:20	14:50	00:05	25	Code	
1996/10/17	14:56	15:27	00:00	31	Compile	
1996/10/17	15:29	16:03	00:08	26	Test	
1996/10/17	16:11	16:34	00:00	23	Postmortem	

Figure 5 – PSPStudio application screenshots

2.4.2 PSP-EAT

It is the use of restructured Excel for the various levels of PSP. In general all processes taught to students that involve calculations are performed manually. **[10]**

2.4.3 PSP-DROPS (PSP Data Repository and Presentation System)

It is an automated online tool developed at Embry-Riddle Aeronautical University. Using PSP-Drops, students can receive online instructions relating to registration data, which reduces the workload on the management and analysis. More importantly, the results analyzed can be displayed graphically. **[11]**

2.4.4 PSP STUDENT WORKBOOK

It is the official software for teaching the PSP in SEI/CMU.

This tool is very comprehensive and complex as a student reaches higher maturity levels. It has an appealing and simple interface where several supporting information exists.

The tool was developed in Access. As mentioned earlier, this tool has been chosen for the teaching of PSP in the course of Software Engineering.

2.4.5 Conclusion

After analyzing various points already described, the conclusions were clear.

After studying the impact of PSP in academic level, it was possible to identify that PSP teaching is not intrinsic to the teaching of software engineering and most of the available tools are made in the local environment.

Of several tools none of them supports the automatic verification of checklists, just another opportunity for this project to be successful, because a generalized PSPChecker version will be able to be integrated into the existing ones.

Finally, during the analysis of the state of the art a set of points emerged, which were formulated in order to respond and identify future problems and points to be applied.

- Advantages of using PSP (Organization level)
- Ability to check data quality

Chapter 3 - Automatic Verification of Items

This chapter is one of the most important to the success of this project. Since the beginning it was important to understand which kind of information each item should provide to solve and create an effective verification tool. For each level of PSP there are different lists of items to check. These items can range from a simple check to the database to see if is fulfilled or not to a more complicated one where it is needed to make calculations and compare with time of different tables in the database.

The initial study of these items consisted in solving little projects with respective times and data, fulfilling PSP Students Workbook with the corresponding data. After that this data was exported to access file in format .mdb where all the tables of the database were analyzed.

Some of the items were easy to understand, while for some others it was necessary the intervention of some experts to better perceive what was asked.

The items of a checklist for each level of PSP are divided for the amount of templates that should be fulfilled and some points like general information or consistency checks.

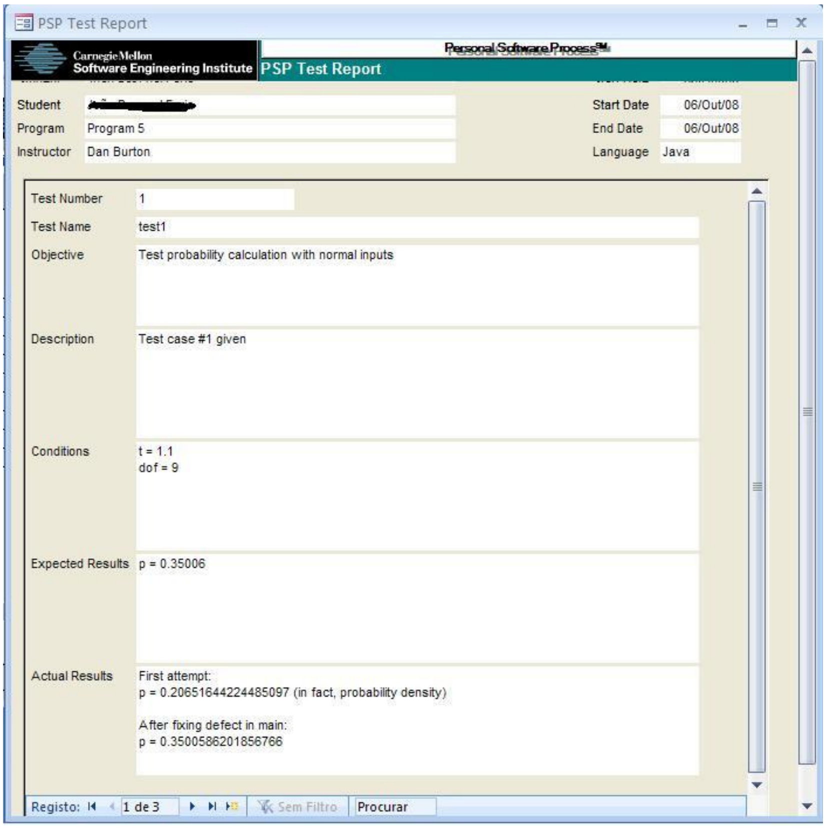
As a result of all the study a table was created. This table contains information about the name of Main items and sub-items as well if is possible to verify automatically and the method to do that. To conclude, it should be referred that there are 94 items in the junction of all PSP levels.

During the presentation some figures will appear describing the forms used by PSP Students Workbook.

Assignment Package		
Item/Process	**Automatic Verification?**	**Method of verification**
PSP0 Project Planning Summary	Yes	Verifies if all manual data is fulfilled.
PSP1 Project Planning Summary	Yes	Verifies if all manual data is fulfilled.
PSP2 Project Planning Summary	Yes	Verifies if all manual data is fulfilled.
PSP2 Design Review Checklist	Yes	It checks if there is a file in Data folder named "`Code_Review_Checklist.doc`" since this document is usually in Doc Format.
PSP2.1 Design Review Checklist	Yes	It checks if there is a file in Data folder named "`PSP2_1_Design_Review_Checklist.doc`" since this document is usually in Doc Format.
PSP2.1 Project Planning Summary	Yes	Verifies if all manual data is fulfilled.
Operational Scenario Template	Yes	It checks if there is a file in Data folder named "`Operational_Scenario_Template.doc`"

		since this document is usually in Doc Format.
Functional Specification Template	Yes	It checks if there is a file in Data folder named “Functional_Scenario_Template.doc” since this document is usually in Doc Format.
State Specification Template	Yes	It checks if there is a file in Data folder named “State_Specification_Template.doc” since this document is usually in Doc Format.
Logic Specification Template	Yes	It checks if there is a file in Data folder named “Logic_Specification_Template.doc” since this document is usually in Doc Format.
Code Review Checklist	Yes	It checks if there is a file in Data folder named “Code_Review_Checklist.doc” since this document is usually in Doc Format.
Task Planning Template	Yes	Verifies if all manual data is fulfilled.
Schedule Planning Template	Yes	Verifies if all manual data is fulfilled.
Test Report	Yes	Verifies if at least one test is reported.
PIP Form	Yes	Verifies if the most important fields are fulfilled (Problem Description and Proposal description).
Size Estimating Template	Yes	Verifies if the most important fields are fulfilled (At least one Part is added).
PROBE Worksheet	Yes	Verifies if the most important fields are fulfilled
Time Recording Log	Yes	Checks if all fields are fulfilled.
Defect Recording Log	Yes	Verifies if at least one defect is reported.
Source Program Listing	No	Checks if there are files with denomination related with programming language.
Test Results	No	It is not possible to verify in an automatic way.

Program and Test Results		
Item/Process	**Automatic Verification?**	**Method of verification**
The program appears to be workable.	No	It is not possible to verify in an automatic way.
All required tests have been run.	No	It is not possible to verify in an automatic way.
The actual output is correct for each test.	No	It is not possible to verify in an automatic way.



Student	[redacted]	Start Date	06/Out/08
Program	Program 5	End Date	06/Out/08
Instructor	Dan Burton	Language	Java

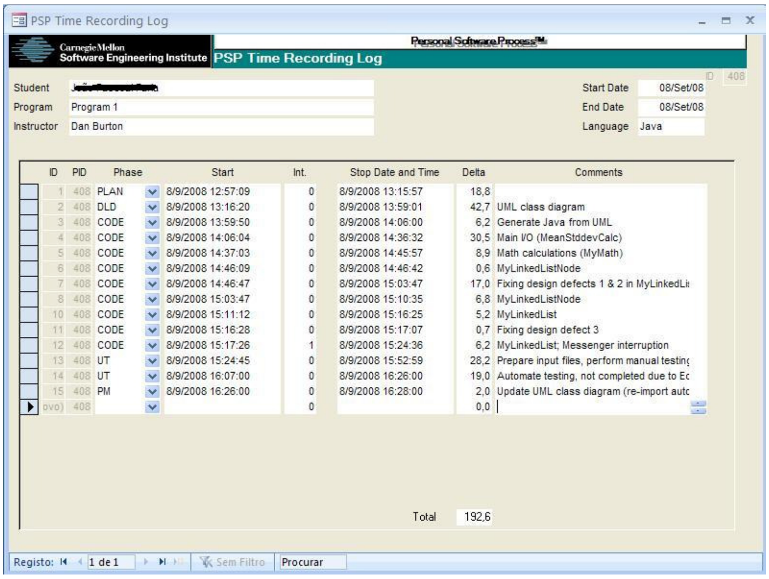
Test Number	1
Test Name	test1
Objective	Test probability calculation with normal inputs
Description	Test case #1 given

Conditions	t = 1.1
	dof = 9

Expected Results	p = 0.35006
Actual Results	First attempt: p = 0.20651644224485097 (in fact, probability density) After fixing defect in main: p = 0.3500586201856766

Figure 6 – PSP Test Report Example

Test Report Template		
Item/Process	**Automatic Verification?**	**Method of verification**
Planned and actual results are included for all tests.	Yes	It check if all data related to planned and actual results are included in the template
All information to repeat the tests is provided.	Yes	Can only verify if this information is provided
The test report is complete.	Yes	Can only verify if all information is provided
All other tests are properly planned and reported.	No	It is not possible to verify in an automatic way. For each test it is necessary to verify if it is valid or not and can only be done in a manual way.



PSP Time Recording Log

Student	[redacted]	Start Date	08/Set/08
Program	Program 1	End Date	08/Set/08
Instructor	Dan Burton	Language	Java

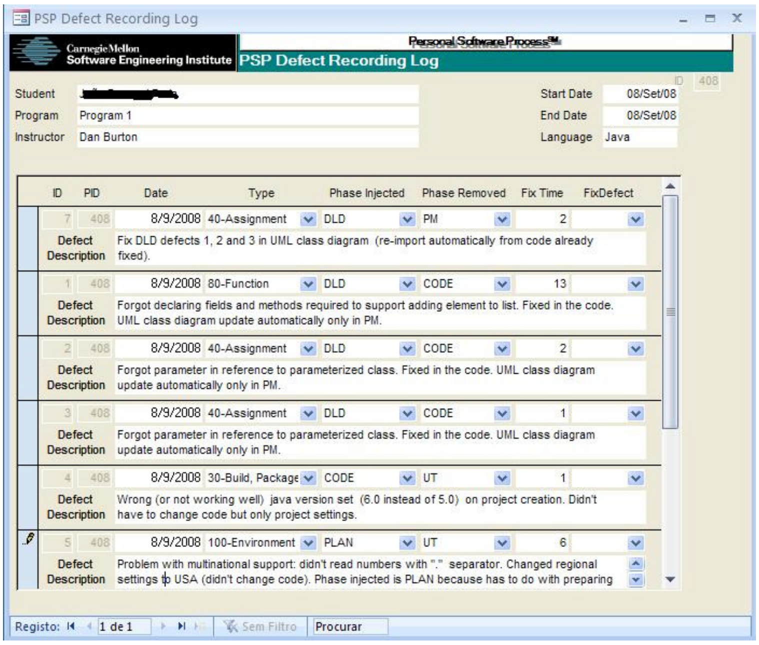
ID 408

ID	PID	Phase	Start	Int.	Stop Date and Time	Delta	Comments
1	408	PLAN	8/9/2008 12:57:09	0	8/9/2008 13:15:57	18,8	
2	408	DLD	8/9/2008 13:16:20	0	8/9/2008 13:59:01	42,7	UML class diagram
3	408	CODE	8/9/2008 13:59:50	0	8/9/2008 14:06:00	6,2	Generate Java from UML
4	408	CODE	8/9/2008 14:06:04	0	8/9/2008 14:36:32	30,5	Main I/O (MeanStddevCalc)
5	408	CODE	8/9/2008 14:37:03	0	8/9/2008 14:45:57	8,9	Math calculations (MyMath)
6	408	CODE	8/9/2008 14:46:09	0	8/9/2008 14:46:42	0,6	MyLinkedListNode
7	408	CODE	8/9/2008 14:46:47	0	8/9/2008 15:03:47	17,0	Fixing design defects 1 & 2 in MyLinkedLi
8	408	CODE	8/9/2008 15:03:47	0	8/9/2008 15:10:35	6,8	MyLinkedListNode
10	408	CODE	8/9/2008 15:11:12	0	8/9/2008 15:16:25	5,2	MyLinkedList
11	408	CODE	8/9/2008 15:16:28	0	8/9/2008 15:17:07	0,7	Fixing design defect 3
12	408	CODE	8/9/2008 15:17:26	1	8/9/2008 15:24:36	6,2	MyLinkedList; Messenger interruption
13	408	UT	8/9/2008 15:24:45	0	8/9/2008 15:52:59	28,2	Prepare input files, perform manual testin
14	408	UT	8/9/2008 16:07:00	0	8/9/2008 16:26:00	19,0	Automate testing, not completed due to Ec
15	408	PM	8/9/2008 16:26:00	0	8/9/2008 16:28:00	2,0	Update UML class diagram (re-import autc
ovo)	408			0		0,0	

Total 192,6

Figure 7 – PSP Time Recording Log Example

Time Log		
Item/Process	**Automatic Verification?**	**Method of verification**
Time data are entered for all process steps.	Yes	Checks if time data was fulfilled in database
Process steps are sequenced appropriately.	Yes	Checks if all process steps are sequenced, for example coding should never come before planning.
Time data are entered against the appropriate process step.	No	Checking is only available manually because this information should be analyzed in a detailed way.
Interrupt time is tracked appropriately.	Yes	Check if interruption time doesn't exceed the correspondent process interval of time
Time data are complete and reasonable.	Yes	Check if distribution of time during process steps is reasonable or not. In this item all the default values should be agreed with experts.
Times were recorded as the work was done.	Yes	Check if times weren´t copied between steps.



ID 408

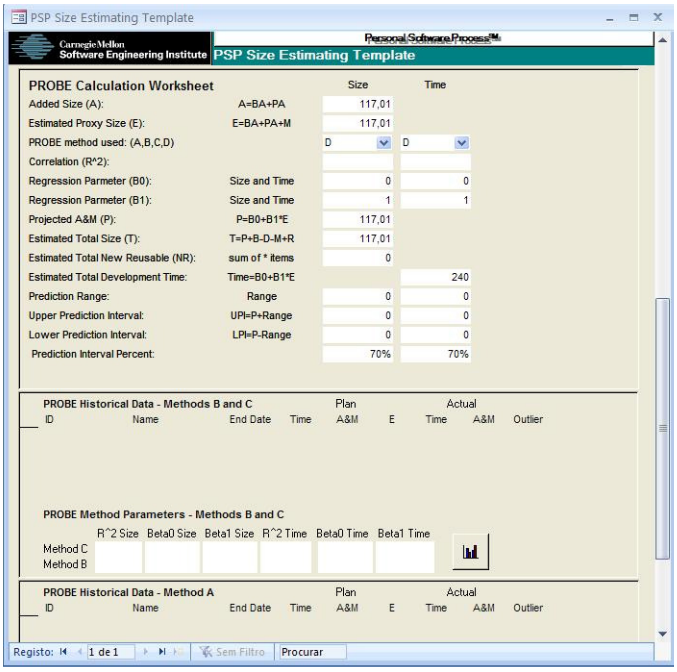
Student [redacted]
Program Program 1
Instructor Dan Burton

Start Date 08/Set/08
End Date 08/Set/08
Language Java

ID	PID	Date	Type	Phase Injected	Phase Removed	Fix Time	FixDefect	Defect Description
7	408	8/9/2008	40-Assignment	DLD	PM	2		Fix DLD defects 1, 2 and 3 in UML class diagram (re-import automatically from code already fixed).
1	408	8/9/2008	80-Function	DLD	CODE	13		Forgot declaring fields and methods required to support adding element to list. Fixed in the code. UML class diagram update automatically only in PM.
2	408	8/9/2008	40-Assignment	DLD	CODE	2		Forgot parameter in reference to parameterized class. Fixed in the code. UML class diagram update automatically only in PM.
3	408	8/9/2008	40-Assignment	DLD	CODE	1		Forgot parameter in reference to parameterized class. Fixed in the code. UML class diagram update automatically only in PM.
4	408	8/9/2008	30-Build, Package	CODE	UT	1		Wrong (or not working well) java version set (6.0 instead of 5.0) on project creation. Didn't have to change code but only project settings.
5	408	8/9/2008	100-Environment	PLAN	UT	6		Problem with multinational support: didn't read numbers with "." separator. Changed regional settings to USA (didn't change code). Phase injected is PLAN because has to do with preparing

Figure 8 – PSP Defect Recording Log

Defect Log		
Item/Process	Automatic Verification?	Method of verification
Every defect has all required data.	Yes	Check if all mandatory fields are fulfilled.
Defects were injected before removed.	Yes	Check if defects injected are removed in the same process step or in the next ones.
Every defect has a fix time.	Yes	Check if value fix time is fulfilled
Defects injected in compile and test have fix numbers.	Yes	Check if value fix time is fulfilled for defects injected in compile and test.
Defects are adequately described.	No	It is not possible to verify in an automatic way.
Defect types are consistent with description.	No	It is not possible to verify in an automatic way.
Defect types are consistent with phase injected.	No	It is not possible to verify in an automatic way.
Defect types are assigned consistently.	No	It is not possible to verify in an automatic way.


PSP Size Estimating Template
Carnegie Mellon
Software Engineering Institute
Personal Software Process℠
PSP Size Estimating Template
Registo: 1 de 1
Sem Filtro
Procurar

PROBE Calculation Worksheet

		Size	Time
Added Size (A):	A=BA+PA	117,01	
Estimated Proxy Size (E):	E=BA+PA+M	117,01	
PROBE method used: (A,B,C,D)		D	D
Correlation (R^2):			
Regression Parmeter (B0):	Size and Time	0	0
Regression Parmeter (B1):	Size and Time	1	1
Projected A&M (P):	P=B0+B1*E	117,01	
Estimated Total Size (T):	T=P+B-D-M+R	117,01	
Estimated Total New Reusable (NR):	sum of * items	0	
Estimated Total Development Time:	Time=B0+B1*E		240
Prediction Range:	Range	0	0
Upper Prediction Interval:	UPI=P+Range	0	0
Lower Prediction Interval:	LPI=P-Range	0	0
Prediction Interval Percent:		70%	70%

PROBE Historical Data - Methods B and C

			Plan			Actual		
ID	Name	End Date	Time	A&M	E	Time	A&M	Outlier

PROBE Method Parameters - Methods B and C

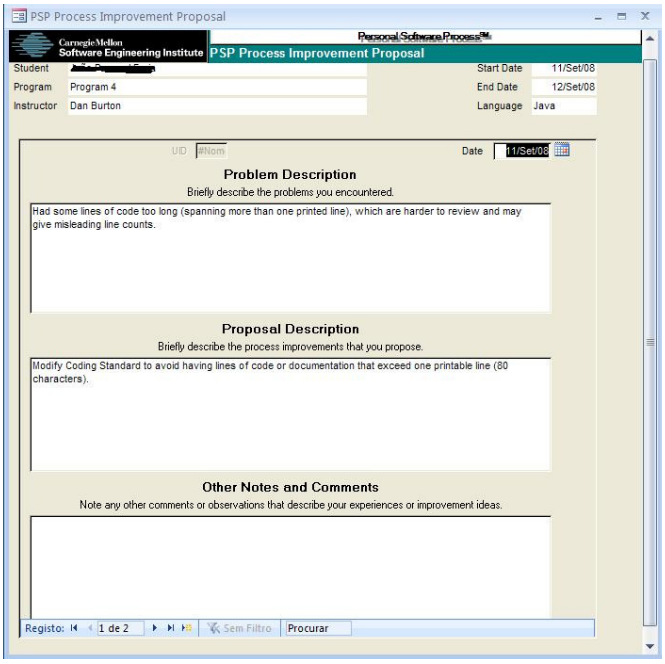
	R^2 Size	Beta0 Size	Beta1 Size	R^2 Time	Beta0 Time	Beta1 Time
Method C Method B						

PROBE Historical Data - Method A

			Plan			Actual		
ID	Name	End Date	Time	A&M	E	Time	A&M	Outlier

Figure 9 – Probe Calculations Example

SET& PROBE Worksheet		
Item/Process	**Automatic Verification?**	**Method of verification**
Plan and actual size data are complete and reasonable.	Yes	Check if information is fulfilled and check if difference between planned and actual is unusual.
The reuse and base measures are used correctly.	No	It is not possible to verify in an automatic way.
A suitable number of new parts are identified.	No	It is not possible verify in an automatic way, can be diferent for each program.
The item sizes are balanced around medium.	Yes	Check information and compare with default values obtained from experts.
The relative size data values are correct and based on historical data	N/A	The Student Workbook tool already calculates the values.
The appropriate PROBE method has been selected.	N/A	The Student Workbook tool already checks if the method is an appropriate one.
The item sizes are balanced around average.	Yes	Check information and compare with default values obtained from experts.
Appropriate historical data were used for the estimate.	N/A	The Student Workbook tool already calculates the values.
The parts additions calculations are correct.	N/A	The Student Workbook tool already calculates the values.
The PROBE worksheet calculations are correct.	N/A	The Student Workbook tool already calculates the values.



PSP Process Improvement Proposal
Carnegie Mellon
Software Engineering Institute
Personal Software Process℠

PSP Process Improvement Proposal

Student	[redacted]	Start Date	11/Set/08
Program	Program 4	End Date	12/Set/08
Instructor	Dan Burton	Language	Java

UID #Nom Date 11/Set/08

Problem Description

Briefly describe the problems you encountered.

Had some lines of code too long (spanning more than one printed line), which are harder to review and may give misleading line counts.

Proposal Description

Briefly describe the process improvements that you propose.

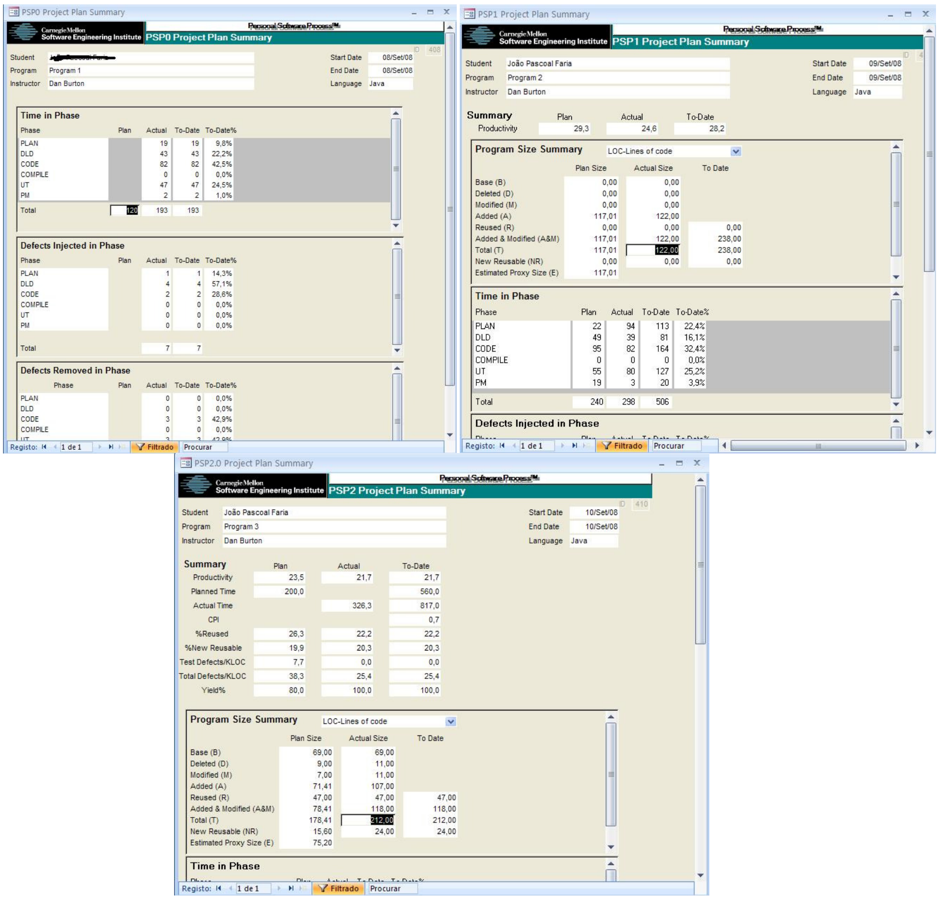
Modify Coding Standard to avoid having lines of code or documentation that exceed one printable line (80 characters).

Other Notes and Comments

Note any other comments or observations that describe your experiences or improvement ideas.

Figure 10 – PIP Form Example

PIP Form		
Item/Process	**Automatic Verification?**	**Method of verification**
The PIP form is completed.	Yes	Check if all important fields are fulfilled (Problem Description and Proposal Description)
The entries show insight and thought.	No	It is not possible to verify in an automatic way.
If yield was low, improvement actions are listed.	No	It is not possible verify in an automatic way.



PSP0 Project Plan Summary

Carnegie Mellon Software Engineering Institute
Personal Software Process℠

PSP0 Project Plan Summary

ID 408

Student	[redacted]	Start Date	08/Set/08
Program	Program 1	End Date	08/Set/08
Instructor	Dan Burton	Language	Java

Time in Phase

Phase	Plan	Actual	To-Date	To-Date%
PLAN		19	19	9,8%
DLD		43	43	22,2%
CODE		82	82	42,5%
COMPILE		0	0	0,0%
UT		47	47	24,5%
PM		2	2	1,0%
Total	120	193	193	

Defects Injected in Phase

Phase	Plan	Actual	To-Date	To-Date%
PLAN		1	1	14,3%
DLD		4	4	57,1%
CODE		2	2	28,6%
COMPILE		0	0	0,0%
UT		0	0	0,0%
PM		0	0	0,0%
Total		7	7	

Defects Removed in Phase

Phase	Plan	Actual	To-Date	To-Date%
PLAN		0	0	0,0%
DLD		0	0	0,0%
CODE		3	3	42,9%
COMPILE		0	0	0,0%

PSP1 Project Plan Summary

Carnegie Mellon Software Engineering Institute
Personal Software Process℠

PSP1 Project Plan Summary

Student	João Pascoal Faria	Start Date	09/Set/08
Program	Program 2	End Date	09/Set/08
Instructor	Dan Burton	Language	Java

Summary	Plan	Actual	To-Date
Productivity	29,3	24,6	28,2

Program Size Summary LOC-Lines of code

	Plan Size	Actual Size	To Date
Base (B)	0,00	0,00	
Deleted (D)	0,00	0,00	
Modified (M)	0,00	0,00	
Added (A)	117,01	122,00	
Reused (R)	0,00	0,00	0,00
Added & Modified (A&M)	117,01	122,00	238,00
Total (T)	117,01	122,00	238,00
New Reusable (NR)	0,00	0,00	0,00
Estimated Proxy Size (E)	117,01		

Time in Phase

Phase	Plan	Actual	To-Date	To-Date%
PLAN	22	94	113	22,4%
DLD	49	39	81	16,1%
CODE	95	82	164	32,4%
COMPILE	0	0	0	0,0%
UT	55	80	127	25,2%
PM	19	3	20	3,9%
Total	240	298	506	

Defects Injected in Phase

PSP2.0 Project Plan Summary

Carnegie Mellon Software Engineering Institute
Personal Software Process℠

PSP2 Project Plan Summary

ID 410

Student	João Pascoal Faria	Start Date	10/Set/08
Program	Program 3	End Date	10/Set/08
Instructor	Dan Burton	Language	Java

Summary	Plan	Actual	To-Date
Productivity	23,5	21,7	21,7
Planned Time	200,0		560,0
Actual Time		326,3	817,0
CPI			0,7
%Reused	26,3	22,2	22,2
%New Reusable	19,9	20,3	20,3
Test Defects/KLOC	7,7	0,0	0,0
Total Defects/KLOC	38,3	25,4	25,4
Yield%	80,0	100,0	100,0

Program Size Summary LOC-Lines of code

	Plan Size	Actual Size	To Date
Base (B)	69,00	69,00	
Deleted (D)	9,00	11,00	
Modified (M)	7,00	11,00	
Added (A)	71,41	107,00	
Reused (R)	47,00	47,00	47,00
Added & Modified (A&M)	78,41	118,00	118,00
Total (T)	178,41	212,00	212,00
New Reusable (NR)	15,60	24,00	24,00
Estimated Proxy Size (E)	75,20		

Time in Phase

Figure 11 - PSP Planning Summary examples

Planning Summary		
Item/Process	**Automatic Verification?**	**Method of verification**
Planned total time has been entered correctly.	Yes	Usually this value is automatic with the exception of the PSP level 0 where there is a need to check if value is reasonable.
Planned and actual size data are entered correctly.	Yes	The Student Workbook tool already checks this point.
Planned and actual size/hour data are reasonable.	Yes	Using an interval of time to check if these values are reasonable or not.
All manual calculations are correct.	N/A	The Student Workbook tool already calculates the values.
Planned times are distributed much like the To Date %.	N/A	The Student Workbook tool already calculates the values.
The planned and actual size values are reasonable.	Yes	Using an interval of time to check if these values are reasonable or not.
The CPI calculations are correct and reasonable.	N/A	The Student Workbook tool already calculates the correct values, but it is interesting to check if values are reasonable (use of intervals of data).

The % Reused and % New Reusable values are proper.	N/A	The Student Workbook tool already calculates the values.
The defect estimates are based on historical data.	N/A	The Student Workbook tool already calculates the values.
The planned review times and rates are reasonable.	Yes	Using an interval of time to check these values are reasonable or not.
The actual review times are reasonable.	Yes	Using an interval of time to check if these values are reasonable or not.
The To Date calculations are restarted with PSP2.	N/A	The Student Workbook tool already calculates the values.
The COQ values are correct and reasonable.	Yes	Using an interval of time to check if these values are reasonable or not.
The size and time prediction intervals are reasonable.	Yes	Using an interval of time to check if these values are reasonable or not.

Consistency Checks

Item/Process	**Automatic Verification?**	**Method of verification**
Defects removed are consistent with compile and test phase time and program size.	Yes	Compare number of defects removed in compile and test phases. Check for example if in compile phase the time is long without any defects, so something wrong is occurring. Here some previous configured interval of values is used.
Total compile defect fix times are less than compile time.	Yes	Use of calculations to check if this item is well done. Use of time data.
Total test defect fix times are less than test time.	Yes	Use of calculations to check if this item is well done. Use of time data.
Defect dates & phases are consistent with the time log.	Yes	Use of calculations to check if this item is well done. Use of time data.
Planning summary is consistent with the time log.	Yes	Use of calculations to check if this item is well done. Use of time data.
Planning summary is consistent with the defect log.	Yes	Use of calculations to check if this item is well done. Use of time data.
Planning Summary values are consistent with the size estimating template values.	Yes	The Student Workbook tool already checks this point.
Actual Added on planning summary close to and no less than actual BA+PA on size estimating template.	Yes	Use of calculations to check if this item is well done. Use of time data.
All manual data entries are consistent among all forms.	Yes	The Student Workbook tool already calculates the values. This point is used for PSP level 2.
Compile times are consistent with the defects found.	Yes	Use of calculations to check if this item is well done. Use of time data. This point is used for PSP level 2.
Test times are consistent with	Yes	Check if information is the same when comparing

the defects found.		planning summary and size estimating template. This point is used for PSP level 2.
The actual size, time, and defect data are reasonable.	No	It is not possible to verify in an automatic way. This point is used for PSP level 2 and for planning summary form.
The PROBE data are correctly entered in the plan form.	N/A	The Student Workbook tool already calculates the values. This point is used for PSP level 2.
Between 2 and 3 defects found per hour of design review.	Yes	It counts the number of defects per hour. This point is used for PSP level 2 and defect recording log form.
Between 5 and 10 defects found per hour of code review.	Yes	It counts the number of defects per hour. This point is used for PSP level 2 and defect recording log form.
Most design defects were injected in the design phase.	Yes	It counts the number of defects and compare with the ones injected in the design phase. This point is used for PSP level 2 and defect recording log form.
Verification methods were used in the design review.	No	It is not possible to verify in an automatic way. This point is used for PSP level 2.

General

Item/Process	Automatic Verification?	Method of verification
Followed the defined process.	Yes	Check for each level of PSP all phases are described and used.
Complete, consistent and accurate process data collected.	Yes	If all information is correct along the forms this item is checked.
The student did his or her own work.	N/A	Use of software to check if information is equal or not. If equal means that data was copied from other work.
Historical data are used in planning the work.	No	It is not possible to verify in an automatic way.
The process data are complete.	Yes	Check if information is correctly fulfilled.
The data are self-consistent.	No	It is not possible to verify in an automatic way.
The exercise report is in the proper order and format.	No	Not possible to verify in an automatic way.
A checked-off code-review checklist is included.	No	Not possible to verify in an automatic way.
A checked-off design-review checklist is included.	No	Not possible to verify in an automatic way.

Design Review Checklist

Item/Process	Automatic Verification?	Method of verification
The checklist entries are based on historical data.	N/A	Not possible to verify in an automatic way.
The checklist was used correctly.	N/A	Not possible to verify in an automatic way.
The checklist is completely checked off.	N/A	Not possible to verify in an automatic way.

Code Review Checklist

Item/Process	Automatic Verification?	Method of verification
The checklist entries are based on historical data.	N/A	Not possible to verify in an automatic way.
The checklist was used correctly.	N/A	Not possible to verify in an automatic way.
The checklist is completely checked off.	N/A	Not possible to verify in an automatic way.

The PSP DS Templates

Item/Process	Automatic Verification?	Method of verification

The PSP design templates were used.	N/A	Not possible to verify in an automatic way.
The templates properly document the design.	N/A	Not possible to verify in an automatic way.
The templates were used in design verification.	N/A	Not possible to verify in an automatic way.

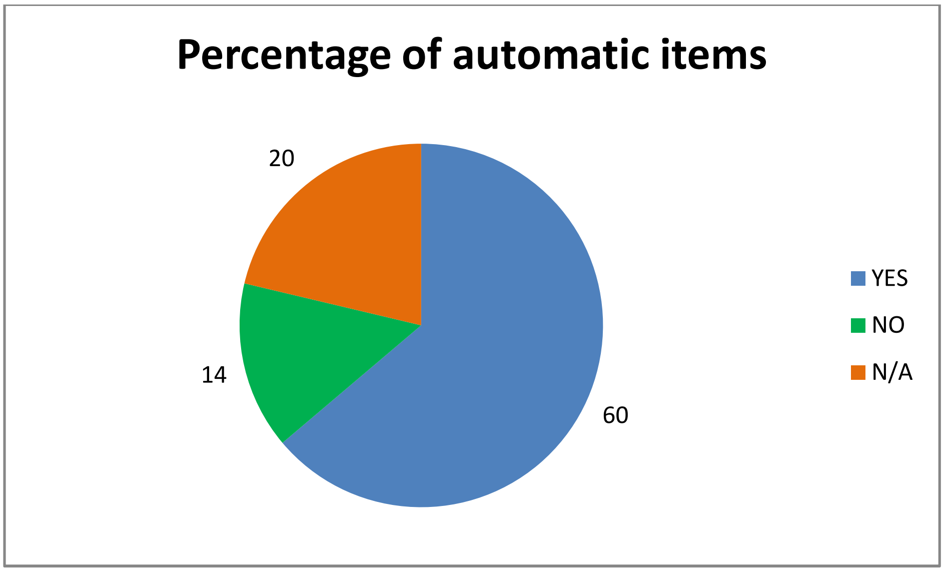


Figure 12– Percentage of items that can be checked automatically

To conclude this chapter, it is important to refer that the values obtained in the previous chart can change, because with a few changes on the PSP Student Workbook forms more points can be automatic or PSPChecker can simply create rules that guarantee that some points are being achieved.

Chapter 4 - PSPChecker Specification

In this chapter all the requirements found for PSPChecker will be described, as well architecture and tests associated. This is the first chapter where all the functionalities will be described.

4.1 PSPChecker Functional Requirements

T he main objective with PSPChecker is to help teachers to be able to make decisions faster, help the students achieve better results and have a better knowledge about PSP.

After a market study of all the existing tools and some discussion with specialists, it was fundamental to create a tool with some of the next functionalities because it provides new points that have never been approached in PSP.

As main functionalities we have:

- **Automatic verification of checklists** – Main functionality of PSPChecker because it is the main reason of this project, since the main issue is to verify checklists regarding the data available in the Access files. Each checklist item has a different verification method, so it is necessary to search and get knowledge from the specialists in PSP (teachers and PSP Book).

  In this automatic verification, if an item in the checklist is completely satisfied, the line in green is shown; otherwise the red line is shown. If it is a special case (not all item can be verified automatically) a message will appear in the screen saying that item should be analyzed in a manual way.

- **Custom processes** – PSP was described since the beginning as a tailor-made process, because it can be used only by using some components that fulfill the needs of a certain course. For example, in PSP level 1 it could only be need to use the criteria for size estimating. So in this way when the tool is initialized the user can choose to use certain items from a specific level of the PSP process. In this way PSPChecker can be adapted with the method of teaching.

- **Import data remotely –** Since the beginning as a possible way of integration with SEI/CMU this functionality was a necessary requirement because all the information is uploaded to a main server in an official way where after that the teacher can download all the files available from a student. So in this way it is fundamental to have an option where the teacher can simply download all the information directly without the need of using a web browser for that purpose. Web services are a simple and secure way to do this operation.

- **Illustrative charts –** In all types of teaching it is important to know if a method is well applied and the best way of seeing that is from the final results. These final results can be transformed afterwards in charts to check percentages of correct points and understand which points can be improved. PSPChecker charts will help teachers and students understand which are the weak and the strong points common in a group of students.

- **Automation of support messages (use of knowledge acquired by specialists) –** This functionality is an add-on to automatic verification because every time an item is checked possible messages related to that item can appear on the screen. This message can be just a simple comment saying that the item was correctly done to a more complex message where the user receives a suggestion of what is wrong or what can be improved. All this information once again is obtained from the experience of specialists and from available literature.

- **Import/Export information –**Import option is planned to receive information in Access format (database format). Exported information will be exported with the result of available checklists (evaluation paper) in formats such as PDF.

- **Modularity and scalability** – As the extra PSPChecker should be modular and scalable, that external team of software development can be able to introduce new features without much effort.

4.2 Requirements of external interface

4.2.1 Interface to users

PSPChecker was planned only to be available to teachers as a support for evaluation and feedback. But depending on the type of teaching this tool can be also provided to students to improve their work. PSPChecker is user friendly.

Users will have access to PSPChecker simply by clicking on the icon related to the application. The interface of the tool was created regarding several issues related to users.

Thus, PSPChecker should provide a simple and intuitive interface that allows users to use it after a short period of learning (2 hours maximum).

Initially PSPChecker works as a desktop application, but in an advanced state one can create the same version as a web page with the same functionalities.

4.2.2 Hardware requirements

Requirement for hardware is a functional computer with a functional operating system. To access data remotely it is necessary an internet connection.

4.3 Non-Functional Requirements

Non-functional requirements are used to specify criteria that judge the several operations of a system. Usually non-functional requirements are related with the quality of a product.

For this project ISO/IEC 9126 [12] was used to check quality of the product .This standard contains a set of characteristics that should be analyzed in the current project.

For this project the achieved characteristics   were:

- **Usability** - Ability of the product to be understood, easy learning and attractive to the user. PSPChecker can be considered that since its interface is easy to understand even for a simple user. All usability sub-characteristics were applied to the project like attractiveness, operability and intelligibility.
- **Maintenance and Expandability –** This was given special attention in relation to good design practices. Easy modification of features and components that constitute the tool as well as introducing new features. This attribute has a particular relevance to the stakeholders, which may at a later stage modify the tool according to their needs.
- **Performance** – It is fundamental that the tool works in all situations since it only depends on the existence of a computer. To improve the performance of the tool some shortcuts will probably be added to help the user achieve the goals quickly.

4.4 Use Case Model

The description of the PSPChecker should behave when a request from user is made.

The use of use case diagram allows a definition of the objectives of the targeted customer. Those are used to capture the behavior of the PSPChecker. All the diagrams were elaborated in ***Enterprise Architect***.

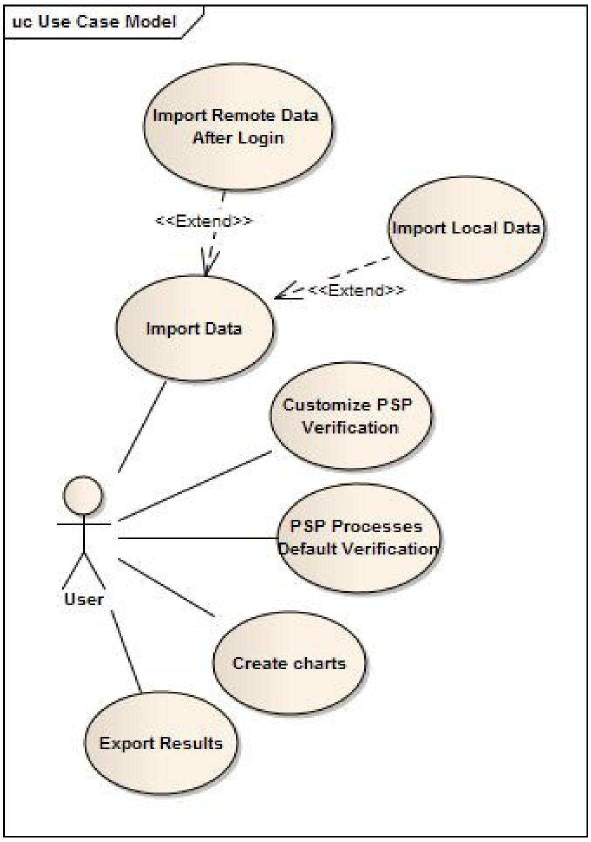


Figure 13- General vision of the use cases

4.4.1 Actors

- **USER** – At this point there is one kind of actor that is called USER that contains teachers (instructors and coach) who can access and control all operations.

  In an advanced state the student will also be able to use PSPChecker but with some restrictions:

  - Retrieve information from web services; since this option is only available in case the user has permission to retrieve the information. This permission will be tested by using a log-in system.
  - Support option to complete evaluation and feedback to students after automatic verification.
  - Aggregation of all students' data.

4.4.2 Use Cases

For a better understanding of all functionalities available in PSPChecker it was necessary to have an initial plan of how data should be accessed. Therefore, the several use cases and sequence diagrams are presented and described in a high-level.

- **Import Data remotely**

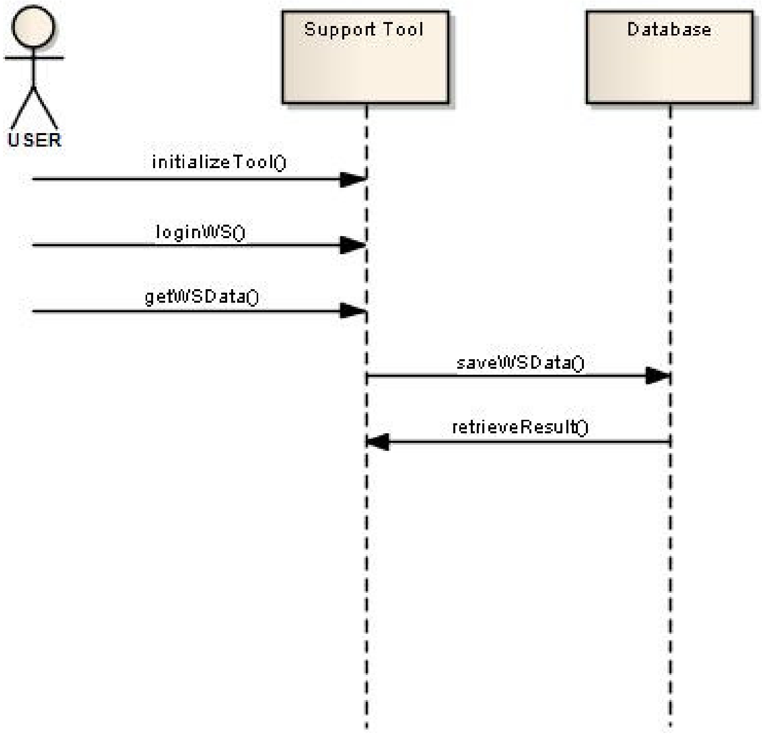


Figure 14- Sequence Diagram for Import Data Remotely

UC_A01 – Import Data Remotely	
This operation allows to retrieve information after the correct fulfillment of a login box. **The information is retrieved using web services.** **After the information is retrieved, a table Result is created in the selected Database.**	
Priority:	**High**
Effort:	**Normal**
Risk:	**Normal**

- **Import Data Locally**

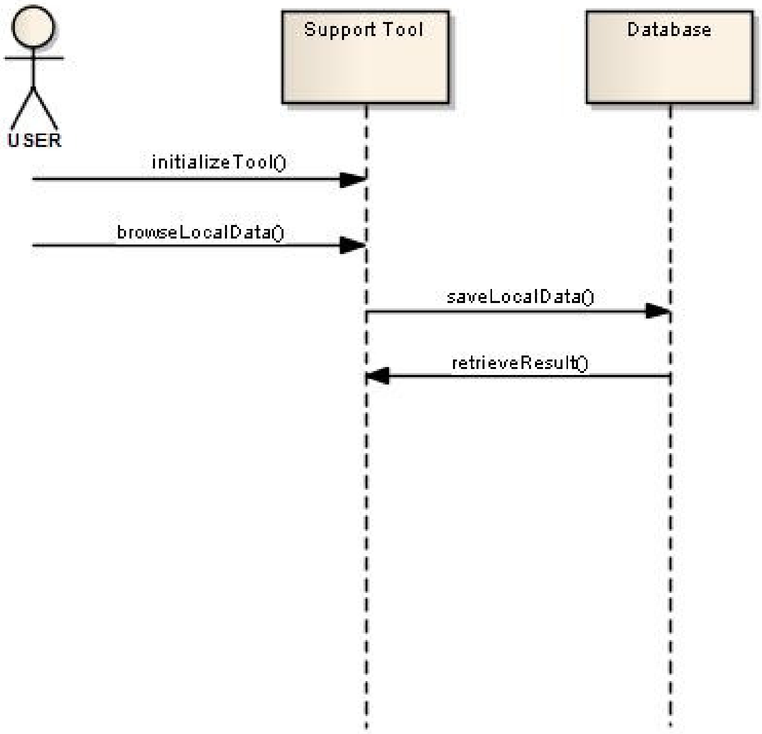


Figure 15- Sequence Diagram for Import Data Locally

UC_A02 – Import Data Locally	
This operation allows to retrieve information after choosing the location of a local Database file. Files should be in .mdb format. **After the information is retrieved, a table Result is created in the selected Database.**	
Priority:	**High**
Effort:	**Normal**
Risk:	**Normal**

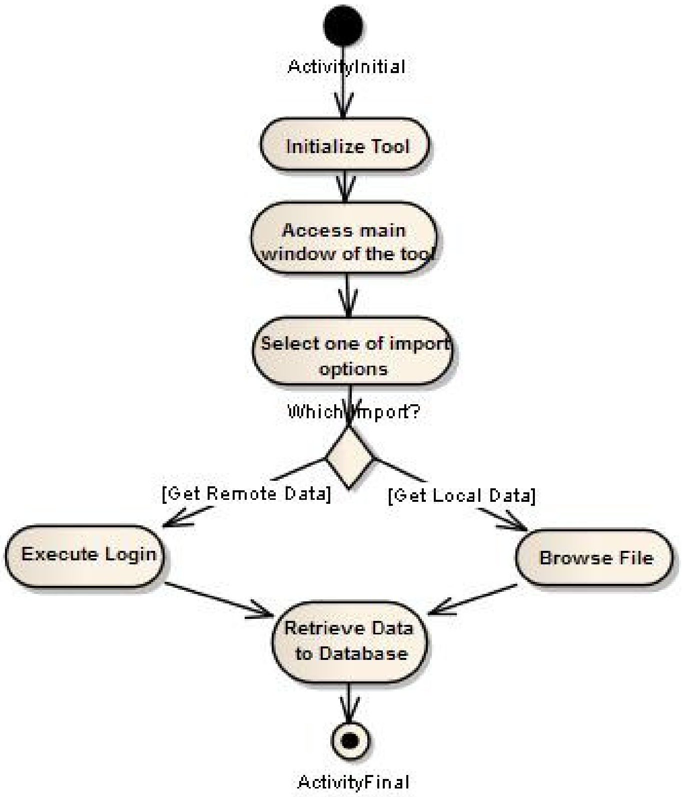


Figure 16- Activity diagram for import data

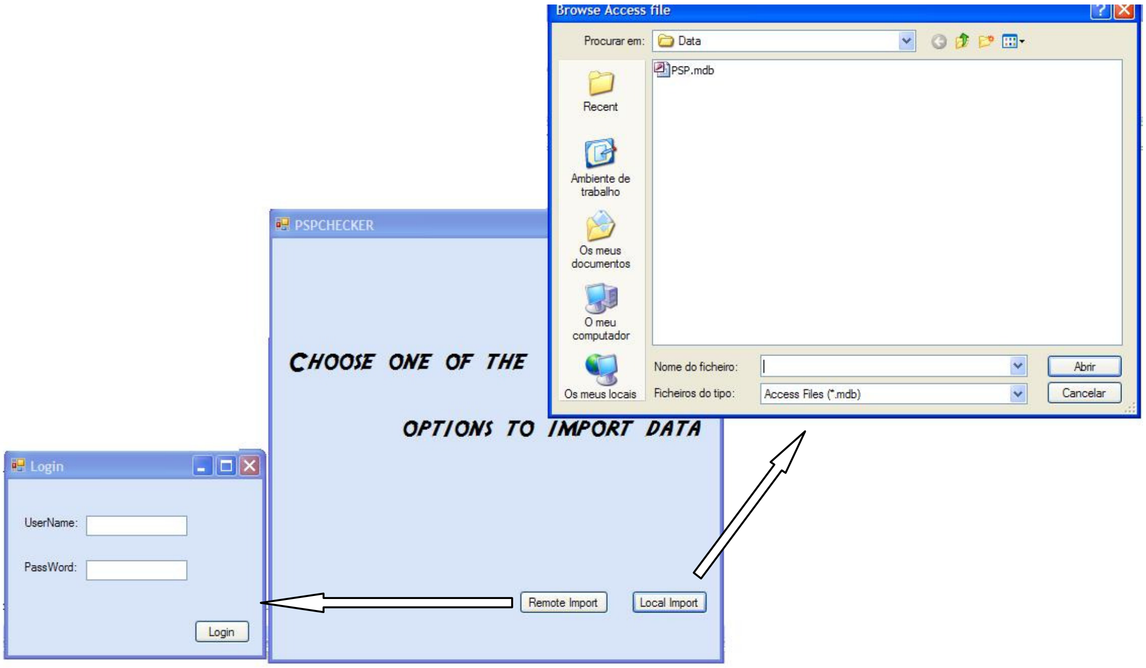


Figure 17- Interface Design (import data)

- **Automatic Verification of a Typical Process**

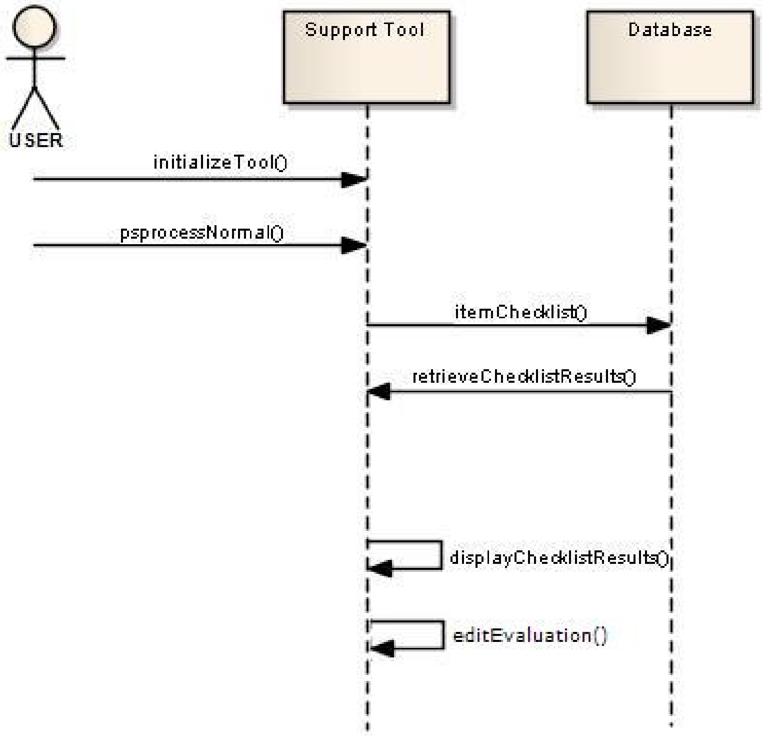


Figure 18- Sequence Diagram for the Verification of a typical PSP process

UC_A03– Automatic Verification of a Typical Process	
Operation can only happen after choosing the location of the Database. **So if that requirement is done, the user can choose one of the level of PSP(0, 1 or 2) already defined with specific check items and check if its own project is going in the right direction or not.** **All the PSP levels are already defined since the initialization of PSPChecker. As a final result all the data is retrieved from the table created in Database with the name Result.**	
Priority:	**Very High**
Effort:	**Very High**
Risk:	**Very High**

Automatic Verification of a Custom Process

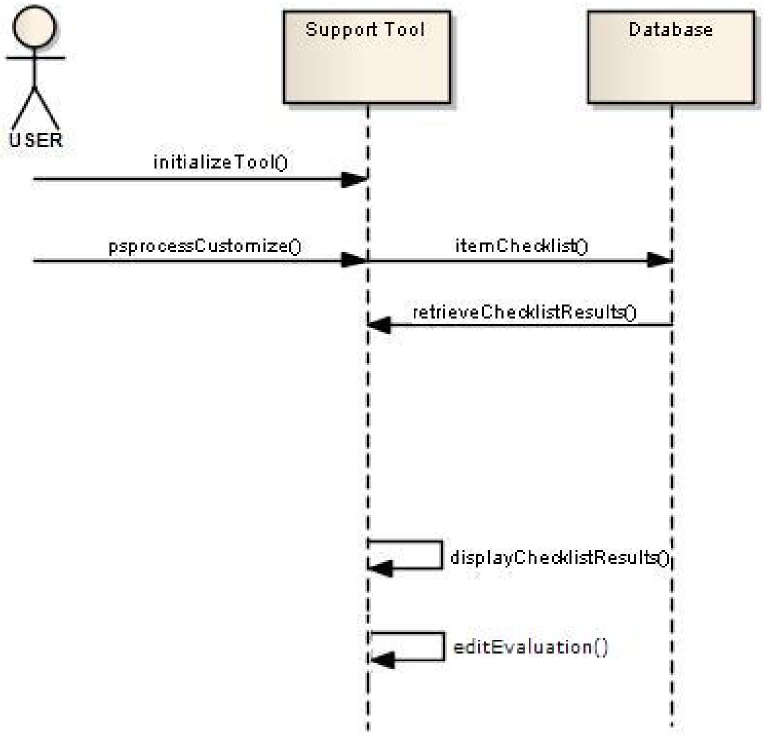


Figure 19- Sequence Diagram for the Verification of a custom PSP process

UC_A04– Automatic Verification of a custom Process	
Operation can only happen after choosing the location of the Database. **So if that requirement is done, the user can choose several items from different levels of PSP (0, 1, 2) and check if its own project is going in the right direction or not.** **All the PSP levels are already defined since the initialization of PSPChecker. As a final result all the data is retrieved from the table created in the Database with the name Result.**	
Priority:	**Very High**
Effort:	**Very High**
Risk:	**Very High**

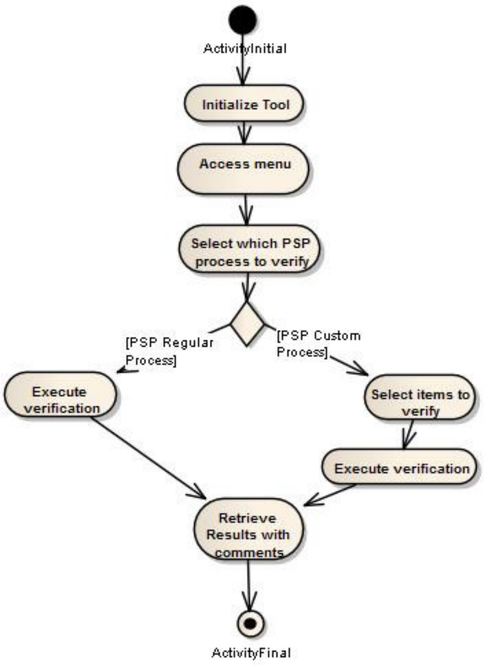
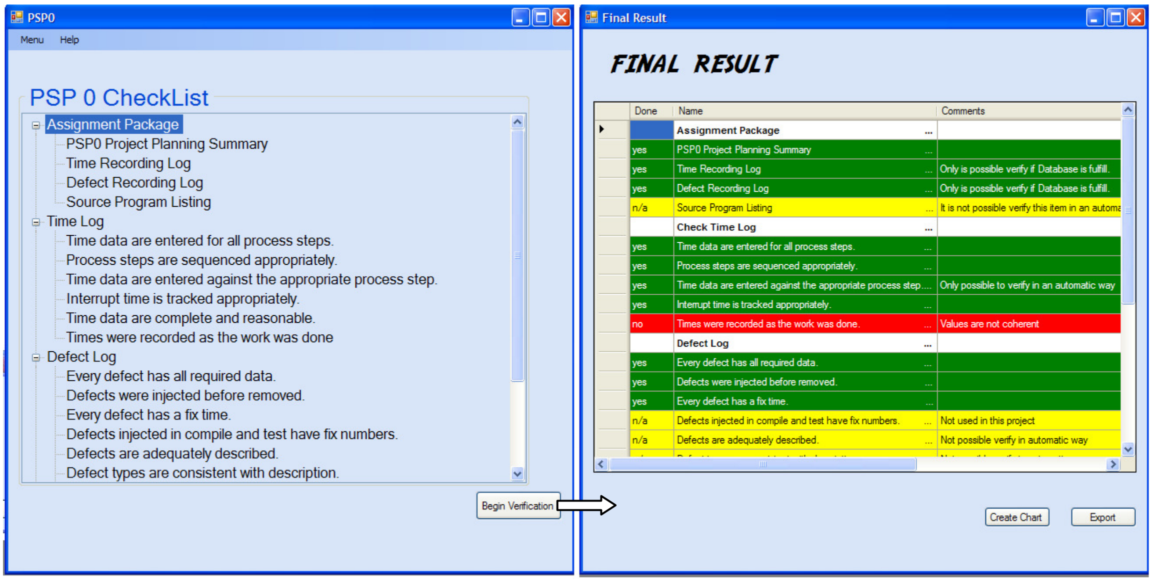


Figure 20- Activity diagram to verify the PSP process



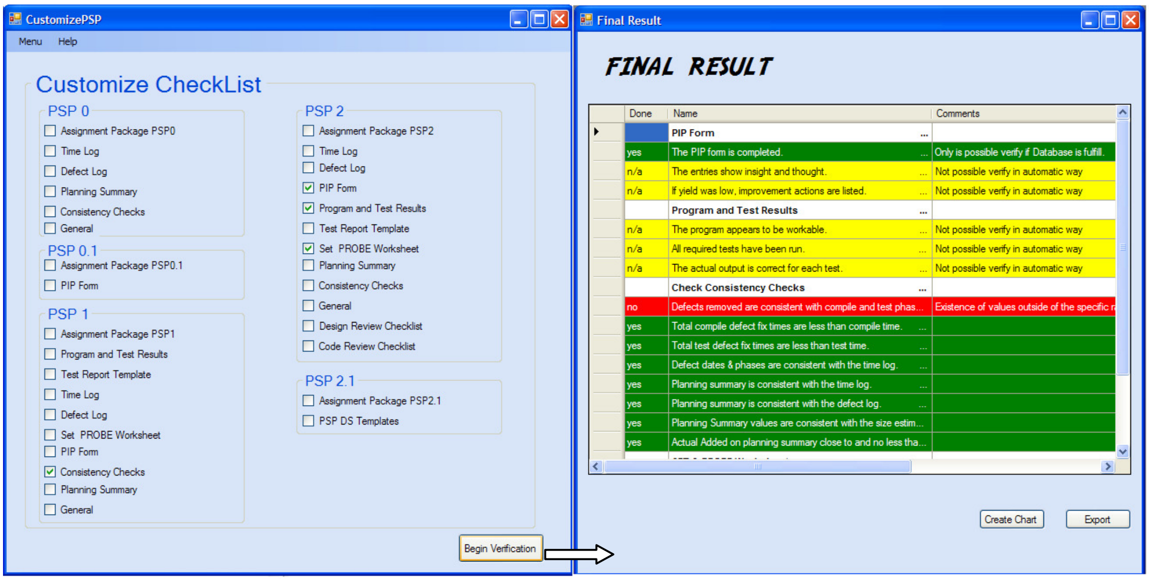
PSP 0 CheckList

- Assignment Package
  - PSP0 Project Planning Summary
  - Time Recording Log
  - Defect Recording Log
  - Source Program Listing
- Time Log
  - Time data are entered for all process steps.
  - Process steps are sequenced appropriately.
  - Time data are entered against the appropriate process step.
  - Interrupt time is tracked appropriately.
  - Time data are complete and reasonable.
  - Times were recorded as the work was done
- Defect Log
  - Every defect has all required data.
  - Defects were injected before removed.
  - Every defect has a fix time.
  - Defects injected in compile and test have fix numbers.
  - Defects are adequately described.
  - Defect types are consistent with description.

FINAL RESULT

Done	Name	Comments
	Assignment Package ...	
yes	PSP0 Project Planning Summary ...	
yes	Time Recording Log ...	Only is possible verify if Database is fulfill.
yes	Defect Recording Log ...	Only is possible verify if Database is fulfill.
n/a	Source Program Listing ...	It is not possible verify this item in an automa
	Check Time Log ...	
yes	Time data are entered for all process steps. ...	
yes	Process steps are sequenced appropriately. ...	
yes	Time data are entered against the appropriate process step....	Only possible to verify in an automatic way
yes	Interrupt time is tracked appropriately. ...	
no	Times were recorded as the work was done. ...	Values are not coherent
	Defect Log ...	
yes	Every defect has all required data. ...	
yes	Defects were injected before removed. ...	
yes	Every defect has a fix time. ...	
n/a	Defects injected in compile and test have fix numbers. ...	Not used in this project
n/a	Defects are adequately described. ...	Not possible verify in automatic way

Figure 21- Interface Design (Verify PSP data)



CustomizePSP

Menu Help

Customize CheckList

PSP 0

- □ Assignment Package PSP0
- □ Time Log
- □ Defect Log
- □ Planning Summary
- □ Consistency Checks
- □ General

PSP 0.1

- □ Assignment Package PSP0.1
- □ PIP Form

PSP 1

- □ Assignment Package PSP1
- □ Program and Test Results
- □ Test Report Template
- □ Time Log
- □ Defect Log
- □ Set PROBE Worksheet
- □ PIP Form
- ☑ Consistency Checks
- □ Planning Summary
- □ General

PSP 2

- □ Assignment Package PSP2
- □ Time Log
- □ Defect Log
- ☑ PIP Form
- ☑ Program and Test Results
- □ Test Report Template
- ☑ Set PROBE Worksheet
- □ Planning Summary
- □ Consistency Checks
- □ General
- □ Design Review Checklist
- □ Code Review Checklist

PSP 2.1

- □ Assignment Package PSP2.1
- □ PSP DS Templates

Begin Verification

Final Result

FINAL RESULT

Done	Name	Comments
	PIP Form ...	
yes	The PIP form is completed. ...	Only is possible verify if Database is fulfill.
n/a	The entries show insight and thought. ...	Not possible verify in automatic way
n/a	If yield was low, improvement actions are listed. ...	Not possible verify in automatic way
	Program and Test Results ...	
n/a	The program appears to be workable. ...	Not possible verify in automatic way
n/a	All required tests have been run. ...	Not possible verify in automatic way
n/a	The actual output is correct for each test. ...	Not possible verify in automatic way
	Check Consistency Checks ...	
no	Defects removed are consistent with compile and test phas...	Existence of values outside of the specific r
yes	Total compile defect fix times are less than compile time. ...	
yes	Total test defect fix times are less than test time. ...	
yes	Defect dates & phases are consistent with the time log. ...	
yes	Planning summary is consistent with the time log. ...	
yes	Planning summary is consistent with the defect log. ...	
yes	Planning Summary values are consistent with the size estim...	
yes	Actual Added on planning summary close to and no less tha...	

Create Chart Export

Figure 22- Interface Design (Verify PSP data - custom process)

- **Export Results**

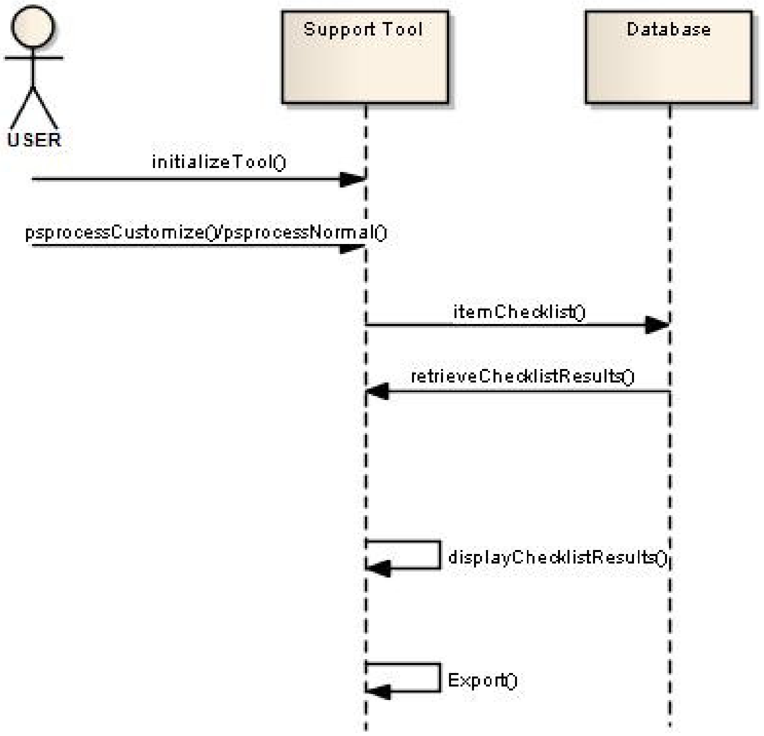

USER
Support Tool
Database
initializeTool()
psprocessCustomize()/psprocessNormal()
itemChecklist()
retrieveChecklistResults()
displayChecklistResults()
Export()

Figure 23- Sequence Diagram for export results

UC_A05– Export results	
After obtaining the results of the verification, it export that information in different formats such as PDF or Excel.	
Priority:	Normal
Effort:	Normal
Risk:	Low

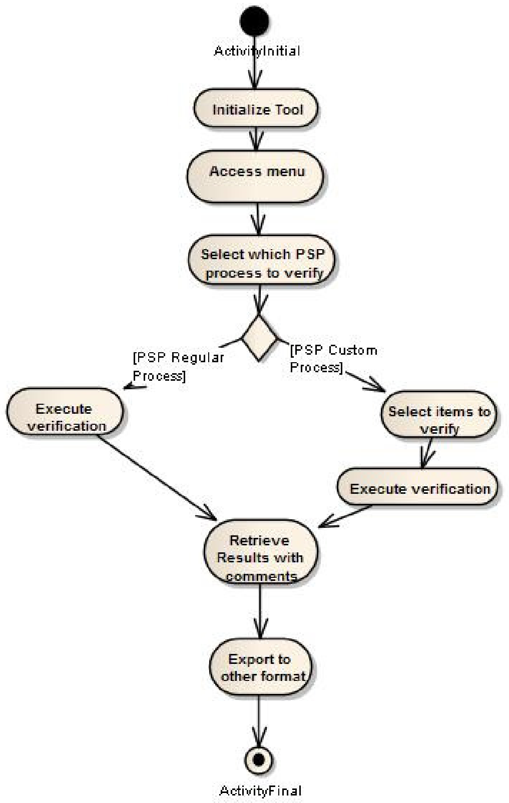
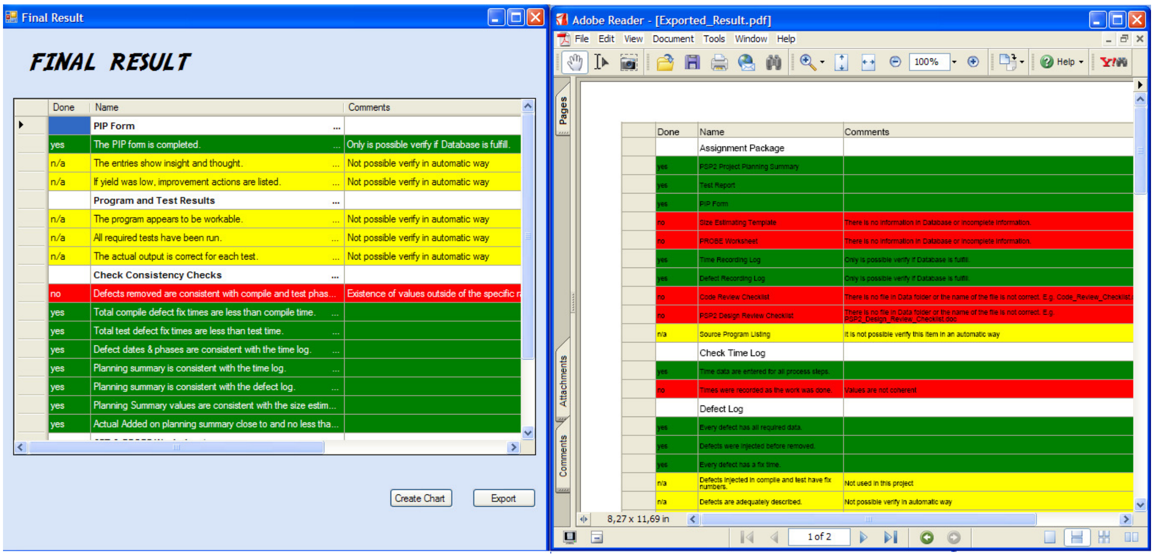


Figure 24- Activity diagram for export results



Final Result

FINAL RESULT

Done	Name	Comments
	PIP Form ...	
yes	The PIP form is completed. ...	Only is possible verify if Database is fulfill.
n/a	The entries show insight and thought. ...	Not possible verify in automatic way
n/a	If yield was low, improvement actions are listed. ...	Not possible verify in automatic way
	Program and Test Results ...	
n/a	The program appears to be workable. ...	Not possible verify in automatic way
n/a	All required tests have been run. ...	Not possible verify in automatic way
n/a	The actual output is correct for each test. ...	Not possible verify in automatic way
	Check Consistency Checks ...	
no	Defects removed are consistent with compile and test phas...	Existence of values outside of the specific r
yes	Total compile defect fix times are less than compile time. ...	
yes	Total test defect fix times are less than test time. ...	
yes	Defect dates & phases are consistent with the time log. ...	
yes	Planning summary is consistent with the time log. ...	
yes	Planning summary is consistent with the defect log. ...	
yes	Planning Summary values are consistent with the size estim...	
yes	Actual Added on planning summary close to and no less tha...	

Create Chart
Export
Adobe Reader - [Exported_Result.pdf]
File Edit View Document Tools Window Help
100%
Help
Pages
Attachments
Comments
8,27 x 11,69 in
1 of 2

Done	Name	Comments
	Assignment Package	
yes	PSP2 Project Planning Summary	
yes	Test Report	
yes	PIP Form	
no	Size Estimating Template	There is no information in Database or incomplete information.
no	PROBE Worksheet	There is no information in Database or incomplete information.
yes	Time Recording Log	Only is possible verify if Database is fulfill.
yes	Defect Recording Log	Only is possible verify if Database is fulfill.
no	Code Review Checklist	There is no file in Data folder or the name of the file is not correct. E.g. Code_Review_Checklist.
no	PSP2 Design Review Checklist	There is no file in Data folder or the name of the file is not correct. E.g. PSP2_Design_Review_Checklist.doc
n/a	Source Program Listing	It is not possible verify this item in an automatic way
	Check Time Log	
yes	Time data are entered for all process steps.	
no	Times were recorded as the work was done.	Values are not coherent
	Defect Log	
yes	Every defect has all required data.	
yes	Defects were injected before removed.	
yes	Every defect has a fix time.	
n/a	Defects injected in compile and test have fix numbers.	Not used in this project
n/a	Defects are adequately described.	Not possible verify in automatic way

Figure 25- Interface Design (Export Results)

- **Create charts**

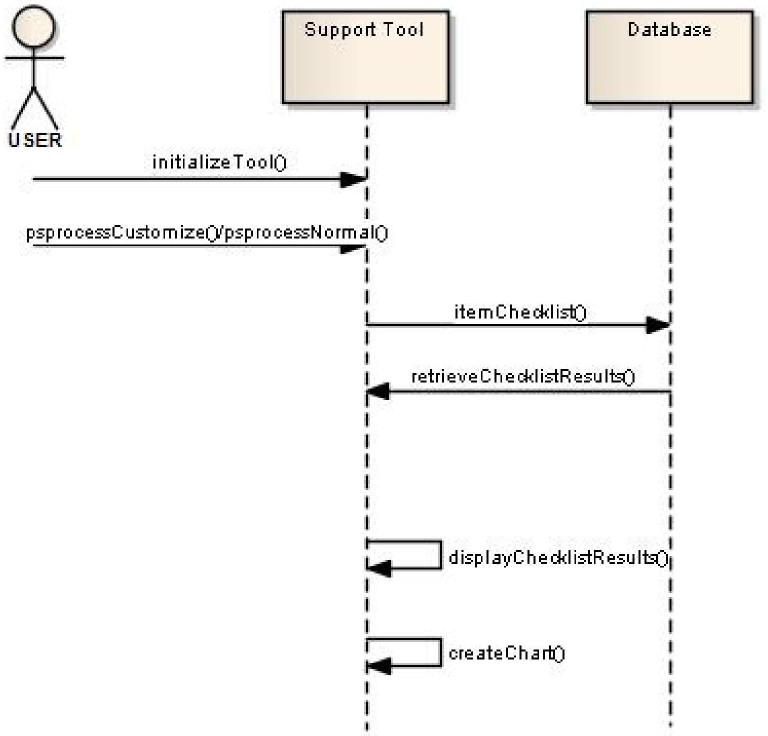

Support Tool
Database
USER
initializeTool()
psprocessCustomize()/psprocessNormal()
itemChecklist()
retrieveChecklistResults()
displayChecklistResults()
createChart()

Figure 26- Sequence Diagram to create charts

UC_A06– Create Charts	
It is ability after obtaining the results of the verification. It creates different kinds of charts with info related with the percentage of correct points or other info. This information is related to a programmer but can be improved applying to a project.	
Priority:	Normal
Effort:	Normal
Risk:	Low

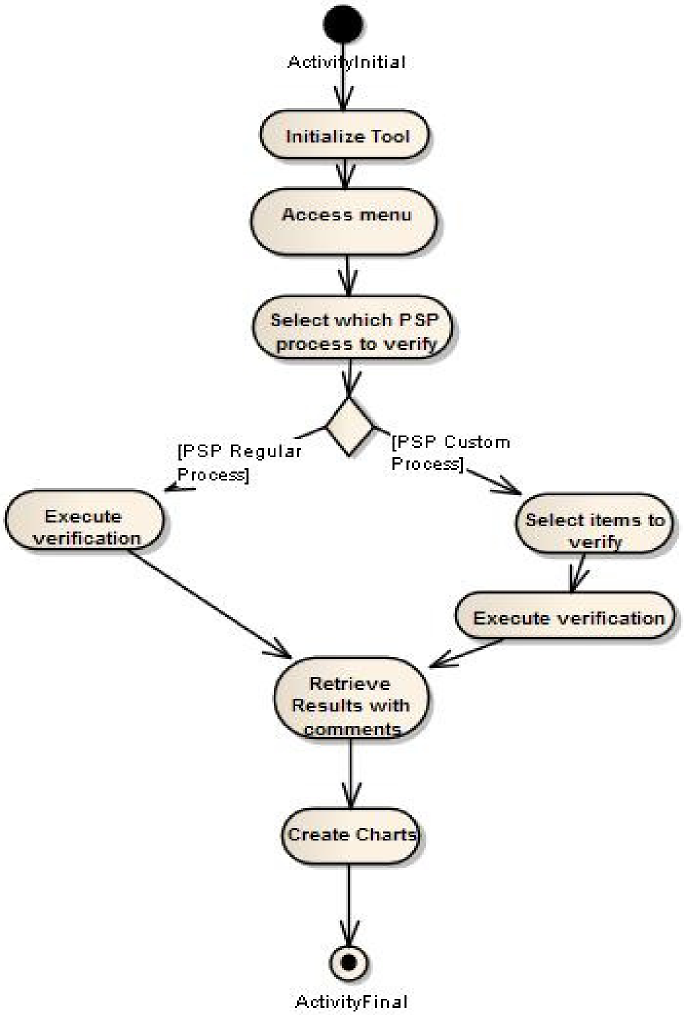
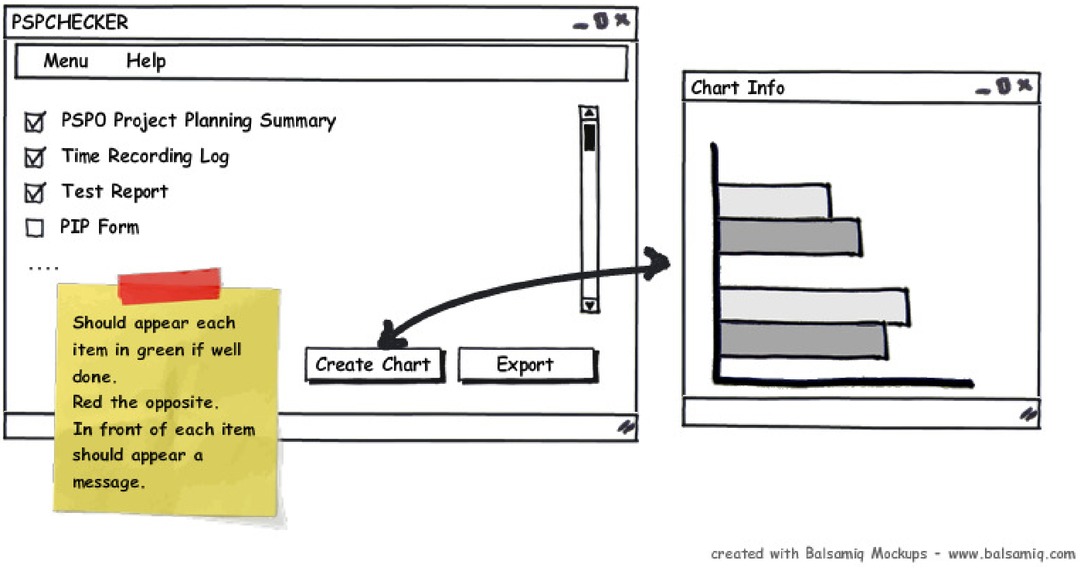


Figure 27- Activity diagram to create charts



☑ PSPO Project Planning Summary
☑ Time Recording Log
☑ Test Report
□ PIP Form
....

Should appear each item in green if well done.
Red the opposite.
In front of each item should appear a message.

Figure 28- Interface Design (Create Charts)

4.5 Architecture

The software architecture contains a set of significant decisions concerning the organization of a software system. This point of the report presents the architecture (logical and physical) of PSPChecker.

It also illustrates the organization of the system and the set of decisions that have relevance to its structure.

The structure of the architectural description will be the following:

- Representation of the logical and physical architecture.
- Design decisions and technology choices.

4.5.1 Logical Architecture

Logical architecture of a system is the set of decomposed logic modules and relationships between them that satisfy all system requirements. This architecture is specified in UML diagrams using software packages.

The presentation of logical architecture includes a horizontal and vertical decomposition of the same. The decomposition describes horizontal layers, such as the user interface or database. The vertical decomposition hierarchy defines the subsystems that constitute PSPChecker; each subsystem corresponds to a set of features.

- **Horizontal View**

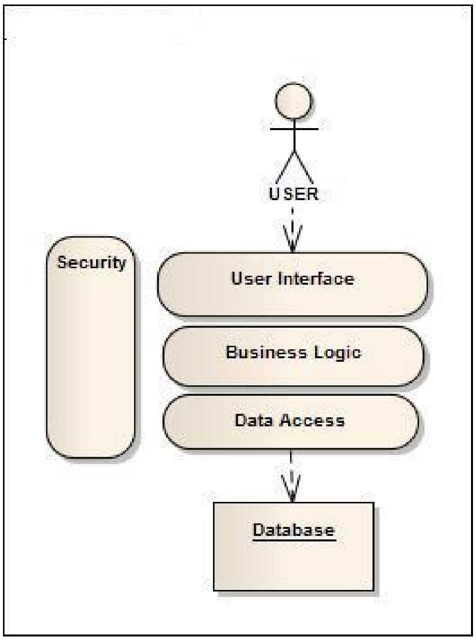


Figure 29 – Horizontal view of PSPCHecker

- **Vertical view**

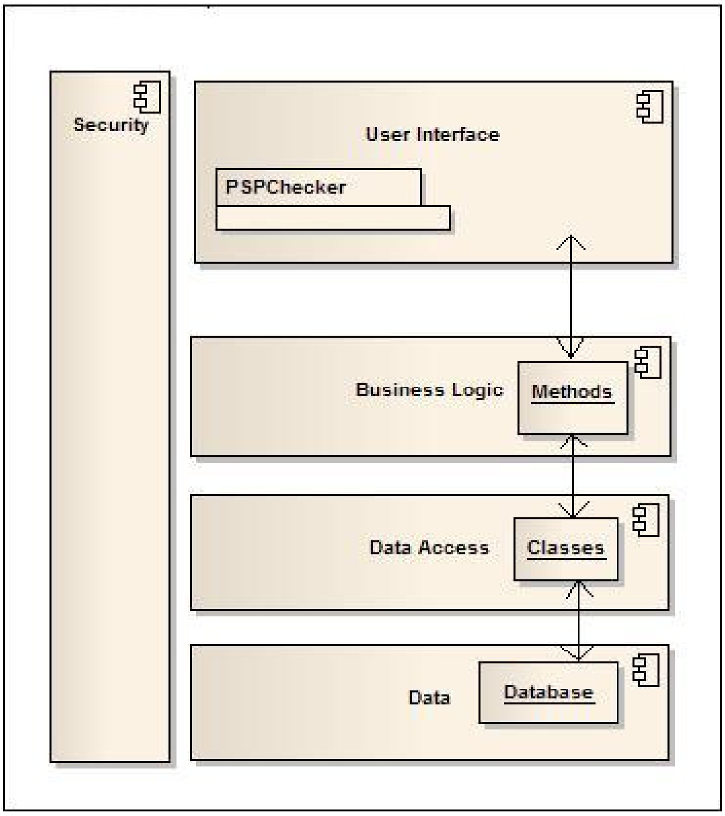


Figure 30 – Vertical view

User Interface

This layer is the only one visible to the final user. It is through the user interface that the user will interact with the tool, accessing all features offered by this. This layer will communicate with the lower, Business Logic, using all methods.

Business Logic

Being a middle layer, the Business Logic Layer communicates with two layers: the User Interface Layer and the Data Access. These communications are designed to provide features that are based on business rules set out in Requirements Specification. This module receives requests from the layer of user interface and processes according to certain rules of business already established, and then sends requests to the Data Access layer.

Data Access

This layer is also an intermediate layer and stays between the layers of Business Logic and Data. It is a layer that allows the communication between the objects and the database.

Data

It is the layer where all crucial information to the system is communicated with the layer of Data Access. It consists of the database that stores the tables in the relational model and its data.

Security

In every layer there is a need to ensure the security of the system.

Thus, this layer serves to ensure that the various users of the system only access to features that are reserved and have permission to perform a specific action. An example of this is that the retrieval of data from the remote server is only available after the fulfillment of a valid login.

4.5.2 Physical Architecture

The next figure demonstrates the physical architecture of the system and components that assemble its architecture.

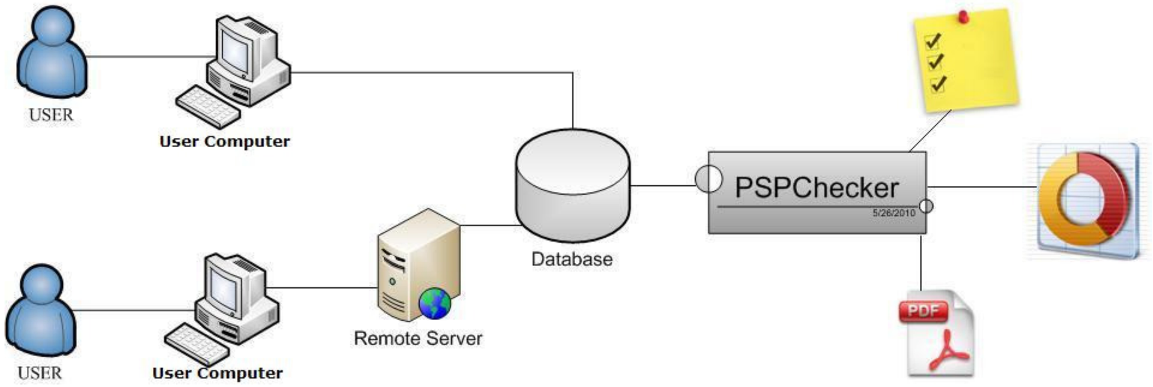


Figure 31- Physical Architecture of the System

The essential components of architecture are:

- **User Computer** – Machine where the user can access to PSPChecker. If a user wants to retrieve information from the remote server, he should have an internet connection.
- **Remote Server** – Use of web services to communicate with SEI/CMU system and retrieve all information associated to an account of a teacher. Each teacher can access to all information of a student.
- **Database** – Database that contains all information to verify if a process is being followed. After selecting the database, a table Result will be created, which will save all the information related to the checklist verification (each item result will be saved here). The database is in Access format.
- **PSPChecker** – Main system where a user can select several options related to PSP process checking. This system was elaborated in C# and runs on the user computer.

4.5.3 Design Decisions

Database

In this project the Database used was not created, because PSPChecker will get the Data already fulfilled. This happens because when the main tool, the PSP Students Workbook, is complete the export of that information is done on an access format file.

When the PSPChecker selects the database, it will create a new table called Result, if this table doesn't exist. But if it exists it will clean all data of the table.

To better understand which information is dealt by the PSPChecker, all the tables used by PSPChecker to verify the PSP processes will be described in a very summary way.

Table Name	Description
DefectType	Description of major categories to classify a defect. E.g. "Data", "interface",…
LOCTypeStandard	Nomenclature of different expressions used in collecting data. E.g. Base (B) or Reuse (R).
LOGDDetail	All information related to defects. E.g. which phase has injected and removed a defect, description and date.
LOGTDetail	Time Log where all information about life cycle of a project, like phase (planning,…), start and stop time and comments is provided.
Parts	Data here present consists on parts added or reuses to a project that allow know with a bigger accuracy estimating size.
PartyType Standard	Standard information related to the previous point.
PhaseData	Data automatic created for each phase of a project. Most of information here presented is in minutes and is related to time phases, Plan defects injected in phase, Actual defects removed in phase summarized from defect log entries.
Phases	Phases of a project. Here contains a short description of possible phases of a project in PSP. E.g. Planning, Coding, Detailed Design,..
PIP	Description of possible improvement to the project or process written by the student.
Processes	List of levels of PSP processes.
ProcessPhase	Relation table between Process and Phase.
ProcessesTypes	List of levels of PSP processes.
ProgramSize	One of the main tables of the database, here most of information is automatic and allows instructors to understand the state of the project, since the information here is the same as the project planning summary.
Projects	List of projects of one student and contains information related to the start and the end of a project and if it was achieved or not.
PSPAsgData	Assignment of projects to a student and information's like number of lines of code, and info about actual and planned time for each phase.

SetADD	Info about parts added to a project.
SetBase	Info about parts used as a base in a project.
SetReuse	Info about parts reused from other projects in the actual one.
SizeMeasure	Selection of method to measure size, usually is LOC (lines of code).
Tasks	Table that contains a list of tasks that should be accomplished.
TaskSchedule	Planning of tasks during one period of time
TestReports	List of all tests done to guarantee the well functioning of a project. Here description, condition and expected results should be present.
UserProfiles	Information related to a student. Thus, information is more technical than in the Users one. Here experience years and other aspects are presented.
Users	Personal information about a user or student.
Result	Table created for this project. This table is fulfilled with the result of the automatic verification of process. This table contains 3 columns: Done, Name and Comments. All these columns are strings. Every time the PSPChecker is initialized this table is created or cleans it.

Table 4 - PSPChecker DataBase components

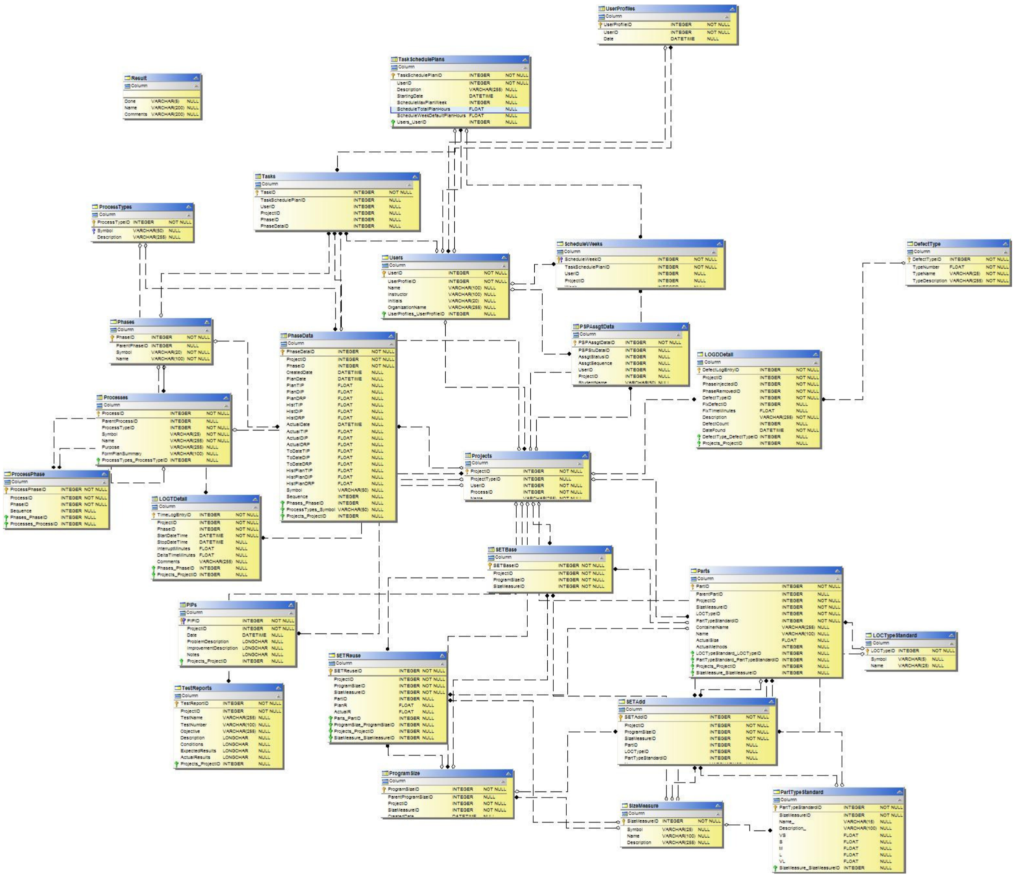

UserProfiles
Column
UserProfileID INTEGER NOT NULL
UserID INTEGER NOT NULL
Date DATETIME NULL
Result
Column
Done VARCHAR(5) NULL
Name VARCHAR(200) NULL
Comments VARCHAR(200) NULL
TaskSchedulePlans
Column
TaskSchedulePlanID INTEGER NOT NULL
UserID INTEGER NOT NULL
Description VARCHAR(255) NULL
StartingDate DATETIME NULL
ScheduleMaxPlanWeek INTEGER NULL
ScheduleTotalPlanHours FLOAT NULL
ScheduleWeekDefaultPlanHours FLOAT NULL
Users_UserID INTEGER NULL
Tasks
Column
TaskID INTEGER NOT NULL
TaskSchedulePlanID INTEGER NULL
UserID INTEGER NULL
ProjectID INTEGER NULL
PhaseID INTEGER NULL
PhaseDataID INTEGER NULL
ProcessTypes
Column
ProcessTypeID INTEGER NOT NULL
Symbol VARCHAR(50) NULL
Description VARCHAR(255) NULL
Users
Column
UserID INTEGER NOT NULL
UserProfileID INTEGER NOT NULL
Name VARCHAR(100) NULL
Instructor VARCHAR(100) NULL
Initials VARCHAR(20) NULL
OrganizationName VARCHAR(255) NULL
UserProfiles_UserProfileID INTEGER NULL
ScheduleWeeks
Column
ScheduleWeekID INTEGER NOT NULL
TaskSchedulePlanID INTEGER NOT NULL
UserID INTEGER NULL
ProjectID INTEGER NULL
DefectType
Column
DefectTypeID INTEGER NOT NULL
TypeNumber FLOAT NOT NULL
TypeName VARCHAR(25) NOT NULL
TypeDescription VARCHAR(255) NOT NULL
Phases
Column
PhaseID INTEGER NOT NULL
ParentPhaseID INTEGER NULL
Symbol VARCHAR(20) NOT NULL
Name VARCHAR(100) NOT NULL
PhaseData
Column
PhaseDataID INTEGER NOT NULL
ProjectID INTEGER NOT NULL
PhaseID INTEGER NOT NULL
CreatedDate DATETIME NULL
PlanDate DATETIME NULL
PlanTIP FLOAT NULL
PlanDIP FLOAT NULL
PlanDRP FLOAT NULL
HistTIP FLOAT NULL
HistDIP FLOAT NULL
HistDRP FLOAT NULL
ActualDate DATETIME NULL
ActualTIP FLOAT NULL
ActualDIP FLOAT NULL
ActualDRP FLOAT NULL
ToDateTIP FLOAT NULL
ToDateDIP FLOAT NULL
ToDateDRP FLOAT NULL
HistPlanTIP FLOAT NULL
HistPlanDIP FLOAT NULL
HistPlanDRP FLOAT NULL
Symbol VARCHAR(50) NULL
Sequence INTEGER NULL
Phases_PhaseID INTEGER NULL
ProcessTypes_Symbol VARCHAR(50) NULL
Projects_ProjectID INTEGER NULL
PSPAssgtData
Column
PSPAssgtDataID INTEGER NOT NULL
PSPStuDataID INTEGER NULL
AssgtStatusID INTEGER NULL
AssgtSequence INTEGER NULL
UserID INTEGER NULL
ProjectID INTEGER NULL
StudentName VARCHAR(50) NULL
LOGDDetail
Column
DefectLogEntryID INTEGER NOT NULL
ProjectID INTEGER NOT NULL
PhaseInjectedID INTEGER NOT NULL
PhaseRemovedID INTEGER NOT NULL
DefectTypeID INTEGER NOT NULL
FixDefectID INTEGER NULL
FixTimeMinutes FLOAT NULL
Description VARCHAR(255) NOT NULL
DefectCount INTEGER NULL
DateFound DATETIME NOT NULL
DefectType_DefectTypeID INTEGER NULL
Projects_ProjectID INTEGER NULL
Processes
Column
ProcessID INTEGER NOT NULL
ParentProcessID INTEGER NULL
ProcessTypeID INTEGER NOT NULL
Symbol VARCHAR(25) NOT NULL
Name VARCHAR(255) NOT NULL
Purpose VARCHAR(255) NULL
FormPlanSummary VARCHAR(100) NULL
ProcessTypes_ProcessTypeID INTEGER NULL
ProcessPhase
Column
ProcessPhaseID INTEGER NOT NULL
ProcessID INTEGER NOT NULL
PhaseID INTEGER NOT NULL
Sequence INTEGER NULL
Phases_PhaseID INTEGER NULL
Processes_ProcessID INTEGER NULL
LOGTDetail
Column
TimeLogEntryID INTEGER NOT NULL
ProjectID INTEGER NOT NULL
PhaseID INTEGER NOT NULL
StartDateTime DATETIME NOT NULL
StopDateTime DATETIME NULL
InterruptMinutes FLOAT NULL
DeltaTimeMinutes FLOAT NULL
Comments VARCHAR(255) NULL
Phases_PhaseID INTEGER NULL
Projects_ProjectID INTEGER NULL
Projects
Column
ProjectID INTEGER NOT NULL
ProjectTypeID INTEGER NULL
UserID INTEGER NOT NULL
ProcessID INTEGER NOT NULL
Name VARCHAR(255) NOT NULL
SETBase
Column
SETBaseID INTEGER NOT NULL
ProjectID INTEGER NOT NULL
ProgramSizeID INTEGER NOT NULL
SizeMeasureID INTEGER NOT NULL
Parts
Column
PartID INTEGER NOT NULL
ParentPartID INTEGER NULL
ProjectID INTEGER NOT NULL
SizeMeasureID INTEGER NOT NULL
LOCTypeID INTEGER NOT NULL
PartTypeStandardID INTEGER NOT NULL
ContainerName VARCHAR(255) NULL
Name VARCHAR(100) NULL
ActualSize FLOAT NULL
ActualMethods INTEGER NULL
LOCTypeStandard_LOCTypeID INTEGER NULL
PartTypeStandard_PartTypeStandardID INTEGER NULL
Projects_ProjectID INTEGER NULL
SizeMeasure_SizeMeasureID INTEGER NULL
PIPs
Column
PIPID INTEGER NOT NULL
ProjectID INTEGER NOT NULL
Date DATETIME NULL
ProblemDescription LONGCHAR NULL
ImprovementDescription LONGCHAR NULL
Notes LONGCHAR NULL
Projects_ProjectID INTEGER NULL
LOCTypeStandard
Column
LOCTypeID INTEGER NOT NULL
Symbol VARCHAR(5) NULL
Name VARCHAR(25) NULL
SETReuse
Column
SETReuseID INTEGER NOT NULL
ProjectID INTEGER NOT NULL
ProgramSizeID INTEGER NOT NULL
SizeMeasureID INTEGER NOT NULL
PartID INTEGER NULL
PlanR FLOAT NULL
ActualR FLOAT NULL
Parts_PartID INTEGER NULL
ProgramSize_ProgramSizeID INTEGER NULL
Projects_ProjectID INTEGER NULL
SizeMeasure_SizeMeasureID INTEGER NULL
TestReports
Column
TestReportID INTEGER NOT NULL
ProjectID INTEGER NOT NULL
TestName VARCHAR(255) NULL
TestNumber VARCHAR(100) NULL
Objective VARCHAR(255) NULL
Description LONGCHAR NULL
Conditions LONGCHAR NULL
ExpectedResults LONGCHAR NULL
ActualResults LONGCHAR NULL
Projects_ProjectID INTEGER NULL
SETAdd
Column
SETAddID INTEGER NOT NULL
ProjectID INTEGER NOT NULL
ProgramSizeID INTEGER NOT NULL
SizeMeasureID INTEGER NOT NULL
PartID INTEGER NULL
LOCTypeID INTEGER NULL
PartTypeStandardID INTEGER NULL
ProgramSize
Column
ProgramSizeID INTEGER NOT NULL
ParentProgramSizeID INTEGER NULL
ProjectID INTEGER NOT NULL
SizeMeasureID INTEGER NOT NULL
CreatedDate DATETIME NULL
SizeMeasure
Column
SizeMeasureID INTEGER NOT NULL
Symbol VARCHAR(25) NULL
Name VARCHAR(100) NULL
Description VARCHAR(255) NULL
PartTypeStandard
Column
PartTypeStandardID INTEGER NOT NULL
SizeMeasureID INTEGER NOT NULL
Name_ VARCHAR(15) NULL
Description_ VARCHAR(100) NULL
VS FLOAT NULL
S FLOAT NULL
M FLOAT NULL
L FLOAT NULL
VL FLOAT NULL
SizeMeasure_SizeMeasureID INTEGER NULL

Figure 32 - Database Model

4.5.4 Technologies

The choice of a technology influences the quality and the development time of a software project. This selection process is essentially carried out based on personal knowledge and time of learning.

So after analyzing these points, it was decided to use the following technologies:

- C# (CSharp)
- WebServices
- Microsoft Access Database
- OleDB
- DataGridViewExtension

To better understand these choices, each of these options will be describednext .

1. **C#** - Is a multi-paradigm language that contemplates functional programming, imperative and object-oriented. Was created and developed by Microsoft as part of this initiative. NET.

   As main advantages of this language we have: extensive documentation and examples, easy use, simple, modern, general-purpose and reduced learning curve.

2. **WebServices** - Typically application programming interfaces (API) or web APIs that are accessed via Hypertext Transfer Protocol and executed on a remote system hosting the requested services **[13].**

3. **Microsoft Access Database** - Pseudo-relational database management system from Microsoft that combines the relational Microsoft Jet Database Engine with a graphical user interface and software-development tools. Microsoft Access is used to create simple database solutions. Access tables support a variety of standard field types, indices, and referential integrity. Access also includes a query interface, forms to display and enter data, and reports for printing. Users can create tables, queries, forms and reports, and connect them together with macros. All database tables, queries, forms, reports, macros, and modules are stored in the Access Jet database as a single file. Microsoft Access applications can adopt split-database architecture. The database can be divided into a front end database that contains the application objects (queries, forms, reports, macros, and modules), and is linked to tables stored in a back end shared database containing the data. Microsoft Access offers several ways to secure the application while allowing users to remain productive. **[14]**

4. **OleDB** - Object Linking and Embedding Database, separates the data store from the application that needs access to it through a set of abstractions that include the datasource, session, command and rowsets. This was done because different applications need access to different types and sources of data and do not necessarily want to know how to access functionality with technology-specific methods. OLE DB is conceptually divided into consumers and providers. The consumers are the applications that need access to the data, and the provider is the software component that implements the interface and therefore provides the data to the consumer. **[15]**

5. **DataGridViewExtension -** Component that extends the standard DataGridView control, with functionalities such as Export (MS Excel**,** HTML and PDF). The design time support allows you to integrate it in any existing application. **[16]**

4.6 Test Planning

For a good test plan on PSPChecker it was necessary to do a careful planning of all stages of testing.

The level of the test (when it is applied), the technique being used, the criteria for classification of tests and the type of software to be used for testing should be defined.

It should be clear which testing levels will be applied:

- Conversion test;
- Interface tests that ensure the good use of the interface of the product;
- Security tests to ensure that the developed product meets all the requirements of confidentiality and information security;
- Performance tests that will confirm or not the minimum performance required.

For this project the IEEE Std 829-1983 and the IEEE Standard for Software Test Documentation were used as references.

4.6.1 Test Items (Modules)

In this first phase of testing it is necessary to identify the items to be tested.

Project modules

Identifier	Module	Description
MOD_TEST_A	Module Import Data	Tests related to options of importing data locally or remotely.
MOD_TEST_B	Module Verification	Tests about the automatic verification on-process PSP.
MOD_TEST_C	Module Export	Tests related with export Data in a different format.
MOD_TEST_D	Module Charts	Tests about the creation of charts.

Participants

All the items here described should be analyzed and tested by the developers and knowledge experts in the subject PSP. In this project participants will have to be very active and do tests in a comprehensive manner to avoid future failures.

4.6.2 Test Items (Sub-Modules)

For each module described before there are some tests for the related sub-modules. This description will be presented next.

Module Import Data

- Check Valid File

Identifier	**TEST_A_01**
Module Identifier	MOD_TEST_A
Description	**Checks if a file is valid (Option Locally)**
Pre-Condition	The file should be in Access format
Expected result	"File successfully upload"
Risk	**High**

- Valid Login

Identifier	**TEST_A_02**
Module Identifier	MOD_TEST_A
Description	**Checks if it is a valid information**
Pre-Condition	Fulfill the Text Area
Expected result	"Successfully login"
Risk	**Very High**

- Invalid Login

Identifier	**TEST_A_03**
Module Identifier	MOD_TEST_A
Description	**Performs user authentication system with one of the wrong fields**
Pre-Condition	Fulfill the Text Area
Expected result	"Invalid login"
Risk	**Very High**

- Import Data

Identifier	**TEST_A_04**
Module Identifier	MOD_TEST_A
Description	**Imports data locally or remotely**
Pre-Condition	Choose Data (remote or local)
Expected result	"Import data successfully"
Risk	**Very High**

- Verify data in Database

Identifier	**TEST_A_05**
Module Identifier	MOD_TEST_A
Description	**Verifies if data is well fulfilled in the Database.**
Pre-Condition	Database being fulfilled.
Expected result	-
Risk	**Very High**

Module Verification

- Choose Process

Identifier	**TEST_B_01**
Module Identifier	MOD_TEST_B
Description	**Chooses a valid process (PSP 0,1,2) and checks if the item information is correct.**
Pre-Condition	-
Expected result	-
Risk	**Medium**

- Create a custom Process

Identifier	**TEST_B_02**
Module Identifier	MOD_TEST_B
Description	**Creates a custom process and verify if the items are being added to the final list for verification.**
Pre-Condition	-
Expected result	-
Risk	**High**

- Check automatic verification

Identifier	**TEST_B_03**
Module Identifier	MOD_TEST_B
Description	**Does an automatic verification of the checklist to see if the items are verified.**
Pre-Condition	Choose a typical process or custom one.
Expected result	"Successfully check of items"

Risk	**Very High**

- Check information of automatic Verification

Identifier	**TEST_B_04**
Module Identifier	MOD_TEST_B
Description	**Checks if the information that appears in screen is equivalent to the expected result.**
Pre-Condition	Choose a typical process or custom one.
Expected result	-
Risk	**Very High**

Module Export

- Execute Export

Identifier	**TEST_C_01**
Module Identifier	MOD_TEST_C
Description	**Executes an export of the final result from the automatic verification.**
Pre-Condition	Had already done the automatic verification.
Expected result	"Successfully export to PDF Format"
Risk	**High**

- Verify document created

Identifier	**TEST_C_02**
Module Identifier	MOD_TEST_C
Description	**Verify if document is with the correct format and information**
Pre-Condition	Execute before an exportation
Expected result	-
Risk	**Low**

Module Charts

- Create Chart

Identifier	**TEST_D_01**
Module Identifier	MOD_TEST_D
Description	**Creates a chart with statistics from the final results of verification. Also here the type of chart can be chosen.**
Pre-Condition	Had already done the automatic

	verification.
Expected result	"Successfully created the chart"
Risk	**Medium**

- Verify format of chart

Identifier	**TEST_D_02**
Module Identifier	MOD_TEST_D
Description	**Verifies if a chart is in the correct format chosen before**
Pre-Condition	Had already done the option "create chart" and chosen the format.
Expected result	-
Risk	**Low**

4.6.3 Acceptance Tests

Testing Acceptance uses project documentation to prepare design, procedures and test cases. This methodology allows us to verify accuracy and understand information in the documentation in the areas covered by the tests and determines if the project is being developed according to specification.

Conversion Testing

In order to verify if the imported files in Access format are well allocated in the database tests to verify data integrity are made.

These tests consist of verifying that data entered in the database correspond to the information contained in the Access file.

Interface Tests

In order to verify interoperability of the interface between the user and the system an interface testing tool will be used, which will determine whether, given certain user actions, the system behaves as expected.

Security Tests

The availability of this data system entails a responsibility to remain private on the option remote data. The access with invalid sets of user / password will be tested. Apart from these tests it will also be checked if it is possible to download information without being authenticated.

4.6.4 Criteria Accepted/Denied

Performed tests must meet certain criteria for approval or disapproval to guarantee the reliability of the tool. For this reason, these criteria should be well defined prior to testing and should clarify what kinds of results raise approval:

- If a test result is consistent with the expected result.
- If the test was unable to prove a failure.

Or, what kinds of results lead to failure:

- Failure to comply with established quality standards.
- Results of a test are not compatible with the expected.
- Faults found in software.

If a test is approved we can proceed with the validation of that functionality. In case of disapproval to correct the problem some changes must be made up in the code depending on the type of failure found.

4.6.5 Test Procedures

In order to achieve success, in the testing process it was needed to plan a series of tasks to be performed.

These tasks were:

1. Plan the test.
2. Identify requirements for the test.
3. Assess the risk.
4. Develop strategies to implement the test.
5. Generate test plan.
6. Identify and describe test cases.
7. Identify and structure test procedures.
8. Check test coverage.
9. Implement test.
10. Identify the specific functionality in the test model design and implementation.
11. Run test.
12. Evaluate the test run.
13. Check the results.
14. Deal with unexpected results.
15. Register anomalies.
16. Determine if the completion criteria and success criteria of the test were met.

Chapter 5 - Experimentation

This chapter will present the product developed according to the specification in the previous chapters. This chapter will provide an overview of the user experience on working with PSPChecker.

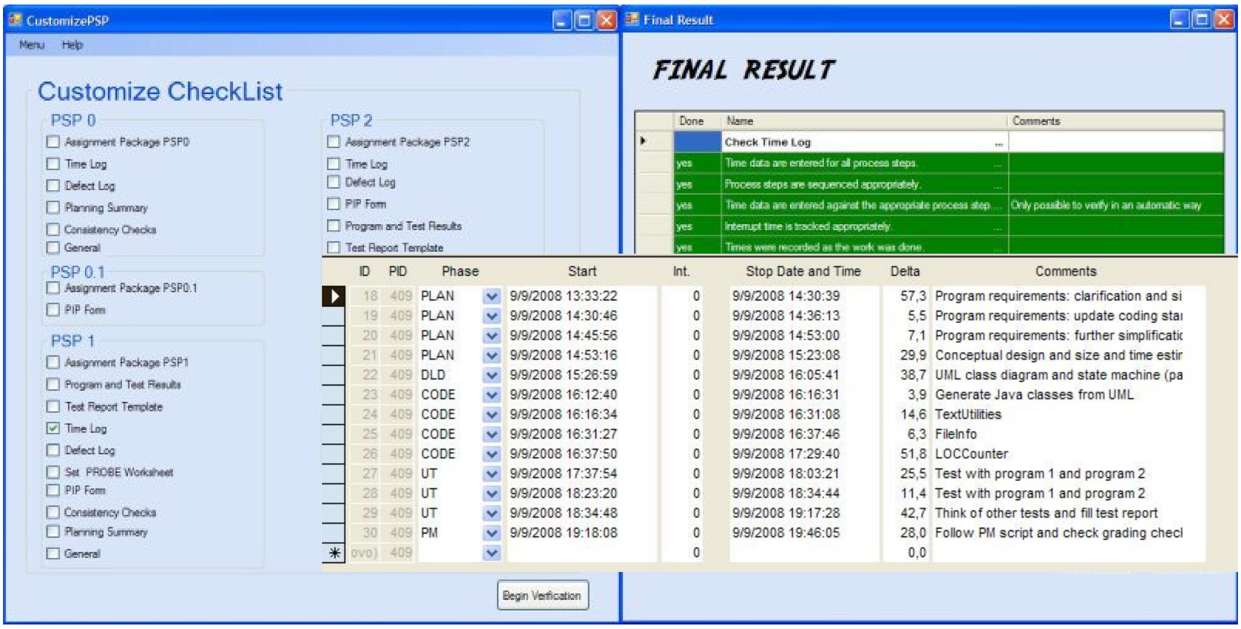
For accessing the tool the user should click on PSPChecker executable. Once entered in the tool the user can choose from remote or local import of data. The difference between those two has already been described earlier. Since in the end of this report a user's guide will be presented, this chapter will describe several conditions that could happen when selecting one PSP level with a database selected. These results are different depending on the information detailed on PSP Students Workbook.

For an easier comprehension some situations and expected results will be described next.

For these examples the level PSP customized will be used, because in this way it is easier to understand what is showed in the figures.

As a first example, the set of items from Time Log will be used, the items of most importance to check there are: *Process steps are sequenced appropriately and Times were recorded as the work was done.* So after selecting on "PSP customize", one of the several Time Log, the initial result should be like the next figure, where all data is correctly introduced.

All the following figures are part of the PSPChecker or PSP Student Workbook.



Customize CheckList

PSP 0
☐ Assignment Package PSP0
☐ Time Log
☐ Defect Log
☐ Planning Summary
☐ Consistency Checks
☐ General

PSP 0.1
☐ Assignment Package PSP0.1
☐ PIP Form

PSP 1
☐ Assignment Package PSP1
☐ Program and Test Results
☐ Test Report Template
☑ Time Log
☐ Defect Log
☐ Set PROBE Worksheet
☐ PIP Form
☐ Consistency Checks
☐ Planning Summary
☐ General

PSP 2
☐ Assignment Package PSP2
☐ Time Log
☐ Defect Log
☐ PIP Form
☐ Program and Test Results
☐ Test Report Template

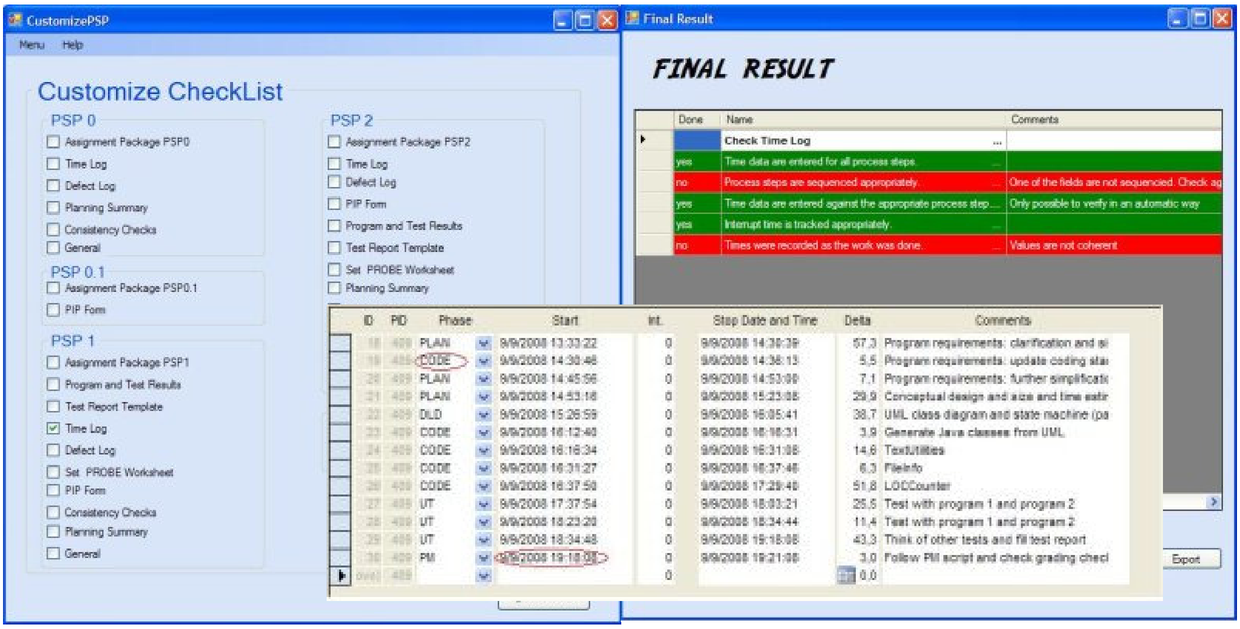
FINAL RESULT

Done	Name	Comments
	Check Time Log ...	
yes	Time data are entered for all process steps. ...	
yes	Process steps are sequenced appropriately. ...	
yes	Time data are entered against the appropriate process step....	Only possible to verify in an automatic way
yes	Interrupt time is tracked appropriately. ...	
yes	Times were recorded as the work was done. ...	

ID	PID	Phase	Start	Int.	Stop Date and Time	Delta	Comments
18	409	PLAN	9/9/2008 13:33:22	0	9/9/2008 14:30:39	57,3	Program requirements: clarification and si
19	409	PLAN	9/9/2008 14:30:46	0	9/9/2008 14:36:13	5,5	Program requirements: update coding sta
20	409	PLAN	9/9/2008 14:45:56	0	9/9/2008 14:53:00	7,1	Program requirements: further simplificati
21	409	PLAN	9/9/2008 14:53:16	0	9/9/2008 15:23:08	29,9	Conceptual design and size and time esti
22	409	DLD	9/9/2008 15:26:59	0	9/9/2008 16:05:41	38,7	UML class diagram and state machine (pa
23	409	CODE	9/9/2008 16:12:40	0	9/9/2008 16:16:31	3,9	Generate Java classes from UML
24	409	CODE	9/9/2008 16:16:34	0	9/9/2008 16:31:08	14,6	TextUtilities
25	409	CODE	9/9/2008 16:31:27	0	9/9/2008 16:37:46	6,3	FileInfo
26	409	CODE	9/9/2008 16:37:50	0	9/9/2008 17:29:40	51,8	LOCCounter
27	409	UT	9/9/2008 17:37:54	0	9/9/2008 18:03:21	25,5	Test with program 1 and program 2
28	409	UT	9/9/2008 18:23:20	0	9/9/2008 18:34:44	11,4	Test with program 1 and program 2
29	409	UT	9/9/2008 18:34:48	0	9/9/2008 19:17:28	42,7	Think of other tests and fill test report
30	409	PM	9/9/2008 19:18:08	0	9/9/2008 19:46:05	28,0	Follow PM script and check grading checl
ovo)	409			0		0,0	

Figure 33– Example 1 - working with time Log

Now we have the same example but some data is changed, like Coding Phase is used in the middle of a planning phase and one of the copied time is exactly equal in different phases. This second point can indicate that the times were copied between them instead or calculate the exact start or finish of a task or phase. The next figure shows that the red lines mean that the information is wrong, because of the reasons previously presented.



Customize CheckList

PSP 0
- ☐ Assignment Package PSP0
- ☐ Time Log
- ☐ Defect Log
- ☐ Planning Summary
- ☐ Consistency Checks
- ☐ General

PSP 0.1
- ☐ Assignment Package PSP0.1
- ☐ PIP Form

PSP 1
- ☐ Assignment Package PSP1
- ☐ Program and Test Results
- ☐ Test Report Template
- ☑ Time Log
- ☐ Defect Log
- ☐ Set PROBE Worksheet
- ☐ PIP Form
- ☐ Consistency Checks
- ☐ Planning Summary
- ☐ General

PSP 2
- ☐ Assignment Package PSP2
- ☐ Time Log
- ☐ Defect Log
- ☐ PIP Form
- ☐ Program and Test Results
- ☐ Test Report Template
- ☐ Set PROBE Worksheet
- ☐ Planning Summary

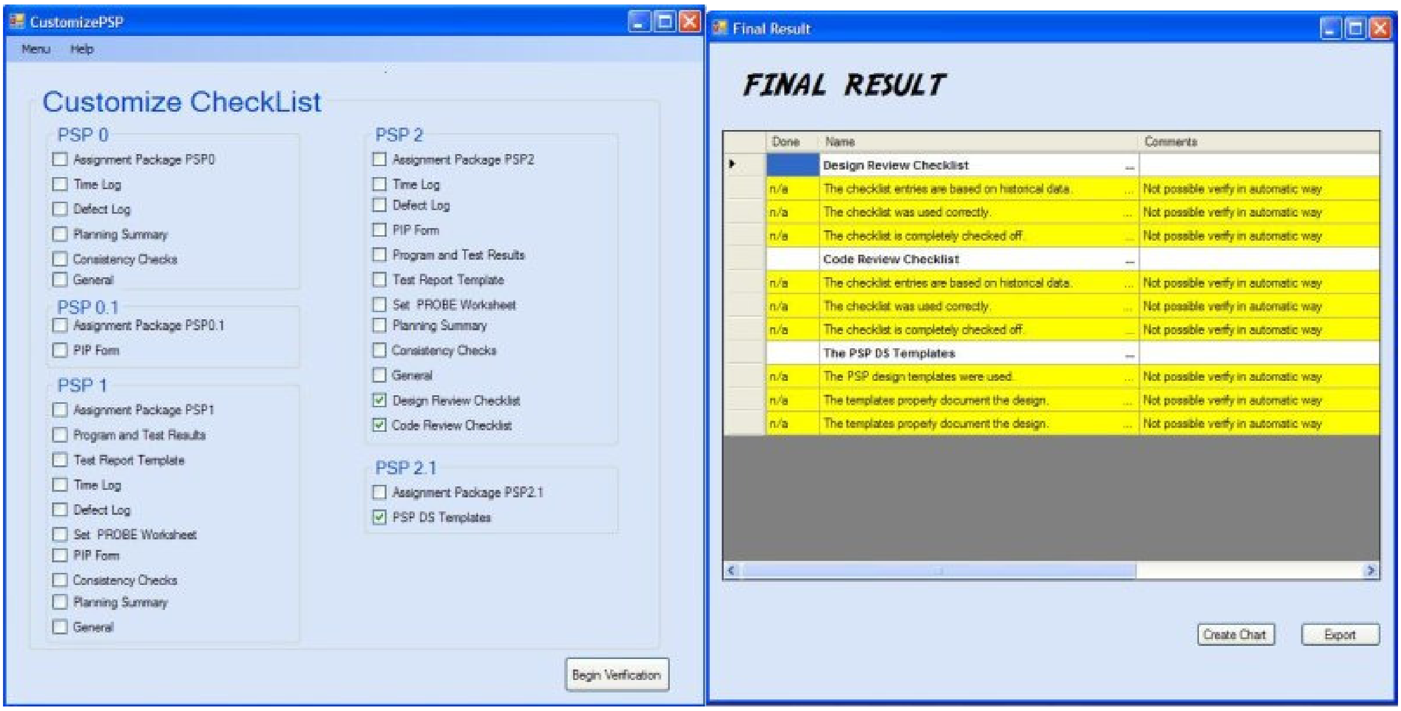
FINAL RESULT

Done	Name	Comments
	Check Time Log ...	
yes	Time data are entered for all process steps. ...	
no	Process steps are sequenced appropriately. ...	One of the fields are not sequencied. Check ag
yes	Time data are entered against the appropriate process step ...	Only possible to verify in an automatic way
yes	Interrupt time is tracked appropriately. ...	
no	Times were recorded as the work was done. ...	Values are not coherent

ID	PID	Phase	Start	Int.	Stop Date and Time	Delta	Comments
18	409	PLAN	9/9/2008 13:33:22	0	9/9/2008 14:30:39	57,3	Program requirements: clarification and si
19	409	CODE	9/9/2008 14:30:48	0	9/9/2008 14:36:13	5,5	Program requirements: update coding sta
20	409	PLAN	9/9/2008 14:45:56	0	9/9/2008 14:53:00	7,1	Program requirements: further simplificati
21	409	PLAN	9/9/2008 14:53:18	0	9/9/2008 15:23:08	29,9	Conceptual design and size and time esti
22	409	DLD	9/9/2008 15:26:59	0	9/9/2008 16:05:41	38,7	UML class diagram and state machine (pa
23	409	CODE	9/9/2008 16:12:40	0	9/9/2008 16:16:31	3,9	Generate Java classes from UML
24	409	CODE	9/9/2008 16:16:34	0	9/9/2008 16:31:08	14,6	TextUtilities
25	409	CODE	9/9/2008 16:31:27	0	9/9/2008 16:37:46	6,3	FileInfo
26	409	CODE	9/9/2008 16:37:58	0	9/9/2008 17:29:46	51,8	LOCCounter
27	409	UT	9/9/2008 17:37:54	0	9/9/2008 18:03:21	25,5	Test with program 1 and program 2
28	409	UT	9/9/2008 18:23:20	0	9/9/2008 18:34:44	11,4	Test with program 1 and program 2
29	409	UT	9/9/2008 18:34:48	0	9/9/2008 19:18:08	43,3	Think of other tests and fill test report
30	409	PM	9/9/2008 19:18:08	0	9/9/2008 19:21:08	3,0	Follow PM script and check grading checl
(new)	409			0		0,0	

Figure 34- Example 2 - working with time Log (Error Info)

In this tool there are some items that are not available to check automatically at this point and others are not available because it is just impossible to check the quality of data since these points can change for each project. The easier points to notice that are: PSP DS Templates, Code Review Checklist and Design Review Checklist. In these items PSP refers that it is important to notice if these second checklists are related to the project at that moment and if those checklists and templates makes sense. This can only be checked manually so the tool indicates that those points are N/A to be checked automatically.



Customize CheckList

PSP 0
☐ Assignment Package PSP0
☐ Time Log
☐ Defect Log
☐ Planning Summary
☐ Consistency Checks
☐ General

PSP 0.1
☐ Assignment Package PSP0.1
☐ PIP Form

PSP 1
☐ Assignment Package PSP1
☐ Program and Test Results
☐ Test Report Template
☐ Time Log
☐ Defect Log
☐ Set PROBE Worksheet
☐ PIP Form
☐ Consistency Checks
☐ Planning Summary
☐ General

PSP 2
☐ Assignment Package PSP2
☐ Time Log
☐ Defect Log
☐ PIP Form
☐ Program and Test Results
☐ Test Report Template
☐ Set PROBE Worksheet
☐ Planning Summary
☐ Consistency Checks
☐ General
☑ Design Review Checklist
☑ Code Review Checklist

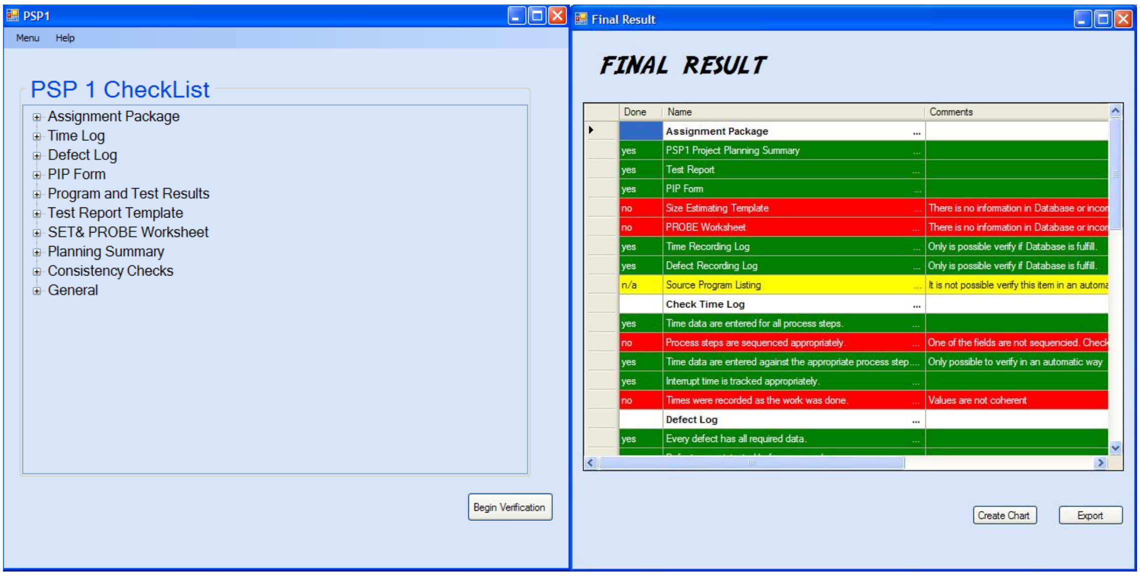
PSP 2.1
☐ Assignment Package PSP2.1
☑ PSP DS Templates

FINAL RESULT

Done	Name	Comments
	Design Review Checklist	
n/a	The checklist entries are based on historical data.	Not possible verify in automatic way
n/a	The checklist was used correctly.	Not possible verify in automatic way
n/a	The checklist is completely checked off.	Not possible verify in automatic way
	Code Review Checklist	
n/a	The checklist entries are based on historical data.	Not possible verify in automatic way
n/a	The checklist was used correctly.	Not possible verify in automatic way
n/a	The checklist is completely checked off.	Not possible verify in automatic way
	The PSP DS Templates	
n/a	The PSP design templates were used.	Not possible verify in automatic way
n/a	The templates properly document the design.	Not possible verify in automatic way
n/a	The templates properly document the design.	Not possible verify in automatic way

Figure 35 - Example 3 - working with not available options

Until now all examples were only using customize checklists, so this next example will show the structure of a PSP level 1, with the fulfillment of previous values related with previous projects. Some values were not fulfilled, so in this way PSPChecker will show points that were achieved and points that were not achieved.



PSP 1 CheckList

- Assignment Package
- Time Log
- Defect Log
- PIP Form
- Program and Test Results
- Test Report Template
- SET& PROBE Worksheet
- Planning Summary
- Consistency Checks
- General

FINAL RESULT

Done	Name	Comments
	Assignment Package	
yes	PSP1 Project Planning Summary	
yes	Test Report	
yes	PIP Form	
no	Size Estimating Template	There is no information in Database or incor
no	PROBE Worksheet	There is no information in Database or incor
yes	Time Recording Log	Only is possible verify if Database is fulfill.
yes	Defect Recording Log	Only is possible verify if Database is fulfill.
n/a	Source Program Listing	It is not possible verify this item in an automa
	Check Time Log	
yes	Time data are entered for all process steps.	
no	Process steps are sequenced appropriately.	One of the fields are not sequencied. Check
yes	Time data are entered against the appropriate process step.	Only possible to verify in an automatic way
yes	Interrupt time is tracked appropriately.	
no	Times were recorded as the work was done.	Values are not coherent
	Defect Log	
yes	Every defect has all required data.	

Figure 36 – Example 4 – Use of PSP Level 1 example

Examples like export data or import data are already referred in the requirements section. In the next examples the information that each menu contains (Menu and Help) will be introduce. In the first menu the following options can be chosen: PSP 0, PSP 0.1, PSP 1, PSP 2, PSP 2.1 and customize. The second contains the Help contents. This option was still not made because the knowledge of experts is necessary to agree which information should contain this option.

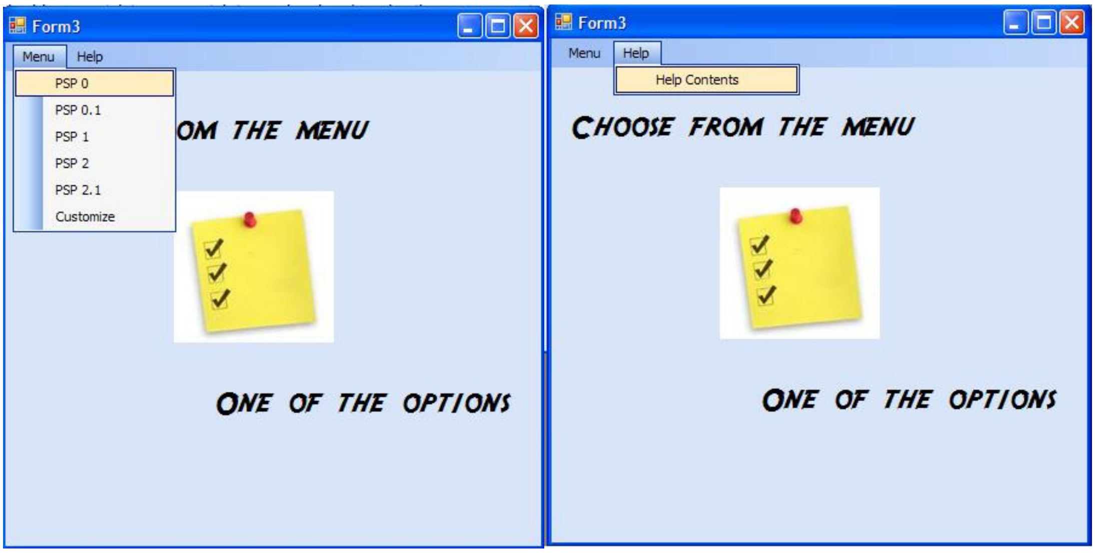


Figure 37 - Example 5 – Menu Options

Chapter 6 - Conclusions

Since the beginning of this project the main goal was clearly to help determine the main requirements and functionalities of PSPChecker. Some requirements were prioritized but in the end most of them were made to help teachers reduce feedback time and improve the teaching of PSP. So we conclude that the overall result is positive, since most of the functionalities were implemented and the ones that were not, was because there must be an agreement with a third entity (SEI/CMU, e.g. import remote data).

When this project started, a main issue was that teachers had to spend 30 minutes with each student since they had to check all items manually. However, after the creation of PSPChecker the time will be reduced because now only some manual information needs to be checked, for instance to see if tests or defects are corrected.

During the development of this project several issues appeared but they were easily solved, either with the help of advisors or internet search.

To conclude this chapter it is important to refer some interesting points for future work. As main points we add:

1. **Benefits of using PSP (business environment)** – Like the previous study of teaching PSP in universities, it would be interesting to check the benefits of using this tool as a complement to good practices in companies of different sizes.
2. **Creation of a globally tool system** – To guarantee the creation of an online version with the same characteristics as this one.
3. **Possibility of verifying data quality** – To check the data quality in database to guarantee that all information is strictly the necessary one.

References

[1]- Watts Humphrey, W.S.: A Discipline for Software Engineering. Addison-Wesley, Reading, Mass. (1995)

[2]- CMMI official page. Available From: http://www.sei.cmu.edu/cmmi/start/index.cfm

[3]- TSP official page. Available From: http://www.sei.cmu.edu/tsp/

[4]- PSP official page. Available From: http://www.sei.cmu.edu/tsp/

[5]- W. S. Humphrey, Introduction to the Personal Software Process. Addison-Wesley, 1997.

[6]- UPSP official page (Swedish university project webpage). Available From: http://www8.cs.umu.se/~jubo/UPSP/

[7]- Teaching PSP: Challenges and Lessons Learned - IEEE SOFTWARE Magazine, September/October 2002

[8]- Zeljka Car : A METHOD FOR TEACHING A SOFTWARE PROCESS BASED ON THE PERSONAL SOFTWARE PROCESS, University of Zagreb, Faculty of Electrical Engineering and Computing

[9]- M. Postema, M. Dick, J. Miller, S. Cuce, Tool Support for Teaching the Personal Software Process. Computer Science Education, Vol. 10, No. 2, 2000, pp.179-193.

[10]- D. Rosca, C. Li, K. Moore, M. Stephan, S. Weiner, PSP-EAT – Enhancing a Personal Software Process Course. Proceedings of the 31st ASEE/IEEE Frontiers in Education Conference, October 10 - 13, 2001 Reno, Nevada

[11]- I. Syu, A. Salimi, M. Towhidnejad, T. Hilburn, A Web-Based System for Automating a Disciplined Personal Software Process (PSP). Proceedings of the 10th Conference on Software Engineering Education and Training (CSEET '97), April 13 - 16, 1997, Virginia Beach, VA.

[12]- ISO/IEC 9126 .Definition Available from: http://pt.wikipedia.org/wiki/ISO/IEC_9126

[13]- WebService. Definition. Available From: http://en.wikipedia.org/wiki/Webservice

[14]- Access Database wiki source. Available From: http://en.wikipedia.org/wiki/Microsoft_Access

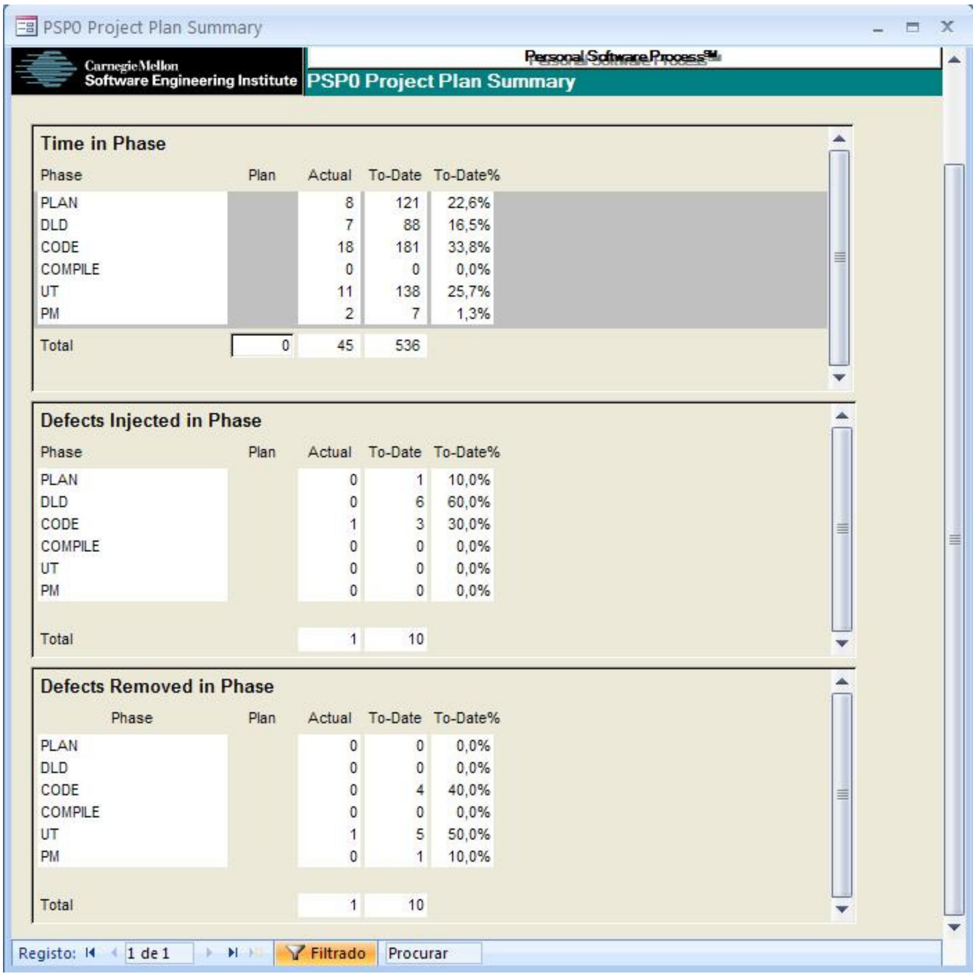
[15]- OLeDB Definition. Available From: http://en.wikipedia.org/wiki/OLEDB

[16]- DataGridViewExtension official page. Available From: http://completit.com/communityserver/blogs/dgve/default.aspx

Appendix A

A.1 Personal experience with PSP

During the realization of this project was necessary to gain knowledge relative with usage of PSP templates. From the several levels of PSP was chosen the level 0 for being the easiest to understand and fulfill all the required fields. So was decided to create as a mini project the elaboration of one of the items related defect log in this case "*Total compile defect fix times are less than compile time*". So what will be presented next is some figures with fulfill of data related with the mini project. All figures belongs to PSP Student Workbook.


PSP0 Project Plan Summary
Carnegie Mellon
Software Engineering Institute
Personal Software Process℠
PSP0 Project Plan Summary
Registo: 1 de 1
Filtrado
Procurar

Time in Phase

Phase	Plan	Actual	To-Date	To-Date%
PLAN		8	121	22,6%
DLD		7	88	16,5%
CODE		18	181	33,8%
COMPILE		0	0	0,0%
UT		11	138	25,7%
PM		2	7	1,3%
Total	0	45	536	

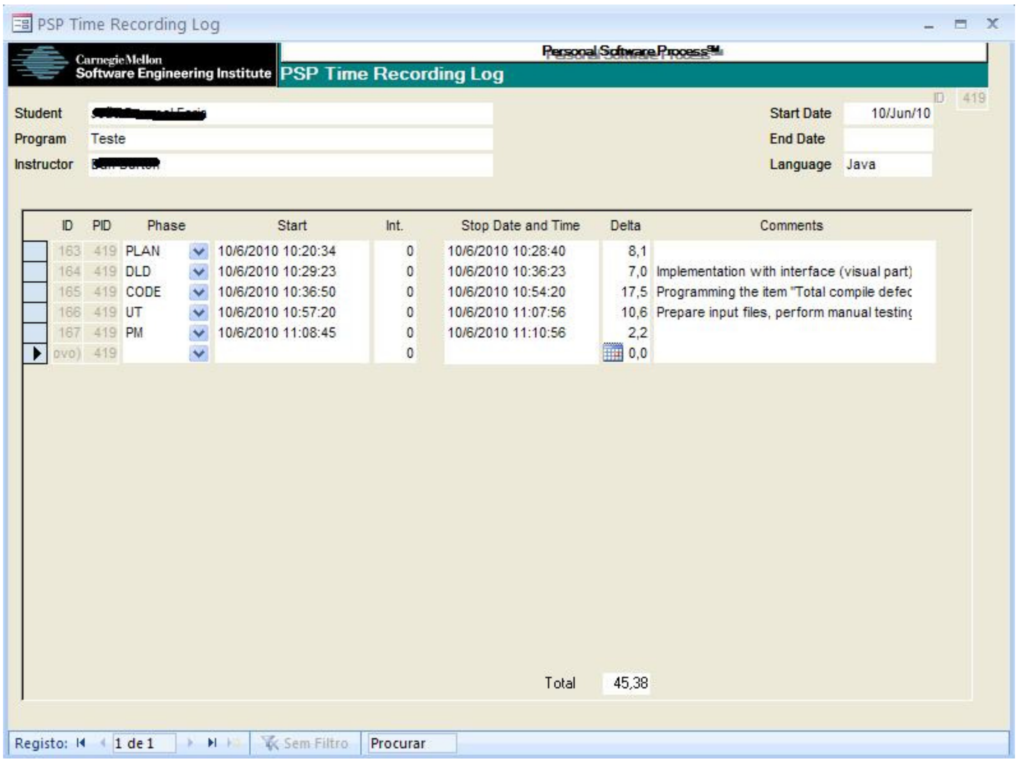
Defects Injected in Phase

Phase	Plan	Actual	To-Date	To-Date%
PLAN		0	1	10,0%
DLD		0	6	60,0%
CODE		1	3	30,0%
COMPILE		0	0	0,0%
UT		0	0	0,0%
PM		0	0	0,0%
Total		1	10	

Defects Removed in Phase

Phase	Plan	Actual	To-Date	To-Date%
PLAN		0	0	0,0%
DLD		0	0	0,0%
CODE		0	4	40,0%
COMPILE		0	0	0,0%
UT		1	5	50,0%
PM		0	1	10,0%
Total		1	10	

Figure 38- PSP Level 0 - Project Summary Example



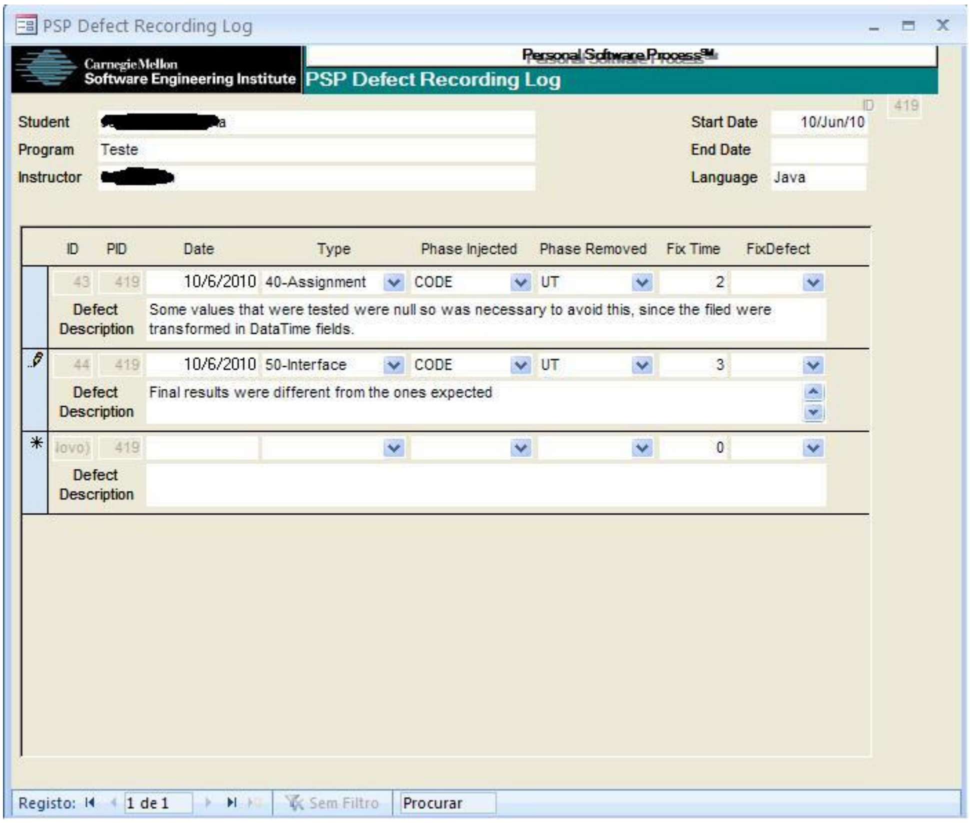
Student	[redacted]	Start Date	10/Jun/10
Program	Teste	End Date	
Instructor	[redacted]	Language	Java

ID 419

ID	PID	Phase	Start	Int.	Stop Date and Time	Delta	Comments
163	419	PLAN	10/6/2010 10:20:34	0	10/6/2010 10:28:40	8,1	
164	419	DLD	10/6/2010 10:29:23	0	10/6/2010 10:36:23	7,0	Implementation with interface (visual part)
165	419	CODE	10/6/2010 10:36:50	0	10/6/2010 10:54:20	17,5	Programming the item "Total compile defec
166	419	UT	10/6/2010 10:57:20	0	10/6/2010 11:07:56	10,6	Prepare input files, perform manual testin(
167	419	PM	10/6/2010 11:08:45	0	10/6/2010 11:10:56	2,2	
ovo)	419			0		0,0	

Total 45,38

Figure 39 - PSP Level 0 – Time Log Example



Student	[redacted]	Start Date	10/Jun/10
Program	Teste	End Date	
Instructor	[redacted]	Language	Java

ID 419

ID	PID	Date	Type	Phase Injected	Phase Removed	Fix Time	FixDefect
43	419	10/6/2010	40-Assignment	CODE	UT	2	
	Defect Description	Some values that were tested were null so was necessary to avoid this, since the filed were transformed in DataTime fields.					
44	419	10/6/2010	50-Interface	CODE	UT	3	
	Defect Description	Final results were different from the ones expected					
lovo)	419					0	
	Defect Description						

Figure 40- PSP Level 0 – Defect Log Example

As final conclusion on using PSP methodologies, personal opinion is that PSP really helps organize the realization of a project. Help improve accuracy and avoid made all the time the same mistakes. When time and other information is saved in some kind of template is easier to reach this data and understand what went wrong.