

[0009] Another embodiment of the present disclosure provides a computer-executable model reverse engineering method and apparatus, capable of expressing all information of a source code in a model without any loss of information and confirming all contents of the source code using the model only.

[0010] Yet another embodiment of the present disclosure provides a computer-executable model reverse engineering method and apparatus, capable of converting a source code into an expression of an abstracted format to abstractly expressing all information without being restricted to the programming language with which a source code is prepared.

[0011] Yet another embodiment of the present disclosure provides a computer-executable model reverse engineering method and apparatus, capable of generating a program that performs the same function with a variety of programming language using a model.

[0012]

[0013] According to embodiments of the present disclosure, there is provided a computer-executable model reverse engineering method, including (a) receiving a source code; (b) receiving a conversion rule or a separate conversion rule for source code information that cannot be expressed in a Unified Modeling Language (UML) to convert the source code into a goal model; (c) parsing the source code using a parser related to the source code to store parsing data having a specific material structure in a memory; (d) converting the stored parsing data into a meta model; (e) converting the meta model into the goal model according to the received conversion rule or separate conversion rule; and (g) excluding the source code that is not concerned so as not to be converted according to the conversion rule or separate conversion rule.

[0014] According to one embodiment, the method may further comprise (f) converting the converted goal model in a form adapted to a UML (Unified Modeling Language) standard again to be stored in a file.

[0015] According to one embodiment, the step (b) may comprise receiving the conversion rule configured of a source conversion rule, a model conversion rule, a naming rule and a relationship rule. Here, the step (e) may comprise designating and

converting a specific element of the source code or a specific element of the goal model according to the conversion rule.

[0016] According to one embodiment, the step (c) may comprise parsing the source code into data having different material structures depending on programming language types of the source code.

[0017] According to one embodiment, the step (d) may comprise converting the parsing data into the meta model showing all elements of the source code in the specific material structure regardless of the source code types and the programming language types.

[0018]

[0019] In another embodiment, the method may further comprise generating a message in at least one of a message lifeline, an exception lifeline, a validation lifeline or a code lifeline, with respect to the source code information that cannot be expressed in the UML.

[0020] According to one embodiment, the method may further comprise verifying whether all information of the source code is included in the goal model.

[0021] According to one embodiment, the step (e) may comprise (e-1) converting the meta model into a source model according to the received conversion rule, (e-2) receiving a model-to-model conversion rule to convert the source model into the goal model, and (e-3) converting the source model into the goal model according to the model-to-model conversion rule.

[0022] According to one embodiment, the method may further comprise (i) receiving the goal model to perform a configuration to generate the source code, (j) reading elements of the goal model to confirm whether the elements are used to generate the source code, (k) validating the model elements used to generate the source code according to a validation rule that the elements have, and (l) generating the source code according to a result of the validation.

[0023] According to one embodiment, the step (i) may comprise

[0024] registering elements of the goal model and conversion rules for respective elements and designating an execution condition of the conversion rule to perform the configuration.

[0025] According to embodiments of the present disclosure, there is provided a model reverse engineering apparatus, comprising a memory configured to store a model reverse engineering available program therein, and a processor configured to execute the program, wherein the processor parses a received source code into parsing data having a specific material structure using a parser, converts the parsed data into a meta model, converts the meta model into a goal model according to a received conversion rule or a received separate conversion rule for source code information that cannot be expressed in a Unified Modeling Language (UML), generates a source code using the converted goal model, and excludes the source code that is not concerned so as not to be converted according to the conversion rule or separate conversion rule.

[0026] According to one embodiment, the processor may receive the source code and the conversion rule or the separate conversion rule for source code information that cannot be expressed in the UML to convert the source code into the goal model, and stores parsing data having the parsed specific material structure in a memory. In another embodiment, the processor may parse the received source code into data having different types of material structures depending on programming language types of the source code.

[0027] According to one embodiment, the processor converts the parsing data into the meta model showing all elements of the source code in the specific material structure regardless of the source code types and the programming language types.

[0028] According to one embodiment, the processor may convert the source model into the goal model according to a model-to-model conversion rule which converts the source model received into the goal model. In another embodiment, the processor may receive a model-to-model conversion rule that is configured of a qualified name of the source model and a qualified name of the goal model.

[0029] According to one embodiment, the processor may receive the goal model to perform a configuration to generate the source code, read elements of the goal model to confirm whether the elements are source code generation target elements, verify the generation target model elements according to a validation rule that the elements have, and generate the source code according to a result of the validation verification. In another embodiment, the processor may register elements of the goal model and conversion rules

for respective elements and designates an execution condition of the conversion rule to perform the configuration.

[0030] According to embodiments of the present disclosure, there is provided a computer-readable storage medium in which a program for executing a model reverse engineering method in a computer is stored, the method comprising (a) receiving a source code, (b) receiving a conversion rule or a separate conversion rule for source code information that cannot be expressed in a Unified Modeling Language (UML) to convert the source code into a goal model, (c) parsing the source code using a parser related to the source code to store parsing data having a specific material structure in a memory, (d) converting the stored parsing data into a meta model, (e) converting the meta model into the goal model according to the received conversion rule or separate conversion rule, and (f) excluding the source code that is not concerned so as not to be converted according to the conversion rule or separate conversion rule.

[0031]

[0032] The disclosed technology may have following effects. However, since it does not mean that a specific embodiment should include all or only the effects, it should not be understood that the scope of the disclosed technology is limited thereto.

[0033] According to one embodiment of the present disclosure, a computer-executable server model reverse engineering method and apparatus is capable of converting an existing source code into a desired type of model to perform an analysis, a structural understanding and a redesigning of a program with ease.

[0034] According to another embodiment of the present disclosure, a computer-executable model reverse engineering method and apparatus is capable of expressing all information of a source code in a model without any loss of information and confirming all contents of the source code using the model only.

[0035] According to yet another embodiment of the present disclosure, a computer-executable model reverse engineering method and apparatus is capable of converting a source code into an expression of an abstracted format to abstractly expressing all information without being restricted to the programming language with which a source code is prepared.

[0036] According to yet another embodiment of the present disclosure, a computer-

converter 210, a model-to-model converter 220 and a code generator 230. The model reverse engineering program executor 112 is an apparatus that can generate and convert a model through a source code reverse engineering, capable of receiving a source code or a source model to generate and convert a model of a user's desired form, and designating flexible and diverse verification rules when preparing and applying conversion rules. Further, the model reverse engineering program executor 112 stores all information of a source code in a model to be converted into a programming language. The source code model converter 210, the model-to-model converter 220 and the code generator 230 will be described below with reference to FIGS. 3 to 7.

[0064]

[0065] FIG. 3 is a view illustrating a configuration and operation of a source code model converter in FIG. 2.

[0066] Referring to FIG. 3, a model converter 210 according to an embodiment of the present disclosure includes a conversion rule receiving unit 212, a parser 214, a meta model converting unit 216 and a model converting unit 218. When the source code is inputted, the conversion rule receiving unit 212 may receive a conversion rule, the parser 214 may parse the source code to store the data in a memory, and the meta model converting unit 216 may fetch the data stored in the memory to store source code information in a meta model which is independent from the program language. The model converting unit 218 may receive the conversion rule and convert the meta model into a goal model, calculating a model which meets a UML standard. The source code is a code to be converted into a UML model, which may be one of sources that are prepared of diverse programming languages, for example, Java, C and COBOL.

[0067] The conversion rule receiving unit 212 may receive a conversion rule to convert the received source code into a goal model. When there is a naming convention in the source code, the conversion rule receiving unit 212 may understand the convention and receive a conversion rule including a configuration of a conversion goal model. The conversion rule receiving unit 212 may receive the conversion rule including information on the inputted source code, information on

goal model to be converted, information on the naming convention and information on a relationship. The conversion rule will be described below with reference to FIG. 4.

[0068] The parser 214 may parse the source code to store the parsing data having a specific material structure in a memory. That is, the parser may parse the source code to upload the data having a material structure of a tree format on the memory. The parser 214 may have different forms depending on what programming language the source code has. The parsing is a step of a procedure where a compiler or a translator translates a source code into a machine language in a computer, which is a procedure to analyze a grammatical construction or syntax of each sentence. That is, it is a task to receive a string of token appearing in a source program to configure it as a parse tree, which is correct grammatically. Accordingly, the parser 214 may have different forms depending on the sort of the source code, and parse the source code with data having different material structure depending on a sort of a programming language of the source code.

[0069] The meta model converting unit 216 may convert the stored parsing data into a meta model. The meta model converting unit 216 may convert the data prepared by a variety of source codes into a meta model material structure. The meta model converting unit 216 enables the model reverse engineering apparatus 10 to generate the UML model having the same quality and form when changing the parser 214 regardless of what programming language the source code has. Accordingly, the meta model converting unit 216 may convert the data into a meta model showing all elements of the source code as a specific material structure regardless of sorts of the source code and programming languages.

[0070] The model converting unit 218 may convert the meta model converted by the meta model converting unit 216 into a goal model according to the conversion rule received by the conversion rule receiving unit 212. The model converting unit 218 may convert the converted goal model into a form meeting the UML standard again to write it on a file. The model converting unit 218 may receive a separate conversion rule for a syntax of a simple expression series which is not expressed as a UML model, to convert it into a UML model. The model converted through the

CLAIMS

1. A computer-executable model reverse engineering method, comprising:
 - (a) receiving a source code;
 - (b) receiving a conversion rule or a separate conversion rule for source code information that cannot be expressed in a Unified Modeling Language (UML) to convert the source code into a goal model;
 - (c) parsing the source code using a parser related to the source code to store parsing data having a specific material structure in a memory;
 - (d) converting the stored parsing data into a meta model;
 - (e) converting the meta model into the goal model according to the received conversion rule or separate conversion rule; and
 - (g) excluding the source code that is not concerned so as not to be converted according to the conversion rule or separate conversion rule.
2. The method of claim 1, further comprising (f) converting the converted goal model in a form adapted to a UML (Unified Modeling Language) standard again to be stored in a file.
3. The method of claim 1, wherein the step (b) comprises receiving the conversion rule configured of a source conversion rule, a model conversion rule, a naming rule and a relationship rule.
4. The method of claim 3, wherein the step (e) comprises designating and converting a specific element of the source code or a specific element of the goal model according to the conversion rule.
5. The method of claim 1, wherein the step (c) comprises parsing the source code into data having different material structures depending on programming language types of the source code.

6. The method of claim 1, wherein the step (d) comprises converting the parsing data into the meta model showing all elements of the source code in the specific material structure regardless of the source code types and the programming language types.
7. The method of claim 1, wherein the method further comprises generating a message in at least one of a message lifeline, an exception lifeline, a validation lifeline or a code lifeline, with respect to the source code information that cannot be expressed in the UML.
8. The method of claim 1, wherein the method further comprises verifying whether all information of the source code is included in the goal model.
9. The method of claim 1, wherein the step (e) comprises:
 - (e-1) converting the meta model into a source model according to the received conversion rule;
 - (e-2) receiving a model-to-model conversion rule to convert the source model into the goal model; and
 - (e-3) converting the source model into the goal model according to the model-to-model conversion rule.
10. The method of claim 1, further comprising:
 - (i) receiving the goal model to perform a configuration to generate the source code;
 - (j) reading elements of the goal model to confirm whether the elements are used to generate the source code;
 - (k) validating the model elements used to generate the source code according to a validation rule that the elements have; and
 - (l) generating the source code according to a result of the validation.
11. The method of claim 10, wherein the step (i) comprises registering elements of the goal model and conversion rules for respective elements and designating an execution condition of the conversion rule to perform the configuration.
12. A model reverse engineering apparatus, comprising:

a memory configured to store a model reverse engineering available program therein; and

a processor configured to execute the program,

wherein the processor parses a received source code into parsing data having a specific material structure using a parser, converts the parsed data into a meta model, converts the meta model into a goal model according to a received conversion rule or a received separate conversion rule for source code information that cannot be expressed in a Unified Modeling Language (UML), generates a source code using the converted goal model, and excludes the source code that is not concerned so as not to be converted according to the received conversion rule or separate conversion rule.

13. The apparatus of claim 12, wherein the processor receives the source code and the conversion rule or the separate conversion rule for source code information that cannot be expressed in the UML to convert the source code into the goal model, and stores parsing data having the parsed specific material structure in a memory.

14. The apparatus of claim 13, wherein the processor parses the received source code into data having different types of material structures depending on programming language types of the source code.

15. The apparatus of claim 13, wherein the processor converts the parsing data into the meta model showing all elements of the source code in the specific material structure regardless of the source code types and the programming language types.

16. The apparatus of claim 12, wherein the processor converts the source model into the goal model according to a model-to-model conversion rule which converts the source model received into the goal model.

17. The apparatus of claim 16, wherein the processor receives a model-to-model conversion rule that is configured of a qualified name of the source model and a qualified name of the goal model.

18. The apparatus of claim 12, wherein the processor receives the goal model to perform a configuration to generate the source code, reads elements of the goal model to confirm whether the elements are source code generation target elements, verifies the generation target model elements according to a validation rule that the elements have, and generates the source code according to a result of the validation verification.

19. The apparatus of claim 18, wherein the processor registers elements of the goal model and conversion rules for respective elements and designates an execution condition of the conversion rule to perform the configuration.

20. A computer-readable storage medium in which a program for executing a model reverse engineering method in a computer is stored, the method comprising:

- (a) receiving a source code;
- (b) receiving a conversion rule or a separate conversion rule for source code information that cannot be expressed in a Unified Modeling Language (UML) to convert the source code into a goal model;
- (c) parsing the source code using a parser related to the source code to store parsing data having a specific material structure in a memory;
- (d) converting the stored parsing data into a meta model;
- (e) converting the meta model into the goal model according to the received conversion rule or separate conversion rule; and
- (f) excluding the source code that is not concerned so as not to be converted according to the conversion rule or separate conversion rule.