International Journal of Scientific Research in Science and Technology
Print ISSN: 2395-6011 | Online ISSN: 2395-602X (www.ijsrst.com)
doi : https://doi.org/10.32628/IJSRST218645

A Security-Specific Knowledge Modelling Approach, Software Practices, and Data Centre Infrastructure for Securing Software Engineering Technologies

Abdul Joseph Fofanah*[1], Habibu Rasin Bundu*[2], Jonathan Gibrill Kargbo*[3] , Ahmed Fofana*[1]

*[1]Department of Mathematics and Computer Science, Milton Margai Technical University, Freetown, Western Area Rural, Sierra Leone

[2]Department of Integrated Science, Milton Margai Technical University, Freetown, Western Area Rural, Sierra Leone

[3]Department of Information and Communication Technology, Milton Margai Technical University, Freetown, Western Area Rural, Sierra Leone

Article Info
Volume 8, Issue 6
Page Number : 324-342

Publication Issue
November-December-2021

Article History
Accepted : 01 Nov 2021
Published : 07 Dec 2021

ABSTRACT

The advancement of emerging technological tools in software engineering is an important element in the design and development of software systems. In this paper, we present an analysis of theory and practice including methodology of software products for both large and complex requirements and development analysis, and synthesis. The paper is presented in two folds: Part-I describes a security-specific knowledge of modelling approach for securing software engineering and typical projects implemented in data centre infrastructure. In relation to software engineering practice and theory, we analysed the key parameters indicators of software development projects and the elements of a system that encapsulate the customer, developer, and the researcher as stakeholders in a software development project, whereas the elements of a system entail computer, data validation, mailroom, and computation with paychecks and pay-information. The modelling process and life cycle model includes some major processes in software development such as users' resources, production of the final product, subprocesses with hierarchy links, process activity, guiding principles, and outcomes of a software requirement specification. In Part-II, an overview of data centre infrastructure and with some schematic illustration for each phase of the construction and implementation of a data centre. The project involves a system and process that creates it with *prepare, design, acquire, and implement* as a process model whereas actors create the project model. In the context of data centre life cycle model, *prepare and design* form the *construct* or *build* phase, and *maintainability* and *optimization* form the *engineering* phase. All these

formulates the project model as the building blocks of data centre. The business need for the construction of the data centre (*prepare, design, acquire, and implement*) are the knowledge-based of the process model phases to produce an overall system we called the four phases of data centre project process.

Keywords: Software engineering, approach, requirement, modelling, theory and practice, SDLC, architecture, data centre

I. INTRODUCTION

Software Engineering Modelling Approach

The theory and practice method of software cannot be achieved if one fails to clarify some of the important components in software engineering techniques and how software is manufactured for the end-user. Software is a multi-disciplinary field of study, encapsulating many social and technological bridges and boundaries. Software engineers have complex tasks to understand the mechanism of software construct and how to maintain complex systems developing software systems, we need to investigate not just the tools and processes they use, but also the social and cognitive processes surrounding them. This requires the study of human activities. We need to understand how individual software engineers develop software, as well as how teams and organizations coordinate their efforts [1].

However, because of the importance of software development to human activities, many of the research methods and implementation methods that are appropriate to software engineering are drawn from disciplines that study human behaviour such as psychology that deals with humans and its behavioural level and lastly sociology that emphasizes on the team and organizational levels. This means that there are different methods that can be employed in the development of any system. Making a logical argument here will enhance the technical development of any software development methods and practice.

In focusing on the main topic, I would like to define key terminologies in this critical research work in quidding any developer or software engineer in practice by theory and method of software engineering. The guide is a laydown process that should be followed in implementing any project. Software practice has defined parameters that have been defined with evidence that if those principles if carefully followed can produce better output required by the end user's requirement. Furthermore, a theory is a defined concept that has the supporting hypothesis that has been postulated and has been proven right by researchers. Methods are the procedures that should be followed to solve a scientific problem. Putting the entire concept explains that how software should be developed by following the underlining principles of software theories and methods using especially the software development life cycle and how these SDLC should be used to define the software models such as the spiral model, and rapid application development (RAD) prototyping model to solve the software problem.

A scientific theory identifies and defines a set of phenomena in software development, concepts, methods, and makes assertions about the nature of those phenomena and the relationships between their core parameters. A good theory precisely defines the

theoretical terms, consequently, the set of scientist communities can observe, deliberate, and measure them. A good theory also explains why certain relationships occur between them. Positivists expect their theories to have strong predictive power, and so look for generalized models of cause-and-effect as the basis for theories. In contrast, constructivists expect theories to strengthen their understanding of complex situations, and so tend to make more use of categorizations and analogies. Theories are also judged for aesthetic value. Often there is more than one theory that explains empirical observations, so the theories that are simpler, or more elegant are preferred [2].

For brevity, a method can be defined as a set of organizing core principles with empirical data collected and further analysed. A multiplicity of methods can be applied to any research problem, and it is necessary to use a sequence of techniques to fully understand the problem and methods. The choice of methods depends upon the theoretical stance of the researcher(s), access to resources (e.g., students or professionals as subjects/participants), and how closely the method aligns with the question(s) that have been presented. Research Design is the process of selecting a method for a particular research problem, tapping into its strengths, while mitigating its weaknesses. The cogency of the results depends on how well the research design offsets the weaknesses of the methods utilized in software engineering approaches.

Finally, in the review of the software engineering papers, the researcher wants to present the concept of software engineering as the application of software development lifecycle (SDLC) and its models that are used to define the practical approach and the theories of SDLC that all software engineers need to follow. The analysis of this problem such as the theory and methodology used to solve software problems can be solved by breaking down the models into the implementation of any software approach. The following points can be best be utilized to guide software practice by theory and method of software engineering.

Data Centre Infrastructure

In order to design and build a primary data centre project in any setting, it can be large or small, new construction or retrofit, complete or partial. It can involve a modification in various parameters such as physical room size or layout or electrical capacity, an increase in power density, a redesign of power and or cooling architecture, any number of other vicissitudes to the physical infrastructure of the data centre. Irrespective of the size or nature of the project, successful implementation depends not only upon the purchase and installation of the equipment of the physical system. Nonetheless, equally upon the process that pilots the project through its development and realization, from concept to commissioning. An illustration of this concept of a project as the combination of a system plus the process that creates it is indicated in Figure 4.

The idea of a formalized process to guide the creation of a system is not new, but its significance to the success of data centre physical infrastructure interoperability projects is just beginning to be understood. Objectively as standardization of the physical system improves reliability and speeds deployment, a standardized process contributes significantly to the overall success and predictability of the project and the system it creates.

It takes incredible time for the combined experience of a maturing industry to evolve toward standardization. Especially in an industry with a long tradition of custom system design such as but the benefits of the standardized process to both users and providers can be wide-ranging and profound. For the end-user, a dependable and repeatable process delivers the system more quickly, with less expense

and fewer defects. For the provider of engineering services or physical equipment, a dependable and repeatable process releases up time and resources for the real business at hand. Such system design and implementation with an increasing the scalability of the provider's core capability. The goal of a standardized process is not to eclipse or minimize system expertise, but to facilitate it.

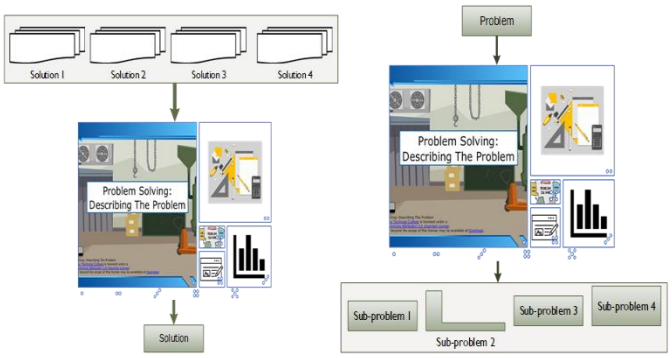

Figure 1: The analysis process of solving the problem (left) and synthesis of solving a problem (right)

PART I : A SECURITY-SPECIFIC KNOWLEDGE MODELLING APPROACH

This part entails a brief discussion of software engineering approaches such as a security-specific knowledge modelling approach, software practices, and theory for securing software engineering. The rationale of software engineering, review method of software engineering approaches, modelling the process and life cycle model, planning and managing the software project, and capturing software project requirements.

Why Software Engineering?

This is a tough question to answer but however, we will attempt to present our discussions in theory and practice (method). Software products are large and complex that require development analysis and synthesis. Analysis decomposes a large problem into smaller, understandable problems and abstraction is the in-analysis decomposition. Synthesis on the other hand builds the software from the building blocks. Figure 1 indicates the analysis process of solving problems (left) and the synthesis process of solving the problem (right).

Furthermore, solving a problem requires four main instruments; namely, method refers to a formal procedure, tool refers to an instrument or automated system for accomplishing something in a better way, procedure refers to a combination of tools and techniques to produce a product and paradigm refers to the philosophy or approach for building a product. Since computer science focuses on computer hardware and programming languages whilst software engineering focuses on computers as problem-solving tools.

There are many success stories we have gained in the development of new tools in tackling emerging challenges in software engineering. In view of the successes such as performance tasks more quickly and effectively (word processing, spreadsheets, e-mail), the support advances in medicine, agriculture, transportation, multimedia education, and most other industries. However, the argument still remains valid that software is not without problems.

Another important point I want to highlight is why software engineering? This question cannot complete without talking about quality assurances of software deliverables. There are basically different views about the perspective quality of software.

- The *transcendental view*: this view purported that quality is something we can recognize but not define.
- The *user view*: quality is fitness for purpose, and most be satisfiable.
- The *product view*: quality tied to inherent product characteristics.
- The *value-based view* depends on the amount the customers are willing to pay for the software.

Good software engineering must always include a strategy for producing quality software such as the quality of the product, the quality of the process, and the quality of the product in the context of the business environment. Quality of the development and maintenance process is as important as the product hence the development process needs to be modelled. Does modelling address questions such as where to find a particular kind of fault? How to find faults earlier? How to build fault tolerance and what are the alternative activities. According to Khraiwesh et al. [3], it is important not also that the quality of software depends on the software underlining principles such as the models for process improvement, SEI's Capability Maturity Model (CMM), ISO 9000, and Software Process Improvement and Capability Determination (SPICE).

In the context of the business environment, quality has different underlying parameters that business-oriented institutions are looking for in software production. Since business value is as important as technical value, business value (in relation to technical value) must be quantified, a common approach that will return on investment (ROI) – what is given up for other purposes and ROI is interpreted in different terms such as reducing costs, predicting savings, improving productivity, and costs in relation to efforts and resources [4].

However, the participants (stakeholders in software development projects also are key in why the theory and practice will support the development approach. By identifying activities and objects to be done, defining the system boundary, and considering nested systems, systems interrelationship. For clarity purpose, an activity is an event initiated by a trigger, and objects or entities are elements involved in the activities. Relationships and the system boundaries are the interaction among entities and activities, system boundaries determine the origin of input and destinations of the output, respectively. Some systems are dependent on other systems (the interdependence may be complex) it is possible for one system to exist inside another system and if the boundary definitions are detailed, building a larger system from the smaller ones is relatively easy [5]. The system approach emerges from a layered system that encapsulates reporting system for data, a data management system for collecting data, a communication system from remote sites to central, a calculation system for remote data, and a remote data collection system. The system building blocks for the engineering approach from requirement analysis and definition, system design, program design, writing the programs, unit testing, integration testing, system testing system delivery, and maintenance will be explained throughout this paper. This paper will enhance different approaches to solving software engineering problems. Figure 2 indicates the stakeholder in a software development project (a) and the elements in the software system approach (b).

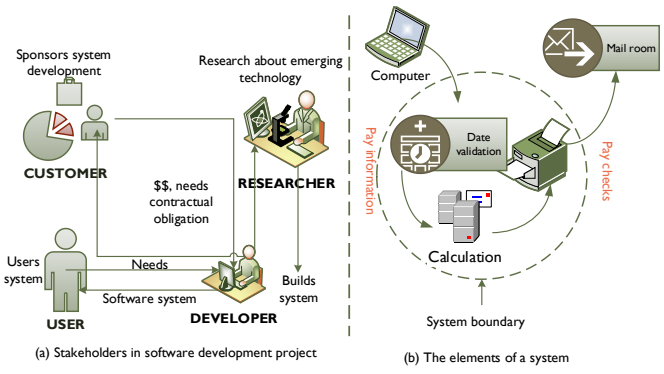

Figure 2 : The stakeholders in a software development project (a) and the elements of a system (b)

II. RESEARCH REVIEW METHODOLOGY

According to Kanniah et al. [6], Karim et al. [7], and Piessens and Verbauwhede et al. [8], there has been a paradigm shift of 'Building Security In' and the identification of critical needs for software development and security. The paradigm shift requires software security to be addressed in all the phases of the SDLC. Further research revealed that most security vulnerabilities outcome from a defect unintentionally introduced during the software design phase and the development phase [9]. According to McGraw, there are two best practices to mitigate the security vulnerabilities in software systems: code reviews and architectural risk analysis. In the traditional SDLC, these best practices normally called security touchpoints are related to the artifacts during the implementation phase (codebase) and the design phase (design documents). For an organization with mature software development processes, software artifacts are disconnected from each other [10]. However, existing tools are inherent with limited capabilities of identifying specific security-related software between the artifacts generated in each phase of the SDLC. This vulnerability exposes a critical research gap of degenerative semantically interconnecting the artifacts that originated at the implementation phase of the design phase. A conceptual framework and a proof-of-concept implementation to semantically interconnect architectural level security flaws and code-level security bugs [11]. The security flaws are identified based on the STRIDE threat categorization model propounded by Microsoft which helps to identify threats from the attacker's perspective by classifying attackers' goals into six threat categories. On the other hand, security bugs are determined based on OWASP's Top 10 vulnerabilities [12]. The ten most critical web application security risks and enhancing a great awareness for web application security. There is an interconnection between security flaws and bugs by employing a knowledge-modelling-based method, which enhances inferring the relations that are manually difficult to resolve.

Abeyrathna A. et al. [13], revealed that OWASP Top 10 vulnerabilities consist of three main approaches; identifying security flaws and inferring relationships among flaws and bugs threat modelling has been used as the architectural risk analysis method due to several noteworthy reasons [13] such as the capability of working with high-level design diagrams, to employ different contexts (simplicity) and explicitly support tools using a threat-modelling process through initialling data flow diagram (DFD). The threat modelling process consists of three phases: decomposition- where potential attackers could interface with the application, determination, and ranking of threats. The threat is determined and categorized according to techniques, and countermeasures and mitigation and threats are identified for ranked and threats. The benefits of the existing static analysis tools in isolation for security-specific analysis of multiple large-scale software systems are highly questionable [14]. The logical relationship between STRIDE and OWASP is obtained between flaws and bugs STRIDE mostly while Application Security Frame (ASF) on a threat of categorization model that supports in identifying the threats from the defensive perspective. The relationship between STRIDE and ASF indicated that a knowledge base is used to identify between threat and bug categories through a semantic text similarity matching model. The summarized proactive control describe were used to achieve the semantic similarity between ASF and proactive controls. Abeyrathna A. et al. [13] explained security measures of software engineering approaches such as knowledge base (querying the knowledge base (rule 1) and discovering associated threat category using the bug category (rule 2); proof-of-concept implementation that has to do with Connexus framework with it four parameters; threat-based processing, bug-based

processing, association inferencing, and knowledge base.

This paper covers projects involving new construction or upgrades to the data centre's physical infrastructure and software engineering advanced techniques with the power, cooling, and other physical systems that house and protect the data centre's IT equipment. Nevertheless, the power consumption and physical size of the IT equipment drive the design of the physical infrastructure system that supports it, the design and architecture of the IT layer of the data centre are an intuitive part of the data centre and software engineering perspective.

A. Modelling the Process and Life Cycle Model

In software development, modelling the entire process is one of the most difficult things to achieve because it is the main method that must be adopted to achieve the required result. A process I meant that software development products, processes, and resources and several models of the software development process must be adopted. Tools and techniques for process modelling are fundamental during the development process and should be highly documented during determining which model should be best adopted. A process is a series of steps involving activities and intended output of some kind and quires adequate tools and techniques. The characteristics of so many processes are involved during software production but for the purpose of this study we would like to prescribe some of the major processes in software development as follows:

- Uses resources, subject to a set of constraints such as schedule
- Produces intermediate and final products
- May be composed of subprocesses with hierarchy or multiple links
- Each processing activity has entry and exit criteria
- Activities are organized in sequence, so timing is clear
- Each process has quidded principles, including goals of each activity
- Constraints may apply to an activity, resource, or product

Therefore, the meaning and importance of process can best be described as impose consistency and structure on a set of predetermine activities and that must be accomplished within specified time. This guides us to understand, control, examine and improve the activities and enable us to capture our experiences and pass them along Again, it should be noted that software process models have several reasons for modelling an entire software life cycle (SDLC). It forms a common understanding of all software engineers to follow and adopt an agreeable method of solving software emerging issues, to find inconsistencies, redundancies, omissions during the development process. The process also models also find and evaluate appropriate activities for reaching process goals and to overall tailor a general process for a particular situation in which it will be used. The software life cycle model is critical and as such when a process involves building software, the process has to undergo some of the paradigms; requirements analysis and definition, system architecture (design), program design (detailed or procedural), testing such as unit, integration and system testing, system delivery (deployment) and maintenance of the software unit the end-user no longer satisfy the with a product which may require reverse re-engineering.

B. Planning and Managing the Software Project

Planning and software management is important in the development of any software outcome. Basically, by tracking the progress of activities through a clear understanding of customer's problems and needs, designing a system to solve customer's satisfaction, and understanding the time and cost of developing the software. Planning also helps you track the project schedule by describing the software development for a particular project by enumerating

the phases or stages of the project and breaking each phase into discrete tasks or activities to be completed. Hence by portraying the interactions among the activities and estimating the times that each task or activity will take to be completed. Understanding customers' needs by listing all the project deliverables such as documents, demonstrations of functions, subsystems, accuracy and reliability, and performance or security will enhance the project milestones and activities to produce the deliverables.

The project milestones and activities basically include activities that take place over a period, milestones that emphases the completion of activity at a specific time, precursor an event or set of events that must occur in order for an activity to start, the duration that determines the length of time needed to complete an activity and due date by which an activity must be completed. Another way could be employing a work breakdown structure (WBS) that depicts the project as a set of discrete pieces of work and activity graphs that depict the dependencies among activities such as nodes (project milestones) and lines (activities involved).

Planning cannot be done without determining the software project Critical Path Method (CPM). The minimum amount of time it will take to complete a project that reveals those activities that are most critical to completing the project on time. Real-time (actual time) estimate amount of time required for the activity to be completed, availability of time that is the amount of time available in the schedule for the activity's completion, and slack time which is the difference between the available time and real-time for that activity.

Lastly, effort estimation is one of the crucial aspects of project planning and management. Estimating cost has to be done as early as possible during the project life cycles such as the facilities and this may include hardware space, furniture, telephone, etc., methods and tools include software, tools for designing software (Computer-Aided Software Engineering or CASE), and staff (effort) which is the biggest component of cost. However, estimation should be done repeatedly because uncertainty early in the project can affect the accuracy of cost and size estimations. However, their many parameters that cause inaccurate estimates such as a frequent request for change by users, overlooked tasks, user's lack of understanding of the requirements, insufficient analysis when developing the estimate, lack of coordination of system development, technical services, operations, data administration, and other functions during development and lack of an adequate method or guidelines for estimating. Therefore, a clear understanding of software planning and management will guide your software methods and theory.

C. Capturing Software Project Requirements

A requirement is an expression of desired behaviour. Because the emergency of software development is virtually increasing there are many methods being adopted to address user requirements by deploying many software tools in defining the requirement such as UML, DFD, and many more. A requirement deals with objects or entities, the state they can be in, and functions that are performed to change states or object characteristics [15-16]. Requirement focus on the customer needs, not on the solution or implementation by stating the designate what behaviour, without saying how that behaviour will be realized.

Why are requirements important? The question seems to be simple but however, always shows up at the end of software project completion whether those requirements are met and how the customer is going to react on that. The top factors that cause the project to fail are incomplete requirements, lack of user involvement, unrealistic expectations, lack of executive support, changing requirements and

specifications, lack of proper planning, and the system no longer needed. Some part of the requirements process is involved in almost all these causes and requirements error can be expensive if not detected early and this requirement is commonly requested by the system analyst who determines the final outcome is a software requirements specification (SRS) document (Figure 3).

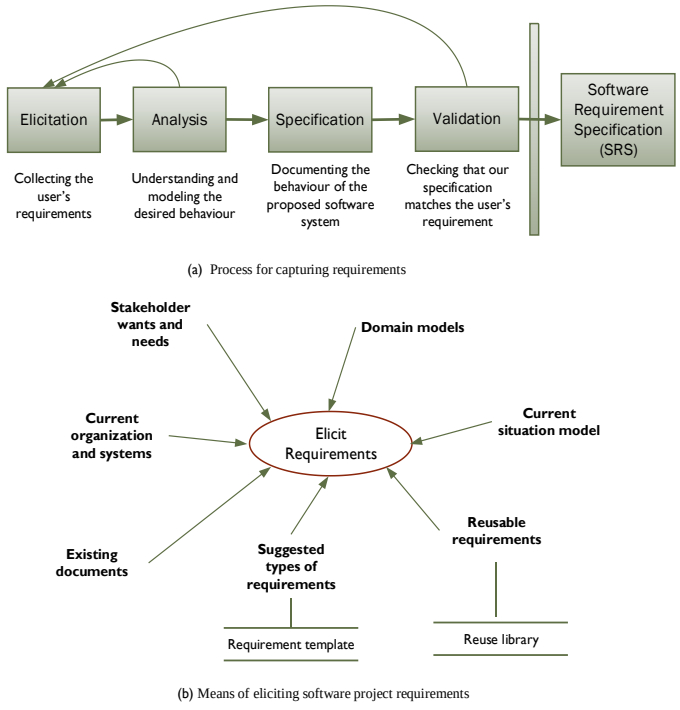

Figure 3: The outcomes of software requirement specification

PART II: DATA CENTRE PROJECT

This part of the paper presents a detail exploration of the infrastructure of the data centre for engineered projects understanding key highlights for this part such as constituents of project, context within data centre life cycle, requirements for standardized process, the basic architecture of the project, project planning phase, phases, steps, and milestones, indispensable features of the process, and custom engineered to-order-projects.

a. Constituents of a Project

In the context of this review, a project is any change significant enough to need an orderly flow of tasks and a process to coordinate and manage its execution. Building a new data centre or server room is clearly a project and challenging task. By adding racks of new blade servers is usually a project development and life cycle model and conversely, adding a single rack to an existing data centre is probably not a project (Figure 4).

The following features will generally upheaval a data centre upgrade to "project" status due to the following reasons: change in power or cooling architecture (for example, converting from centralized to row-based), the introduction of risk, need for planning or coordination, and need to shut down equipment [17].

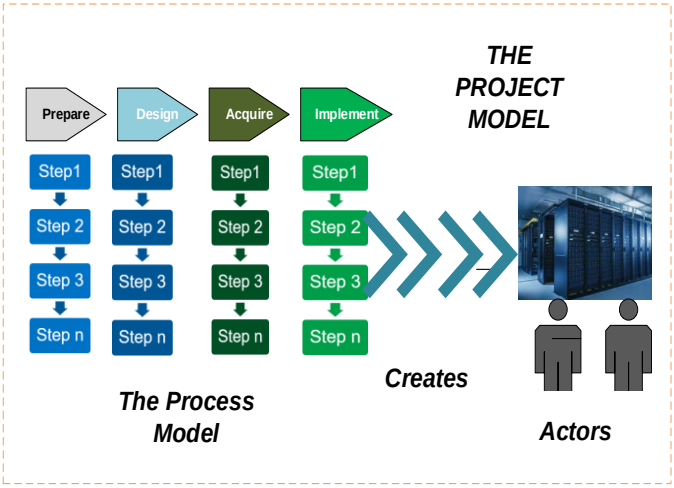

Figure 4: A project encapsulate a SYSTEM plus the PROCESS that creates it

b. The context within the data centre life cycle

The context within the data centre life cycle is an important phase in the development process. This process involves planning and building, which constitute the beginning of the data centre life cycle. Figure 5 shows this context within the complete life cycle model of a data centre.

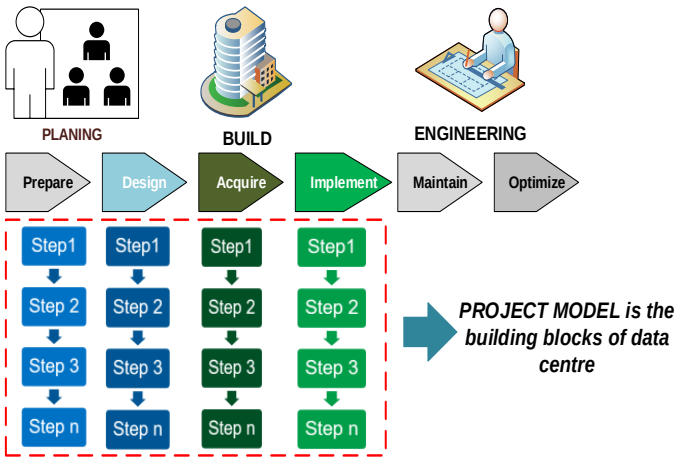

Figure 5: The project process within the context of the data centre life cycle model

c. Requirements for standardized process

A major problem common to many data centre projects is wasted time, wasted money, or defects due to flaws in the process includes dropped handoffs, ambiguous responsibility, misinformed decisions, and other errors of communication or execution [18]. This is not necessarily due to flaws in the activity of the various parties to the process of the end-user, the hardware provider(s). The design engineers and but rather to the lack of an overarching, shared process guiding all parties as a team, clarifying responsibilities and communication. The hazards of a non-standardized or non-existent process span the familiar spectrum of unnecessary expense, delays, and frustration: reduced quality, higher cost, wasted time, poor documentation, inadequate testing, and degraded service [19].

Most defects that ultimately turn up in the latter stages of a data centre project, or even after the project is completed including the ultimate defect, failed business results. Subsequently, they are caused not by problems in the physical components of the system that was built. Rather by decisions that were made in planning the overall system and flaws in the process by which the system was deployed. A well-designed, standardized process has built-in intelligence and structure to avoid such problems, both in the planning stages and at every step along the way to project completion [20]. The result is reduced rework, accelerated cycle time, and a system that is ultimately deployed as expected, with no surprises.

Value of Communication Skills: We can address issues related to communication and adequate understanding of building a data centre [21]. Essentially, despite the clarity, repeatability, and efficacy of its implementation, a standardized process proposes additional protection against miscommunication and waste: *a common language*. Many of the pitfalls and mistakes that typically occur during the course of a project can be avoided by using standard and familiar terminology. In project communications among the vendors, partners, and users who have a stake in its success [22].

Standardization and Customization: The standardized process described in this review does not mean that every project is the same, or that every process must be exactly like the one we proposed. It does, however, offer a best-practice framework and guideline for essential process architecture [23] that can be adapted to the project at hand, whether wiring closet or multi-megawatt data centre. Not all the phases in this process description will be executed for every project. In regard to the agile system, the process is organized into modular units (steps, and tasks within steps) which can be selectively configured or eradicated, according to the requirements of the data centre and requirements project.

Customization through the configuration of a modular, standardized architecture is a time-tested strategy – Lego® blocks are a familiar example. Moving toward modularization, standardization design in equipment hardware such as the implemented "system" of Figure 4 in order to achieve efficient, predictable, and reliable results, the data

centre physical infrastructure industry is necessary. Therefore, similar business feedbacks accumulate from a standardized, modular process to build that system [24].

d. Basic Architecture of the Project

The project process starts with a business requirement, which may be an insecurely articulated interpretation of a business concern, or some other overall statement, such as "*I need a backup data centre*." As the project advances through well-defined process phases such as *prepare, design, acquire, and implementation.* The tasks are executed, time dependencies are managed, information is passed to where it is needed at the right time, handoffs are coordinated, and the final outcome of the process is a fully deployed and operational system. A summary of the sequence of activity through the four phases of a data centre project is illustrated in Figure 6. The first two phases constitute the *PLANING* portion of the process, which translates the originally stated need into a detailed design and a list of components on a purchase order. The remaining last two phases are the *BUILD* portion of the process, taking the project from hardware attainment to a functioning and structured system.

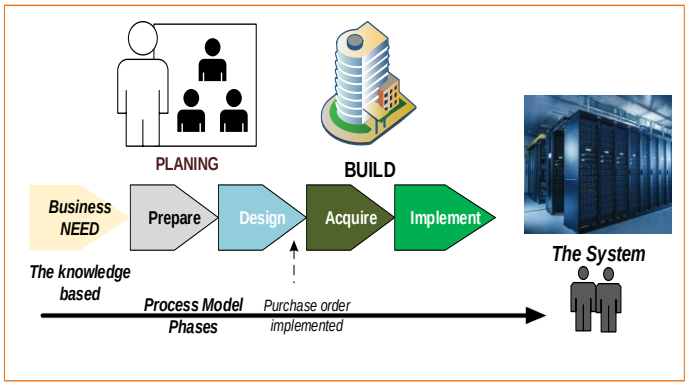

Figure 6: The four phases of the data centre project process

e. Project Planning Phase

The PLAN portion of the process places the critical foundation for everything that follows. Hitherto contempt this crucial importance to the success of the project, planning has historically provided the greatest opportunity for *confusion, misunderstanding, and miscommunication*. Errors made here will magnify and broadcast through the latter *BUILD* phases. The characteristic result is interruptions, restarts, cost overruns, unexploited time, frustration, and eventually a compromised system. Appropriate consideration must be given to the planning phases, using suitable expertise to safeguard that design rudiments are specified in a way that delivers the essential and sufficient information to the downstream BUILD portion of the process and consequently to assure a fruitful outcome.

The practical and commercial and business considerations include variables, trade-offs, and constraints that can be intimidating to even the greatest experienced professional. Despite an expert consultant betrothed in system planning, there is a critical hierarchical order of user communication and input that can be modelled by a homogeneous methodology that minimizes backtracking and unexploited effort by all parties. Since planning activity is so vital to the accomplishment of the project and so prone to unintentional distraction and errors. It is important for all the equipment, experts required to participate during the development of the data centre. When the *PLAN* phases have been successfully implemented, the most critical part is complete. The remaining *BUILD* phases can be carried out in a deterministic such as almost automated and managed, on condition that they are under the control of a difficult, well-defined process executed by a qualified project team.

f. Indispensable Features of the Process

Irrespective of the specific methodology rummage-sale, the process must conduct the project efficiently, dependably, and justifiably, with safeguards in place to eradicate difficulties such as missed handoffs, vague accountability, and lost information. It should include strategies for the management of unplanned

occurrences such as project changes and defects. It should be done with two main concepts: *modularized and configurable* consequently it can be adapted to projects of different types and sizes.

A homogeneous process that meets the above overall requirements will encapsulate the following characteristics:

- There are no "cracks" or dead space between steps such as every step is linked to prerequisite and subsequent steps by its inputs and outputs. Once a step has received all its inputs, it can complete its tasks and make its outputs available to other steps that depend on them.
- There are special "asynchronous" functions that remain on standby during the course of the project, to systematically deal with unplanned changes or defect correction.
- Steps can be deleted to configure the process appropriately for the project at hand.
- A Web-based tracking and status system is accessible to all stakeholders (both the customer and any parties providing project services), for shared documentation, data, and reports.
- Every activity necessary for the completion of the project is included in the process.
- Each step has clearly defined inputs and outputs.
- Every output produced is either the input to another step, or is a final output of the project.
- No effort is wasted on extraneous outputs that do not contribute to the progress or ultimate outcome of the project.
- Every step of the process has clearly assigned ownership responsibility, so there is no "dropping the ball" due to unassigned or ambiguous ownership of steps.

g. Phases, steps, and milestones

Prior to this, Figure 6 showed the four main phases of the process that occur sequentially. From the original idea of business requirements to the completed building of the physical system by carrying out the project from left to right. Each of those four phases consists of several steps listed below it, which occur sequentially going down as illustrated in Figure 7. The process advances to the next phase, all the steps of a phase are completed to the right and consequently, the end of each phase is marked by a milestone.

Non-parallel Activities: In addition to the process steps that navigate the expected course of the project, it is essential to have a built-in process structure to handle the unexpected. These or non-parallel or asynchronous activities can be triggered at any time during the project.

Project changes: Changes should be an expected part of a project. The process must be designed to accommodate changes without creating process defects, delays, or unnecessary costs. Changes can result from new information that was not previously recognized, changes to vendors' equipment or services, or changes in the user's system requirements.

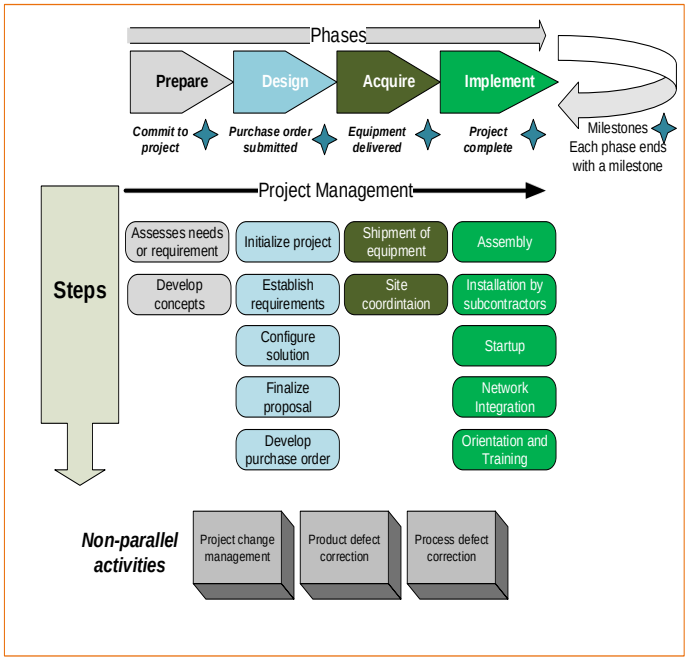

Figure 7: Process "map" showing basic elements of the project process

Product defect correction: Part of the system may be found missing, damaged, or unsuccessful, at any period after delivery. However, the user's project (data centre) process must be prepared to interface with the supplier and manage postponements during defect correction. Whereas the responsibility for correcting these defects will principally rest with the product supplier as part of the supplier's project process (Figure 7).

Process defect correction: Predominantly a new one, any process, should be measured as a proving ground for evolutionary development. Missing data, sequencing errors, and even missing steps may be discovered during the course of the project. With a pre-planned recovery strategy, the postponement and cost of process faults can be mitigated. These asynchronous activities must be explicitly assigned to an owner in order to ensure process continuity when the unexpected arises, as required with the sequential process steps. Pre-defined non-parallel procedures are indispensable to a well-organized and successful process, whether defined and handled as a separate activity or incorporated into project management duties.

h. Custom Engineered-to-Order Projects (ETO)

The process labelled in the previous section undertakes a system configured from standard hardware and software components. It does not include the extra steps needed for a project including engineered-to-order (ETO), or highly customized equipment or services. A highly customized project, for example, a unique supercomputer installation will require additional steps for engineering design, factory acceptance test. To verify that the system operates as designed, and commissioning and or post-installation whole-system testing to confirm the correct operation in the context of the on-site environment, which can be incorporated into this process as illustrated in Figure 8. Therefore, the project process can be modified for a specific requirement by adding or eliminating steps from the standard process model.

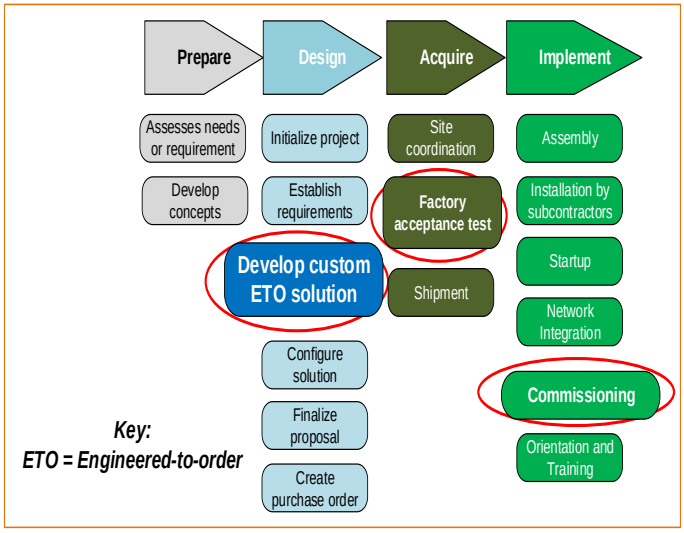

Figure 8: A standard process model of engineered-to-order projects (ETO)

The anatomy of the steps is a collection of associated tasks that together achieve the main goal of the step as referred to in this research. For instance, Figure 9 indicates the tasks within the "*Start-up*" step whereas, Figure 10 illustrates each step as follows:

Ownership: The ownership is the party responsible for the execution of the step. This could be provided as an outsourced service by the equipment vendor or third-party service provider as ownership could probably be within the user's organization. Additionally, it provides insurance against missed handoffs we called dropped balls and the things falling through cracks which explicitly assigned ownership for every step executed.

Task-list: A task-list is a description of the work that requires to be done to accomplish the step. The actual work of the project is defined by the tasks. Each task has worked instructions and a checklist of specific actions to be accomplished. The type of the project and the physical infrastructure elements involved are determined by tasks within each step. For instance, tasks related to cooling will not be present if cooling is not part of the project. As a set of drawings, each

checklist contains one or more data elements that may be a simple as a date or as complex of that architecture design. However, a task is accomplished when all its checklist items are finalized.

Inputs: Input is the data needed in order for the work of the step to be accomplished. Each input to a step is an output of a step that precedes it.

Outputs: An output is data produced by the step, needed as input to subsequent steps in the process.

a. Pull-based Process Architecture

The pull-based process architecture is an efficient process design that prescribes that every output enhanced by a step is produced at the right time and in the right form that is to be utilized by a subsequent *downstream* step as an input or serve as a final output of the entire process which is otherwise the output amounts to misguided work. At the final expected result and asking specific questions like *what is directly needed to achieve this result*? Which by designing a process this manner requires looking at the intended outcome. Consequently, working backward through each step of the process asking *what does this step requires from previous steps*? By ensuring that there is no misguided work or outputs to nowhere and enabling the process to flow efficiently from step to step, requires providing what is necessary and sufficient for each step. This *Pull-Based* technique to information flow which was downstream steps called *pulling* only the information they need from upstream steps and which is a cornerstone strategy of this, or any other, efficient and effective process design [25].

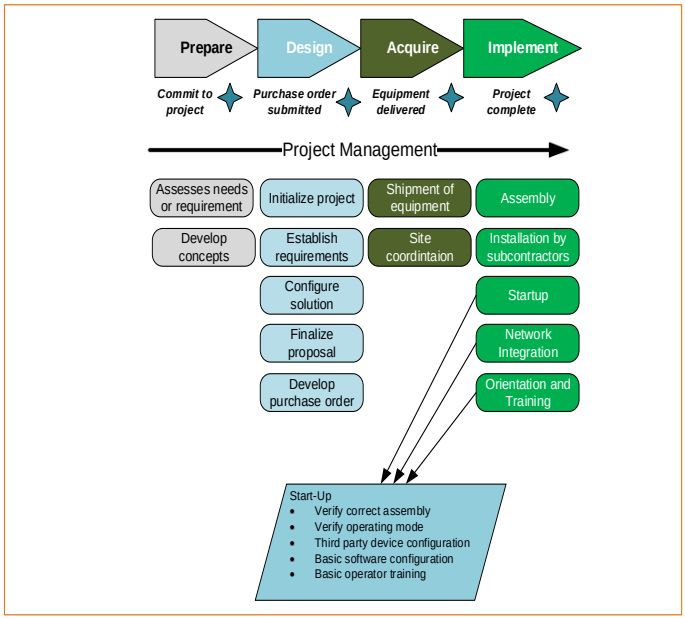

Figure 9: Detail of tasks within a step (Detail of Start-up step showing five tasks).

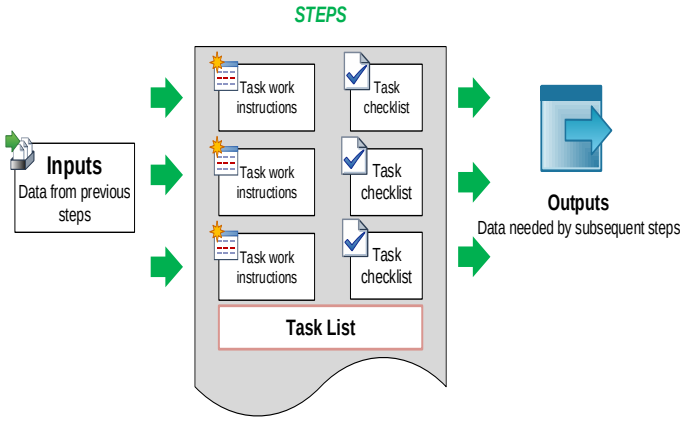

Figure 10: Detail of step anatomy

Project Management: As may require from any business project, a data centre project requires devoted and skilled oversight, with documented procedures to address project-critical activities such as *continuity, scheduling, resources, budget, system changes, process defects, and status reporting*. The delegation of project management duties is a significant element of process design that must be considered and determined upfront, well before the time comes to execute them [26].

Tracking Responsibilities: With complete clarity in regards to who is doing what, it is indispensable that

all the roles in a project be well defined and allocated. Every block diagram process as illustrated in Figure 7 is work that must be accomplished. Consequently, each one must be explicitly assigned to a person or party who will be responsible for executing it. Whether managed within or outsourced to a service provider such as either the primary equipment vendor or a third party. The third party is vital every element of the process is clearly reported by creating a responsibility list. A clear and agreed-upon list of assignees for every element of the process provides protection from surprises, delays, and common but unwelcome remarks. *We* supposed somebody *ELSE* was doing that *Responsibility* does not imply that the named object is the only contributor in the allocated effort; it simply designates accountability for ensuring that the work gets done [27].

Learning Process: For those involved in the deployment of data centre physical infrastructure such as either as self-architect of the project or as the customer of a service provider. The service provider is an informed engagement that ranges from a matter of interest to a critical prerequisite, depending upon the level of responsibility for the result [28]. As with another acquainted instance of a complex product, the automobile, the amount of interest and involvement in the product's creation depends upon the resources, skills, and temperament of the new owner. The new owner from completely do-it-yourself, to ordering from a list of standard options, and to simple off the- lot selection. The type of knowledge required is different at each phase in the process (Figure 11).

However, it can be noticed that in an established industry, customers are not typically involved in design and construction, so they do not need detailed knowledge in those areas. They trust the manufacturer to have the knowledge required for the design and construction of the data centre. A data centre physical infrastructure is poignant in this direction, with standardized designs that can be ordered and configured much as a car is configured from standard possibilities [29]. However, for data centre customers who wish to do their design and construction, education is obtainable in the form of research papers and e-learning courses. Customers can also involve the services of specialized consultants to contribute to the design and construction of the model centre.

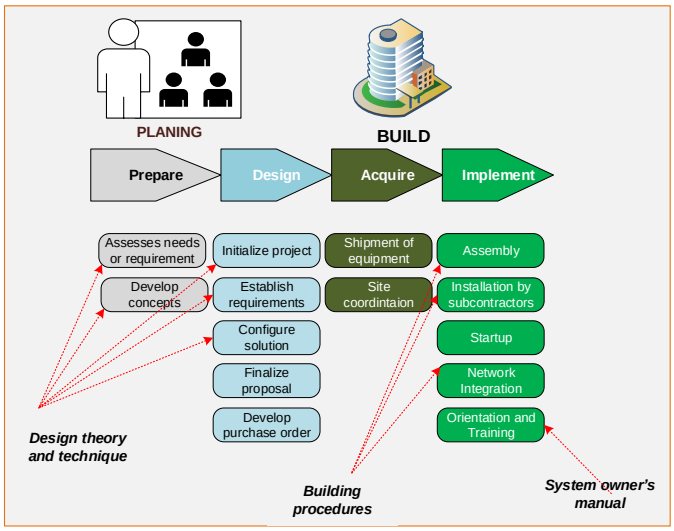

Figure 11: Types of learning at different places in the process

III. CONCLUSION

1. Security-Specific Knowledge Modelling Approach and Software Practices

By given a problem to solve, one must analyse it and synthesize a solution. Understand that requirements may change, must view quality from several different perspectives, use fundamental software engineering concepts (abstraction and measurements), and keep system boundary in mind. Process development involves activities, resources, and products and may include organizational, functional, behavioural, and other perspectives. A process model is useful for guiding team behaviour, coordination, and collaboration.

Looking at what it means to design a system, design begins at a high level, with important decisions about system architecture based on system requirements, desired attributes, and the long-term intended use of the system. The need to keep in mind the several characteristics as we build a design i.e., modularity and level of abstraction, coupling and cohesion and fault tolerance, prototyping, and user interface. Though use cases are embellished to generate a system design, painting a high-level portrait of how the system will solve the problem at hand through handling program design implements non-functional requirements. UML is becoming a de facto standard way of describing the OO system because there are many metrics that are commonly used to measure the size and characteristics of OO artifacts.

Writing codes are cost-effective to the programmer but it is important to consider organizational standards and guidelines, reusing code from other projects, writing code to make it reusable on future projects, and using the low-level design as an initial framework, and moving in several iterations from design to code. It is important to understand the difference between faults and failures. The goal of testing is to find faults, not to prove correctness.

Testing should be anticipated from the very beginning of the system life cycle and system functions should be thought of during requirements analysis. The use of fault-tree analysis, failure modes, and effect analysis during design considers all possible test cases during testing and building safety cases during design and code reviews. Training and documentation should be planned and tracked from the project's beginning and should be integrated with the regular system software. All training and documentation modules and documents should consider the varying needs of different audiences.

The more a system is linked to the real world, the more likely it will change and the more difficult it will be to maintain. Maintainers have many jobs in addition to software developers because measuring maintainability is difficult to achieve. The impact analysis builds and tracks links among the requirements, design, code, and test cases. Finally, software rejuvenation involves documenting, restructuring, reverse engineering, and reengineering are complex, therefore guiding our software practice by theory and method of software engineering can be achieved by following the underlining principles of software engineering.

2. Data Centre Infrastructure

The data centre by utilizing standardization, data centre projects have not historically implemented and documented processes as discussed in this paper. In the realm of art rather than science, with one-time engineering, ad hoc management, and exceptional system design, rather data centre builds and upgrades have characteristically been implemented. The concept of the standardized process will integrate into the data centre physical infrastructure industry, as it has in other areas of business activity, as research and development continue to make progress in this domain especially artificial intelligence with its full potential in designing and implementation of the smart data centre. As services will become available to provide the time-tested advantages of a standardized but configurable process, much like in the automobile industry. Custom construction of ordinary data centres will someday be a similar historical curiosity. Currently, the process of constructing one's own car is relegated to the garages of an adventurous few.

A more standardized paradigm for the construction of a data centre projects will make planning and building more predictable, efficient, and scalable for both the provider and the user, as this can be accounted for in other industries. To be redirected away from what has become routine and more toward the continuing emergence of unusual, complex, or very large projects that will require specialized or

exceptional expertise as a further standardized process enable design talent.

End-users may not be particularly interested in the comprehensive workings of a homogenous project process, but they want a dependability solution to occur as rapidly and professionally as imaginable. A process such as the one described in this paper is an insurance policy for that outcome. A generic outline for the conduct and construction of data centre projects, from concept growth to solution deployment. Notwithstanding the specific consortium and identification of the project tasks, whether the different steps of the process are joint or split, implemented within or subcontracted to a service provider. It is the integrity of the fundamental procedure that is vital to the accomplishment of the scheme. If all tasks are obviously defined, allocated, and correctly associated via the precise inputs and outputs, the process can be trusted to work.

The process described in this paper is a representation of research work to meet the requirements of effective project execution for their customers, who may select to do some or all of the procedure themselves or hire facilities to perform selected portions. An eminent and complete definition of process fundamentals enables steps to be captured as declarations of effort and obtainable as service modules, for customers who wish to representative scheme or project responsibilities. Other institutions may have their own portrayal of this same process, with different vocabulary and tasks consortium, but with the same project consequence or result. A well-presented and communicated process should be the standard operating procedure for any user-directed project and demanded of any service provider or hired consultant. A homogenous, predictable, and comprehensible methodology assures a lean, foreseeable process that speeds placement, enables communication, minimizes cost, energies out flaws, and abolishes waste.

This paper major contribution is the design and construction of a model, architecture of a typical data centre project, and overall, software engineering for a security-specific knowledge modelling approach, software practices, and theory for securing software engineering project.

IV. REFERENCES

[1]. Easterbrook, M.J. and Vignoles, V.L., 2015. When friendship formation goes down the toilet: Design features of shared accommodation influence interpersonal bonds and well-being. British Journal of Social Psychology, 54(1), pp.125-139.

[2]. Jain, L.C., Favorskaya, M.N., Nikitin, I.S. and Reviznikov, D.L., 2020. Advances in Theory and Practice of Computational Mechanics. Springer Singapore.

[3]. Khraiwesh, M., 2020. Measures of Organizational Training in the Capability Maturity Model Integration (CMMI®). International Journal of Advanced Computer Science and Applications, 11(2).

[4]. Molléri, J.S., Felderer, M., Mendes, E. and Petersen, K., 2019. Reasoning about Research Quality Alignment in Software Engineering. Journal of Systems and Software.

[5]. Saeedi, K. and Visvizi, A., 2021. Software Development Methodologies, HEIs, and the Digital Economy. Education Sciences, 11(2), p.73

[6]. Kanniah, S.L. and Mahrin, M.N.R., 2016. A review on factors influencing implementation of secure software development practices. International Journal of Computer and Systems Engineering, 10(8), pp.3032-3039.

[7]. Karim, N.S.A., Albuolayan, A., Saba, T. and Rehman, A., 2016. The practice of secure software development in SDLC: an

investigation through existing model and a case study. Security and Communication Networks, 9(18), pp.5333-5345.

[8]. Abeyrathna, A., Samarage, C., Dahanayake, B., Wijesiriwardana, C. and Wimalaratne, P., 2020. A security-specific knowledge modelling approach for secure software engineering. Journal of the National Science Foundation of Sri Lanka, 48(1).

[9]. Foster, E.C., 2021. Software engineering: a methodical approach. Auerbach Publications.

[10]. Giray, G., 2021. A software engineering perspective on engineering machine learning systems: State of the art and challenges. Journal of Systems and Software, p.111031.

[11]. Wang, Y., Wang, R., Jing, J. and Wang, H., 2021. Implementing a high-efficiency similarity analysis approach for firmware code. Plos one, 16(1), p.e0245098.

[12]. Dahanayake, B., Wimalaratne, P., Wijesiriwardana, C., Samarage, C. and Abeyratne, A., 2020. A security specific knowledge modelling approach for secure software engineering.

[13]. Abeyrathna, A., Samarage, C., Dahanayake, B., Wijesiriwardana, C. and Wimalaratne, P., 2020. A security specific knowledge modelling approach for secure software engineering. Journal of the National Science Foundation of Sri Lanka, 48(1).

[14]. Wijesiriwardana, C. and Wimalaratne, P., 2017, May. On the detection and analysis of software security vulnerabilities. In 2017 International Conference on IoT and Application (ICIOT) (pp. 1-4). IEEE.

[15]. Gómez, F.J., Aguilera, M.A., Olsen, S.H. and Vanfretti, L., 2020. Software requirements for interoperable and standard-based power system modeling tools. Simulation Modelling Practice and Theory, 103, p.102095.

[16]. Alenezi, M. and Almuairfi, S., 2019. Security risks in the software development lifecycle. International Journal of Recent Technology and Engineering, 8(3), pp.7048-7055.

[17]. Baudry, K., 2021. Data Center Site Search and Selection. Data Center Handbook: Plan, Design, Build, and Operations of a Smart Data Center, pp.367-380.

[18]. Zhang, X. and Wang, Y., 2021. Research on intelligent medical big data system based on Hadoop and blockchain. EURASIP Journal on Wireless Communications and Networking, 2021(1), pp.1-21.

[19]. Tennant, R., 2021. Supporting Caregivers in Complex Home Care: Towards Designing a Voice User Interface (Master's thesis, University of Waterloo).

[20]. Geng, H., 2021. SUSTAINABLE DATA CENTER: STRATEGIC PLANNING, DESIGN, CONSTRUCTION, AND OPERATIONS WITH EMERGING TECHNOLOGIES. Data Center Handbook: Plan, Design, Build, and Operations of a Smart Data Center, pp.1-13.

[21]. Veltsos, J.R. and Hynes, G.E., 2021. Managerial communication: Strategies and applications. SAGE Publications, Incorporated.

[22]. Andersson, J. and Sandberg, K., 2021. Potential Pitfalls in the Implementation Process of an Information System: A Framework for Identifying Pitfalls for Companies in the Startup Phase Aiming to Implement an Information System.

[23]. Kasonde, V. and Chembe, C., 2021. An Assessment of the Best Practices in the Implementation of Data Centers (A case of Zambia). Zambia ICT Journal, 5(1), pp.8-15.

[24]. Alsufyani, A., Alotaibi, Y., Almagrabi, A.O., Alghamdi, S.A. and Alsufyani, N., 2021. Optimized intelligent data management framework for a cyber-physical system for computational applications. Complex & Intelligent Systems, pp.1-13.

[25]. Mosin, V., Durisic, D. and Staron, M., 2021. Applicability of Machine Learning

Architectural Patterns in Vehicle Architecture: A Case Study.

[26]. Gurusubramani, S., Mouleeswaran, S.K., Srinivas, P. and Aruna, R., 2021, July. A Data Centre Configurable Data Mining Document Management Information System. In Journal of Physics: Conference Series (Vol. 1964, No. 4, p. 042095). IOP Publishing.

[27]. Maguire, J. and Ross Winthereik, B., 2021. Digitalizing the state: Data centres and the power of exchange. Ethnos, 86(3), pp.530-551.

[28]. Gurusubramani, S., Mouleeswaran, S.K., Srinivas, P. and Aruna, R., 2021, July. A Data Centre Configurable Data Mining Document Management Information System. In Journal of Physics: Conference Series (Vol. 1964, No. 4, p. 042095). IOP Publishing.

[29]. Wang, X., Wang, Y., Tao, F. and Liu, A., 2021. New paradigm of data-driven smart customisation through digital twin. Journal of manufacturing systems, 58, pp.270-280.

Cite this article as :

Abdul Joseph Fofanah, Habibu Rasin Bundu, Jonathan Gibrill Kargbo , Ahmed Fofana, "A Security-Specific Knowledge Modelling Approach, Software Practices, and Data Centre Infrastructure for Securing Software Engineering Technologies", International Journal of Scientific Research in Science and Technology (IJSRST), Online ISSN : 2395-602X, Print ISSN : 2395-6011, Volume 8 Issue 6, pp. 324-342, November-December 2021. Available at doi : https://doi.org/10.32628/IJSRST218645
Journal URL : https://ijsrst.com/IJSRST218645