International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 4, Issue 2, pp 13-23, February-2020

Knowledge Engineering and Traditional Information System

[1]Dr. Oye, N. D., [2]Jemimah Nathaniel

[1]Department of Computer Science, MAUTECH- Yola, Nigeria
[2]Postgraduate Student of Computer Science (MSc), University of Jos, Plateau State, Nigeria

Abstract - **The discipline of knowledge engineering grew out of the early work on expert systems in the seventies. The purpose is not necessarily to develop systems that replace humans, but to allow the use of systems that increase human effectiveness and efficiency. Another issue that frequently comes up in discussions about problem-solving methods is their correspondence with human reasoning. Invocation of the propose task produces one new parameter assignment, the smallest possible extension of an existing design. Domain-specific, search-control knowledge guides the order of parameter selection, based on the components they belong to. Ontology is specified describing the categories of the domain knowledge and the relationships between these categories. Knowledge roles link the components of the method to elements of the application domain; ontologies provide guidelines for building domain conceptualizations, such as the construction of subsumption hierarchies. The traditional methodologies such as System Development Life Cycle (SDLC), Object oriented analysis and design (OOAD), Waterfall and Rapid Application Development (RAD) have struggled to accommodate the web-specific aspects into their method and work practices. Most of the websites are based on graphical hypermedia systems, database-driven information system, and also security systems.**

Keywords: Knowledge; Engineering; Information; System; Traditional.

I. Introduction

The discipline of knowledge engineering grew out of the early work on expert systems in the seventies. With the growing popularity of knowledge-based systems (as these were by then called), there arose also a need for a systematic approach for building such systems, similar to methodologies in main-stream software engineering. Over the years, the discipline of knowledge engineering has evolved into the development of theory, methods and tools for developing knowledge-intensive applications (Allen, 2016). In other words, it provides guidance about when and how to apply particular knowledge-presentation techniques for solving particular problems. Principles that have become the baseline of modern knowledge engineering may include the common distinction made in knowledge engineering between task knowledge and domain knowledge (Artale, Franconi, & Pazzi, 2017).

The Traditional Methodologies such as System Development Life Cycle (SDLC), Object oriented analysis and design (OOAD), Waterfall and Rapid Application Development (RAD) have struggled to accommodate the web specific aspects into their method and work practices (Boose, 2017). Most of the websites are based on graphical hypermedia systems, database-driven information system, and also security systems. Web based information system requires a collection of website development techniques together with traditional development competences in program design and database design. Traditional methodology has some problems to accommodate web specific aspect in the terms of their methods and implementation. Many traditional methodologies are based on outmoded concepts dating back to the 1970s (Brachman, & Levesque, 2018). These methodologies are being utilized to develop web sites, and not surprisingly, they are limited since they were never intended to be used for this purpose (Schmolze, & Brachman, 2017). Before selecting best methodology for web development, we discuss the traditional methodologies and their applicability to this process. This paper discusses some of these methodologies, providing a brief explanation of their strengths and weaknesses in relation to the Web development process (Chandrasekaran, 2012).

II. Literature Review

2.1 Knowledge Engineering

Knowledge engineering (KE) is the application of machine systems to problems of human endeavor. The purpose is not necessarily to develop systems that replace humans, but to allow the use of systems that increase human effectiveness and efficiency (Hamid & Chandrasekaran, 2010). The goal is to encourage humans to do what they do best, whatever this might be, at a time deemed appropriate, and to allow machines to assume the functions best suited to them, such as power saws, mechanical arms, and payroll computers do. KE has many major functions that are discussed here (Clancey, 2017). Some of the fundamental KE issues addressed are:

- Ergonomics: Human-machine interaction
- Problem-solving and decision-making
- Thinking
- Communication: People-people, people-machine, and machine-machine
- Dark-side problems (fear, conflict, privacy, etc.)
- Intelligence
- The future

The main techniques that are being used in knowledge engineering and examples of their use are as follows:

2.1.1 Baseline

The early expert systems were based on an architecture which separated domain knowledge, in the form a knowledge base of rules, from a general reasoning mechanism. This distinction still is still valid in knowledge engineering practice. In the early eighties a number of key papers were published that set the scene for a systematic approach to knowledge engineering (Yang et al., 2002). In 1982 Newell published a paper on “The Knowledge Level” in which he argued the need for a description of knowledge at a level higher the level of symbols in knowledge-representation systems. The knowledge-level was his proposal for realizing a description of an AI system in terms of its rational behavior: why does the system (the “agent”) perform this “action”, independent of its symbolic representation in rules, frames or logic (the “symbol” level). Descriptions at the knowledge level have since become a principle underlying knowledge engineering (Shrobe, Davis, & Szolovits, 2011). Two other key publications came from Clancey. His “Epistemology of a rule-based system” can be viewed as a first knowledge-level description of a knowledge-based system, in which he distinguished various knowledge types. Two years later his article “Heuristic classification” appeared which described a standard problem-solving pattern in knowledge-level terms. Such patterns subsequently became an important focus of knowledge-engineering research; these patterns typically serve as reusable pieces of task knowledge (Fitzgerald, 1997). In the nineties, the attention of the knowledge-engineering shifted gradually to domain knowledge, in particular reusable representations in the form of ontologies. A key paper, which also quite wide attention outside the knowledge-engineering community was Gruber’s paper on portable ontologies. During this decade, ontologies are getting widespread attention as vehicles for sharing concepts within a distributed community such as the Web. Similar to task knowledge, patterns also play an important role on modeling domain knowledge (Powell, 1998).

2.1.2 Tasks and Problem-Solving Methods

In the early expert systems task knowledge was embedded in the reasoning engine and in rules in the knowledge base. The key point of Clancey’s “epistemology” paper was to explicate the underlying problem-solving method. Since then, the knowledge-engineering community has developed a range of such problem-solving methods. We can define a problem-solving method as follows:

A problem-solving method (PSM) is a knowledge-level specification of a reasoning pattern that can be used to carry out a knowledge-intensive task. To categorize problem-solving-methods we need a typology of knowledge-intensive tasks. Various (partial) task typologies have been reported in the literature. (Stefik, 2012) distinguishes between “diagnosis”, “classification” and “configuration”. Chandrasekaran has described a typology of” design tasks (including configuration). (McDermott, 2012) describes taxonomy of problem types. Table 1.1 shows the typology of task types distinguished by (Schreiber et al. (2004). In this section, we describe two problem-solving methods in more detail: one method for configuration design and one method for assessment. In general, there does not need to be a one-to-one correspondence between methods and tasks, although in practice there often is.

2.2 Two sample problem-solving methods

Propose-and-revise: The method was used to solve a configuration-design task, namely elevator design (the so-called VT case study (Marcus, 1988). The data for this case study was subsequently used in a comparative study in which different knowledge-engineering approaches were used to solve this problem. The results of the study were published in special issue of Human-Computer Studies (Schreiber & Birminghamm, 1996). This issue provides a wealth of information for readers interested in details of modern knowledge engineering. We will come back to this study in the next section, as reuse of the pre-existing ontology was a prime focus of this study (Powell, 1998).

Unlike some other methods, which undo previous design decisions, P&R fixes them. P&R does not require an explicit description of components and their connections. Basically, the method operates on one large bag of parameters (Britton & Doake, 2003). Invocation of the propose task produces one new parameter assignment, the smallest possible extension of an existing design. Domain-specific, search-control knowledge guides the order of parameter selection, based on the components they belong to. The verification task in P&R applies a simple form of constraint evaluation. The method performs domain-specific calculations (Vidgen, Avison, Wood & Wood-Harper, 2002).

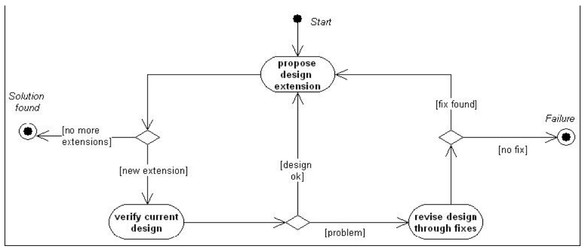

Figure 1: Top-level reasoning strategy of the P&R method in the form of a UML activity Diagram provided by the constraints

In P&R, a verification constraint has a restricted meaning, namely a formula that delivers a Boolean value. Whenever a constraint violation occurs, P&R's revision task uses a specific strategy for modifying the current design (Vidgen, Avison, Wood & Wood-Harper, 2002). To this end, the task requires knowledge about fixes, a second form of domain-specific, search-control knowledge. Fixes represent heuristic strategies for repairing the design and incorporate design preferences. The revision task tries to make the current design consistent with the violated constraint. It applies combinations of fix operations that change parameter values, and then propagates these changes through the network formed by the computational dependencies between parameters. Applying a fix might introduce new violations. P&R tries to reduce the complexity of configuration design by disallowing recursive fixes. Instead, if applying a fix introduces a new constraint violation, P&R discards the fix and tries a new combination. (Motta, Stutt, Zdrahal, Hara & Shadbolt, 1996) have pointed out that, in terms of the flow of control, P&R offers two possibilities. One can perform verification and revision directly after every new design or after all parameter values have been proposed.

The original P&R system used the first strategy, but Motta argues that the second strategy is more efficient and also comes up with a different set of solutions. Although this method has worked in practice, it has inherent limitations. Using fix knowledge implies that heuristic strategies guide the design revisions. Fix knowledge implicitly incorporates preferences for certain designs. This makes it difficult to assess the quality of the method's final solution. Assessment is a task not often described in the AI literature, but of great practical importance (Weaver, 2004). Many assessment applications have been developed over the years, typically for tasks in financial domains, such as assessing a loan for mortgage application, or in in the civil-service area, such as assessing whether a permit can be given. The task is often confused with diagnosis, but where diagnosis is always considered with some faulty state of the system, assessment is aimed at producing a decision: e.g. yes/no to accept a mortgage application. During the Internet hype at the start of this decade every bank was developing such applications to be able to offer automated services on the Web. Assessment starts off with case data (e.g., customer data about a mortgage application). As a first step the raw case data is abstracted into more general data categories (e.g., income into income class). Subsequently, domain-specific norms/criteria are retrieved (e.g. "minimal income') and evaluated against the case data (Klein, 2008). The method then checks whether a decision can be taken or whether more norms need to be evaluated. This basic method is typically enhanced with domain-s-specific knowledge, e.g. select inexpensive (e.g., in terms of data acquisition) first. The resulting decision categories are also domain-specific; for example, for a mortgage application this could be "accepted", "declined", or "flag for manual assessment"1. A detailed example of the use of this method can be found in the Common KADS book (Romi, 2000). (Valente and L¨ockenhoff , 1993) have published a library of different assessment methods.

2.3 The notion of "knowledge role"

Above we showed two examples of methods for different tasks. These methods cannot be applied directly to a domain; typically, the knowledge engineer has to link the components of the method to elements of the application domain. Problem-solving methods can best be viewed as patterns: they provide template structures for solving a problem of a particular type. Designing systems with the help of patterns is in fact a major trend in software engineering at large; see for example the work of (Gamma, Helm, Johnson & Vlissides, 1995) on design patterns2. The knowledge-engineering literature provides a number of proposals for specification frameworks and/or languages of problem-solving methods. These include the "Generic Task"approach (Avison & Fitzgerald, 2006), "Role-Limiting Methods", (Gangemi, 2005) "Components of Expertise" (Steels, 1990). Although there differences at a detailed level between these approaches, the one important commonality: all rely on the notion of "knowledge role": (Wielinga & Breuker, 1992).

Definition: A knowledge role specifies in what way particular domain knowledge is being used in the problems solving process. Typical knowledge role in the assessment method are "case data", "norm" and "decision". These are method-specific names for the role that pieces of domain knowledge play during reasoning (Puerta, et al, 1992). From a computational perspective, they limit the role that these domain-knowledge elements can play, and therefore make problem solving more feasible, when compared to old "old" expert-systems idea of one large knowledge-vase with a uniform reasoning strategy (Schreiber et al, 1994). In fact, the assumption behind PSM research is that the epistemological adequacy of the method gives one a handle on the computational tractability of the system implementation based on it. This issue is of course a long-standing debate in knowledge representation at large (see e.g., Dix et al, 2003)). Another issue that frequently comes up in discussions about problem-solving methods is their correspondence with human reasoning. Early work on KADS used problem solving methods as a coding scheme for expertise data (Wielinga & Breuker, 1984).

Over the years the growing consensus has become that, while human reasoning can form an important inspirational source for problem-solving method and while it is used to use role cognitively-plausible terms for knowledge role, the problem-solving strategy may well be different. Machines have different qualities than humans. For example, a method that requires a large memory space cannot be carried out by a human expert but presents no problem to a computer program. In particular methods for synthetic tasks, where the solution space is usually large, problem-solving methods often have no counterpart in human problem solving.

2.4 Specification languages

In order to put the notions of "problem-solving method" and "knowledge role" on a more formal footing, the mid '90s saw the development of a number of formal languages that were specifically designed to capture these notions. The goal of such languages was often two fold. First of all, to provide a formal and unambiguous framework for specifying knowledge models. This can be seen as analogous to the role of formal specification languages in Software Engineering, which aim to use logic to describe properties and structure of software in order to enable the formal verification of properties. Secondly, and again analogous to Software Engineering, some of these formal languages could be made executable (or contained executable fragments), which could be used to simulate the behaviour of the knowledge models on specific input data. Most of the language that were developed followed the maxim of structure preserving specification (Harmelen, Van and Aben, 1996): if the structure of the formal specification closely follows the structure of the informal knowledge model, any problems found during verification activities performed on the formal model can be easily translated in terms of possible repairs on the original knowledge model.

In particular the Common KADS framework was the subject of a number of formalization attempts; for an extensive survey. Such languages would follow the structure of Common KADS model into (1) a domain layer, where an ontology is specified describing the categories of the domain knowledge and the relationships between these categories. Knowledge roles link the components of the method to elements of the application domain; (3) inference steps that are the atomic elements of a problem solving method, and (4) a task definition which imposes a control structure over the inference steps to complete the definition of the problem solving method. A simplified example is shown in Fig. 1.3, using a simplification of the syntax of $(ML)^2$ F. (Van Harmelen & Balder 1993):

The domain layer specifies a number of declarative facts in the domain. These facts are already organized in three different modules.

- The knowledge roles impose a problem-solving interpretation on these neutral domain facts: any statement from the patient-data module is interpreted as data, any implication from the symptom-definition module is interpreted as an abstraction rule, and any implication from the symptomatology module is interpreted as a causal rule.
- The inference steps then specify how these knowledge roles can be used in a problem solving method: an abstraction step consists of a deductive (modus ponens) step over an abstraction rule, whereas a hypothesis step consists of an addictive step over a causation rule.
- Finally, the task model specifies how these atomic inference steps must be strung together procedurally to form a problem solving method: in this a sequence of a deductive abstraction step followed by an abductive hypothesis step.

The impact of languages such $(ML)^2$ (Richard & Vidgen, 2008), and many others (see (Sommerville, 2007)) was in one sense very limited: although the knowledge modelling methods are in widespread use, the corresponding formal languages have not received widespread adoptation. Rather than direct adoption, their influence is perhaps mostly seen through the fact that they forced a much more precise formulation of the principles behind the knowledge modelling methods.

2.5 Ontologies

During the nineties ontologies become popular in computer science. (Gruber, 1993) defines ontology as an "explicit specification of a conceptualization. Several authors have made small adaptations to this. A common definition nowadays is:

2.5.1 Ontology specification languages

Many of the formalisms can be said to be useful for specifying ontology. An insightful article into the ontological aspects of KR languages is the paper by Davis and colleagues (Preece et al, 2002). They define five roles for a knowledge representation, which we can briefly summarize as follows:

1. A surrogate for the things in the real world
2. A set of ontological commitments
3. A theory of representational constructs plus inferences it sanctions/recommends
4. A medium for efficient computation
5. A medium for human expression

One can characterize ontology-specification languages as KR languages that focus mainly on roles 1, 2 and 5. In other words, ontologies are not specified with a particular reasoning paradigm in mind. There have been several efforts to define tailor-made ontology-specification languages. In the context of the DARPA Knowledge Sharing Effort Gruber defined Ontolingua (Gruber, 1993). Ontolingua was developed as an ontology layer on top of KIF, which allowed frame style definition of ontologies (classes, slots, subclasses...). Additional software was provided to enable the use of Ontolingua as a mediator between different knowledge representation languages, such as KIF and LOOM. Ontolingua provided a library service where users share their ontologies.

2.5.2 Types of ontologies

Ontologies exist in many forms. Roughly, ontologies can be divided into three types:

Foundational ontologies Foundational ontologies stay closest to the original philosophical idea of "ontology". These ontologies aim to provide conceptualizations of general notions, such as time, space, events and processes. Some groups have published integrated collections of foundational ontologies. Two noteworthy examples are the SUMO (Suggested Upper Merged Ontology) and DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering). Ontologies for part-whole relations have been an important area of study. Unlike the subsumption relation, part-whole relations are usually not part of the basic expressivity of the representation language. In domains dealing with large structures, such as biomedicine, part-whole relations are often of prime importance. Such typologies are of practical importance as transitivity of the part-whole relation does not hold when different part-whole relations are mixed (""I'm part of a club, my hand is part of me, but this doesn't imply my hand is part of the club"). WordNet defines a semantic network with 17 different relation types between concepts used in natural language. The original Princeton WordNet targets the English-American language; WordNets now exist or are being developed for almost all major languages.

Domain-specific ontologies: Although foundational ontologies are receiving a lot of attention, the majority of ontologies are domain-specific: they are intended for sharing concepts and relations in a particular area of interest. Other well-known biomedical ontologies are the Unified Medical Language System8 (UMLS), the Simple Bio Upper Ontology9, and the Gene Ontology10. Domain ontologies vary considerably in terms of the level of formalization. Communities of practice in many domains have published shared sets of concepts in the form of vocabularies and thesauri. Such concept schemes typically have a relatively weak semantic structure, indicating many hierarchical (broader/narrower) relations, which most of the time loosely correspond to subsumption relations. This has triggered a distinction in the ontology literature between weak versus strong ontologies. The SKOS model11, which is part of the W3C Semantic Web effort, is targeted at allowing thesaurus owners to publish their concept schemes in an interoperable way, such that sharing of these concepts on the Web becomes easier. In practice, thesauri are important sources for information sharing (the main goal of ontologies in computer science). For example, in the cultural-heritage domain thesauri such as the Getty vocabularies12 (Art & Architecture Thesaurus, Union List of Artist Names, Thesaurus of Geographic Names) and Icon Class (concepts for describing image content) are important resources. Current efforts focus therefore on making such vocabularies available in ontology-representation formats and enriching ("ontologizing") them.

Task-specific ontologies: A third class of ontologies specifies the conceptualizations needed for carrying out a particular task. Data of configuration-design of an elevator system were used in the first ontology-reuse experiment in the nineties (Schreiber & Birmingham, 1996). In general, conceptualizations of domain information needed for reasoning algorithms typically takes the form of a task-specific ontology. For example, search algorithms typically operate on ontology of states and state transitions. Tate's plan ontology is another example of a task-specific ontology.

2.5.3 Ontology Engineering

Ontology engineering is the discipline concerned with building and maintaining ontologies. It provides guidelines for building domain conceptualizations, such as the construction of subsumption hierarchies. An important notion in ontology engineering is ontological commitment. Each statement in an ontology commits the user of this ontology to a particular view of the domain. If a definition in an ontology is stronger than needed, than we say that the ontology is overcommitted or example, if we state that the name of a person must have a first name and a last name we are introducing a western bias into the ontology and may not be able to use the ontology in all intended cases. Ontology engineers usually try to define ontology with a minimal set of ontological commitments. One can translate this into an (oversimplified) slogan: "smaller ontologies are better!" (Gruber, 1993) gives some principles for minimal commitments. Construction of subsumption hierarchies is seen as a central activity in ontology engineering. The Onto Clean method of Guarino and Welty define a number of principles for this activity, based on three meta-properties of classes, namely rigidity, unity and identity. Central in the Onto Clean method is the identification of so-called "backbone "classes of the ontology. Gangemi has published a set of design patterns for a wide range of modeling situations (Gangemi, 2005).

2.5.4 Ontologies and Data Models

The difference between ontologies and data models does not lie in the language being used: you can define an ontology in a basic ER language (although you will be hampered in what you can say); similarly, you can write a data model with OWL. Writing something in OWL does not make it an ontology! The key difference is not the language the intended use. A data model is a model of the information in some restricted well-delimited application domain, whereas an ontology is intended to provide a set of shared concepts for multiple users and applications. To put it simply: data models live in a relatively small closed world; ontologies are meant for an open, distributed world (hence their importance for the Web). So, defining a name as consisting of a first name and a last name might be perfectly OK in a data model, but may be viewed as incorrect in an ontology. It must be added that there is a tendency to extend the scope of data models, e.g. in large companies, and thus there is an increasing tendency to "ontologize" data models.

2.6 Knowledge Elicitation Techniques

Although this entire Handbook is devoted to the formal and symbolic representation of knowledge, very few if any of its chapters are concerned with how such representations are actually obtained. Many techniques have been developed to help elicit knowledge from an expert. These are referred to as knowledge elicitation or knowledge acquisition (KA) techniques. The term "KA techniques" is commonly used. The following list gives a brief introduction to the types of techniques used for acquiring, analyzing and modeling knowledge:

- Protocol-generation techniques include various types of interviews (unstructured, semi-structured and structured), reporting techniques (such as self-report and shadowing) and observational techniques
- Protocol analysis techniques are used with transcripts of interviews or other text based information to identify various types of knowledge, such as goals, decisions, relationships and attributes. This acts as a bridge between the use of protocol-based techniques and knowledge modelling techniques.
- Hierarchy-generation techniques, such as laddering, are used to build taxonomies or other hierarchical structures such as goal trees and decision networks.
- Matrix-based techniques involve the construction of grids indicating such things as problems encountered against possible solutions. Important types include the use of frames for representing the properties of concepts and the repertory grid technique used to elicit, rate, analyse and categorise the properties of concepts.
- Sorting techniques are used for capturing the way people compare and order concepts, and can lead to the revelation of knowledge about classes, properties and priorities.
- Limited-information and constrained-processing tasks are techniques that either limits the time and/or information available to the expert when performing tasks. For instance, the twenty questions technique provides an efficient way of accessing the key information in a domain in a prioritized order.
- Diagram-based techniques include the generation and use of concept maps, state transition networks, event diagrams and process maps. The use of these is particularly important in capturing the "what, how, when, who and why" of tasks and events.

Specialized tool support has been developed for each of these techniques. This wide variety of techniques is required to access the many different types of knowledge possessed by experts.

This is referred to as the Differential Access Hypothesis, and has been shown experimentally to have supporting evidence. The vertical axis on the figure represents the dimension from object knowledge to process knowledge, and the horizontal axis represents the dimension from explicit knowledge to tacit knowledge. The details of these techniques are described in a number of survey articles and textbooks, such as (Boose, 2017).

III. Traditional Information System Development Methodologies

The traditional methodologies such as System Development Life Cycle (SDLC), Object oriented analysis and design (OOAD), Waterfall and Rapid Application Development (RAD) have struggled to accommodate the web-specific aspects into their method and work practices.

3.1 System Development Life Cycle (SDLC)

System Development Life cycle (SDLC) is a technique that is used to divide the system development to many several stages. By following these stages the manager will find easy to manage any project. It is divided the system development into many stages which are: (Britton & Doake, 2003).

Problem definitions

- Feasibility study
- Requirements Engineering
- Design
- Implementation
- Maintenance

Strengths

- It follows sequence development steps and more strict controls for ensuring to get the documents to make the proper analysis and design reviews to make sure of the quality, reliability and maintainability for the system development software.
- It is ideal for supporting minimum experienced developer and project manager.
- System development activities are measurable.
- Conserve the resources.
- Requirements will be less.

Weaknesses

- Due to the sequence steps, the system development processes are inflexible, more time consuming and costly.
- The requirement inconsistencies and missing system objects are discovered during the analysis and design phases
- The problems cannot be discovered until the system testing
- System performance cannot be identified until the system is completed.

3.2 Waterfall Model

Waterfall methodology is "split in a sequence of consecutive phases. The output of each phase is the input for the next phase."(Vidgen, Avison, Wood, & Wood-Harper, 2002). The waterfall model is appropriate for large projects that its requirements well understood or very complex. (Weaver, 2004). In waterfall model, the developer cannot move from one stage to another stage except when the stage is completed (Klein, 2008).

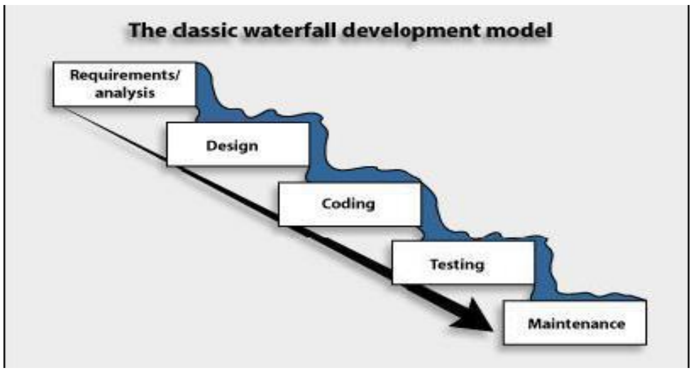

Figure 2: Classic Waterfall Development Model

Strengths

- It is like sequential steps which makes the methodology very simple, easy to understand, and simple to implement.
- It is suitable for large project that has well-defined requirements.
- In the waterfall model, each stage produces documentation and this makes understanding the product very simple.

Weaknesses

- It is time consuming to complete the project because of many steps.
- In waterfall model the developer cannot go back to the previous stage because, the waterfall is a sequential process. So, if any previous stage has gone wrong, the modification will be very complicated to fix or modify it.
- It is not suitable for small projects.
- Small changes or errors that arise in the completed software may cause a lot of problems.
- The Developer is communication with the client is extremely limited being either at the beginning or at the end of the development.
- There is minimal feedback between the phases.

3.3 Object Oriented Analysis and Design (OOAD)

An Object Oriented Analysis and Design (OOAD) techniques has been prominent since 1990. The OOAD techniques utilized the same basic principles as OOP languages such as Java, C++, and other programming languages. An object is the concept of all OO approaches. An object is "An abstraction of the real world based on objects and their interactions with other objects".(RomiSatriaWahono, 2000). An OO approach represents a stand-alone package, the process and data related with the real-world object. An object can communicate with other objects via messages.

There are three processes which are:

1. System Analysis
2. System Design
3. System Implementation

Strengths

- It is more reliable and flexible than SDLC because it is depended on the object and object's behaviors. This makes the requirements clearer.
- It is a modelling method which makes the system better in quality (increase or improve quality)

Weaknesses

- It is difficult to learn or understand it for the developer.
- Time consuming to draw objects (or using UML techniques e.g.10 diagrams)

3.4 Rapid Application Development Methodology (RAD)

The goal of RAD methodology is to enable the developer to develop a system and produce information system more quickly or much faster.

- Requirements Planning
- User Design
- Construction
- CutOver (Avison, & Fitzgerald, 2006)

Strengths

- It produces high quality system.
- It requires high level of commitment from the stakeholder, developer and customer.
- The customer views are very important to focus on the system elements.
- Based on customer demands, it provides the ability to change the system design.
- It makes tight fit between the user requirements and system specifications.
- Easy to use.
- The development and process in any of project is faster because of the shortened life cycles as compared with difference life cycles.
- Increases the project's quality in short time.

IV. Proposed Methodology for Web Development Process

Now a day's most of the web development processes adapt new web technology. The web technologies focus on technical skills and customer expectations. Building a web site is relatively easy, because the barriers to entry are low and development is largely uncomplicated. But building a web application is a hard work. Because of the rich content and its importance to the business. The developer should have to deal with many different stakeholders. Web developer should construct the website that is user friendly, it must also provide security and information consistency about businesses.

4.1 Wisdm Methodology

Web site developer uses the WISDM methodology for the modification of multiview aspects used locally and uniquely for web development in practice. Multiview‘s fundamental assumption: A system methodology that relies much over an engineering approach and technical rationality is, by itself, an insufficient foundation for IS development. The foundations of multiview needs of computer artifacts, organizations and individuals need to be considered jointly. The major concern of multiviews is the negotiations between technological, organizational, social and human aspects of system development (Richard, 2008). The WISDM mainly focuses on Organizational analysis, Information analysis, Technical design, Human computer interaction and Work design. There is no prior ordering of the five aspects of the said method. Each method has been emphasized alone during the website project development.

4.2 Organizational analysis

Organizational analysis is represented in multi-view soft systems methodology which is mainly focused on relevant situations characterized by complexity and stakeholder interests. Most of the projects get from the client organizations. The projects are web based in terms of Ecommerce which is related to Marketing and Sales. The stakeholder's goal is to make the business more successful in terms of measure and ability to generate the revenue. The E-commerce project aims to market and sell products and give valuable services to customers. The organizational analysis consists of building an e-commerce strategy and conducting a market survey to get the feedback from the customer to improve the organization strength and future improvements. The E-commerce survey was concerned with the development project with business strategy. The market surveys focused on customers and identify the perceptions to the Internet, E-commerce and also get the organizational information requirements. The Organizational analysis is concerned with value creation because the website is made public, it also may be of interest to customer who can access the website globally(McDermott. 2012). Web developer collects the user requirements from the organization. The user requirement has objectives of new system information for developing the Website.

The requirement might be in the form of documents with graphical notations and also be in software prototype. The indicative approach in WISDM uses UML Model. This model should represent the proposal functionality of the new system. The complete requirements are known at the outset and have strong initial ideas in our head. These are complete and require in depth exploration. The web developer should analyses the specific user requirements correctly and finding out exactly what is needed. The project plan is based on information analysis. The requirements Specification are met and consistent with client goal which includes all the functionality of the system.

4.3 Work design

The aim of work design is establishing relationship between customer and employee. It has been broaden from job satisfaction to a more general concern with user satisfaction in the e-commerce environment. The organization has managements. The management defines job requirements. These requirements fit for an employee's job expectations and job requirements to make the employee jobs satisfied. Most of the projects should target external customers and have minimal impact on working practices within the organization. The employee designs the website according to the customer needs. They always focus on usability, information quality and accessibility. The customer satisfaction is very important in web information system which enables the designer to reach other organizations to find external user during the development processes.

4.4 Technical Design

Technical design supports the website development. This design represents in the programming design and data structure to design the website. The website development can be categorized into static and dynamic website developments. The static websites are Standard HTML pages. It contains HTML code, which defines the structure and content of the Web page.

Each time an HTML page is loaded, it looks the same. The dynamic websites contains server-side code, which allows the server to generate unique content each time the page is loaded. The website developments are based on client (Browser) and server (Web server) model. Most of the websites are hosted on the server. The web browser can request particular website to a server.

The server processes the request and gives the response. Many people are involved during the website development. The Genuine participation involves customer, project managers, website developers and stakeholders who influence each other's plans policies and decisions which will affect the future outcomes of the website.

The website development needs programming design, database design and server software which is indicated in Table-1.

TABLE 1
Web Technology Softwares

Programming design software	HTML, XML, XSD, XST Java Script, VB Script, ASP.Net, JSP, PHP, CSS, FrontPage, Macromedia Dreamweaver with Fireworks and Flash for graphics and animation
Database design Software	MS Access, Oracle and SQL Server
Server	Websever, Email Server, Database server, Personal Web Server (Windows 98), IIS(Windows NT/2000/XP/2003), Apache, iPlanet Web Server, Roxen and Zeus

The web developer designs the website as to the look and feel for the customer. When the developer designs the website, they should focus on following points:

- Purpose of the website
- Subject of the website
- Target what the customer feels about the website (Graphical User Interface)

TABLE 2
Web Service Standards [10]

Web Technology	XML, ASP, JSP
Support	WS-Security, WS-Addressing
Services Definition	UDDI(Universal Description Discover and Integration) WSDL(Web Service Definition Language)
Messaging	SOAP
Transport Protocols	HTTP, HTTPS, SMTP, IMAP

In the technical design, web service-oriented architecture provides the facilitating of inter-organizational computing and executes these services geographically through distributed computing. This architecture allows flexibility as web services can be provided locally or through an external provider. Table 2 indicates the web service standards (Sommerville, 2007).

V. Conclusion

Over the years, the discipline of knowledge engineering has evolved into the development of theory, methods and tools for developing knowledge-intensive applications. In other words, it provides guidance about when and how to apply particular knowledge-presentation techniques for solving particular problems (Allen, 2016).. Principles that have become the baseline of modern knowledge engineering may include the common distinction made in knowledge engineering between task knowledge and domain knowledge (Artale, Franconi, & Pazzi, 2017). While on the other hand, the Traditional Methodologies such as System Development Life Cycle (SDLC), Object oriented analysis and design (OOAD), Waterfall and Rapid Application Development (RAD) have struggled to accommodate the web specific aspects into their method and work practices (Boose, 2017). Most of the websites are based on graphical hypermedia systems, database-driven information system, and also security systems. Each web site will have different goals and objectives and a unique set of problems, thus any methodology will require adaptation to the contingencies of each situation. The methodologies used for traditional systems development or web development have their uses and also their limitations.

REFERENCES

[1] Allen J., (2016). Maintaining knowledge about temporal intervals. *Communications of the ACM,* 26:832–843.

[2] Artale E., Franconi A., & Pazzi L. (2017). Part-whole relations in object-centered systems: An overview. *Data and Knowledge Engineering,* 20 (347–383).

[3] Avison, D. & Fitzgerald, G. (2006). Information Systems Development: Methodologies, Techniques, and Tools. *Fourth Edition. UK: McGraw Hill Education.*

[4] Boose. A. (2017). Survey of knowledge acquisition techniques and tools. *Knowledge Acquisition,* 1(1):3–37.

[5] Brachman R. J. & Levesque H. J. (2018). The tractability of sub sumption in frame-based description languages. *In AAAI 84.*

[6] Brachman R. J & Schmolze J. G. (2017). An overview of the KL-ONE knowledge representation system. *Cognitive Science, 9*: 171–216.

[7] Britton, C. & Doake, J. (2003). Software System Development. Third Edition. USA: *McGraw-Hill Education.*

[8] Chandrasekaran B. (2012). Generic tasks in knowledge based reasoning: High level building blocks for expert system design. *IEEE Expert,* 1(3):23–30.

[9] Chandrasekaran B. & Hamid (2010). Design problem solving: A task analysis. *AI Magazine,* 11:59–71.

[10] Clancey W. J. (2017). The epistemology of a rule based system -a framework for explanation. *Artificial Intelligence,* 20:215–251.

[11] Davis R., Shrobe H., & Szolovits P. (2011). What is a knowledge representation? AI 16 1, *Knowledge Engineering Magazine, Spring:*17–33.

[12] Dix et al,(2003) "Human Computer Interaction" *2nd Edition, Prentice Hall.*

[13] Fitzgerald B. (1997). Time to turn update the clock, in Wojtkowski G, Wojtkowski W, Wrycza S and Zupancic J (eds.) *Systems Development Methods for the Next Century, Plenum Press, New York.*

[14] Gamma E., Helm R., Johnson R., & Vlissides J. (1995). Design Patterns: Elements of Reusable Object-Oriented Software. *Addison-Wesley, Reading, MA.*

[15] Gruber T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition,* 5:199–220.

[16] Gangemi A. (2005). Ontology design patterns for semantic web content. In International Semantic Web

Conference ISWC'05, Galway, Ireland, LNCS, pages 262–276. *Springer-Verlag.*

[17] Klein, F. J. (2008). The Waterfall Model of Software Development *[Online].*

[18] Marcus S., editor (1988). Automatic knowledge acquisition for expert systems. *Kluwer, Boston.*

[19] McDermott J. (2012). Preliminary steps towards a taxonomy of problem-solving methods. In S. Marcus, editor, *Automating Knowledge Acquisition for Expert Systems,* pages 225–255. Kluwer, Boston.

[20] Motta E., Stutt A., Zdrahal Z., O'Hara K., & Shadbolt N. R. (1996). Solving VT in VITAL: a study in model construction and reuse. *Int. J. Human-Computer Studies,* 44(3/4):333–372.

[21] Powell, T A. (1998), Web Site Engineering. *New Jersey. Prentice Hall.*

[22] Preece et al (2002). "Interaction Design - Beyond Human-Computer Interaction" *Wiley.*

[23] Puerta A. R., Egar J., Tu S., & Musen M. (1992). A multiple-method shell for the automatic generation of knowledge acquisition tools. *Knowledge Acquisition,* 4:171–196.

[24] Richard Vidgen (2008), "Constructing a web information system development methodology" P 247-261, *Blackwell Science Ltd.*

[25] RomiSatriaWahono. (2000). Object-Oriented Analysis and Design Methodology. *[Online].*

[26] Schreiber A. Th., Akkermans J. M., Anjewierden A. A., de Hoog R., Shadbolt N. R., Van de Velde W., and Wielinga B. J. (2004). Knowledge Engineering and Management: The Common KADS Methodology. *MIT Press, Cambridge, MA,* December.

[27] Schreiber A. Th & Birmingham W. P. (1996). The Sisyphus-VT initiative. Int. J. Human-Computer Studies, 43(3/4):275–280. *Editorial special issue.*

[28] Steels L. (1990). Components of expertise. *AI Magazine, summer.*

[29] Stefik M. (2012). Introduction to Knowledge Systems. Los Altos, *CA. Morgan Kaufmann.*

[30] Vidgen, R., Avison, D., Wood, B. & Wood-Harper, T. (2002). Developing Web Information Systems. *Oxford: Butterworth Heinemann.*

[31] Van Harmelen F. & Aben M. (1996). Structure preserving specification languages for knowledge-based systems. *International Journal of Human Computer Studies,* 44:187–212.

[32] Van Harmelen F. & Balder J. R. 1993). (ML)2: a formal language for KADS models of expertise. Knowledge Acquisition, 4(1). Special issue: 'The KADS approach to knowledge engineering', reprinted in KADS: *A Principled Approach to Knowledge-Based System Development, Schreiber,* A. Th. et al. (eds.).

[33] Valente A. & L¨ockenhoff C. (1993). Organization as guidance: A library of assessment models. *In Proceedings of the Seventh European Knowledge Acquisition Workshop (EKAW'93),* pages 243–262.

[34] Weaver, P. (2004). Success in Your Project: A Guide to Student System Development Projects. USA: *Prentice Hall.*

[35] Wielinga B. J. & Breuker J. A. (1992). Models of expertise. In Proceedings ECAI–86, 18 1. *Knowledge Engineering.*

[36] Wielinga B. J. & Breuker J. A. (1984). Interpretation of verbal data for knowledge acquisition. In T. O'Shea, editor, Advances in Artificial Intelligence, pages 41–50, Amsterdam, The Netherlands,. ECAI, Elsevier Science. Also as: Report 1.4, *ESPRIT Project 12, University of Amsterdam.*

[37] Yang et al., (2002). Heuristic classification. *Artificial Intelligence,* 27:289–350.

Citation of this Article:

Dr. Oye, N. D., Jemimah Nathaniel, "Knowledge Engineering and Traditional Information System" Published in *International Research Journal of Innovations in Engineering and Technology - IRJIET*, Volume 4, Issue 2, pp 13-23, February 2020.

****