

METHOD FOR MANAGING DEVELOPMENT OF A SOFTWARE APPLICATION

TECHNICAL FIELD

- 5 This invention relates generally to method for managing development of a software application for achieving a business objective.

Background

10 Large language models (LLMs) have been increasingly touted for software development, which can generate, optimize, and troubleshoot code through natural language processing. These models, trained on vast datasets, offer an unprecedented capability to automate portions of the software development process, enabling faster and more accessible programming for a broader audience. However, despite their potential, using LLMs in software development has notable limitations.

15 One of the primary shortfalls of LLMs in software development is their dependence on existing code patterns and data. LLMs generate code based on probabilistic patterns observed in the training data, which can lead to issues such as overfitting, repetition of outdated practices, or the inclusion of suboptimal code snippets. Moreover, they may struggle with understanding the nuanced requirements of complex software projects, leading to code that, while syntactically correct, fails to meet specific functional or security requirements. The lack of contextual awareness and the potential for generating insecure or inefficient code further underscore the limitations of relying solely on LLMs for software development.

25 Given the current state of LLM maturity, human oversight should be maintained throughout software development process to maintain effectiveness, relevance, and appropriateness. Nonetheless, some new skills are required to use LLMs effectively and coding capabilities are required to validate their results. As users may come from various backgrounds and have different levels of technical expertise, an intuitive and visual interface-based approach

30

to software development is desirous without traditional coding knowhow. There is therefore a need for a method and system for addressing the foregoing problems.

Summary

5 In accordance with a first aspect of the invention, there is disclosed a method for managing the development of a software application comprising generating first project codes of a software application by a code language model based on developmental requirements being provided thereto, the first project codes being in a first code format, and rendering the first project codes into at least one of components, relational vertices, code segments, functional
10 descriptors of code segments and natural language representation of code segments in a user-interface (UI) environment and presented on a UI of a computing device. At least one of the components, relational vertices, code segments, functional descriptors of code segments and natural language representation of code segments being editable via the UI to effect a corresponding change in the first project codes.

15 In accordance with a second aspect of the invention, there is disclosed a system for managing the development of a software application, the system comprising cloud-based storage circuitry storing a code language model, and cloud-based control circuitry configured to generate first project codes of a software application by a code language
20 model based on developmental requirements being provided thereto, the first project codes being in a first code format. The system further comprising cloud-based input/output circuitry configured for communication with a user interface (UI) to render the first project codes into at least one of components, relational vertices, code segments, functional descriptors of code segments and natural language representation of code segments in a
25 user-interface (UI) environment and presented on a UI of a computing device. At least one of the components, relational vertices, code segments, functional descriptors of code segments and natural language representation of code segments being editable via the UI to effect a corresponding change in the first project codes.

In accordance with a third aspect of the invention, there is disclosed a non-transitory computer readable medium comprising instructions that when executed by one or more processors cause operations comprising generating first project codes of a software application by a code language model based on developmental requirements being provided thereto, the first project codes being in a first code format, and rendering the first project codes into at least one of components, relational vertices, code segments, functional descriptors of code segments and natural language representation of code segments in a user-interface (UI) environment and presented on a UI of a computing device. At least one of the components, relational vertices, code segments, functional descriptors of code segments and natural language representation of code segments being editable via the UI to effect a corresponding change in the first project codes.

Brief Description of the Drawings

FIG. 1 shows an exemplary process flow diagram of a method for managing the development of a software application according to an aspect of the invention;

5

FIG. 2 shows a system block diagram of a system for utilizing the development method of FIG. 1;

FIG. 3 shows an exemplary language model configuration of the system of FIG. 2;

10

FIG. 4 shows a process flow diagram of a step of generating first project codes in the method of FIG. 1; and

FIG. 5 shows a process flow diagram of a step of converting the first project codes into the second project codes of the method of FIG. 1.

15

Detailed Description

An exemplary embodiment of the present invention, a method for managing the development of a software application (“the method 100”) for achieving a business objective applied through a system 20 for managing the development of a software application (“the system 20”) is described hereinafter with reference to FIGS. 1 to 5.

Implementations of the present disclosure are generally directed to the method 100, as shown in FIG. 1, and the associated system 20 to provide a comprehensive platform system to facilitate delivery of services in, but not limited, business-to-business (B2B), business-to-consumer (B2C), consumer-to-consumer (C2C), business-to-government (B2G) and government-to-business (G2C) arrangements. The B2B applies also to individuals requiring operating in a business framework. The services provided can be adapted through a choice of B2B gateway services. The services can be aggregated by and channeled through third-parties.

An exemplary system architecture of the system 20 that can execute implementations of the present disclosure, can include one or more of a user computing device 22 associated with a user and a system controller 24 (controller module 24). As shown in FIG. 2, the system 20 further comprises a database module 26 in data communication with the controller module 24. The user computing device 22 can communicate with the controller module 24 over a network. In some implementations, the system 20 may represent a client/server system supporting multiple computer systems (e.g., the controller module 24) including one or more clients (e.g., the user computing device 22) that are connectively coupled for communication with one another over the network.

The user computing device 22 can represent various forms of processing devices including, but not limited to, a desktop computer, a laptop computer, a handheld computer, a personal digital assistant (PDA), a smartphone, a smart tablet, a cellular telephone, a network appliance, a camera, a smart phone, an enhanced general packet radio service (EGPRS) mobile phone, a media player, a navigation device, an email device, a game console, or a

combination of any two or more of these data processing devices or other data processing devices. The user computing device may access application software on one or more of the control computer systems 26.

- 5 The controller module 24 can represent various forms of server systems including, but not limited to a web server, an application server, a proxy server, a network server, or a server farm. For example, one or more of the servers can be an application server that executes software accessed by the user computing device 22. In some implementations, a user can invoke applications available on one or more of the servers in a web browser or a mobile application running on a client (e.g., the user computing device 22). Each application can individually access data from one or more repository resources (e.g. the database module 26).

- 15 The network can be a large computer network, such as a local area network (LAN), wide area network (WAN), the Internet, a cellular network, or a combination thereof connecting any number of mobile clients, fixed clients, and/or servers. In some implementations, each client (e.g., the user computing device 22) can communicate with one or more of the controller module 24 via a virtual private network (VPN), Secure Shell (SSH) tunnel, or other secure network connection. In some implementations, the network can include the Internet, a wireless service network and may include the Public Switched Telephone Network (PSTN). In other implementations, the network may include a corporate network (e.g., an intranet) and one or more wireless access points.

- 25 The user computing device 22 can establish its own session with the controller module 24. Each session can involve two-way information exchange between the controller module 24 and the user computing device 22. For example, a Hypertext Transfer Protocol (HTTP) session can allow the association of information with individual users. A session can be stateful session, in which at least one of the communicating parts (e.g., the controller module 24 or the user computing device) stores information about the session history in

order to be able to communicate. Alternatively, stateless communication during a stateless session includes independent requests with associated responses.

In some embodiments, user interface 28 (UI 28) may comprise a user interface on the user computing device 22 for accessing a comprehensive automation and support toolset for software application development and for generating project codes and the user interface environment, or an editor environment, for editing the project codes at least one of textually and graphically. For example, user interface 100 represents an illustrative user interface for requesting development, providing user requirements and for interacting with a knowledge graph residing on the database module 26 (e.g. stored on storage circuitry and accessed through input/output circuitry). The UI 28 preferably presents a low code environment of an instance of an application that guides and assists a user through the development process, while still allowing for user control and customization when required.

The system 20 may comprise numerous components to generate the UI 28 and for controlling the control module 24. For example, the system 20 may include one or more lambda functions (e.g., executed by control circuitry 27A) to provide provision and data collection as well as reporting. The system 20 further comprises an artificial intelligence engine 29 (AI engine 29), for example a machine learning based calculation and analysis engine, used to facilitate and enhance the development of. In some embodiments, the system 20 may be built using a cloud-based approach and access a central repository 27B. The central repository 27B may receive and store one or more memory modules for the language models 34, reference templates 36, knowledge graphs and related data. The system 20 can comprise cloud-based input/output circuitry 27C configured to generate at the UI 28 various outputs and for receiving, via the UI 28, various inputs from and interactions with the user.

Large language model (LLM) agents are advanced AI systems designed for creating complex text that needs sequential reasoning. They can think ahead, remember past conversations, and use different tools to adjust their responses based on the situation and style needed. Beyond just looking up facts or information, the agents are required to

understand new rules, how they affect tasks or sub tasks to which they are applied. different companies, and finding out what courts have said about it all. When there is a need to connect the rules to the tasks through multi-stage reasoning, the agent can break down its tasks into subtasks like so. These subtasks may involve the retrieval of necessary information and rules and, the establishment of a historical baseline of how similar issues were previously handled, and to collate observed patterns. To complete these subtasks, the LLM agent requires a structured plan, an ability to track progress, and access to necessary tools.

At the core of an LLM agent is a language model that processes and understands language based on a vast amount of data it's been trained on. Through the provision of an input, or a prompt, the agent is guided on how to respond, what tools to use, and the goals it should aim to achieve during the interaction.

Additionally, agents can be customized with a specific persona. This means setting up the agent with certain characteristics and expertise that make it better suited for particular tasks or interactions. Essentially, the core of an LLM agent combines advanced processing abilities with customizable features to effectively handle and adapt to various tasks and interactions. Through the use of the method 100 and the system 20, one or more agents in a particular workflow may be tuned to perform tasks in a way that feels right for the situation.

An exemplary embodiment of the method 100 comprises a step 110 of generating first project codes 30 of a software application by a code language model 32 based on developmental requirements 34 being provided thereto. The developmental requirements 34 preferably comprises business requirements and at least one of a plurality of user journeys and a plurality of user stories. The developmental requirements 34 can further comprise a project descriptor, project requirements, project details and project concerns for development of the software application. The developmental requirements 34 may be prepared and provided by a user, be provided by an upstream system in a software developmental pipeline or be generated by preceding language models based on input and

feedback provided by the user. The developmental requirements 34 can also be from textual input, generated through keystroke input, for example keyboard input, speech to text input, handwriting to text input, electrical signals transmitted from another signaling or communication device, from preceding one or more LLMs or LLM agents, a preceding
5 data-generating or data storage module, or any combination thereof.

The first project codes 30 are in a first code format. The method 100 further comprises a step 112 of rendering the first project codes 30 into at least one of components, relational
10 vertices, code segments, functional descriptors of code segments and natural language representation of code segments in a user-interface environment and presented on a UI 28 of a computing device 22. At least one of the components, relational vertices, code segments, functional descriptors of code segments and natural language representation of code segments being editable via the UI 28 to effect a corresponding change in the first project
15 codes 30.

The method 100 further comprises a step 114 of converting the first project codes 30 from the first code format into a second code format by a code converter 36 prior to the first project codes 30 being rendered in the UI environment and presented on the UI 28 of the
20 computing device 22. In various embodiments, the code language model 32 uses at least one of a pre-tuned model, live-tuned model, a retrieval augmented generation (RAG) system and a code reference template for generating the first project codes 30. The code reference template provides a code format guideline for generating the first project codes 30 based on the context of the developmental requirements 34

25

In some implementations, the code convertor 36 comprising an abstract syntax tree (AST) for parsing the first project codes 30 into syntactic components for subsequent mapping and converting of the first project codes 30 from the first code format into the second code format.

30

Additionally or alternatively, the code language model 32 is provided with constraint-based instructions for generating the first project codes 30. The constraint-based instructions can comprise the aforesaid code format guideline. The code language model 32 may be provided with one-shot examples to aid reference or inspiration in generating of the first project codes 30.

Examples of constraint-based instructions comprise:

“You must always follow data structures and interfaces, and program call flows. Don’t change any design.” and “Write out every code detail. Don’t leave TODO”, and “Use one shot example as inspiration for your design and UI.”.

Preferably, the rendered components comprise control elements, action descriptors, text and corresponding properties with the relational vertices representing relational keys and flow between components. In various embodiments, the first code format is a web application framework comprising at least one of HTML, CSS and Javascript (JS) codes, and the second code format is a Javascript Object Notation (JSON).

The method 100 further comprises a step 120 of generating a plurality of tasks corresponding to the developmental requirements 34, and a step 122 of associating each of the plurality of tasks to one of a plurality of data items, the plurality of data items being based on the dependency order thereamongst. Each of the plurality of tasks having a portion of the first project codes 30 corresponding therewith. Following the step 122, each portion of the first project codes 30 is assigned to the one of the plurality of data items corresponding with the one of the plurality of tasks associated therewith in a step 124. Preferably, the plurality of data items comprises at least one of variables with pre-defined variable names and files having pre-defined file names and pre-defined file formats.

The method 100 further comprises a step 130 of deriving a test case and test scripts from a plurality of reference test case and reference test scripts, a knowledge graph comprising the plurality of reference test case and reference test scripts, and a step 132 of applying the

derived test scripts to the first project codes 30 in the first code format to obtain test results therefrom. The first project codes 30 are then debuggable based on the test results. In a step 134, the first project codes 30 may be further evaluated based on its evaluating constraints and dependencies to further determine or influence the test results. Based on the outcome of the test results, the first project codes 30 are debugged in a step 140 before re-applying the derived test scripts to the first project codes 30 to re-obtain test results therefrom in a step 142.

Debugging of the first project codes 30 may be performed by the user via the UI 28, by the code language model 32 based on feedback input, for example textual input, captured from the user or by a standalone review language model 38, constituting a review LLM agent, in a step 124. The review language model 38 learns the developmental, review and debugging preferences of the user, including the tools utilized, and performs debugging of the first project codes 30 as a proxy of the user. The review language model 38 can be pre-configured with structured or unstructured rule-based constraints, have debugging approaches and requirements generated based on the needs or demands of the test results, or based on reference debugging guidelines and process flow.

Preferably, the code converter 36 is a specific LLM agent that uses a language model which augments the conversion based on is supplemented by information from, for example, a vector database using retrieval augmented generation (RAG). The first project codes 30 in the first code format become second project codes 40 once it has been converted by the code converter 36 into the second code format in the step 114.

As the user-interface environment in which the first project codes 30 in the second code format, now also known as the second project codes 40, are rendered in can be an editor of a software development platform 42, for example a no-code or low-code development platform, specific consideration has to be taken to the functions of the software development platform and those that exist within the first project codes 30 in the first code format.

Hence, selected segments of the first project codes 30 that correspond with selected functions of the software development platform are first converted for code validation and code refinement in a step 150 by the user, or by a validation language model 44 of a validation LLM agent functioning as a validation and code refinement proxy for the user, before the rest of or all of the first project codes 30 in the first code format are converted to the second project codes 40 in a step 152.

Hence, once the first project codes in the first format have been converted into the second project codes 40, the second project codes 40 are then rendered in the UI environment of the software development platform 42 for editing and further development via the UI 28 of the computing device 22. On the software development platform 42, editing at least one of the components, relational vertices, code segments, functional descriptors of code segments and natural language representation of code segments in the user-interface environment of via the UI 28 will effect a corresponding change in the second project codes 40.

Once editing and further development of the second project codes 40 on the software development platform 42 has been completed, the second project codes 40 may be provided as the final software application or part thereof.

The method 100 and the system 20 may be implemented using a non-transitory computer readable medium comprising instructions that when executed by one or more processors cause operations comprising generating first project codes 30 of a software application by a code language model 32 based on developmental requirements 34 being provided thereto, the first project codes 30 being in a first code format, and rendering the first project codes 30 into at least one of components, relational vertices, code segments, functional descriptors of code segments and natural language representation of code segments in a user-interface (UI 28) environment and presented on a UI 28 of a computing device 22, at least one of the components, relational vertices, code segments, functional descriptors of code segments and natural language representation of code segments being editable via the UI 28 to effect a corresponding change in the first project codes 30.

Aspects of particular embodiments of the present disclosure address at least one aspect, problem, limitation, and/or disadvantage associated with existing computer-implemented methods and systems. While features, aspects, and/or advantages associated with certain
5 embodiments have been described in the disclosure, other embodiments may also exhibit such features, aspects, and/or advantages, and not all embodiments need necessarily exhibit such features, aspects, and/or advantages to fall within the scope of the disclosure. It will be appreciated by a person of ordinary skill in the art that several of the above-disclosed structures, components, or alternatives thereof, can be desirably combined into alternative
10 structures, components, and/or applications. In addition, various modifications, alterations, and/or improvements may be made to various embodiments that are disclosed by a person of ordinary skill in the art within the scope of the present disclosure, which is limited only by the following claims.

Claims

1. A method for managing the development of a software application comprising:
generating first project codes of a software application by a code language
model based on developmental requirements being provided thereto, the first project
5 codes being in a first code format; and
rendering the first project codes into at least one of components, relational
vertices, code segments, functional descriptors of code segments and natural
language representation of code segments in a user-interface (UI) environment and
presented on a UI of a computing device, at least one of the components, relational
10 vertices, code segments, functional descriptors of code segments and natural
language representation of code segments being editable via the UI to effect a
corresponding change in the first project codes.
2. The method as in claim 1, further comprising:
15 converting the first project codes from the first code format into a second
code format by a code converter prior to the first project codes being rendered in the
UI environment and presented on the UI of the computing device.
3. The method as in claim 2, the code converter comprising an abstract syntax tree
20 (AST) for parsing the first project codes into syntactic components for subsequent
mapping and converting of the first project codes from the first code format into the
second code format.
4. The method as in claim 1, wherein the code language model uses at least one of a
25 pre-tuned model, live-tuned model, a retrieval augmented generation (RAG) system
and a code reference template for generating the first project codes.
5. The method as in claim 4, further comprising:
providing the code language model with constraint-based instructions for
30 generating the first project codes, the code reference template providing a code

format guideline for generating the first project codes based on the context of the developmental requirements.

- 5 6. The method as in claim 1, the components comprising control elements, action descriptors, text and corresponding properties with the relational vertices representing relational keys and flow between components.
- 10 7. The method as in claim 1, the first code format being a web application framework comprising at least one of HTML, CSS and Javascript (JS) codes, and the second code format being a Javascript Object Notation (JSON).
- 15 8. The method as in claim 1, further comprising:
 - deriving a test case and test scripts from a plurality of reference test case and reference test scripts, a knowledge graph comprising the plurality of reference test case and reference test scripts; and
 - applying the derived test scripts to the first project codes to obtain test results therefrom,
 - wherein the first project codes are debuggable based on the test results.
- 20 9. The method as in claim 8, further comprising:
 - evaluating constraints and dependencies of the first project codes.
- 25 10. The method as in claim 8, further comprising:
 - debugging the first project codes; and
 - re-applying the derived test scripts to the first project codes to obtain test results therefrom,
- 30 11. The method as in claim 1, further comprising:
 - generating a plurality of tasks corresponding to the developmental requirements;

associating each of the plurality of tasks to one of a plurality of data items, the plurality of data items being based on the dependency order thereamongst, each of the plurality of tasks having a portion of the first project codes corresponding therewith; and

- 5 assigning each portion of the first project codes to the one of the plurality of data items corresponding with the one of the plurality of tasks associated therewith, wherein the plurality of data items comprises at least one of variables with pre-defined variable names and files having pre-defined file names and pre-defined file formats.

10

12. A system for managing the development of a software application, the system comprising:

cloud-based storage circuitry storing a code language model;

cloud-based control circuitry configured to:

15

generate first project codes of a software application by a code language model based on developmental requirements being provided thereto, the first project codes being in a first code format; and

cloud-based input/output circuitry configured for communication with a user interface (UI) to:

20

render the first project codes into at least one of components, relational vertices, code segments, functional descriptors of code segments and natural language representation of code segments in a user-interface (UI) environment and presented on a UI of a computing device, at least one of the components, relational vertices, code segments, functional descriptors of code segments and natural language representation of code segments
25 being editable via the UI to effect a corresponding change in the first project codes.

13. A non-transitory computer readable medium comprising instructions that when
30 executed by one or more processors cause operations comprising:

generating first project codes of a software application by a code language model based on developmental requirements being provided thereto, the first project codes being in a first code format; and

rendering the first project codes into at least one of components, relational vertices, code segments, functional descriptors of code segments and natural language representation of code segments in a user-interface (UI) environment and presented on a UI of a computing device, at least one of the components, relational vertices, code segments, functional descriptors of code segments and natural language representation of code segments being editable via the UI to effect a corresponding change in the first project codes.

14. The non-transitory computer readable medium of claim 13, wherein the instructions further cause operations comprising

converting the first project codes from the first code format into a second code format by a code converter prior to the first project codes being rendered in the UI environment and presented on the UI of the computing device.

15. The non-transitory computer readable medium of claim 14, the first code format being a web application framework comprising at least one of HTML, CSS and Javascript (JS) codes, and the second code format being a Javascript Object Notation (JSON),

wherein the code convertor comprises an abstract syntax tree (AST) for parsing the first project codes into syntactic components for subsequent mapping and converting of the first project codes from the first code format into the second code format.

16. The non-transitory computer readable medium of claim 13, wherein the code language model uses at least one of a pre-tuned model, live-tuned model, a retrieval augmented generation (RAG) system and a code reference template for generating the first project codes.

17. The non-transitory computer readable medium of claim 16, wherein the instructions further cause operations comprising:

5 providing the code language model with constraint-based instructions for generating the first project codes, the code reference template providing a code format guideline for generating the first project codes based on the context of the developmental requirements.

- 10 18. The non-transitory computer readable medium of claim 13, the components comprising control elements, action descriptors, text and corresponding properties with the relational vertices representing relational keys and flow between components.

- 15 19. The non-transitory computer readable medium of claim 13, wherein the instructions further cause operations comprising:

 deriving a test case and test scripts from a plurality of reference test case and reference test scripts, a knowledge graph comprising the plurality of reference test case and reference test scripts;

20 applying the derived test scripts to the first project codes to obtain test results therefrom;

 evaluating constraints and dependencies of the first project codes;

 debugging the first project codes; and

 re-applying the derived test scripts to the first project codes to obtain test results therefrom,

25

20. The non-transitory computer readable medium of claim 13, wherein the instructions further cause operations comprising:

 generating a plurality of tasks corresponding to the developmental requirements;

associating each of the plurality of tasks to one of a plurality of data items, the plurality of data items being based on the dependency order thereamongst, each of the plurality of tasks having a portion of the first project codes corresponding therewith; and

5 assigning each portion of the first project codes to the one of the plurality of data items corresponding with the one of the plurality of tasks associated therewith,

 wherein the plurality of data items comprises at least one of variables with pre-defined variable names and files having pre-defined file names and pre-defined file formats.