ABSTRACT
In our daily engagement with technology, we interact with software in many aspects, rendering software engineering field a very robust area with a lot of dynamism. In this paper we have comprehensively surveyed on software engineering approaches for designing knowledge based systems. First we looked at knowledge based systems in detail, then at software engineering basics, and finally match the various software engineering approaches that have been used in the field of knowledge based systems in the current research and surveys. Some of the current approaches in knowledge approaches that we surveyed include: case-based approach, software reuse approach, Model driven software engineering approaches, ontology based approach, Open metadata approach, cloud based approach, agile approach. Traditional integration approach, Automatic software generation code approach, process based approach, and the Knowledge based system approach.

Keywords
Knowledge based systems, Software engineering, Software engineering approaches.

1. INTRODUCTION
Knowledge based systems are increasingly achieving technological advancements, with a view to helping human beings solve some intelligence problems in our days to day working environments. This has been partly achieved due to the advancements of software engineering, which have greatly played a role in developing of this knowledge based systems. On the other hand software’s are also becoming more and more sophisticated in the way they operate and engineered, implying that both software engineering approaches and knowledge based systems have been advancing side by side. In view of this we surveyed the current trends in software engineering approaches for designing knowledge based systems. In achieving the same, we first had an overview of knowledge based systems, and then looked at software engineering approaches that have been used in developing these knowledge based systems. Our contribution in this paper is a survey of the current software engineering approaches that are being used in developing knowledge based systems.

2. KNOWLEDGE BASED SYSTEMS
Knowledge based systems as a system that contains significant amount of knowledge in explicit declarative form, which can be through mimicking the inductive and deductive reasoning of a subject matter expert. They are sometimes referred to as expert systems [16].

2.1 Components of a knowledge based systems
Reference [12] and [18] elaborates that knowledge based systems have the following components: -

2.1.1 The Knowledge Base
This is a centralized and organised repository of a knowledge based system organised by a computer system. It is in the knowledge base that all the facts about certain domain knowledge used by the system, to find either new knowledge or existing knowledge, are kept.

2.1.2 The Inference Engine
This is part of an expert system that uses artificial intelligence to draw out conclusions from the facts and rules stored in a knowledge base. In ascertaining new knowledge, or solving a problem, the inference engine could either use the facts to establish the goal through forward chaining, or it could use the goal to establish the facts through backward chaining.

2.1.3 The User Interface,
This is the part of a knowledge based system that the user interacts with in order to use the knowledge based system. It is through the user interface that the knowledge based system user can either get results of a problem, or feed the system with a task to search for their solutions. User interfaces range from screens, mouse, terminals, keyboards etc.

Figure 1 shows basic components of knowledge based system, and their interaction.
2.2 Methods of representing knowledge on knowledge based systems
Reference [1], [4] and [18] points out that there are several ways of representing knowledge in knowledge based systems, some of these representation methods include:

2.2.1 Case based reasoning
Process of solving new problems based on the solutions of similar past problems in the knowledge base. It involves going through the knowledgebase and comparing the current problem with the already existing solved problems stored in the knowledgebase. Then it uses the solutions made earlier to solve the current problem, if there exists similar scenarios. They have been used in making recipes, judgments etc.

2.2.2 Propositional logic
This is a symbolic reasoning whereby proposition stand for units and logics as their connectives. In this method, there has to be propositions which are statements that can either be true or false. These propositional statements should be structures which follow the syntax of a logic language referred to as well-formed formulae (wff). Hence in making a decision, the problem has to be mapped as proposition and hence a solution could be derived from the wff.

2.2.3 Decision trees
A schematic tree-shaped structure used to determine a course of action, where each branch of the decision tree represents a possible decision or occurrence. In developing decision trees, write the a decision that you intend to make, then draw out lines towards the right for each possible solution for the decision tree, then write that solution along the line. Thus to find a solution for a particular problem, one can traverse through the tree and reach a desired solution.

2.2.4 Bayesian network
Refers to a directed graph in which each node is annotated with quantitative probability information in the form of U = \{X1, X2, ..., Xn\}. In Bayesian network, not all edges point in the same direction. This is because different edges have different alternative paths that represent possible influence within the variables.

2.2.5 Ontology
This is a formal representation of knowledge about a given domain. When using ontology it is possible to create knowledge bases that can be shared, extended and reused in different scenarios. The ontologies usually offer common agreed vocabulary that agents can use to make decisions. When the agents interact with these vocabularies, then they can make a certain decision based on the problem.

2.3 Characteristics of good knowledge based systems
There are several knowledge based systems that have been designed, several of them have good characteristics while others could have a few of the good characteristics. A good knowledge based system is expected to have the following characteristics. It should be complete, concise, and transparent. It should as well allow for incremental refinement and extension of the knowledge, facilitates computation, and is computable by an existing procedure[5] and [22].

2.4 Benefits of knowledge based systems
As suggested by[1] and [14], there are a number of benefits that users derive from knowledge based systems, from their expert knowledge. These benefits include:

2.4.1 Availability
Knowledge based systems and expert systems are highly available easily due to mass production of them in software.

2.4.2 Cheaper
The cost of providing knowledge based systems is cheaper compared to training of the experts themselves.

2.4.3 Reduced danger:
They can be used in any risky environments where humans cannot work with ease and expected performance.

2.4.4 Permanence
The knowledge based system performance will last long as long as they are still relevant.

2.4.5 Multiple expertise in one system
One system can be designed to have knowledge of many experts.

2.4.6 Explanation
They are capable of explaining in detail the reasoning that led to a certain conclusion.

2.4.7 Fast response
They can respond at great speed due to the inherent advantages of computers over humans.

2.4.8 Unemotional and response at all times
Unlike humans, they do not get tense, fatigue or panic and work steadily at all the situations.

2.5 Disadvantages of knowledge based systems
On contrary [1] and [22] further illustrates that knowledge based systems have disadvantages and challenges too. Some of the disadvantages include:

2.5.1 Common sense
In addition to a great deal of technical knowledge, human experts have common senses which lacks in the knowledge based systems, and it is still unclear how to impart the senses on them

2.5.2 Creativity
Human experts can respond creatively to unusual situations, knowledge based systems and expert systems cannot.

2.5.3 Learning
Human experts automatically adapt to changing environments; knowledge based systems and expert systems must be explicitly updated. Case-based reasoning and neural networks are methods that can incorporate learning, which is not so possible on other ways of representing knowledge.

2.5.4 Degradation
Knowledge based systems and expert systems are not good at recognizing when no answer exists or when the problem is outside their area of expertise.

2.6 Selected applications of knowledge based systems
A number of knowledge based system, as outlined by [18], [21] and [14], have been used in the development of intelligent systems. These systems have performed favorably in the areas like, medical diagnostics, digital equipments, business predictions etc. some of the notable applications of the knowledge based systems and expert systems that gained attention include:-
2.6.1 MYCIN
An expert system build in 1970 by Buchanan and Shortliffe that used rule based system with heuristics focusing on the medical domain for selecting therapies for infectious blood diseases.

2.6.2 ALVINN (automatic land vehicle in neural networks)
It is an intelligent system that took images from cameras and laser ranges as input and from the knowledge base it provided directions for the vehicle. It was generally a neural network robotic for vehicle driving.

3. SOFTWARE ENGINEERING
Reference [20] and [15] defines software engineering as an engineering discipline that is concerned with all aspects of software production, which should deliver the required functionality and performance to the user and should be maintainable, dependable, and usable.

3.1 Software engineering activities
Software engineering has four basic fundamental activities in the software process, though different software has different requirements. Below are the fundamental software engineering fundamental activities [15] and [20]:

3.1.1 Software specification
A stage where customers and engineers define software that is to be produced and the constraints on its operation

3.1.2 Software development
A stage where the software is designed and programmed

3.1.3 Software validation
A stage where the software is checked to ensure that it is what the customer requires

3.1.4 Software evolution
A stage where the software is modified to reflect changing customer and market requirements

3.2 Software process
Software process is a framework for building high quality software, and defines the approach that is taken as software is engineered. But software engineering also encompasses technologies that populate the process [15]. Reference [11] stresses that professional system developers and the customers they serve share a common goal of building information systems that effectively support business process objectives. In order to ensure that cost effective, quality systems are developed which address an organization’s business needs, developers employ some kind of system development process model to direct the project’s lifecycle. The most difficult task in software engineering is to select an appropriate software process model, which will perfectly solve the software requirements objectives.

4. SOFTWARE ENGINEERING APPROACHES FOR KNOWLEDGE BASED SYSTEMS
There are many different types of software systems, from simple embedded systems to complex, worldwide information systems. It is pointless to look for universal notations, methods, or techniques for software engineering because different types of software require different approaches. Developing an organizational information system is completely different from developing a controller for a scientific instrument. Neither of these systems has much in common with a graphics-intensive computer game [20]. All of these applications need software engineering; they do not all need the same software engineering techniques. Hence there is need to give each information system its comparative and workable software, which will enable the comprehensive working environment. Some of the software engineering approaches that could be employed to the various applications include the following:

4.1 A case-based approach
Case-based approach allows reuse of software without the usual and significant effort for making software explicitly reusable. The case based approach even supports such reuse for only partially developed requirements, since it allows reuse already without the need to develop a complete specification first. The solution information of the most similar problems to the developed problem can then be taken for reuse and adapted to the newly specified requirements. The specification of these new requirements can be facilitated, since the retrieved software case contains related requirements, which may be reused as well [9].

4.2 Software reuse approach
Software reuse is often just addressed at the level of code or low-level design. He stresses that software reuse deals with requirements reuse in the context of product lines. It makes the relations among product line requirements explicit, so that single system requirements in this product line can be derived consistently. It elaborates that lines with commonality and variability across different products could be reused to engineer the requirements for innovative new products. The second approach for software reuse involves case-based reasoning. Instead of explicit relations between requirements, similarity metrics are employed for finding the most similar software case in a repository to a given set of requirements [8], [9], and [21].

They further stresses that software reuse approach even works when a single envisioned usage scenario is specified; it also allows reusing requirements from retrieved cases. The third approach believed to be still under development strives for (partly) automating software development for certain business applications through reusing business knowledge and software, where both are tightly connected. It involves automated reuse of business processes, and software executing them, based on ontological knowledge to develop taxonomies for business models.

4.3 Model driven software engineering approaches
References [3] and [6] points out that businesses rely on software solutions to preserve their position in a highly competitive market, the need for reliable and robust software systems is extremely vital to remain in business. Of late there has been a significant interest in building software using models as their main artifacts. Unlike traditional development techniques which tend to be code-centric, model driven approaches. Model driven software engineering approach stress the usage of models at all levels of the software development life-cycle. This paradigm shift in software engineering has impacted not only the way software is built but also the way software engineering is being taught. They argue that this standard introduced a significant body of knowledge that should be integrated in a software engineering curriculum.
4.4 Ontology based approach
References [19] and [23] explain that ontology is formal representations of knowledge about a given domain. Based on them, it is possible to create more expressive knowledge bases that can be shared, extended and reused in different scenarios. In software engineering, ontologies can contribute in many different ways to improve the models, methods, techniques and processes of software development in our days today use. However, ontologies have three major challenges in software engineering, namely:-
- Difficulty in communicating and sharing information;
- Effective management of software development phases; and
- Development techniques and environments to support the production of semantic software through an interdisciplinary approach.

4.5 Open metadata approach
Reference [3] introduces a situational method engineering called OPEN. He argues that OPEN is a well-established, standardized and popular approach for situational method engineering. It has a large repository of reusable method fragments called OPF, that method engineers can select and assemble them according to the requirements of a project to construct a new project-specific software development methodology.

4.6 Cloud based approach
Cloud is the in thing during these times, with the flexibility among other demands such as in manufacturing, for rapid-change-over to new product variants. On the other hand robots are flexible machines that potentially can be adapted to a large variety of production tasks; while robot programs are hardly reusable from an application point-of-view to improve the situation, a knowledge-based approach exploiting distributed declarative information and cloud computing offers many possibilities for knowledge exchange and reuse, and it has the potential to facilitate new business models for industrial solutions [9]. They further explain that there are many unresolved questions yet, such as reliability, consistency, or legal responsibility with the cloud based approach. They conclude by advising that distributed cloud-based approaches offer many possibilities, but there is still a need for further research and better infrastructure before this approach can become industrially attractive.

4.7 Agile approaches
Nowadays’ projects have to cope with increasingly dynamic and turbulent environmental conditions, claiming that agile approaches are one possibility to successfully face this challenge. Combining agile with more traditional process models seems to be usual software development practice in industry, which in most cases lacks scientific reflection. The agile information management provides process designers with a tool suite that consists of roles, values and principles and a set of various methods and that implements iterative and incremental processes in small steps [17]. Early results of a case study confirm the appropriateness of the approach for challenging frequent changes due to changing markets, user needs or vague requirements, interdisciplinary cooperation and communication between the involved roles makes agile combined with traditional approaches in a better position for this.

4.8 Traditional integration approach
Reference [14] and [17] advises that the engineering of complex production automated systems involve experts from several backgrounds, such as mechanical, electrical and software engineering. This production automation expert knowledge embedded in these tools and data models, which are sometimes unfortunately and insufficiently integrated across the expert disciplines, due to semantically heterogeneous data structures and terminologies. Hence, they stresses that traditional integration approaches to data integration using a common repository are limited as they require an agreement on a common data schema by all project stakeholders. They explain how data from real-world use cases from the production automation domain on data exchange between tools and model checking across tools.

4.9 Automatic software generation code approach
Development of avionics software demands more and more human intervention. The software development process is rather highly expensive and risky since it must undergo several certifications. Hence the automation which deals with automatic code generation, of such a process can reduce errors and human interaction. These approaches are actually based on pre-defined fixed rules and policies to automatically generate software code which still demands considerable hand-coded software. The automated decision-making process relies on a reasoner, and an ontological database that captures development skills from human developers as well as the dynamic configurations and capabilities of the system under development. The knowledge captured enables the high level reasoning to automatically generate the control software code [7].

4.10 Process based approach
Reference [2] argues that the documentation of architecture and design decisions lies at the backbone of building a comprehensive architectural knowledge basis within a company. As a consequence, a plethora of supporting frameworks has been lately proposed by the research community. The existing frameworks focus on capturing the rationale that lies behind a certain decision, but less on sustaining the collaborative process that architects employ when making decisions. They advise that this approach is still in development, but advises that the processes should be kept in some knowledgebase to be used for software engineering, instead of defining the process afresh.

4.11 Knowledge based system approach
Currently companies are turning their attention to the increasing market competition and the quality of their products is directly related to the organizational processes that should be well defined and adopted. They advise that in software development it is important that the choice of a development method is aimed at matching organization needs and its production culture. The micro and small enterprises face many problems and one of the largest ones is the lack of policies that can help improving development processes [13]. Hence, they are suggesting a set of good practices in software development that closely match company needs and culture represented as a profile.
5. DISCUSSION

We ran a brief survey on the frequency of use of these software engineering approaches, by various knowledge based systems software developers in Kenya. The survey was in form of a short open questionnaire. The questionnaire’s main question was tasking the developers to tick the various software approaches they had used in designing the latest twenty (20) knowledge base systems; as well as to indicate the number of times they used each software approach in their designs of knowledge based systems. In case where there was more than one approach in designing a particular knowledge based system, then they should consider the software approach which was mainly utilized in the design.

Table 1 below shows a summary of results that we received from ten local system developers in Kenya, in designing their latest 20 knowledge based systems.

<table>
<thead>
<tr>
<th>No.</th>
<th>System developer number / Approach used</th>
<th>01</th>
<th>02</th>
<th>03</th>
<th>04</th>
<th>05</th>
<th>06</th>
<th>07</th>
<th>08</th>
<th>09</th>
<th>10</th>
<th>Total usage</th>
<th>% Usage</th>
<th>Rank on usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A case-based approach</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>29</td>
<td>14.5%</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>Software reuse approach</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>36</td>
<td>18.0%</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>Model driven software engineering</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>27</td>
<td>13.5%</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>Ontology based approach</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>14</td>
<td>7.0%</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Open metadata approach</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>08</td>
<td>4.0%</td>
<td>9</td>
</tr>
<tr>
<td>6</td>
<td>Cloud based approach</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>16</td>
<td>8.0%</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>Agile approaches</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>10</td>
<td>5.0%</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>Traditional integration approach</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>25</td>
<td>12.5%</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>Automatic software generation code</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>17</td>
<td>8.5%</td>
<td>5</td>
</tr>
<tr>
<td>10</td>
<td>Process based approach</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>09</td>
<td>4.5%</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>Knowledge based system approach</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>09</td>
<td>4.5%</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td><strong>TOTALS</strong></td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>200</td>
<td>100%</td>
<td></td>
</tr>
</tbody>
</table>

From table 1, assuming that the same data applies across the board, then the following conclusions could be arrived at. Software reuse approach has the highest use, while agile metadata approach has the least use in designing knowledge based systems. Further rankings on the software approach use, could be seen on table1.

6. CONCLUSIONS AND FURTHER RESEARCH

The integration of matured artificial integration methods and techniques, with conventional software engineering remains difficult and poses both implementation problems and conceptual problems. A closer look points out that both disciplines of Artificial Intelligence and Software engineering have many commonalities. They both deal with modeling real world objects from the real world like business processes, expert knowledge, or process models. Several research directions of both disciplines come closer together and are beginning to build new research areas. Some of these research areas include; Software Agents, which play an important role as research objects in distributed artificial intelligence; as well as in agent oriented software engineering, with intense Knowledge-Based Systems (KBS) investigations for learning software organizations as well as knowledge engineering [10].

We are also in a position to say that knowledge based systems are still in development and more so how to implement them in the current software engineering paradigms, as illustrated in our survey. Some of the current software engineering approaches in designing knowledge bases which we surveyed include: case-based approach, software reuse approach, Model driven software engineering approaches, ontology based approach, Open metadata approach, cloud based approach, agile approach, Traditional integration approach, Automatic software generation code approach, process based approach, and the Knowledge based system approach.

A summary of the frequency and popularity of the usage of the surveyed software engineering approaches, from the most used to the least used, is as follows: software reuse approach, case-based approach, Model driven software engineering approaches, Traditional integration approach, Automatic software generation code approach, cloud based approach, ontology based approach, agile approaches, Open metadata approach, process based approach, and the Knowledge based system approach.

We thereby recommend that a study should be conducted to ascertain why there is very high usage of Software reuse approach, and a very low usage of Knowledge based system approach in designing knowledge based systems.
7. REFERENCES


