INFORMATION SYSTEMS AND SOFTWARE ENGINEERING

[1]VO KY QUANG, [2]NOR HIDAYATI ZAKARIA, [3]BUI TRONG HIEU

[1]Asia e University, Malaysia. [2]Information Systems Department, Faculty of Computing, University Teknologi Malaysia, Malaysia
[3]Faculty of Information Technology, HoChiMinhCity University of Transport, Ho Chi Minh City, Vietnam
E-mail: [1]quang.vo668@gmail.com, [2]hidayati@utm.my, [3]hieubt@hcmutrans.edu.vn

Abstract- In my research paper, we want to understand more about Information Systems and Software Engineering. We understand what software engineering is and why it is important; the development of different types of software systems may require different software engineering techniques and some ethical and professional issues that are important for software engineers. With information systems, We understand more about it by learning about the concepts, components of information systems, Computer hardware, Computer software, and Telecommunications, Database and data warehouse.

Keywords- Information Systems, Software Engineering, Computer hardware, Computer software, Telecommunications, Databases, Data Warehouse.

I. INTRODUCTION

Software engineering is concerned with all aspects of software production from the early stages of system specification through to maintaining the system after it has gone into use. In this chapter, we will explain the following:

• the definition of computer science and software engineering and how the two are different
• software engineering is similar to other engineering disciplines and what that means for software engineers
• the unique challenges of software engineering
• software development models and processes and their component parts, software development practices

Software engineering has progressed very far in a very short period of time, particularly when compared to classical engineering field (like civil or electrical engineering). In the early days of computing, not much more than 50 years ago, computerized systems were quite small. Most of the programming was done by scientists trying to solve specific, relatively small mathematical problems. Errors in those systems generally had only “annoying” consequences to the mathematician who was trying to find “the answer.” Today we often build monstrous systems, in terms of size and complexity.

What is also notable is the progression in the past 50 years of the visibility of the software from mainly scientists and software developers to the general public of all ages.

“Today, software is working both explicitly and behind the scenes in virtually all aspects of our lives, including the critical systems that affect our health and well-being.” (Pfleeger, 1998) Despite our rapid progress, the software industry is considered by many to be in a crisis.

Like all engineering discipline, software engineering is driven almost by three major factors: cost, schedule, and quality. The cost of developing a system is the cost of the resources used for the system, which in the case of software, are the manpower, hardware, software, and other support resources. Generally, the manpower component is predominant, as software development is largely labor-intensive and the cost of computing systems is now quite low. Hence, the cost is considered to be the total number of Person-months spent in the project. Schedule is an important factor in many projects. Business trends are dictating that the time to market of a product should be reduced; that is, the cycle time from concept to delivery should be small.

Though high quality, low cost (or high productivity), and small cycle time is the primary objectives of any project, for an organization there is another goal: consistency. An organization involved in software development does not just want low cost and high quality for a project, but it wants these consistently. In other words, a software development organization would like to produce consistent quality with consistent productivity. Consistency of performance is an important factor for any organization, it allows an organization to predict the outcome of a project with reasonable accuracy, an to improve its processes to produce higher-quality products and to improve its productivity.

Information system: an integration set of components for collecting, storing, and processing data and for delivering information, knowledge, and digital products. Business firms and other organizations rely on information systems to carry out and manage their operations, interact with their customers and suppliers, and compete in the marketplace.

The value of research, when study subjects will help readers understand more about Information Systems and Software Engineering. And understand more about concepts, components of information systems, Computer hardware, Computer software, Telecommunications, Databases and data warehouses,… We know what software engineering is and why it is important; the development of different types of software systems may require different software engineering techniques and some

ethical and professional issues that are important for software engineers.

II. LITERATURE

1. Software engineering

The wider context for this study is that of investigating the use of the evidence-based paradigm in software engineering. The possibility of applying the evidence-based paradigm to the software engineering field was raised, discussed and enthusiastically supported at ICSE 2004 (Kitchenham et al., 2004). The goal of evidence-based software engineering (EBSE) is summarized by Kitchenham et al., as being: "to provide the means by which current best evidence from research can be integrated with practical experience and human values in the decision making process regarding the development and maintenance of software".

Performing a systematic review involves several discrete activities, which can be grouped into three main phases: planning; conducting the review; and reporting the review. Fig. 1 illustrates the overall 10-stage review process.

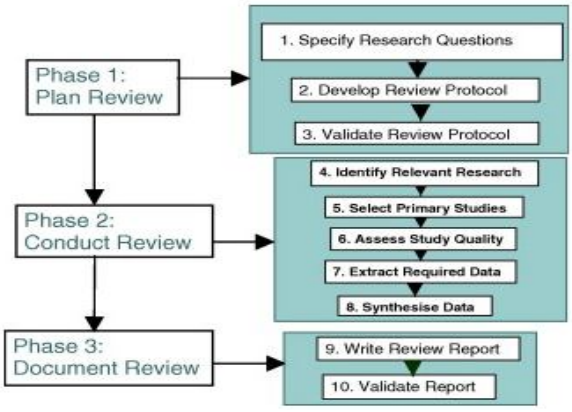

Fig 1. Systematic literature review process.

Systematic literature reviews are primarily concerned with the problem of aggregating empirical evidence which may have been obtained using a variety of techniques, and in (potentially) widely differing contexts—which is commonly the case for software engineering. While they are used in information systems research (Webster and Watson, 2002), they are less common in software engineering (however, see (Glass et al., 2002) as an example of a secondary study that samples literature within the software engineering domain). Indeed, at the present time, outside of information systems research, reviews in any form, as well as review journals are really not part of the computing research culture, which focuses almost entirely on publishing primary studies.

To understand the role of evidence, we need to recognize that, across a wide spectrum of disciplines of study, there is a common requirement to find objective practices that can be employed for aggregating the outcomes of different empirical studies in a consistent manner. The range of forms and issues is very wide: at the one end, aggregating (say) experimental studies measuring the mass of the electron is largely a matter of using mathematically based transformations to adjust for variations in experimental conditions; whereas drawing together the results from a set of surveys, that may have employed different sets of questions and been administered to rather different populations, presents a much less mathematically tractable problem. One of the key issues underlying this difference is the role of the human in the process of data collection: in the former the only involvement is as an external observer, while in the latter, the human is a participant in the treatment itself. In order to investigate the applicability of systematic literature reviews to the software engineering domain, the authors (with others) have undertaken, or are in the process of undertaking, a number of reviews which aim to address a range of software engineering questions. These reviews are summarized in the following sections using the structured abstract headings (context, objectives, methods, results, conclusions) which form part of the recommendations for reporting systematic reviews (Khan et al., 2001). In addition, for each of the reviews, there is a brief introduction and a description of its current state. The reviews have used Kitchenham's guidelines as described in (Kitchenham, 2004) or, in some cases, in earlier versions of the report.

2. Information Systems

A computer (-based) information system is essentially an IS using computer technology to carry out some or all of its planned tasks. The basic components of computer based information system are:

- Hardware-these are the devices like the monitor, processor, printer and keyboard, all of which work together to accept, process, show data and information.
- Software-are the programs that allow the hardware to process the data.
- Databases-are the gathering of associated files or tables containing related data.
- Networks- are a connecting system that allows diverse computers to distribute resources.
- Procedures-are the commands for combining the components above to process information and produce the preferred output.

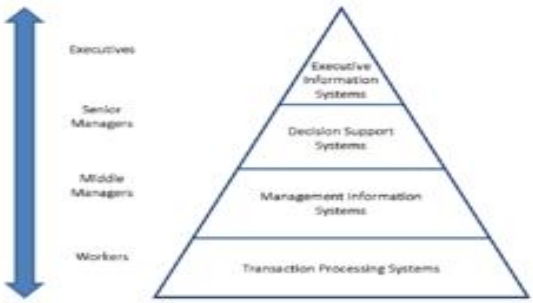

Fig 2. A four level of Information Systems

The first four components (hardware, software, database, and network) make up what is known as the information technology platform. Information technology workers could then use these components to create information systems that watch over safety measures, risk and the management of data. These actions are known as information technology services.

3. **Difficulties in the development of Software Engineering [26]**

Errors, mistakes and fails in software are common, usually a fail cause inconvenience but no serious long-term damages or something as serious as huge money loss or even health damage. However, in some systems failure can have very big and serious consequences. This type of system is called critical system. There are three main types of critical systems:

- Safety-critical systems. Fails in this system may result in injury, death or environmental damage. For example, space shuttle with astronauts on board. If something goes wrong with navigation system, people may die.
- Mission-critical systems. Fails in this system may result in failure of some goal-directed activity and main objective of the system may not be reached. The same space-shuttle is an example of mission-critical system, because even without astronauts taken in count, the whole mission might be failed.
- Business-critical systems. Fails may cause loss of money for customers using this system. Bank money management system is an example.

4. **Solutions for the development of System Engineering [26]**

System functional requirements may be generated to define error checking and recovery facilities and features that provide protection against system failures. Non-functional requirements may be generated to define the required reliability and availability of the system.

Critical systems specifications' objective is to understand the risks faced by the system and generate dependability requirements to cope with them. The process of risk analysis consists of four steps:

Risk identification: Potential risks that might arise are identified. These are dependent on the environment in which the system is to be used. In safety-critical systems, the principal risks are hazards that can lead to an accident. Experienced engineers, working with domain experts and professional safety advisors, should identify system risks. Group working techniques such as brainstorming may be used to identify risks.

Risk analysis and classification: The risks are considered separately. Those that are potentially serious and not implausible are selected for further analysis. Risks can be categorized in three ways:

- Intolerable. The system must be designed in such a way so that either the risk cannot arise or, if it does arise, it will not result in an accident. Intolerable risks are those that threaten human life or the financial stability of a business and which have a significant probability of occurrence.
- As low as reasonably practical (ALARP). The system must be designed so that the probability of an accident arising because of the hazard is minimized, subject to other considerations such as cost and delivery. ALARP risks are those which have less serious consequences or which have a low probability of occurrence.
- Acceptable. While the system designers should take all possible steps to reduce the probability of an 'acceptable' hazard arising, these should not increase costs, delivery time or other non-functional system attributes

Risk decomposition: Each risk is analyzed individually to discover potential root causes of that risk. Different techniques for risk decomposition exist. The one discussed in the book is Fault-tree analysis, where analyst puts hazard at the top and place different states which can lead to that hazard above. States can be linked with 'or' and 'and' symbols. Risks that require a combination of root causes are usually less probable than risks that can result from a single root cause.

Risk reduction assessment: Proposals for ways in which the identified risks may be reduced or eliminated are made. Three possible strategies of risk deduction that can be used are:

- Risk avoidance. Designing the system in such a way that risk or hazard cannot arise.
- Risk detection and removal. Designing the system in such a way that risks are detected and neutralized before they result in an accident.
- Damage limitation. Designing the system in such a way that the consequences of an accident are minimized.

III. METHODOLOGY

Depending on the kind of problem to solve and the context of the problem, Science or Engineering, different research methods are used [4], [6]. Moreover, scientific research methods cannot always be applied to engineering research problems [11].

Scientific research problems are similar to problems broached in traditional sciences and can have either an empirical or a cultural and social nature. When the Science has an empirical nature, quantitative research methods can be applied [9]; these methods try to solve problems like: "what model method is more efficient?".

When the Science has a social and cultural nature, qualitative research methods can be applied [12] and these methods can seek to answer questions like: "what factors make a given software process unacceptable to the company?" or "why is one information systems development tool more acceptable than another?". In both, it is necessary

certain knowledge of the reality: the object of study is an existing object in the world. Thus, this kind of problems use the research methods proposed by traditional sciences, as they study phenomena and objects of the world regardless of how they were created.

However, there is not any precise method to broach Engineering research problems and the search for a method appropriate to this field is becoming a research field in its own right [5], [7], [8], [12], [13]. The solution of problems purely concerning Engineering requires methods of a different kind since in these cases it is directly possible to apply neither empirical methods nor methods which have to do with social and cultural component as the object of study does not yet exist [14]. Furthermore, in the case of Engineering, it is necessary a major component of creativity, which makes it difficult to draw up a universal method for solving problems within this field. For instance, "what research method would be valid for the specification of a new methodology for software development?". It would be necessary to study existing methodologies, reflecting on them to determine their advantages and disadvantages and proposing a new one, which, while retaining the advantages of the methodologies studied, would, as far as possible, lack their shortcomings. Arriving at a better final proposition would largely depend on the creativity and common sense applied to the construction of the new method. This method is proposed by [10] and it is stated the method it is applied in Engineering consist in the formulation of experiences and the identification of the best practices.

1. Types of Research Questions

Research questions may be about methods for developing software, about methods for analyzing software, about the design, evaluation, or implementation of specific systems, about generalizations over whole classes of systems, or about the sheet feasibility of a task. Table 1 shows the types of research questions software engineers ask, together with some examples of specific typical questions.

Type of question	Examples
Method or means of development	How can we do/create (or automate doing) X? What is a better way to do/create X?
Method for analysis	How can I evaluate the quality/correctness of X? How do I choose between X and Y?
Design, evaluation, or analysis of a particular instance	What is a (better) design or implementation for application X? What is property X of artifact/method Y? How does X compare to Y? What is the current state of X / practice of Y?
Generalization or characterization	Given X, what will Y (necessarily) be? What, exactly, do we mean by X? What are the important characteristics of X? What is a good formal/empirical model for X? What are the varieties of X, how are they related?
Feasibility	Does X even exist, and if so what is it like? Is it possible to accomplish X at all?

Table 1: Research questions in software engineering

2. Types of Research Results

Research yields new knowledge. This knowledge is expressed in the form of a particular result. The result may be a specific procedure or technique for software development or for analysis. It may be more general, capturing a number of specific results in a model; such models are of many degrees of precision and formality. Sometimes, the result is the solution to a specific problem or the outcome of a specific analysis. Finally, as Brooks observed, observations and rules of thumb may be good preliminary results. Table 2 lists these types, together with some examples of specific typical questions.

Type of result	Examples
Procedure or technique	New or better way to do some task, such as design, implementation, measurement, evaluation, selection from alternatives, Includes operational techniques for implementation, representation, management, and analysis, but not advice or guidelines
Qualitative or descriptive model	Structure or taxonomy for a problem area; architectural style, framework, or design pattern; non-formal domain analysis Well-grounded checklists, well-argued informal generalizations, guidance for integrating other results,
Empirical model	Empirical predictive model based on observed data
Analytic model	Structural model precise enough to support formal analysis or automatic manipulation
Notation or tool	Formal language to support technique or model (should have a calculus, semantics, or other basis for computing or inference) Implemented tool that embodies a technique
Specific solution	Solution to application problem that shows use of software engineering principles – may be design, rather than implementation Careful analysis of a system or its development Running system that embodies a result; it may be the carrier of the result, or its implementation may illustrate a principle that can be applied elsewhere
Answer or judgment	Result of a specific analysis, evaluation, or comparison
Report	Interesting observations, rules of thumb

Table 2: Research results in software engineering

3. Types of Research Validations

Good research requires not only a result, but also clear and convincing evidence that the result is sound. This evidence should be based on experience or systematic analysis, not simply persuasive argument or textbook examples. Table 3 shows some common types of validation, indicating that validation in practice is not always clear and convincing.

Type of validation	Examples
Analysis	I have analyzed my result and find it satisfactory through ...ormal analysis) ... rigorous derivation and proof (empirical model) ... data on controlled use(controlled ... carefully designed statistical experiment) experiment
Experience	My result has been used on real examples by someone other than me, and the evidence of its correctness / usefulness / effectiveness is ...alitative model) ... narrative(empirical model, ... data, usually statistical, on practice (notation, tool) ... comparison of this with similar results in technique) actual use
Example	Here's an example of how it works on (toy example) ... a toy example, perhaps motivated by reality (slice of life) ...a system that I have been developing
Evaluation	Given the stated criteria, my result... (descriptive model) ... adequately describes the phenomena of interest ... (qualitative model) ... accounts for the phenomena of interest... (empirical model) ... is able to predict ... because ..., or ... gives results that fit real data ... Includes feasibility studies, pilot projects
Persuasion	I thought hard about this, and I believe... (technique) ... if you do it the following way, ... (system) ... a system constructed like this would ... (model) ... this model seems reasonable Note that if the original question was about feasibility, a working system, even without analysis, can be persuasive
Blatant assertion	No serious attempt to evaluate result

Table 3: Validation techniques in software engineering

IV. COLLETION AND ANALYSIS DATA

1. Software Engineering

The challenge of collecting software engineering data is to make sure that the collected data can provide useful information for project, process, and quality management and, at the same time, that the data collection process will not be a burden on development teams. Therefore, it is important to consider carefully what data to collect. The data must be based on well-defined metrics and models, which are used to drive improvements. Therefore, the goals of the data collection should be established and the questions of interest should be defined before any data is collected. Data classification schemes to be used and the level of precision must be carefully specified. The collection form or template and data fields should be pretested. The amount of data to be collected and the number of metrics to be used need not be overwhelming. It is more important that the information extracted from the data be focused, accurate, and useful than that it be plentiful. Without being metrics driven, over-collection of data could be wasteful. Over collection of data is quite common when people start to measure software without an a priori specification of purpose, objectives, profound versus trivial issues, and metrics and models.

Basili and Weiss (1984) propose a data collection methodology that could be applicable anywhere. The schema consists of six steps with considerable feedback and iteration occurring at several places:

- Establish the goal of the data collection.
- Develop a list of questions of interest.
- Establish data categories.
- Design and test data collection forms.
- Collect and validate data.
- Analyze data.

The importance of the validation element of a data collection system or a development tracking system cannot be overemphasized.

2. Information Systems

Information System is an information system in which part of the collection, transmission, storage and data processing is done using the elements or components of IT, that means modern computing and communications, specialized software, procedures and techniques plus specific specialized personnel.

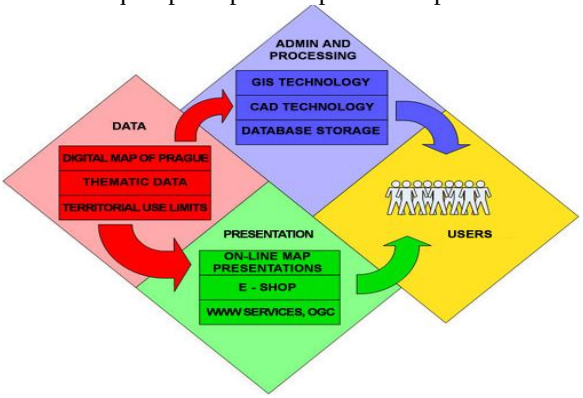

Fig 6. information system map

An information system can be defined technically as a set of interrelated components that collect (or retrieve), process, store, and distribute information to support decision making and control in a organization. In addition to support decision marking, coordination, control, information systems may also help managers and workers analyze problems" according to Laudon and Laudon.

Types of Information Systems:

- Transaction Processing Systems (TPS) - System that performs or records daily routine transactions such as sales order entry, payroll, employee record keeping, and shipping.
- Management Information Systems (MIS) - It designates a specific category of information to middle management. It is to monitor and control the business and predict future performance.
- Decision-Support Systems (DSS) - Support no routine decision making for middle management. Example of a no routine decision: What would be the impact on production schedules if we were to double sales in the month of December?
- Executive Support Systems (ESS) - Help Senior management to address strategic issues and long-term trends, both in the firm and in the external environment. Examples: What will employment levels be in five years? What products should we be making in five years?

The computer system is part of the information system, namely that part which includes collection, processing and automated data transmission and information from the information system, integrated computer system – specific to certain areas of activity – e.g. system failure, fine, banking) is the only system that provides data entry and processing multiple thereof depending on the various forms of requests by users.

Information technology is a contemporary term that describes a combination of computing technologies – equipment and software – communication technology – data transmission networks, images and voice.

Management information systems – management models a domain regroups own procedures. In practical activities to identify a series of domain-specific models, such as: – manufacturing technologies, specific sales, accounting.

System analysts are those specialists who understand both aspects of facilities and limits offered by information technology and data processing requirements necessary information-decision process of economic agents.

Transaction processing systems (SPT) are applications of information system that lets the collection, storage and processing of data resulting from the conduct daily transactions, providing the database.

The complexity of a large software system surpasses the comprehension of any one individual. To better control the development of a project, software managers have identified six separate stages through

which software projects pass; these stages are collectively called the software development life cycle:

* Requirements analysis; Specification; Design; Coding; Testing; Operation and maintenance.

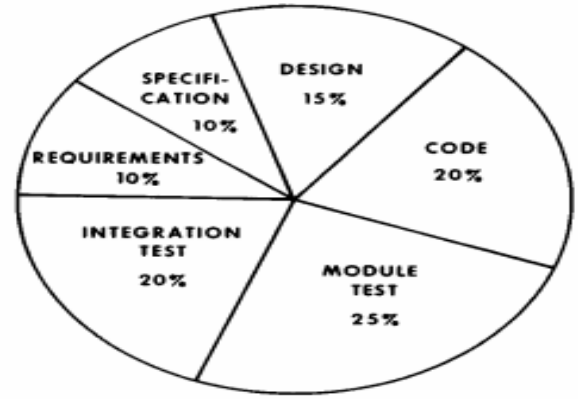

Fig 7: Effort required on various development activihes (excluding maintenance)

Figure 7 shows the disposition of software costs in developing a new project.

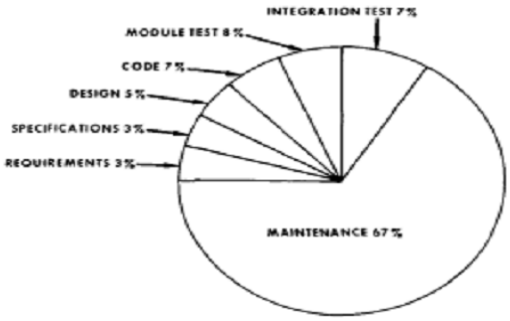

Fig 8: True effort on many large-scale software systems

The division of effort indicated in Figure 8 greatly affects system development. Because of hidden maintenance costs, techniques that rush development and provide for very early initial implementation may be trading early execution for a much more extensive maintenance operation. The maintenance problem is sometimes referred to as the "parts number explosion.".

To avoid this growth, systems often receive updates, called releases, at flexed intervals. A useful tool for dealing with myriad maintenance problems is a "systems database" started during the specifications stage. This database records the characteristics of the different installations. It includes the procedures for reporting, testing, and repairing errors before distributing the corrections.

V. DISCUSION OF RESULTS

We understood more about Information Systems and Software Engineering. We understand what software engineering is and why it is important; the development of different types of software systems may require different software engineering techniques and some ethical and professional issues that are important for software engineers. With information systems, We understand more about it by learning about the concepts, components of information systems, Computer hardware, Computer software, Telecommunications, Databases and data warehouses.

At the same time, We also better understand the difficulties in the development of an information system or software. So that we may have plans to develop them better.

Information Systems and Software Engineering are two areas are so broad that I cannot learn in detail and carefully about 2 system this large. Concepts related to them a lot and I think I will try to learn them as well as the history and process of system operation.

CONCLUSION

Although coming from different backgrounds and using different approaches, the students produced similar and compatible information systems. The scheduling team's familiarity with the project was only a small advantage. The scheduling (information systems) team used a database approach. First the database tables were constructed and populated with test data. Then the queries necessary for the forms and reports were built. Finally the system was tested.

The billing (software engineering) team's knowledge of UML allowed them to understand the design and build the desired system. They used the use cases, associated classes, and sequence diagrams extensively to black-box program the required modules. After all the modules were completed and tested they were combined into the system.

In general, software engineers adopt a systematic and organized approach to their work, as this is often the most effective way to produce high-quality software. However, engineering is all about selecting the most appropriate method for a set of circumstances so a more creative, less formal approach to development may be effective in some circumstances. Less formal development is particularly appropriate for the development of web-based systems, which requires a blend of software and graphical design skills

Technically the resulting system was a great success. I learned a great deal about systems requirements elicitation, system design, software and database construction, and systems implementation. They learned the importance of good design documentation. They also learned about the difficulties in dealing with a real client unfamiliar with computers and automation. The resulting system was designed to improve on the existing system, and was fundamentally built to the design. Clearly it was a technical and educational success. There is a shortage of skilled software professionals to fill the needs of industry (Information Technology Association of America, 2001). Graduates of schools that have programs for conceptualizers, developers and modifiers are well prepared to fill positions in

software development and database development. It matters little whether the education is in Software Engineering, Information Systems, or Computer Science.

REFERENCES

[1]. https://ifs.host.cs.standrews.ac.uk/Books/SE9/SampleChapters/PDF/Chap1-Introduction.pdf
[2]. http://agile.csc.ncsu.edu/SEMaterials/Introduction.pdf
[3]. http://www.britannica.com/topic/information-system
[4]. Chalmers, A. (1984). La Ciencia y cómo se Elabora. Siglo XXI de España Editores, S. A.
[5]. Madrid.
[6]. Dobson, P. J. (2001) The Philosophy of Critical Realism-An Opportunity for Information
[7]. Systems Research. Information Systems Frontiers, 3(2), pp. 199-210.
[8]. Fetzer, J. H. (1993) Philosophy of Science. Paragon House. United States.
[9]. Glass, R.L., Vessey, I. Ramesh, V. (2002) Research in Software Engineering: an analysis of
[10]. the literature. Information and Software Technology. Elsevier Science B.V. N. 44, pp. 491-506.
[11]. Gregg, D. G., Kulkarni, U. R. and Vinzé, A. S. (2001) Understanding the Philosophical
[12]. Underpinnings of Software Engineering Research in Information Systems. Information
[13]. Systems Frontiers, 3(2), pp. 169-183.
[14]. Juristo N. and Moreno A. M (2001) Basics of Software Engineering Experimentation.
[15]. Kluwer Academic Publisher.
[16]. Klein H. and Hirschheim R. (2003) Crisis in the IS Field. A Critical Reflection on the State
[17]. of the Discipline. Journal of AIS, 4, 10.
[18]. Marcos, E. (2002) Investigación en Ingeniería del Software vs Desarrollo Software. Actas
[19]. de 1er Workshop en Métodos de Investigación y Fundamentos Filosóficos en IS y SI.
[20]. November, pp. 136-149.
[21]. Miles M. B. and Huberman, A. M. (1984). Quality Data Analysis: A sourcebook of New
[22]. Methods. SAGE. NewBury Park-CA.
[23]. Myers, M. D. (1997) Qualitative Research in Information Systems. MIS Quarterly, 21(2),
[24]. June, pp. 241-242.
[25]. 24. Wohlin C. Et al. (2000) Experimentation in Software Engineering. An introduction. Kluwer
[26]. 25.http://web2.aabu.edu.jo/tool/course_file/lec_notes/902340_SE%20Part%201.pdf
[27]. 26. http://freetonik.com/text/software-engineering-notes/

★★★