Developing an Intelligent User Interaction Development Engine

Ashit Kumar DUTTA

Department of Computer Science and Information System., Shaqra University, KSA

Abstract

This paper presents an intelligent user interaction development (IUID) engine that helps to enhance the structure of the relational database by using artificial intelligence. The IUID consists of two phases, the first phase enhances the user database by comparing it with a well structured pre-defined database, and the second phase gives the user the ability to use the artificial intelligence by writing java code and organize it in a tree structure. The main objectives of the IUID engine are to allow user to use the expert knowledge to upgrade his/her database, and increasing the speed of the development process by appending a new artificial intelligence layer at the user application that is represented by a package to run in the application

1. Introduction

Databases are gaining prime importance in a huge variety of application areas employing private and public information systems. Databases are built with the objective of facilitating the activities of data storage, processing, and retrieval associated with data management in information systems. An intelligent database is a full-text database that employs artificial intelligence (AI), interacting with users to ensure that returned items (hits) contain the most relevant information possible. This is in contrast to a traditional database, which is searchable only by keywords and verbatim phrases connected by Boolean operations such as AND, OR, and NOT. Intelligent database technology is in its infancy, and is evolving as AI becomes more advanced. An Intelligent Database System is endowed with a data management system able to manage large quantities of persistent data to which various forms of reasoning can be applied to infer additional data and information. This includes knowledge representation techniques, inference techniques, and intelligent user interfaces - interfaces which extend beyond the traditional query language approach by making use of Artificial Intelligence. This technique plays an important role in enhancing databases systems.In this paper we have designed an engine with artificial intelligence algorithm which can upgrade the user database.

2. Related Work

User interface design has been a topic of considerable research, including on its aesthetics. In the past standards have been developed, as far back as the eighties for defining the usability of software products. One of the structural basis has become the IFIP user interface reference model. The model proposes four dimensions to structure the user interface:

- The input/output dimension (the look)
- The dialogue dimension (the feel)
- The technical or functional dimension (the access to tools and services)
- The organizational dimension (the communication and co-operation support)

This model has greatly influenced the development of the international standard ISO 9241 describing the interface design requirements for usability. The desire to understand application-specific UI issues early in software development, even as an application was being developed, led to research on GUI rapid prototyping tools that might offer convincing simulations of how an actual application might behave in production use.[3] Some of this research has shown that a wide variety of programming tasks for GUI-based software can, in fact, be specified through means other than writing program code.

Research in recent years is strongly motivated by the increasing variety of devices that can, by virtue of Moore's Law, host very complex interfaces.

There is also research on generating user interfaces automatically, to match a user's level of ability for different kinds of interaction.

3. The Intelligent User Interaction Development Engine

The IUID consists of two stages the database stage and artificial intelligence algorithms (AI) stage, each stage consists of 3 phases as shown in figure 1.

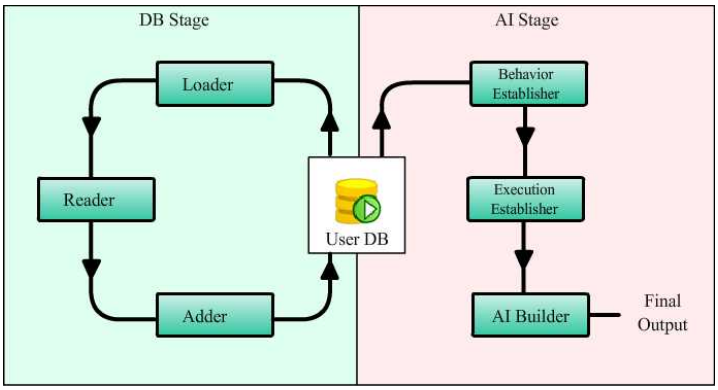

Figure 1: The architecture of the IUID engine

The database stage consists of Loader, Reader and Adder. It starts by loading the user database into java scope as objects. The Reader concerns about reading the user database and understand every table and its columns. The table and all column names must be known by the reader. At this phase the user asked to modify any unknown table or column names. In the adder phase, the user database will be compared with a well structured relational database recommended some suggestions about the user database related of adding or deleting some features from it. If the user doesn't have a database then our engine will help him/her to build a new a well structured relational database.

The output of the first stage is a well-structured identified database uses as input to the second stage. The three stages appear in figure 2.

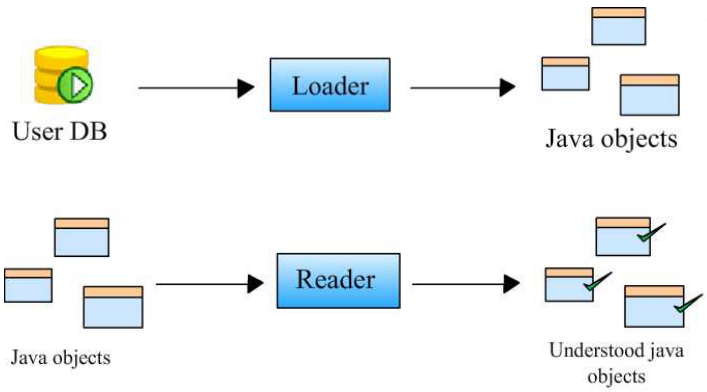

Figure 2: The structure of Loader, Reader, and Adder

The AI stage consisted of three phases: Behavior Establisher, Execution Establisher, and the AI builder. The behavior establisher starts working on the user database after enchantment by the previous phase. The user can create nodes called behaviors; these behaviors are java codes that are connected to entities from the user database, all the behaviors calculation will reflect on the user DB. The behavior establisher presented in Figure 3.

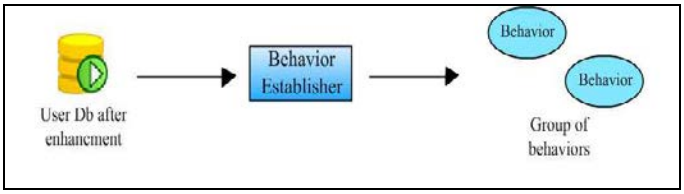

Figure3 Creating the behaviors in Behavior Establishment phase.

The execution Establisher phase main purpose is to group the behaviors made by user in a tree structure that the user will have the ability to define some rules to navigate throw it.
Many types of trees are available like: list, Recursion tree and conditional tree. The execution establisher presented in figure 4.

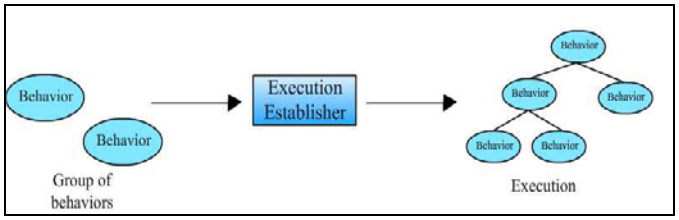

Figure 4 Building the execution tree by using group of behaviors in Execution Establishment.

At the AI stage the user can design tree algorithms to use in his/her application. In order to establish the tree algorithm, the user has to build it step by step starting with:

i. Creating an execution which represent the tree of the algorithm
ii. The tree of the execution contains nodes that hold the user written-by-wizard code.
iii. These node
iv. s defined in the execution called AI-behaviors.
v. Each AI-behavior (node) should have a parameters list and a return variables list.

vi. Each of these lists have a group of variables that represent a cell in the user database, these variables are called AI-cells.

There are three types of executions (tree); conditional node tree, recursion tree and execution list.
The conditional node tree should have conditional nodes that are used to control the run path of the tree.

Each conditional node should have a group of enter points[1] ,one for each child behavior, that are represented by a Boolean method that the user should code, this enable the user to use his logic(variables and methods in his own application) to make the conditional node decision. The AI builder stage presented in figure 5.

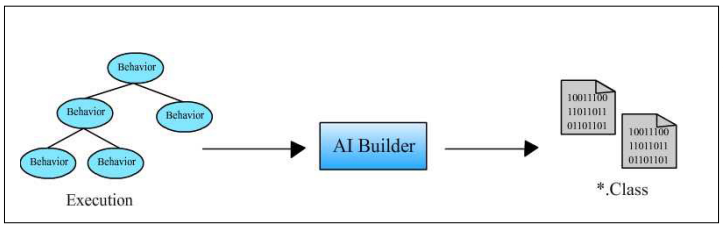

Figure 5 Generating .class files in AI Building Phase.

The AI builder phase is the last phase that works on the executions and transforms them into java classes; each class represents a tree of execution.
Finally the output classes will be grouped into a package and will be given to the user, so he can use these classes in his project.

4. Implementation

Our system was developed in Oracle 10g enterprise edition release , MySql Server , and NetBeans were implemented on an Intel® Pentium® M dual core with speed 1.8 GH processor and DDR@ I GBRAM, running on the Microsoft Windows XP Professional Service Pack 2.
Our system work space arranged to design and developed a high quality structural database by using the AI structure algorithm. Figure 1 presents the workspace that includes menus, variety of tools and palettes for viewing, editing and adding new elements to your project.

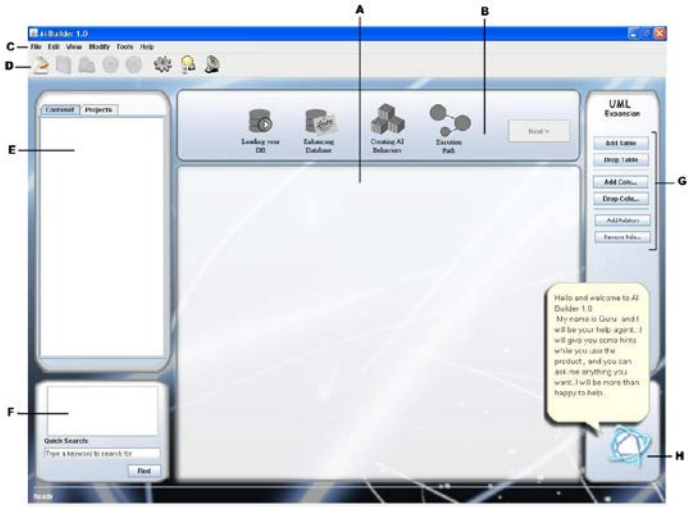

Figure 6: Main screen **A.** Project work Area **B.** Progress palette **C.** Menu bar **D.** Tools bar **E.** Project contents **F.** Search palette **G.** Tools palette **H.** Help agent

When the user starts a new project or open an old one, the user selects the database category type and import his\her database that should be installed in a database service provider (like oracle…). After the user has started his\her project the project shall have a directory folder saved in the pre-defined work space of AI-Builder.

The project starts with loading the user database very efficiently from Oracle, MySql or SqlServer into java scope (project scope) using the Loader Class, some tests has been made in a modern pc to insure the speed reliability of Loader and one of the results was :
167 table each with (on average) 10 columns and at least 50 rows have been loaded successfully in 3.8 seconds.
The Loader class has the ability to load a database from deferent database service providers include: Oracle, MySql and SqlServer
Which are the most commonly used around the world and that insures the flexibility of the resulted database. The Loader is a dynamically implemented means that it can load the tables whatever its structure design was or its number of tables or columns, and this is a needed feature in AI-Builder to be able to handle any user data base. However the class was built in a specific design so it can store the tables in a way that separate the structure [1] and data from each other insuring Loader ease of use and performance stability. (See figure 7)

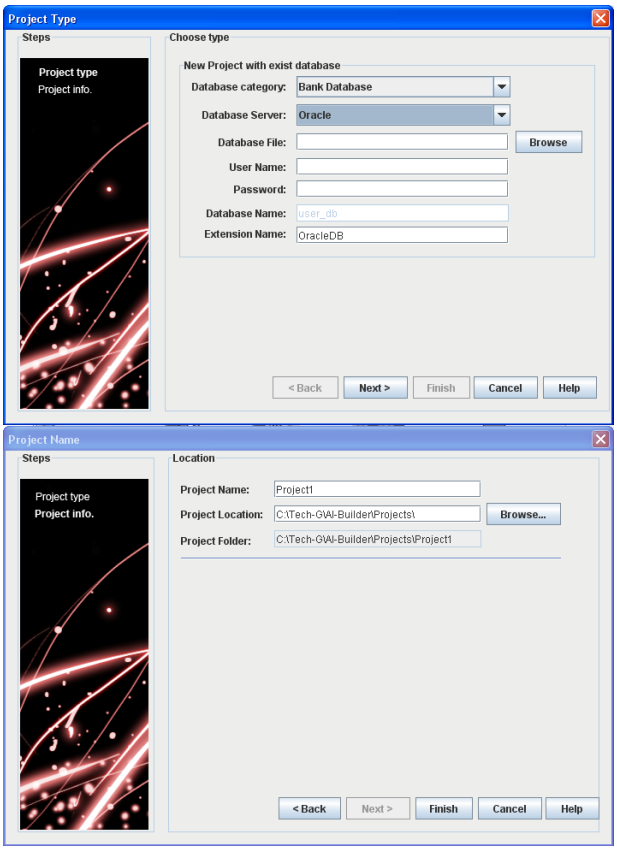

Figure 7: these figures shows some information the user has to fill to complete the connection to the database like database type, database provider, user name, and password.

The Reader starts with reading all the loaded tables, this done so AI-Builder can recognize what does the user mean (its type) by any selected cell in the user database.
When reading a table the reader starts with identifying its name by a text recognition engine [2] that investigate if the table is known to Reader or not known, so AI-Builder can ask the user to identify it. I must point here to the flexibility of the Reader design, the Reader can be bound easily with any text recognition algorithm with minimum code changes that need only one hour effort.

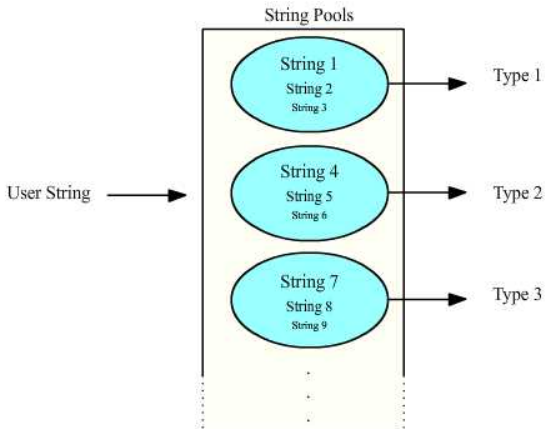

Figure 8: types-pool mapping algorithm

After all the table's names are identified, the Reader starts to read the columns inside the table using text recognition engine in the same way as the tables' names identification. Every table name or column name are identified by the types-pool mapping algorithm[3], and its works in this way: if the name (table's or column's) given from the user is not recognized by text recognition the name is added to a pool of names that are all bound to a specific general type (an answer to the following question: what does the user mean by this table and this column so data in cell are meaningful) that is used later by AI-Builder.(See figure 5.3)

An important feature of Reader is the ability to learn from the user, in other words it never ask the user to identify something twice as Reader store it in the first user identification. (See Figure 5.4)

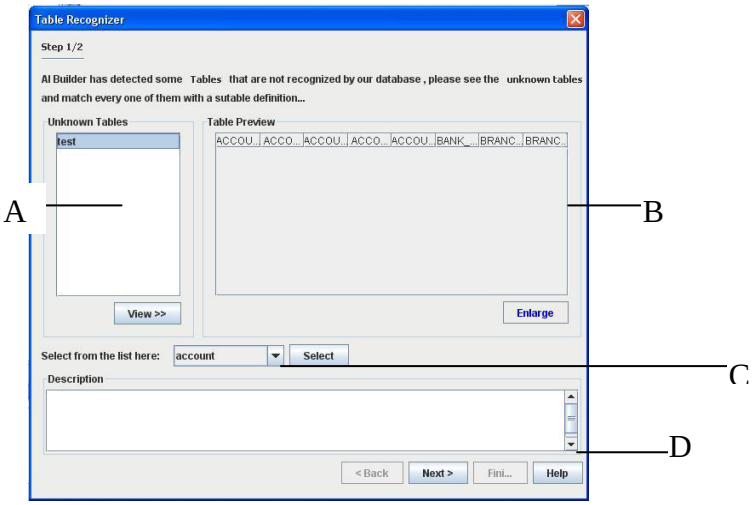

Figure 9: Table recognizer
A. list of unknown tables **B.** preview of the unknown table
C. Table type **D.** Description palette

Before the reading operation the Reader checks the tables and asks the AIHandler [1] to load the needed information from the recognition database, and then start the reading operation, that uses the text recognition engine.
While the reading operation, names that are not understood by Reader are enquired in an updates buffer and treated as new knowledge that needed to be add to the AI database, these updates are stored in the updater class.

The adder engine main job is to enhance the user tables after they are read by the reader engine, all the tables at this point are well understood and ready to enter the adder phase.
It starts by comparing the structure of the user tables with a structure of a general tables in the general database [1] stored in our program, the design of the general database passed throw several steps to insure its effectiveness; this is has been accomplished by an intensive study to database principles and collecting information from deferent resources.

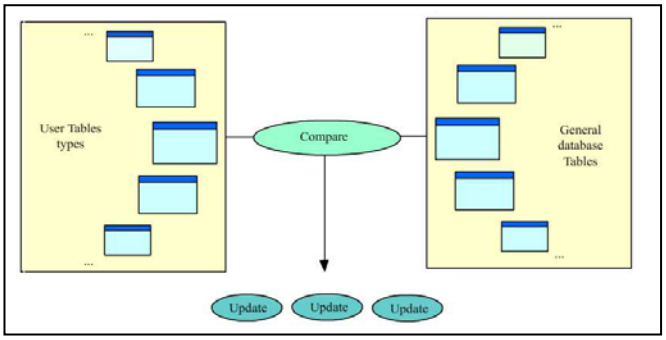

Figure 10: Comparing the user tables with the general database tables.

The comparison is done in two phases; the first one includes comparison the user tables with the general database tables, that means the columns is not included in this phase. (See Figure 5.5)
The next phase of comparison is comparison the columns of each table with another set of columns from the general database, the result from the both phases will be a group of updates that will be given to the user. (See figure 5.6)

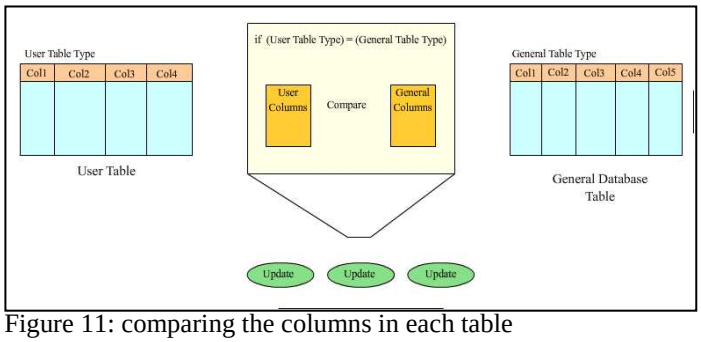

Figure 11: comparing the columns in each table

After the adder has done the comparison, Adder will suggest group of updates that are listed in a buffer and these updates represents the result of the Adder comparing the general database with the user database. Any new addition on the user database will be grouped in a list and will be given to the user to determine if he/she wants to use any of them or not,
The addition to the user database includes adding new tables, new columns and creating new relations between tables; deletion is not an option to the adder so the user will be sure that no lost of tables will happen to his\her database.
The user can add a new general database as a new category that is added to the adder as easy as one click; the user can import his\her database structure into the program and make it as a general database so AI-Builder can use it later to enhance any future work, also he/she can import the database that a viable in any of the most commonly used database service provider in the world; MySql, oracle and SQL Server.
The idea of adding new general database will give the adder a huge advantage by making it able to learn. Every new general database means a new field of experience added to its huge previous experiences, this feature gives a really good advantage at development time, the developers can use, work-on and benefit, so the experience of a single expert is effectively reused.
The ability of learning that is represented in the reader and the adder engines makes AI-Builder a huge resource of valuable and precious information collected throw every use of the program, the more you use the program the cleverer it will be.
After the database has been well defined and enhanced, the user can start to build his\her AI algorithms that he will use later in his\her application, the final output of this stage is a package of classes that represents AI executions (the AI algorithms) so the user can simply execute any of them anywhere in his\her application by a method call.
The major advantage of this binding is that the user can build complex AI algorithms and implement its code and use it in a short time, and the reader may ask what is the benefit ?, what is the main advantage? And the answer is simply that your developers will do things faster easier and the AI expert doesn't have to worry about doing the structure design that will hold his\her AI code.

Behavior Establishment:

The user can build an AI algorithm by first constructing some behaviors so they can be used in the algorithm, a behavior is a list of statements (code lines) that are treated as a method in a programming language, the user write code in a behavior to a specific thing that he want, remember that a behavior is only a part of the code because it represent a node in a tree and one path in that tree will be the code that will execute.
each behavior have two lists of variables that are accessible inside the behavior , first the parameter variables list which represents the parameters used in the behavior and second the return variables list that represent the variables that will be affected after the behavior has completes.
The variables that are defined in the two lists each represents a cell in the user scanned database (this is done in the Reader phase) so the user can write the behavior code to use the values of the parameters and to return new values to the return variables (as update these database cells at user application run time).

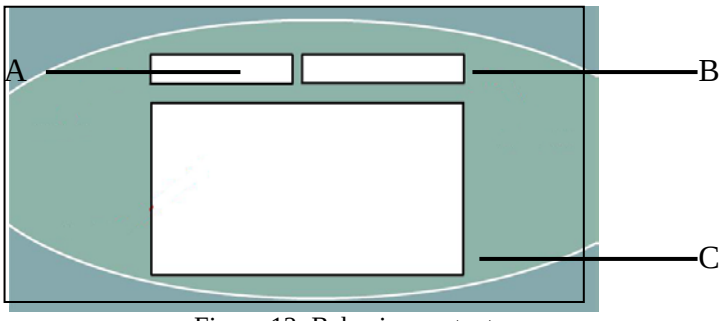

Figure 12: Behavior content
A. Return list **B.** Parameter List **C.** User code

The variables scope is the whole user database; this is means that the user can define a variable on any cell in the database. This is insures that AI-Builder can be used for any application

that depends on the database as major part of design. A variable definition in most development IDEs is represented by declaring the variable name and type and maybe an initial value, in AI-Builder you can define the variable in the same way by giving a name and selecting the table and column of the cell that represent the variable and the cell value will be the variable initial value.

If you look closely u can see that the table type and the column type are bound to gather [(1)] so they represent the variable type, by other words the cell type as an answer of the following question : what does the user mean by this cell?. This feature enables AI-Builder to have ready-for-use behaviors that are generally used on deferent projects that have the same category [(2)],and this insures the reusability of behaviors (built-in or user-defined)in deferent project that are enhanced using AI-Builder.

The binding of behaviors in an AI algorithm (called Execution in AI-Builder) depends on the algorithm type , behaviors are sometimes bounded directly as nodes in a tree[(3)] or bounded to a conditional node [(3)] that are used to enable the user to control the run path of the tree algorithm.

Execution Establishment

When the user have an AI idea that he think will make his\her application better ,as for example he won't his\her application to make some shortcuts to have some the arrangement in the his\her client interface, create a new execution and start to design the AI algorithm so it can do what he want.

When the user have an AI idea that he think will make his\her application better, as for example he won't his\her application to make some shortcuts to have some the arrangement in the his\her client interface, create a new execution and start to design the AI algorithm so it can do what he want, after the user has finished building his\her algorithm he can use it by using its execution file.

for every algorithm there is a .java file that holds the behaviors code and have the execute method that is used to execute the algorithm , that file is call an execution file and is used when the user has finished his\her algorithm design and has established and compiled his\her code.

The types of algorithms that can be built in AI-Builder are structured as a tree of behaviors, by other words AI-Builder gives the ability to write code sections and distribute them on the tree nodes from the root node down to all leaves and all of these nodes will be treated as behaviors (as mentioned before a behavior is a node the contain code to be executed) so he can control what code to be executed, what to pass over. The control decisions can be made at run time as I will explain shortly.

The types of algorithms that can be built in AI-Builder are structured as a tree of behaviors, by other words AI-Builder gives the ability to write code sections and distribute them on the tree nodes from the root node down to all leaves and all of these nodes will be treated as behaviors (as mentioned before a behavior is a node the contain code to be executed) so he can control what code to be executed, what to pass over. The control decisions can be made at run time as I will explain shortly.

There are three types of these tree algorithms:

1. Conditional tree algorithm.
2. Recursion tree algorithm.
3. Single path list algorithm.

Conditional tree algorithm:

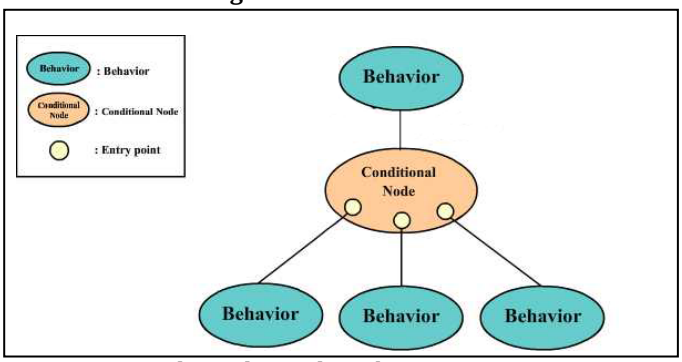

Figure13: Conditional tree algorithm.

This type of tree consists of behaviors and conditional nodes, the behaviors are used to write code in it and the conditional nodes are for control so at run time a single path in the tree is executed.

A conditional node main purpose is to determine the path for the execution, to determine the next node that will be executed. To make that possible a conditional node consists of enter points one of every behavior that is child to the conditional node, each enter point is bounded to a method that return a Boolean and it's the user job to code that method ,these methods are given to the user in a class (that comes with the execution class) that have empty methods that return true by default and every one of these methods must be coded in order to make the control possible[(1)], notice that AI-Builder sets the decisions coding in the user's application scope so the user control can be easily linked with his\her application logic. The enter points are checked from left to right at runtime and the first one that comes with a true result will be consider the valid enter point and its behavior will be the next one to be executed so after that its goon to another conditional node in a lower level of the tree, and its repeated until the last behavior have no conditional node linked to it as a child.

The reader can think of this as that each enters point has a priority, the first one has the highest priority and the last one has the lowest priority. There is a little trick that the user might consider, the user can add an enter point and make its method always return false so it is never entered so that the method can have some code to be executed before the decision process of the conditional node begins, this is can be extremely useful especially if the decision of the conditional node depends on some calculations the is needed to be done first.

Recursion Tree Algorithm:

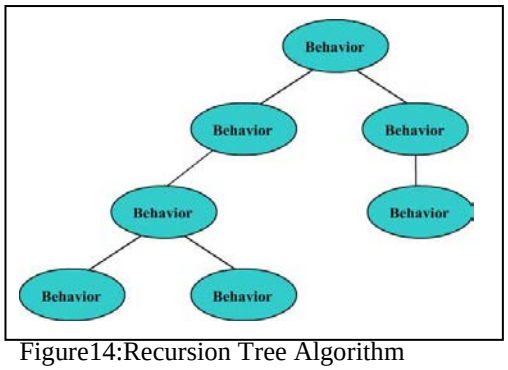

Figure14:Recursion Tree Algorithm

This type of tree consists only of behaviors, the behaviors are used to write code in it and the all of the tree nodes (behaviors) will be call as the tree is a recursion tree.
Generally in recursive tree the root node will call the left child and the right child, in AL-Builder the recursion tree nodes are behaviors and the behaviors call each other recursively. Notice that in this type of tree all behaviors are executed, so the user job is to code his\her defined behaviors and the more important to control the order of execution of these behaviors.
Recursion tree algorithms are generally building a single problem solution; the algorithm handles one main idea and they can be used when ever needed.
There is a little trick that might be very useful, the user can use his\her recursion algorithm (call it's execute method in its execution java file) inside another algorithm, maybe conditional node algorithm, the user can create an empty behavior and call the execute method of that algorithm inside the empty behavior. By this the user can create general algorithms and use it whenever he wants to, this insures that high reusability of the user work and is done by a single line of code.

Behaviors' List Algorithm:

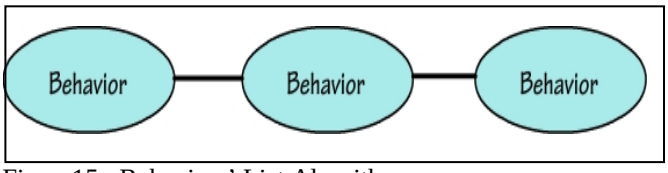

Figure15: Behaviors' List Algorithm

This is the simplest type of algorithm that the user can build; it's simply a list of behaviors that will execute in a linear order.
Generally the behaviors list algorithm is used when the user want a simple group of actions to be done when he call the execute method in his\her application, there is no conditions no recursive calls (handled by execution java file not by user; the user just call it).

Main Output:
The main output is a package of classes that represent the executions and there support classes, the user can use these executions by simply import the output package and call the execute method of a specific execution any where needed in his\her application.
The user can use any conditions in his\her application logic or any build in listeners in java libraries, this insure the usably and high compatibility of AI-Builder.

5. Conclusion

The intelligent user interaction development (IUID) engine makes the development of the artificial intelligence code and the addition to the product is possible and risk free. It provides the ability to use an artificial intelligence to improve the structure of the relational database. It consists of two phases, database phase that includes: loader, reader and adder and artificial intelligence algorithms phase that includes behavior establishment, execution establishment and the artificial intelligence builder.
The output of the first phase is a well-structured identified database uses as input to the second phase. The output of the artificial intelligence algorithms phase is a tree structure that represented as java classes grouped together in a java package. It gives the user the power to develop some clever actions to use in his/her applications without any need to worry about the code structure or object oriented constrains in an easy way.

References

[1]. Bertino, B. Catania, G.P. Zarri, "Intelligent database systems", Reading, Addsion Wesley Professional, 2001.
[2]. Kamran Parsaye, Mark Chignell, Setrag Khoshafian and Harry Wong, "Intelligent databases-object-oriented,deductive hypermedia technologies", New York, JohnWiley& Sons, 1989.
[3]. Androutsopoulos, G.D. Ritchie, and P. Thanisch, Natural Language Interfaces to Databases An Introduction, Journal of Natural Language Engineering
1 Part 1 (1995), 29–81
[4]. Charniak E. 1993, "Statistical Language Learning", MITPress.
[5]. Church K., Mercer R. 1993, "Introduction to the special issue on computational linguistics using large corpora",Computational Linguistics,19 (1), pp. 1-24.
[6]. Miikkulainen R. 1993, "Subsymbolic Natural Language Processing: An Integrated Model of Scripts, Lexicon, and Memory", MIT Press, Cambridge, MA.
[7]. Marcus M., Santorini B., Marcinkiewicz M. 1993,"Building a large annotated corpus of English: The Penn Treebank", Computational Linguistics, 19 (2), pp. 313-330.
[8]. McCarthy J, Lehnert W ,1995, "Using decision trees for coreference resolution", Proceedings of the Fourteenth
International Joint Conference on Artificial Intelligence,
pp. 1050-1055.
[9]. Riloff E. 1993, "Automatically constructing a dictionary for information extraction tasks", Proceedings of the Eleventh
National Conference on Artificial Intelligence,pp. 811-816.
[10].Riloff E.1996, "Automatically generating extraction patterns from untagged text", Proc eedings of the
Thirteenth National Conference on Artificial Intelligence,
pp. 1044-1049.
[11].Majumder P., Mitra M., Chaudhari B.,2002, "N-gram: A

Language Independent Approach to IR and Natural Language Processing", Lecture Notes.
[12].Miikkulainen R., 1997,"Natural language processing with subsymbolic neural networks", Neural Network Perspectives on Cognition and Adaptive Robotics.
[13].Reilly R., Sharkey N. (Eds.),1992 "Connectionist Approaches to Natural Language Processing", Lawrence Erlbaum and Associates, Hilldale, NJ.
[14].Shashtri L.,1997, "A model of rapid memory formation in the hippocampal system", Proceeding of Meeting of cognitive Science Society, Stanford.
[15].Abrahams P. W. et al. "The LISP 2 Programming Language and System", in proceedings of FJCC, No. 29, USA, 1966, pp. 661– 676.
[16].J. McCarthy, "LISP Programmers Manual, Handwritten Draft" MIT AI Lab., Vambridge, USA, 1959.
[17].T. Warren, "A Step toward Man-Computer Symbiosis", Ph.D. Thesis, Massachusetts Institut of Technologie, Project on Mathematics and Computation (MAC), Technical Report MAC-TR-32, Cambridge, MA, USA, 1966.
[18].Rohit J. Kate and Raymond J. Mooney, Using String-Kernels for Learning Semantic Parsers, COLINGACL (2006).
[19].Woods, W. (1973). An experimental parsing system for transition network grammars. In Natural language Processing, R. Rustin, Ed.,Algorithmic Press, New York.
[20].Woods, W., Kaplan, R. and Webber, B. (1972). The Lunar Sciences Natural Language Information System. Bolt Beranek and Newman Inc., Cambridge, Massachusetts Final Report. B. B. N. Report No 2378.
[21].Hendrix, G. (1977). The LIFER manual A guide to building practical natural language interfaces. SRI Artificial Intelligence Center, Menlo Park, Calif. Tech. Note 138.
[22].Hendrix, G., Sacrdoti, E., Sagalowicz, D. and Slocum, J. (1978). Developing a natural language interface to complex data. ACM Transactions on Database Systems, Volume 3, No. 2, USA, Pages 105 – 147
[23].D.L. Waltz., "An English Language Question Answering System for a Large Relational Database
[24].Yunyao Li, Huahai Yang, and H.V. Jagadish, Constructing a Generic Natural Language Interface for an XML Database, EDBT (2006).
[25].Ana-Maria Popescu, Alex Armanasu, Oren Etzioni, David Ko, and Alexander Yates, Modern Natural Language Interfaces to Databases:Composing Statistical Parsing with Semantic Tractability, COLING (2004).
[26].Yuk Wah Wong, Learning for Semantic Parsing Using Statistical Machine TranslationTechniques, Technical Report UT-AI-05-323, University of Texas at Austin, Artificial Intelligence Lab, October 2005.