Chapter 1
An Overview of the Problem-Solving Methodology

The chapter covers:

Unit 3, Area of Study 1: Analysing information problems
An overview of the problem-solving methodology

Key terms:
Information problem, Problem Solving Methodology (PSM), analysis, design, development, evaluation, solution, requirements, constraints, scope, stakeholders, coding, validation, testing, documentation, information system, efficiency, effectiveness, functional and non-functional requirements, evaluation criteria, reporting

Note: Draft chapter 1 only. Permission is given to use these materials in induction programs taking place during November and December 2010. Copies may be made and used in classes, but please collect these from students at the conclusion of induction programs. If you are photocopying this chapter and are involved in a CAL survey, please don’t forget to list this text’s ISBN: 978-0-9750764-5-3.
Solving Information Problems

There is an art to solving information problems. All of you will have an appreciation of this, having used software that addresses particular problems in different ways – with varying success. There is a perception amongst non-programmers that programming is a very mechanical process. The truth is the exact opposite. The process of creating a solution to an information problem is one that involves high levels of creativity and often imagination. The methodology known as the Problem-Solving Methodology gives a framework to this process.

The Problem-Solving Methodology (or PSM) is central to the study of VCE Software Development. Within the discipline of Software Development, many similar methodologies exist. For the purposes of this course, the methodology that we are going to study is the PSM as defined in this chapter.

The Problem-Solving Methodology (PSM)

The Problem-Solving Methodology is a process by which a solution can be found to an information problem that exists within an information system. A software developer

“Problem-Solving Methodology: A systematic way of solving information problems”

VCAA VCE IT Study Design 2011-2014 Glossary

Figure 1: The Problem-Solving Methodology (PSM)
will follow this process to ensure that a design addresses the problems that exist, performs within specifications, integrates, if needed, with the existing system and can be achieved within budget and on time. We will define four stages, each with a number of activities within them:

1. Analysis
2. Design
3. Development
4. Evaluation

What is an information system?

The term ‘information system’ is one that is used to encompass the combination of people, procedures, equipment and data that process data and information. It is an important definition as it is easy to think of an ‘information system’ as simply consisting of hardware and software. The processes that are used by the organisation, the data within the system and the roles that people play are just as important (if not more so), and have a significant impact on the creation of a software solution.

1. Analysis

The analysis stage is where the information system is examined to determine what problems exist or how new elements can be added. There are a number of tools that can be used to analyze a system, but it is important to remember that an information problem involves users, processes, equipment and data and all of these must be considered. Analysis is often about asking targeted questions; together the answers help to build a picture of what is required. Analysis involves three main activities in preparing a solution: determining the requirements, identifying the constraints and determining the scope, all of which are influenced by the needs of the stakeholders (any parties that has a valid link or interest in the system or information problem). These are often documented in the form of a Software Requirements Specification (SRS) which is a type of report that is used by software developers to document the analysis of an information problem and to enable the design stage to begin.

Determining the solution requirements

In determining the solution requirements, the key question that needs to be asked is: What does the solution need to provide?

Any given information problem will contain aspects that will be well understood and others that will not. So what is the best way to gain a thorough understanding of the
problem in order to work out what is required of the solution?

A number of tools can be used to represent the information problem and come to a better understanding of its particular aspects. Tools such as context diagrams, data flow diagrams (DFDs) and use cases all can be used in this way, and often all three together will be used to build a complete picture of what is occurring. Collectively, these tools clearly identify the data that is currently being gathered, how it is being used and by whom, and what information is being produced. By examining the information problem in this way, it can be easier to find out what additional data will be needed to produce the software solution that is required and what functions the solution needs to provide.

The requirements of a software solution can be classified as either functional or non-functional. Functional requirements are directly related to what the software solution is required to do. Non-functional requirements are related to the attributes of the software solution, such as user-friendliness, response rates, reliability, portability, robustness and maintainability. For example, the digital blood pressure monitor shown in Figure 2 performs two very specific functions (taking a person’s blood pressure and their pulse). These are the functional requirements of the software that is running on the device. However, the software has a number of non-functional requirements relating to the way in which the information needs to be presented and the way the device responds. The device displays the pulse as a flashing heart icon and the blood pressure is displayed in a timely manner (usually less than a minute). The monitor is very easy to use with a start and stop function (user friendly) as well as a recall function that immediately displays the last blood pressure reading (response rate).

**Figure 2: Digital blood pressure monitor**

**Identifying the constraints on the solution**

Key to the success of any software solution, will be an understanding of the constraints placed upon it. Probably the most obvious constraint is cost and this will vary widely based on the size of the organization and the scale of the project. Other constraints that need to be taken into account are: speed of processing required (or available), the requirements that users have of the solution, legal requirements, security required (or imposed), compatibility with existing hardware and software within the system, the level of expertise of the users, technical support staff and the software developers themselves, the capacity of the existing system and the
availability of equipment. This is by no means an exhaustive list and sometimes the constraints that are placed on a proposed software solution are hard to predict and particular to the organization or environment.

**Determining the scope of the solution**

Similar to constraints is the topic of scope. The scope defines what the boundaries of the software solution will be. It also identifies what will the solution do, what it won’t do and what particular benefits there will be to users. Benefits are often stated in terms of efficiency and effectiveness.

**What is the difference between efficiency and effectiveness?**

Two terms that are important to understand are efficiency and effectiveness. Efficiency can be measured by examining factors such as the time it takes to complete common tasks, the cost of maintaining the system and the effort required to produce the required information. Effectiveness can be measured by examining whether the goals of the system have been met - that is, how accurate the solution is.

"**Efficiency:** A measure of how little time, cost and/or effort is applied in order to achieve intended results.”

"**Effectiveness:** A measure of how well something works, such as a solution ... and a network, that is, the extent to which it achieves its intended results

VCAA VCE IT Study Design 2011-2014 Glossary

2. **Design**

Once the analysis is complete, the design of the software solution can begin. Often if a software developer is new to a project at this stage in the PSM, they will be provided with a SRS which outlines all of the important aspects from the stage above.

The design stage consists of two main activities: planning how the solution will function given the requirements (and be presented), and determining the criteria that will be used to evaluate the solution.

**Planning solution functionality and appearance**

Designing how the solution will function is a complex activity for a software developer. It involves working out how the data that is required will be named, structured, validated, manipulated and stored. A number of tools can help to accomplish this task. Data dictionaries, data structure diagrams, input-process-output (IPO) charts, pseudocode and object descriptions can all be used to do this. This activity also involves showing how the various components of the solution relate to one another. Tools that can help to accomplish this are: data flow diagrams, context diagrams, structure charts, hierarchy charts and use cases. Lastly, it is important to be able to design how information will be presented in the software solution. This can be represented by using tools such as layout diagrams and annotated diagrams / mock ups.
Each of these tools will be explained in the chapters that follow.

“Design tools: Methods for representing the functionality and appearance of solutions”
VCAA VCE IT Study Design 2011-2014 Glossary

Determining the evaluation criteria

It may seem out of place in the design stage to be considering how the software solution will be evaluated, however, this is vitally important to the success of the project. By translating the requirements of the software solution as set out in the SRS to a number of evaluation criteria, software developers have a better sense of how the success of the solution will be ultimately judged.

3. Development

The development stage of the PSM is the stage during which the software solution is built. It consists of four main activities: coding, validation, testing and documenting. These activities do not necessarily occur in this order, as will be explained.

Coding

Coding a software solution is probably seen as the largest activity from the perspective of a software developer. In a sense this is true, but it is a task that is made much easier if due diligence has been given to the analysis and design stages. While coding a software solution, it is important that a software developer follows good coding conventions for the naming and structure of the code and includes ample internal documentation. While following these sorts of conventions may not affect the final product (and will be invisible to all but those involved in the development now and in the future), not doing these things leads to sloppy and hard-to-read code (and is seen as quite unprofessional). The coder writes code in an appropriate programming language in accordance with the plan, then tests, debugs and modifies the code as required.

Validating

Validating data is the process of determining the reasonableness of the data. The amount of validation that is included and the way in which it operates, can have a profound impact on the effectiveness of the entire software solution. As well as preventing errors from occurring, the process of ‘trapping errors’ can also be used to determine if data is reasonable.

Validation really begins in the design stage. It is in the design stage that important decisions about how data will be collected, processed and output will be made. Validation is also strongly influenced by user interface design. The actual validation coding takes place in the development stage.

Testing

Testing is often an ongoing activity during the development of a software solution, and as a programmer adds elements to the program, they will test them to see that they are working and modify or fix them as needed. However, the formal activity of testing a software solution is usually conducted at the conclusion of the development of the software and is done using an exhaustive grid
Chapter 1: An Overview of the Problem-Solving Methodology

which covers both the valid and invalid possibilities of the software’s use.

When undertaking this task, the first step is to list all of the tests that will be undertaken. This list can be quite long as it will be designed to cover all of the combinations of valid and invalid input as well as use of the software. Test data will be constructed to perform each of these tests and often the expected behaviour or output from each will be documented. The tests will then be carried out and the behaviour of the software solution compared to the expected result in each case.

There is some argument that the best time to compile a list of tests is during the design stage. Doing so gives those coding the solution a valuable insight into the exact parameters of both the input and output.

The final step of the testing process is correcting those errors that have been detected, after which the test data that initially triggered the incorrect result is tested again to ensure that the software solution has been fixed.

**Documentation**

Documentation is then written to support the variety of users of the software solution. Documentation can take a number of forms. Internal documentation is placed inside the program code and assists future programmers who wish to modify the software solution. System support documentation can be in electronic or hardcopy form. Different types of system support documentation can be produced for different groups from those using the system to those maintaining it.

4. Evaluation

The evaluation of a software solution is an important activity that will usually take place after the solution has been in full operation for a while. This time period varies, but it needs to be long enough so that users of the system are comfortable with it and hopefully not resentful of it (as can happen when new software solutions are introduced). An evaluation can (and should) contain many different elements. There are two key activities involved in evaluating a software solution: determining a strategy that will be used find out the extent that the solution meets the required needs and reporting on the success of the solution.

**Determining a strategy**

What are the best ways to find out if the software solution has met the required needs? A strategy to determine this will include a timeline for the evaluation, what data will be collected and by what methods and how the data relates to the evaluation criteria set out in the design stage. Note that this activity is more complex than simply asking a series of questions of the users.

**Reporting**

How does the software solution address those evaluation criteria determined in the design stage? This activity is concerned with reporting on this direct comparison and will document how successful the software solution has been in tackling the information problem.
Context Questions

1. What is the purpose of the Problem-Solving Methodology?
2. What are the four components of an information system?
3. What is a Software Requirements Specification used for?
4. What is the difference between functional and non-functional requirements of a solution?
5. List four common constraints on solutions.
6. What are the benefits in determining the evaluation criteria for a software solution during the design stage?
7. What advantages are there in using coding conventions for the naming of program elements?
8. What is validation?
9. What advantages are there in compiling test data during the design stage?
10. What does the term ‘due diligence’ mean?
11. What possible consequences are there in evaluating a software solution too early?