

A COMPUTER IMPLEMENTED SYSTEM AND METHOD FOR CONFIGURING LOGIC TREE BASED ON JSON

TECHNICAL FIELD

[0001] The present disclosure relates generally to a system and method for configuring logic tree based on Java Script Object Notation (JSON). Embodiments of the disclosure are related to a computer implemented system and method used in financial application to configure or amend logic tree based on JSON.

BACKGROUND

[0002] Logic trees are in common use in a wide range of industries, particularly in financial application, risk assessment or loan applications. A set of rule and logic are combined to represent an essential and reasonable business logic. For example, when a bank is determining whether to lend the money to a borrower, it will check a set of conditions to decide the credit of that borrower.

[0003] JSON stands for JavaScript Object Notion (JSON). It is made for quick lightweight data exchange from the server to the client. JSON is commonly used for web applications development but not restricted to it. It is human-readable and easy-to-learn.

[0004] Patent publication number CN107704265A, filed by University Of Electronic Science And Technology, discloses a business flow configurability-oriented rule generation method. The method is based on an intelligent audit system. The system comprises a system management platform and a rule analysis framework. The method comprises a system management step and a rule analysis step. The rule analysis step comprises the sub-steps that a GRL rule description module transmits rule information between a front-end interface and a rule engine, wherein the rule information comprises GRL rule description. The invention provides a business flow-oriented intelligent audit method capable of shielding related technical problems of the rule engine, program design and the like, so that a user of a project and a similar software system can independently finish business rule definition and maintenance through a

customized UI view, the convenience and efficiency of business change are improved, and the business secrecy is enhanced.

[0005] Patent publication number CN109447485A, filed by Beijing Baifendian Information Science & Tech CO LTD, discloses a kind of rule-based Real-time Decision System and method, system mainly includes Kafka platform, kudu database, hive database, initial data input module, data parsing filtering module, decision tree configuration module, Zookeeper, decision tree resolver, decision forest module, connection pool. It through the invention can be with real-time collecting, the user behaviors log data of monitoring user, according to marketing scene demand, building automation marketing scene rule tree, connect the behavior of user online and offline, then the label of real-time judge user, feature, behavior reach the accurate touching of user by the different channel realizations such as short message, mail, APP PUSH message, the interactive communication with user, achieve the effect that Branding image or promote conversion ratio, realizes the marketing behavior of precision marketing.

[0006] In the current marketplace, solutions to a configurable logic tree execution engine are mostly heavy and exclusive, such as developing a standalone language. Such solutions are expensive to obtain, with steep learning curve and hard to maintain and extend.

[0007] A need, therefore, exists for an improved easily configurable, lightweight logic tree using JSON.

SUMMARY

[0008] The following summary is provided to facilitate an understanding of some of the innovative features unique to the disclosed embodiment and is not intended to be a full description. A full appreciation of the various aspects of the embodiments disclosed herein can be gained by taking into consideration the entire specification, claims, drawings, and abstract as a whole.

[0009] It is, therefore, one aim of the disclosed embodiments to provide for a

computer implemented system for configuring a business logic tree in Java Script Object Notion (JSON) format based on at least one business requirement. The system has at least one computing device with at least one memory for storing at least one logic tree and at least one configuration program; a Graphical User Interface (GUI) for allowing at least one user to configure at least one logic tree based on the configuration program; at least one server comprises at least one storage unit to store at least one execution program and the logic tree configured by the GUI in JSON format. The computing device is connected to the server through a network. The logic tree comprises one or more equations with boolean operators and conditions, based on at least one business requirement. The execution engine retrieves the logic tree from the storage unit based on the execution program. Further the execution engine executes the logic tree to produce a result.

[0010] It is, therefore, one aim of the disclosed embodiments to provide for a computer implemented method for configuring a business logic tree in Java Script Object Notion (JSON) format based on at least one business requirement. The method comprises configuring at least one computing device to: store at least one logic tree and at least one configuration program in at least one memory; allow at least one user to configure at least one logic tree based on the configuration program using at least one Graphical User Interface (GUI). The logic tree comprises one or more equations with boolean operators and conditions, based on at least one business requirement. At least one storage unit of at least one server is used to store at least one execution program and the logic tree configured by the GUI in JSON format. At least one execution engine in server is used to execute the logic tree based on the execution program. The computing device is connected to the server through a network.

[0011] It is, therefore, one aim of the disclosed embodiments to provide for a computer implemented system and method for configuring a business logic tree in Java Script Object Notion (JSON) format based on at least one business requirement. The execution engine is configured to retrieve the logic tree from the storage unit based on the execution program. Further the execution engine is configured to execute the logic tree to produce a result.

[0012] It is, therefore, one aim of the disclosed embodiments to provide for a

computer implemented system and method for configuring a business logic tree in Java Script Object Notion (JSON) format based on at least one business requirement. Storing the logic tree in JSON offers a way to save and express a complicated logic tree with minimum effort for a user to understand. In particular, individuals who wish to configure the logic tree are free of the need to learn a new language and associated principles. In essence, the invention is capable of expressing highly complicated business logic while allowing configuration with ease and accessibility.

[0013] It is, therefore, one aim of the disclosed embodiments to provide for a computer implemented system and method for configuring a business logic tree in Java Script Object Notion (JSON) format based on at least one business requirement. The present invention is focused on having a financial application (i.e. risk assessment or loan applications) but could be applied and adapted for any industry that can utilize a configurable lightweight decision tree for aiding in producing a result.

[0014] It is, therefore, one aim of the disclosed embodiments to provide for a computer implemented system and method for configuring logic tree based on JSON in which the user can configure logic tree based on the business requirement using GUI. The configured logic tree is a logic tree structure comprising one or more equations that are executable with a parent/child hierarchy, with multiple relations as rule conditions. The logic tree is saved in JSON format and allows the execution engine to retrieve the logic tree and produce the result efficiently and reliably.

[0015] It is, therefore, one aim of the disclosed embodiments to provide for a computer implemented system and method for configuring logic tree based on JSON in which logic tree is self-innovated with extreme light weight and not require to import special languages, as rules are stored in JSON format which is understandable and structural.

[0016] It is, therefore, one aim of the disclosed embodiments to provide for computer implemented system and method for configuring logic tree based on JSON, wherein, the logic tree comprises at least one logic tree component comprising: at least one rule tree, at least one type of rule tree, at least one data type of a variable, at least one rule, at least one equation, at least one relation between the equations, at least one

operator between a left hand side and a right hand side of the equation, at least one attribute of an equation and at least one attribute type of the equation.

[0017] It is, therefore, one aim of the disclosed embodiments to configure the logic tree using the logic tree component comprises: creating one or more equations using at least one operator and one or more variables; creating one or more relations of one or more equations and/or one or more variables; creating one or more subsequent level relations using one or more relations and one or more equations; creating one or more next level of relations using one or more relations and/or subsequent level relations; creating one or more rules using one or more equations and/or one or more variables and/or one or more relations and/or one or more subsequent level relations and/or one or more next level relations; and creating the logic tree using one or more rules.

[0018] It is, therefore, one aim of the disclosed embodiments to provide logic tree processing system comprising a visual representation means for visually displaying, configuring and/or editing elements and structure of a logic tree, a coding means for generating and/or parsing said logic tree for configuration in an open standard, language-independent data format, and a logic tree execution engine for extracting said logic tree in said data format, and executing said logic tree to obtain results. The coding means is the configuration program used to configure the logic tree.

[0019] It is, therefore, one aim of the disclosed embodiments to provide logic tree processing system, wherein the visual representation means comprises a graphical user interface (GUI) provided on a computing device disposed to enabling a user to build the logic tree with said elements and structure.

[0020] It is, therefore, one aim of the disclosed embodiments to provide logic tree processing system, wherein the coding means configures the logic tree in Javascript Object Notation (JSON) format.

[0021] It is, therefore, one aim of the disclosed embodiments to provide logic tree processing system, wherein the logic tree execution engine extracts said logic tree in JSON and executes said logic tree accordingly.

[0022] It is, therefore, one aim of the disclosed embodiments to provide logic tree processing system, wherein the GUI is provided on a front-end webpage disposed to enabling the user to display, configure and edit rules in building the logic tree.

[0023] It is, therefore, one aim of the disclosed embodiments to provide logic tree processing system, wherein the coding means is accessible at client-end by the user and is disposed to enable said user to display, configure, edit and storing the tree logic dynamically.

[0024] It is, therefore, one aim of the disclosed embodiments to provide logic tree processing system, wherein the coding means further includes any one or in combination of syntax checker, algorithm, interpreter, translator and compiler as necessary for configuring the logic tree into the designated format.

[0025] It is, therefore, one aim of the disclosed embodiments to provide logic tree processing system, wherein the coding means is operable at server end in generating and parsing logic tree structure.

[0026] It is, therefore, one aim of the disclosed embodiments to provide a logic tree execution engine embodying a logic tree execution process comprising the steps of: extracting the logic tree from its stored format, and executing said logic tree to obtain results, wherein the basic execution unit comprises at least an equation relating to a next or subsequent equation by a second or subsequent operator; or alternatively, said equation is related to a next level equation by a second or subsequent operator in a parent/child hierarchy, enabling a plurality of such equations to form a tree-like rule set.

[0027] It is, therefore, one aim of the disclosed embodiments to provide a logic tree execution engine, wherein the logic tree is extracted in JSON format.

[0028] It is, therefore, one aim of the disclosed embodiments to provide a logic tree execution engine, wherein the coding means is operable at server end.

[0029] It is, therefore, one aim of the disclosed embodiments to provide a logic tree

execution engine embodied as coding means in a logic tree processing.

[0030] It is, therefore, one aim of the disclosed embodiments to provide An application (app) including a computer program embodying a GUI installed on a computing device enabling a user to display, configure and edit rules in building a logic tree to be processed in a logic tree processing system.

[0031] It is, therefore, one aim of the disclosed embodiments to provide a logic tree execution engine configured for assessing business risk.

[0032] It is, therefore, one aim of the disclosed embodiments to provide a logic tree processing system configured for assessing business risk.

BRIEF DESCRIPTION OF THE DRAWINGS

[0033] The summary above, as well as the following detailed description of illustrative embodiments, is better understood when read in conjunction with the appended drawings. For the purpose of illustrating the present disclosure, exemplary constructions of the disclosure are shown in the drawings. However, the disclosure is not limited to specific methods and instrumentalities disclosed herein. Moreover, those in the art will understand that the drawings are not to scale. Wherever possible, like elements have been indicated by identical numbers.

[0034] **FIG. 1A** is an illustration of a block diagram of a computer implemented system **100** for configuring a business logic tree based on at least one business requirement in accordance with the general embodiment of the invention;

[0035] **FIG. 1B** is a block diagram illustrating a computer implemented system **150** for configuring a business logic tree in Java Script Object Notion (JSON) format based on at least one business requirement in accordance with disclosed embodiments;

[0036] **FIG. 2** is a schematic diagram showing a configuration process of a logic tree **200** in accordance with the disclosed embodiments;

[0037] **FIG. 3** is a schematic diagram of GUI **300** showing rule details **306** in

configuration of each component of the logic tree **200** in accordance with the disclosed embodiments;

[0038] **FIG. 4** is an illustration of flow diagram showing a method **400** for configuring a logic tree, in accordance with the disclosed embodiments;

[0039] **FIG. 5** is a schematic diagram of GUI **500** showing rule details **510** in configuration of each component of one another logic tree in accordance with one another disclosed embodiments;

[0040] **FIG. 6** is an illustration of a schematic diagram showing result tree **600** of each rule details **510** of **FIG. 5** in accordance with the disclosed embodiments;

[0041] **FIG. 7A** is a flow diagram of a computer implemented method **650** for configuring a business logic tree based on at least one business requirement, using system **100** of **FIG. 1A** in accordance with the disclosed embodiments;

[0042] **FIG. 7B** is an illustration of a flow diagram of a computer implemented method **680** for executing logic tree using execution of system **100** of **FIG. 1A**, in accordance with the alternate embodiments; and

[0043] **FIG. 7C** is an illustration of a flow diagram of a computer implemented method **700** for configuring a business logic tree in Java Script Object Notion (JSON) format based on at least one business requirement, using system **150** of **FIG. 1B**, in accordance with the disclosed embodiments.

DETAILED DESCRIPTION OF ILLUSTRATIVE EMBODIMENTS

[0044] The particular configurations discussed in the following description are non-limiting examples that can be varied and are cited merely to illustrate at least one embodiment and are not intended to limit the scope thereof.

[0045] **FIG. 1A** is an illustration of a general block diagram of a computer

implemented system **100** for configuring a business logic tree based on at least one business requirement, in accordance with the disclosed embodiments. A logic tree processing system **100** has a visual representation means **105**, a coding means **107** and a logic tree execution engine **106**. The visual representation means **105** is used for visually displaying, configuring and/or editing elements and structure of a logic tree. The coding means **107** can be utilized for generating and/or parsing the logic tree for configuration in an open standard, language-independent data format **109**. The logic tree execution engine **106** is used for extracting the logic tree in the data format **109** and executing the logic tree to obtain results.

[0046] In one embodiment of the invention, the visual representation means **105** comprises a graphical user interface (GUI) provided on a computing device **101** disposed to enabling a user to build the logic tree. The coding means **107** configures the logic tree in Java script Object Notation (JSON) format. The logic tree execution engine **106** extracts the logic tree in JSON and executes the logic tree accordingly.

[0047] In another embodiment of the invention, the GUI is provided on a front-end webpage disposed to enabling the user to display, configure and edit rules in building the logic tree. In one embodiment of the invention, the coding means **107** can be accessible at client-end by the user and is disposed to enable the user to display, configure, edit and storing the tree logic dynamically. The coding means **107** further includes any one or in combination of syntax checker, algorithm, interpreter, translator and compiler as necessary for configuring the logic tree into the designated format. In another embodiment of the invention the coding means **107** is operable at server end in generating and parsing logic tree structure.

[0048] **FIG. 1B** shows a computer implemented system **150** used for configuring a business logic tree in Java Script Object Notion (JSON) format based on at least one business requirement. The system **150** comprises at least one computing device **101** and at least one server **110** connectable to the storage unit **112**, for example a database, through a network. The computing device **101** can be used to configure at least one business logic tree based on at least one business requirement in, for example, finance industry. The computing device **101** comprises at least one memory **102** and Graphical User Interface (GUI) **104**. The memory **102** is configured to store

at least one configuration program **103**. The GUI **104** allows at least one user to configure/amend at least one logic tree based on the configuration program **103**. The logic tree comprises one or more equations with Boolean operators and conditions, based on the business requirement. Using the GUI **104**, the user can drag and drop the various Boolean operators and logical conditions to form one or more equations. The configuration program **103** has instruction set that are used by user to configure at least one logic tree based the business requirement. The logic tree **114** in JSON format configured by the user using GUI **104** is stored in at least one storage unit **112**. The server **110** has at least one execution engine **106**. The execution engine **106** is used to execute the logic tree based on the execution program **107**. Due to the popularity and lightweighness of JSON format, the logic tree can be easily implemented into various scenarios. The business requirement can be approval of financial application, risk assessment or loan applications.

[0049] It should be noted that the computing device **101** is a client device used for configuration or amendment of logic tree by finance industry/company/bank professionals based on their business requirements. The configuration program comprises a set of instruction, along with GUI, render configuration information to user, so that the user can select the required conditions for their business requirement to create a logic tree. The execution program comprises a set of instruction used to execute the logic tree and produce at least one result.

[0050] It should be noted that the GUI **104** is the frontend webpage used by user to configure rules and send rules in JSON format to storage unit **112**. The execution engine **106** retrieves the logic tree in JSON format from the storage unit **112** for further rule execution. The GUI **104** is used for rule display and rule editing purpose.

[0051] **FIG. 2** is an illustration of a schematic diagram showing configuration process of logic tree graph **200**, in accordance with the disclosed embodiments. The present invention uses JSON to store the logic tree **114** to achieve a maximum capability of extension and a lightweight solution. The minimum executable unit of present invention is equation. An equation **204** is consisted of a Left Hand Side (LHS) expression **203** and a Right Hand Side (RHS) expression **201** and an operator **202**. Then, the next level concept is defined as relation **205**, **206** or **207**, which include

equations **204** and their relation. Similarly, the relation **210** is formed using equation **208** and relation set **205**, **206** and **207**. At this level the concept of parent or child hierarchy is introduced, which allows the user to configure a tree-like rule set **211**, **212** or **213**, by having an embedding structure in relation. After this, multiple relations **210** or **209** are included as a rule's condition and use the condition to determine final result. The operator can be for example, lesser than (<), greater than (>), equal (=), and so on, and the relation can be AND or OR. As per the **FIG. 2**, the basic idea is to configure the logic tree or tree rule **214** from top to down, from simple to complex structure. Benefited from this tree structure, this design is capable of expressing highly complicated logic tree while keeping it be amended with ease.

[0052] **FIG. 3** is an illustration of a schematic diagram of GUI **300** showing rule details **306** in configuration of each component of logic/rule tree **114**. As shown in the rule details, the three equations namely, Equation A **301**, Equation B **302** and Equation C **303**, stating various business requirement are selected using the GUI **300**. The relation between equation A **301** and equation B **302** is "OR". While equation C **303** is child of equation B **302**. So, the overall condition of "when" is "A || (B && C)". If result of "A || (B && C)" is true, as at block **304**, the result of derived variable will be assigned as "true" based on "then" condition, otherwise, as at block **305**, the result will be assigned as "false" based on "elseVal" condition.

[0053] LHS of equation A **301** uses function "association_count". Function parameters are variable "Company Name" and variable "Monthly Income". Function arguments are "num" whose value is 1 and "time_interval" whose value is "month". RHS of equation A **301** is constant 2 and operator of equation A is "greater than or equal". The meaning of equation A is that, the number of different "Monthly Income" on same "Company Name" within 1 month is greater than or equal to 2.

[0054] Similarly, referring to the equation B **302** is that, "Loan Amount" greater than 10000. And meaning of equation C **303** is that, "Other Loans Monthly Repayment" greater than 1000.

[0055] A structural sample of the JSON format of the rule tree comprises various components, also referred as, at least one logic tree component or elements, for example, at least one rule tree, at least one rule tree type, at least one variable data type, at least one rule, at least one relation, at least one equation, at least one relation between the equations, at operation (operators) between LHS and RHS.

[0056] The “rule tree refers to a rule tree configured based on the business requirement using the present invention. The rule tree type refers to a type of the rule tree. The variable data type refers to a type of the variable used in configuration of rule tree and is used to define the data type of variable. The “rule refers to rule used in the configuration of logic tree and consists of concepts of “when”, “then” and “elseVal” objects. The relation consists of equations, a relation between equations and a list of relations as a children.

[0057] The rule tree consists of at least one rule tree type, at least one variable data type as return type, and a list of rules. The rule tree type defines the type of the rule tree. The rule tree type is an enumeration, including at least one derived variable, at least one score card, at least one rule, at least one risk rule set, and at least one condition.

[0058] The rule tree returns a result and the result is assigned as value of a variable. The result could be a number, date, Boolean, or string. The rule tree must have when, then, “elseVal” component. As this variable is derived, hence called derived variable. The rule tree will match a score (number) based on result of one or more variables and return a sum as the result of a scorecard. The score card type of rule tree only contains when and then components. The rule tree will re-determine the result of a variable based on “when”, “then”, “elseVal” conditions. The result of rule tree is a number or Boolean only. The rule tree must have “when”, “then”, “elseVal” component. The risk rule set is similar like the score card, but return a map of risk levels, for example, “low”, “medium”, “high”, as result of the risk rule set. The rule tree will return a Boolean, as a result of a condition.

[0059] A rule consists of concepts of when, then, “elseVal objects”. Object “when” is a list of relation. The condition relation between relations are “AND”. The objects “then” and “elseVal” are attribute which will be introduced. If there is more than one rule in logic tree, only “when” and “then” object are exist in each rule.

[0060] The variable data type defines the data type of the variable. The variable data type is an enum, including not limited to a number, a date, a Boolean and a string.

[0061]

[0062] The relation consists of an equation, relation between the equations, and a list of relations as children. The relation between equations is an enum, including “AND”, “OR” operators. The equation consists of attributes LHS, RHS, as well as at least one operation between LHS and LHS.

[0063] The operation or operator between LHS and RHS is an enum, including not limited to “eq” (equal), “ne” (not equal), “gt” (greater than), “gte” (greater than or equal), “lt” (less than), and “lte” (less than or equal).

[0064] Attribute define attribute of an equation. The attribute consists of type of an attribute type, a list of attribute as parameters, a map of type and string as arguments.

[0065] The type of an attribute is an enum, including not limited to a constant, a variable, a function, a score card, an expression, and a risk rule set. A constant, can be a number or string. Such number or string will be used as value of this attribute. A variable can be a predefined variable or derived variable. Value will be the identity of such variable. In execution, real value of variable with such ID will be used as value of this attribute. A function can be a predefined function. Value will be the function code. There will be validation of params and arguments based on different function code. In execution, result of the function will be used as value of this attribute. The scorecard represents an user configured scorecard. Value will be the ID of such scorecard. In execution, real value of scorecard with such ID will be used as value of this attribute. Expression represents an user configured expression, for example, V1+V2, where V1

and V2 stands for two variables. Value of attribute will be “V1+V2”. The “variableNameMap” will be provided to indicate which variable is V1 and which variable is V2. In execution, result of this expression which is calculated by values of variables will be used as value of this attribute. The “risk rule set represents user configured risk rule set. The value will be the ID of such risk rule set. In execution, real value of risk rule set with such ID will be used as value of this attribute.

[0066] FIG. 4 is an illustration of flow diagram showing a method **400** for configuring a logic tree, in accordance with the disclosed embodiments. The logic tree for example the logic tree graph **200** depicted in **FIG. 2** can be configured by user using the method **400**. The logic tree is configured using as at least one logic tree component or elements, for example, at least one rule tree, at least one rule tree type, at least one variable data type, at least one rule, at least one relation, at least one equation, at least one relation between the equations, at operation between LHS and RHS expressions.

[0067] As at step **402**, one or more equations are created using at least one operator and one or more variables. Then as at step **404**, one or more relations using one or more equations and/or one or more variables. As at step **406**, one or more subsequent level relations using one or more relations and one or more equations. As at step **408**, one or more next level of relations are created using one or more relations and/or subsequent level relations. Then as at step **410**, one or more rules are created using one or more equations and/or one or more variables and/or one or more relations and/or one or more subsequent level relations and/or one or more next level relations. Finally, a logic tree using one or more rules, as at step **412**.

[0068] FIG. 5 is an illustration of a schematic diagram of GUI **500** showing rule details **510** in configuration of each component of one another logic tree. User defined logic tree comprises a derived variable whose data type is Boolean. “when” condition includes seven equations, as shown as Equation A **501**, Equation B **502**, Equation C **503**, Equation D **504**, Equation E **505**, Equation F **506** and Equation G **507**. The relation between equation B **502** and equation C **503** is “Or”. Equation D **504** is child of equation C **503**. Equation B **502** or Equation C **503** is child of equation A **501**. Similarly, the relation between equation F **506** and equation G **507** is “Or”. Equation F

506 or Equation G **507** is child of equation E **505**.

[0069] LHS of Equation A **501** uses function “association_count”. Function parameters are variable “Company Name” and variable “Monthly Income”. Function arguments are “num” whose value is 1 and “time_interval” whose value is “month”. RHS of equation A **501** is constant 2 and operator of equation A **501** is “greater than or equal”. The meaning of equation A **501** is that the number of different “Monthly Income” on same “Company Name” within 1 month is greater than or equal to 2.

[0070] Similarly, the Equation B **502** is “Basic Scode” greater than 100. And Equation C **503** is “Loan Term (months)” greater than constant 6. The Equation D **504** is “monthly_repay_ratio” greater than constant 10. The Equation E **505** is “Loan Amount” greater than constant 10000. The Equation F **506** is “Other Loans Monthly Repayment” greater than constant 1000. The Equation G **507** is “Working Experience (years)” less than constant 5.

[0071] As at blocks **508** and **509**, overall condition of “when” is “(((D && C) || B) && A) || ((G || F) && E)”. If result of “(((D && C) || B) && A) || ((G || F) && E)” is true, the result of derived variable will be assigned as “true” based on “then” condition, otherwise the result will be assigned as “false” based on “elseVal” condition.

[0072] FIG. 6 is an illustration of a schematic diagram showing result tree **600** of each rule details **510** of FIG. 5. Initially, as shown in result tree **600**, the Result 1 is calculated from Equation (D && C), then ((Result 1) || (Equation B)) is calculated to get the Result 2. Then, (Result 2 && (Equation A)) is calculated to get Result 3. Similarly, Equation (G || F) 1 is calculated to get Result 4. The ((Result 4) && (Equation E)) is calculated to get Result 5. Finally based on the ((Result 3) || (Result 5)), the final Result 6 is calculated.

[0073] To execute the logic tree, the executor will execute the “when” condition first. If the result of “when” condition equals to true, then it will execute “then” part and assign result of “then” to be the final result. Else execute “elseVal” part and assign result of “elseVal” to be the final result. If there are more than one rule in the logic tree, each rule will have a set of “when” / “then” conditions without any “elseVal” condition.

And the last “when” condition must be “1=1”. The executor will execute the “when” conditions by sequence, if result of any “when” condition equals to true, corresponding “then” condition will be executed and result will be assigned as final result of the logic tree. If none of “when” condition have result of true, the last “when” condition “1=1” for sure will be true and last “then” result will be assigned as final result. Each condition is a combination of lots of equations with “And” or “Or” relationship between each other. The executor will execute equations following sequence of from inner to outer.

[0074] FIG. 7A is a flow diagram of a computer implemented method **650** illustrating the configuring of a business logic tree based on at least one business requirement, using system **100** of **FIG. 1A**. As at step **652**, a basic execution unit, also referred to as an equation, is formed using one or more operators. Then, as at step **654**, a next or subsequent equation is related by a second or subsequent operator. Then, as at block **656**, the next level equation is related by a second or subsequent operator in a parent/child hierarchy. Finally, as at block **658**, a plurality of such equations are enabled to form a tree-like rule set. It should be noted that the business logic tree is configured using the logic tree components. The basic execution unit configured using the logic tree components is an equation.

[0075] FIG. 7B is an illustration of a flow diagram of a computer implemented method **680** for executing logic tree using execution of system **100** of **FIG. 1A**. The logic tree execution engine **106** embodying a logic tree execution process comprising the steps **682** ad **684** of extracting the logic tree from its stored format, and executing said logic tree to obtain results.

[0076] FIG. 7C is an illustration of a flow diagram of a computer implemented method **700** for configuring a business logic tree in Java Script Object Notion (JSON) format based on at least one business requirement. As at block **701**, the Graphical User Interface (GUI) **104** of the computing device **101** allow the user to configure at least one logic tree based on the configuration program. The memory **102** of the computing device **101** is used to store the configuration program. As at block **702**, after configuration, the logic tree in JSON format is saved in storage unit **112**. As at blocks **703**, **704** and **705**, at least one execution engine configured to retrieve the logic tree from memory **102** and execute the logic tree based on the execution program to get

the required result. Storing the logic tree in JSON offers a way to save and express a complicated logic tree with minimum effort for a user to understand. In particular, individuals who wish to configure the logic tree are free of the need to learn a new language and associated principles. The logic tree execution engine thus allows professionals without programming skill to reconfigure and amend a business logic dynamically and promptly, as well as execute the logic tree to get a result.

[0077] In another embodiment, the configuration of logic tree can be performed using a front-end webpage disposed to enabling the user to display, configure and edit rules in building the logic tree. In one another embodiment of the invention an application (app) including a computer program embodying a GUI installed on a computing device is used to enable a user to display, configure and edit rules in building a logic tree to be processed in a logic tree processing system **100** of **FIG. 1A**. It should be noted that the configuration of logic tree can be performed using execution engine and processing system for assessing business risk.

[0078] The present invention can use any programming language to build the configuration webpage using the configuration program. Finally, the configured logic tree should be converted to structure of logic tree in JSON format. The logic tree can be executed in execution engine. By using the present invention, complex rule can be easily configured in JSON format and can easily executed by execution engine.

[0079] While the present invention includes configuring the logic tree from top to down, from simple to complex, with the tree structure design being capable of expressing highly complicated logic tree while keeping it be amended with ease, and storing the logic tree in JSON offers a way to save and express a complicated logic tree with minimum effort for a user to understand. In particular, individuals who wish to configure the logic tree are free of the need to learn a new language and associated principles. More importantly, the present invention includes means for extracting the logic tree from its stored format for execution the logic tree in equation form whereby the equation may relate to the next or subsequent equation by one or more operators, or alternatively, to the next level equation in a parent/child hierarchy, which forms a logic tree-like rule set.

[0080] The logic tree of present invention is self-innovated with extreme light weight and not require to import special languages, as rules are stored in JSON format which is understandable and structural. Although embodiments of the current disclosure have been described comprehensively and in considerable detail to cover possible aspects, those skilled in the art would recognize that other forms of embodiments or hardware configurations to achieve our invention's implementations are also possible, for instance, with the complementary use of reconfigurable digital circuits such as, Field-Programmable Gate Array (FPGAs), Programmable Logic Devices (PLDs) and so on.

[0081] It will be appreciated that variations of the above disclosed and other features and functions, or alternatives thereof, may be desirably combined into many other different systems or applications. Also, various presently unforeseen or unanticipated alternatives, modifications, variations or improvements therein may be subsequently made by those skilled in the art which are also intended to be encompassed by the following claims.

CLAIMS

What is claimed is:

1. A computer implemented system for configuring a business logic tree in a Java Script Object Notion (JSON) format based on at least one business requirement, system comprising:
 - at least one computing device comprising,
 - at least one memory for storing at least one configuration program having at least one instruction set; and
 - a Graphical User Interface (GUI) allowing at least one user to configure the logic tree using the configuration program based on the business requirement,
 - wherein the logic tree comprises at least one logic tree component comprising: at least one rule tree, at least one type of rule tree, at least one data type of a variable, at least one rule, at least one equation, at least one relation between the equations, at least one operator between a left hand side and a right hand side of the equation, at least one attribute of an equation and at least one attribute type of the equation;
 - at least one storage unit to store the logic tree configured by the GUI in JSON format; and
 - at least one server comprising:
 - at least one execution engine comprising at least one execution program, wherein the execution program has a set of instruction to execute the logic tree based on the execution program.
2. The system of claim 1, wherein the GUI allows user to amend the logic tree.
3. The system of claim 1, wherein the execution engine retrieves the logic tree from the storage unit based on the execution program and executes the logic tree to produce a result.
4. A computer implemented method for configuring a business logic tree in Java

Script Object Notion (JSON) format based on at least one business requirement, method comprising:

configuring at least one computing device to:

store at least one configuration program in at least one memory of a computing device;

allow at least one user to configure at least one logic tree based on the configuration program using at least one Graphical User Interface (GUI) of the computing device, wherein the logic tree comprises at least one logic tree component comprising: at least one rule tree, at least one type of rule tree, at least one data type of a variable, at least one rule, at least one equation, at least one relation between the equations, at least one operator between a left hand side and a right hand side of the equation, at least one attribute of an equation and at least one attribute type of the equation, based on at least one business requirement;

configuring at least one storage unit to store the logic tree configured by the GUI in JSON format; and

configuring at least one execution engine of a server to retrieve logic tree from storage unit and to execute the logic tree based on the execution program to produce a result.

5. The method of claim 4, wherein configuring the logic tree using the logic tree component comprises:

creating one or more equations using at least one operator and one or more variables;

creating one or more relations using one or more equations and/or one or more variables;

creating one or more subsequent level relations using one or more relations and/or one or more equations;

creating one or more next level of relations using one or more relations and/or subsequent level relations;

creating one or more rules using one or more of one or more equations, one or more variables, one or more relations, one or more subsequent level relations or one or more next level relations; and

creating the logic tree using one or more rules.

6. A logic tree processing system comprising
 - a visual representation means for visually displaying, configuring and/or editing elements and structure of a logic tree,
 - a coding means for generating and/or parsing said logic tree for configuration in an open standard, language-independent data format, and
 - a logic tree execution engine for
 - extracting said logic tree in said data format, and
 - executing said logic tree to obtain results.
7. The logic tree processing system according to Claim 6 wherein the visual representation means comprises a graphical user interface (GUI) provided on a computing device disposed to enabling a user to build the logic tree with said elements and structure.
8. The logic tree processing system according to Claim 6 wherein the coding means configures the logic tree in Javascript Object Notation (JSON) format.
9. The logic tree processing system according to Claim 6 wherein the logic tree execution engine extracts said logic tree in JSON and executes said logic tree accordingly.
10. The logic tree processing system according to Claim 6 wherein the GUI is provided on a front-end webpage disposed to enabling the user to display, configure and edit rules in building the logic tree.
11. The logic tree processing system according to any one of Claims 6 to 10 wherein the coding means is accessible at client-end by the user and is disposed to enable said user to display, configure, edit and storing the tree logic dynamically.
12. The logic tree processing system according to any one of Claims 6 to 11

wherein the coding means further includes any one or in combination of syntax checker, algorithm, interpreter, translator and compiler as necessary for configuring the logic tree into the designated format.

13. The logic tree processing system according to Claim 6 to 122 wherein the coding means is operable at server end in generating and parsing logic tree structure.
14. A logic tree execution engine embodying a logic tree execution process comprising the steps of
 - extracting the logic tree from its stored format, and
 - executing said logic tree to obtain results, wherein
 - the basic execution unit comprises at least an equation relating to a next or subsequent equation by a second or subsequent operator;
 - or alternatively,
 - said equation is related to a next level equation by a second or subsequent operator in a parent/child hierarchy, enabling a plurality of such equations to form a tree-like rule set.
15. The logic tree execution engine according to Claim 14 wherein the logic tree is extracted in JSON format.
16. The logic tree execution engine according to Claim 14 embodied as coding means in a logic tree processing system according to any one of Claims 11 to 13.
17. The logic tree execution engine according to Claim 16 wherein the coding means is operable at server end.
18. An application including a computer program embodying a GUI installed on a computing device enabling a user to display, configure and edit rules in building a logic tree to be processed in a logic tree processing system according to any

one of Claims 7 to10.

19. A logic tree execution engine according to any one of Claims 14 to 17 configured for assessing business risk.
20. A logic tree processing system according to any one of Claims 6 to 13 configured for assessing business risk.