

SUMMARY OF THE INVENTION

It is therefore an object of the invention to provide such a tool. To achieve the above and other objects, one aspect of the present invention presents a method for generating a software application using a computer system which comprises the steps of: modeling a business process in a graphical design tool implemented in said computer system, said graphical design tool capable of generating models to serve as blueprints in development of the software application; creating at least one Unified Modeling Language (UML) model to formalize the described processes by using one or more state diagrams to represent the modeled business processes, and one or more class diagrams to represent data models and to represent predetermined business rules that are applied to the described business processes; validating at least one UML model by detecting syntax errors using a predetermined syntax, in order to validate either at least one state diagram or class diagram, or combination thereof; transforming at least one validated UML model into metadata that is used to generate functional and non-functional aspects of the software application; generating a new Enterprise Application Archive (Ear) file and/or selecting an existing Ear file, the new Ear file and/or the selected existing Ear file comprising a plurality of Java Archive (JAR) files required for deployment of the software, wherein at least one JAR file is added and/or removed; generating a project that contains said at least one validated UML model, wherein generating said project comprises removing all conflicting stereotypes; and generating code automatically by: converting the UML model into an XML file, using the XML file to generate Java source code, and compiling the generated Java source code using a compiler and creating the at least one JAR .

In one embodiment of the present invention, the transforming step includes generating the software application.

In another embodiment of the present invention, the generated software application is combined with pre-existing libraries.

In another embodiment of the present invention, the generated software application is combined with manually written code.

In another embodiment of the present invention, the metadata includes one or more application code, or constraint models, or automate files, or documentation, or some combination thereof.

In another embodiment of the present invention, the generated application includes a predefined set of functional requirements.

In another embodiment of the present invention, the generated application includes a predefined set of non- functional requirements.

In another embodiment of the present invention, the predefined set of non functional requirements includes security management, or load balancing, or transaction management, or user interfaces, or a skeleton to build the algorithms on, or some combination thereof.

In another embodiment of the present invention, the generating step includes adding manually written code to the generated application.

In another embodiment of the present invention, the method further includes the step of deploying the target application by means of a deployment tool 490.

Another aspect of the present invention presents a method for generating a software application using a computer system which comprises the steps of: modeling business processes in a graphical design tool implemented in said computer system, said graphical design tool capable of generating models to serve as blueprints in development of the software application; creating at least one UML model to formalize the described processes by using state diagrams to represent the modeled business processes, and class diagrams to represent data models and to represent predetermined rules that are applied to the modeled business processes; validating at least one UML model by detecting syntax errors using a predetermined syntax, to validate said state diagrams, said class diagrams, or a combination thereof; transforming at least one UML model into metadata that are used to generate functional and non-functional aspects of the software application; generating said software application such that said software application can be deployed to a server by transforming the validated UML models into source code, combining the transformed code with pre-existing libraries, and configuration files; generating a new Enterprise Application Archive (Ear) file and/or selecting an existing Ear file, the new Ear file and/or the selected existing Ear file comprising a plurality of Java Archive (JAR) files required for deployment of the software, wherein at least one JAR file is added and/or removed, wherein said software application is deployed as a layer that sits on top of a UML visual modeling tool as a plug in; generating a project that contains said at least one validated UML model, wherein generating said project comprises removing all conflicting stereotypes; and generating code automatically by: converting the UML model into an XML file, using the XML file to generate Java source code, and compiling the generated Java source code using a compiler and creating the at least one JAR.

Another aspect of the present invention presents a method for generating a software application using a computer system which comprises the steps of: modeling business processes in a graphical design tool implemented in said computer system, said graphical design tool capable of generating models to serve as blueprints in development of the software application; creating at least one UML model to formalize the described processes by using one or more state diagrams to represent the modeled business processes, and one or more class diagrams to represent data models and to represent predetermined business rules that are applied to the modeled business processes; validating at least one UML model by detecting syntax errors using a predetermined syntax, to validate either said one or more state diagrams or class diagrams, or a combination thereof, producing at least one validated UML model; transforming at least one UML model into metadata that are used to generate functional and non-functional aspects of the software application; generating a new Enterprise Application Archive (Ear) file and/or selecting an existing Ear file, the new Ear file and/or the selected existing Ear file comprising a plurality of Java Archive (JAR) files required for deployment of the software, wherein at least one JAR file is added and/or removed; generating a project that contains said at least one validated UML model, wherein generating said project comprises removing all conflicting stereotypes; deploying the software application as a layer that sits on top of a UML visual modeling tool as a plug in; and generating code automatically by: converting the UML model into an XML file, using the XML file to generate Java source code, and compiling the generated Java source code using a compiler and creating the at least one JAR.

What is claimed is:

1. A method for generating a software application using a computer system, comprising the steps of:

(a) modeling business processes in a graphical design tool implemented in said computer system, said graphical design tool capable of generating models to serve as blueprints in development of the software application;

(b) creating, in said computer system via Unified Modeling Language (UML), at least one UML model to formalize the business processes modeled in step (a) by using one or more state diagrams to represent the modeled business processes, and one or more class diagrams to represent data models and to represent predetermined business rules that are applied to the modeled business processes, wherein one or more stereotypes are added to a class of said one or more class diagrams, wherein a tagged value is used in the data models to define an exclusive dependency between a current field and another field in the same entity type of the data models or in another entity type of the data models with a relation;

(c) validating, in said computer system, the at least one UML model by detecting syntax errors using a predetermined syntax, to validate either said one or more state diagrams or said one or more class diagrams, or a combination of said one or more state diagrams and said one or more class diagrams;

(d) transforming, in said computer system, said at least one UML model validated in step (c) into metadata that are used to generate functional and non-functional aspects of the software application, the software application being a layer that sits on top of a UML visual modeling tool as a plug-in;

(e) generating, via the at least one UML model a new Enterprise Application Archive (Ear) file and/or selecting an existing Ear file, the new Ear file and/or the selected existing Ear file comprising a plurality of Java Archive (JAR) files required for deployment of the software, wherein at least one JAR file is added and/or removed;

(f) generating a project that contains said at least one validated UML model, wherein generating said project comprises removing all conflicting stereotypes; and

(g) generating code automatically by:

converting the UML model into an XML file;

using the XML file to generate Java source code; and

compiling the generated Java source code using a compiler and creating the at least one JAR.

2. The method of claim 1, wherein the transforming step includes generating the software application.

3. The method of claim 2, wherein the generated software application is combined with pre-existing libraries.

4. The method of claim 2, wherein the generated software application is combined with manually written code.

5. The method of claim 2, wherein the generated software application includes a predefined set of functional requirements.
6. The method of claim 2, wherein the generated software application includes a predefined set of non-functional requirements.
7. The method of claim 6, wherein the predefined set of non-functional requirements includes at least one of security management, load balancing, transaction management, user interfaces, and a skeleton on which to build algorithms.
8. The method of claim 2, wherein the generating the software application includes adding manually written code to the generated software application.
9. The method of claim 1, wherein the metadata comprise at least one of application code, constraint models, automata files, and documentation.
10. The method of claim 1, further comprising deploying the software application by means of transforming the validated UML models to be deployed to a server.
11. The system of claim 1, wherein the syntax includes Stereotypes for classes (composite, constrainable, entity, macro, viewable), states (loadable), fields (ASP, code, same as, sequence, transient, unique), and relations (code, list, map, set, transient, unique).
12. The method of claim 1, wherein predefined types of classes and decorations are applied within the one or more class diagrams and the decorations are organized, in part, by tagged values in step (b).
13. A method for generating a software application using a computer system, comprising the steps of:
 - (a) modeling business processes in a graphical design tool implemented in said computer system, said graphical design tool capable of generating models to serve as blueprints in development of the software application;
 - (b) creating, in said computer system via Unified Modeling Language (UML), at least one UML model to formalize the business processes modeled in step (a) by using state diagrams to represent the modeled business processes, and class diagrams to represent data models and to represent predetermined rules that are applied to the modeled business processes, wherein steps in said business processes are expressed by states of said state diagrams and interconnectivity of said states are expressed by transitions, wherein transitions between said states occurs after satisfying a condition, wherein a tagged value is used in the data models to define an exclusive dependency between a current field and another field in the same entity type of the data models or in another entity type of the data models with a relation;

(c) validating, in said computer system, the at least one UML model by detecting syntax errors using a predetermined syntax, to validate said state diagrams, said class diagrams, or a combination of said state diagrams and said class diagrams;

(d) transforming, in said computer system, said at least one UML model validated in step (c) into metadata that are used to generate functional and non-functional aspects of the software application, the software application being a layer that sits on top of a UML visual modeling tool as a plug-in;

(e) generating said software application such that said software application can be deployed to a server by transforming the validated UML models into source code, combining the transformed code with pre-existing libraries, and configuration files;

(f) generating, via the at least one UML model a new Enterprise Application Archive (Ear) file and/or selecting an existing Ear file, the new Ear file and/or the selected existing Ear file comprising a plurality of Java Archive (JAR) files required for deployment of the software, wherein at least one JAR file is added and/or removed, wherein said software application is deployed as a layer that sits on top of a UML visual modeling tool as a plug in;

(g) generating a project that contains said at least one validated UML model, wherein generating said project comprises removing all conflicting stereotypes; and

(h) generating code automatically by:

converting the UML model into an XML file;

using the XML file to generate Java source code; and

compiling the generated Java source code using a compiler and creating the at least one JAR.

14. The method of claim 13, wherein step (e) comprises transforming the validated UML model into metadata that are used to generate functional and non-functional aspects of the software application.

15. The method of claim 13, wherein predefined types of classes and decorations are applied within the class diagrams and the decorations are organized, in part, by tagged values in step (b).

16. A method for generating a software application using a computer system, comprising the steps of:

(a) modeling business processes in a graphical design tool implemented in said computer system, said graphical design tool capable of generating models to serve as blueprints in development of the software application;

(b) creating, in said computer system via Unified Modeling Language (UML), at least one UML model to formalize the business processes modeled in step (a) by using one or more state diagrams to represent the modeled business processes, and one or more class diagrams to represent data models and to represent

predetermined business rules that are applied to the modeled business processes, wherein a tagged value is used in the data models to define an exclusive dependency between a current field and another field in the same entity type of the data models or in another entity type of the data models with a relation;

(c) validating, in said computer system, the at least one UML model by detecting syntax errors using a predetermined syntax, to validate either said one or more state diagrams or said one or more class diagrams, or a combination of said one or more state diagrams and said one or more class diagrams, producing at least one validated UML model, wherein one or more use-cases and allowed actions for each use-case is generated via said one or more state diagrams, facilitating defining actions and drawing use-cases by a user;

(d) transforming, in said computer system, said at least one validated UML model validated in step (c) into metadata that are used to generate functional and non-functional aspects of the software application, the software application being a layer that sits on top of a UML visual modeling tool as a plug-in; and

(e) generating, via the at least one UML model a new Enterprise Application Archive (Ear) file and/or selecting an existing Ear file, the new Ear file and/or the selected existing Ear file comprising a plurality of Java Archive (JAR) files required for deployment of the software, wherein at least one JAR file is added and/or removed;

(f) generating a project that contains said at least one validated UML model, wherein generating said project comprises removing all conflicting stereotypes;

(g) deploying the software application as a layer that sits on top of a UML visual modeling tool as a plug in; and

(h) generating code automatically by:

converting the UML model into an XML file;

using the XML file to generate Java source code; and

compiling the generated Java source code using a compiler and creating the at least one JAR.

17. The method of claim 16, wherein predefined types of classes and decorations are applied within the one or more class diagrams and the decorations are organized, in part, by tagged values in step (b).

18. The method of claim 16, further comprising generating said software application such that said software application can be deployed to a server by transforming the validated UML models into source code, combining the transformed code with pre-existing libraries, and configuration files.

19. The method of claim 18, further comprising adding manually written code to the generated software application.

20. The method of claim 18, wherein the generated software application is combined with pre-existing libraries.