

METHOD FOR MANAGING GENERATING OF DEVELOPMENTAL REQUIREMENTS FOR A SOFTWARE APPLICATION

TECHNICAL FIELD

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

Background

Implementing complex workflows with a Large Language Model (LLM) often leads to
10 inefficiencies and inaccuracies. A single model may not excel at every task required in a diverse workflow, resulting in suboptimal performance. For instance, a model designed for software development may not fully account for the thought process and task that goes into each software development role in a software development workflow. One approach to address this is to employ a multi-LLM, or multi-LLM agent, workflow. This enables the
15 creation of complex text for sequential reasoning.

However, creative content generation might struggle with technical data analysis, leading to errors and inconsistencies. However, this may introduce challenges in maintaining consistency and coherence across different tasks. Without a structured framework, outputs
20 from a single LLM can vary in format and context, making the results unreliable and harder to manage. This lack of standardization complicates the process of scaling operations and ensuring that all tasks are executed according to the desired guidelines to achieve a common business objective. Troubleshooting, or debugging, the errors and inconsistencies across the various LLMs with varying tasks can also be time-consuming and inefficient. There is
25 therefore a need for a method and system for addressing the foregoing problems.

Summary

In accordance with a first aspect of the invention, there is disclosed a method for managing generating of developmental requirements for a software application for achieving a
30 business objective comprising receiving user input comprising input user requirements for

meeting the business objective, and processing the user input by a first language model using a first reference template for generating first artifacts therefrom, the first reference template comprising reference input, a plurality of reference requirements and reference descriptors and the first artifacts comprising output requirements and output descriptors substantially correlated with the reference requirements and reference descriptors. At least a portion of the first artifacts forms a portion of the functional requirements. The first artifacts are further processable by at least one additional language model using at least one additional reference template for generating additional artifacts, the first artifacts and the additional artifacts constituting the developmental requirements to be processed by a code language model for generating at least a portion of codes of the software application therefrom, each of the first language model and the additional language model adopting a corresponding one of a plurality of software development roles, each of the plurality of software development roles having role-specific objectives, capabilities and behavioral pattern.

In accordance with a second aspect of the invention, there is disclosed a system for use in managing generating of developmental requirements for a software application for achieving a business objective, the system comprising cloud-based storage circuitry storing a first language model and a first reference template, and cloud-based control circuitry configured to receive user input comprising input user requirements for meeting the business objective, and process the user input by the first language model using the first reference template for generating first artifacts therefrom, the first reference template comprising reference input, a plurality of reference requirements and reference descriptors and the first artifacts comprising output requirements and output descriptors substantially correlated with the reference requirements and reference descriptors. The system further comprises cloud-based input/output circuitry configured for communication with a user interface (UI) to present at least a portion of the first artifacts on the UI for review by a user, receive feedback input from the user via the UI, and process the user input based on the feedback input by the first language model for updating the first artifacts generated thereby. At least a portion of the first artifacts forming a portion of the functional requirements, the

first artifacts are further processable by at least one additional language model using at least one additional reference template for generating additional artifacts, the first artifacts and the additional artifacts constituting the developmental requirements to be processed by a code language model for generating at least a portion of codes of the software application
5 therefrom, each of the first language model and the additional language model adopting a corresponding one of a plurality of software development roles, each of the plurality of software development roles having role-specific objectives, capabilities and behavioral pattern.

10 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 receiving user input comprising input user requirements for a software application for meeting a business objective, developmental requirements for the software application to be generatable from the user input, and
15 processing the user input by a first language model using a first reference template for generating first artifacts therefrom, the first reference template comprising reference input, a plurality of reference requirements and reference descriptors and the first artifacts comprising output requirements and output descriptors substantially correlated with the reference requirements and reference descriptors. At least a portion of the first artifacts
20 forming a portion of the functional requirements, the first artifacts are further processable by at least one additional language model using at least one additional reference template for generating additional artifacts, the first artifacts and the additional artifacts constituting the developmental requirements to be processed by a code language model for generating at least a portion of codes of the software application therefrom, each of the first language
25 model and the additional language model adopting a corresponding one of a plurality of software development roles, each of the plurality of software development roles having role-specific objectives, capabilities and behavioral pattern.

Brief Description of the Drawings

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

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; and

FIG. 4 shows a process flow diagram of a step of generating first artifacts in the method of FIG. 1.

Detailed Description

An exemplary embodiment of the present invention, a method 100 for managing generating of developmental requirements 21 for a software application (“the method 100”) for achieving a business objective applied through a system 20 for managing generating of developmental requirements 21 for a software application (“the system 20”) is described hereinafter with reference to FIGS. 1 to 4.

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
 10 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
 20 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
 30 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.

5 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 the developmental requirements 21 therefor. For example, user interface 100 represents an illustrative user interface for requesting development, providing user requirements for interacting with a knowledge graph residing on the database module 26 (e.g. stored on storage circuitry and accessed
10 through input/output circuitry). The UI 28 preferably presents a low code or no 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
15 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 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.

25

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
30 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.

20

An exemplary embodiment of the method 100 comprises a step 110 of receiving user input 28 comprising input user requirements for meeting the business objective, and a step 112 of processing the user input 28 by a first language model 30 using a first reference template 32 for generating first artifacts 34 therefrom. The first reference template 32 comprises reference input, a plurality of reference requirements and a plurality of reference descriptors. The first artifacts 34 comprising a plurality of output requirements and a plurality of output descriptors substantially correlated with the reference requirements and reference descriptors of the first reference template 32. When viewed from an LLM agent implementation perspective, steps 110 and steps 112 can be seen as being performed by a LLM agent comprising the first language model 30 and the first reference template 32.

The user input 28 is preferably received from a user via the UI 28. The user input 28 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
 5 signaling or communication device, from preceding one or more LLMs or LLM agents, a preceding data-generating or data storage module, or any combination thereof.

Preferably, each of the output requirements and the reference requirements comprises a plurality of business requirements, and each of the output descriptors and the reference
 10 descriptors comprising at least one of a plurality of user journeys and a plurality of user stories. Each of the output descriptors and the reference descriptors can further comprise a project descriptor, project requirements, project details and project concerns.

An example of the reference input of the first reference template 32 comprises:

15 “Develop a comprehensive field sales application to streamline sales process and enhance efficiency of field sales”.

Examples of the plurality of reference requirements of the first reference template 32 comprises business requirements such as:

20 “Introduce users to the application with an engaging splash page that explains its purpose and functionality;
 Enable efficient data management and access by providing a robust database connection feature; and
 Ensure secure user authentication and account access with reliable login system.”.

25

Examples of the plurality of reference descriptors of the first reference template 32 comprises user stories such as:

 “As a user, I want to see a splash page when I first visit the website, so that I can learn about the purpose of the application;
 30 As a user, I want to connect to a database, so that I can manage data efficiently; and

As a user, I want a login page, so that I can securely access my account”.

Examples of the plurality of reference descriptors of the first reference template 32 can further comprise user journey statements such as:

- 5 “User opens the app and gets greeted with the splash page;
 After a while, user is lead to the login page; and
 User successfully enters login credentials and log in”.

An example of the user requirements, which forms the user input 28, comprises:

- 10 “Build a website regarding musical instruments. Include only string, percussion and
 wind types”.

Examples of the plurality of output requirements of the first artifacts 34 comprises business requirements such as:

- 15 “Introduce users to the website with an engaging landing page that highlights the
 purpose and variety of musical instruments available;
 Enable efficient browsing and filtering options for different instrument types to
 enhance user experience;
 Ensure secure user authentication and account access with a reliable login system.”.

20

Examples of the plurality of output descriptors of the first artifacts 34 comprises user stories such as:

- “As a user, I want to see a captivating landing page showcasing a variety of musical
 instruments;
 25 As a user, I want to easily browse and filter instruments by string, percussion and
 wind categories; and
 As a user, I want a secure login page to access my account and preferences”.

- Examples of the plurality of output descriptors of the first artifacts 34 can further comprise
 30 user journey statements such as:

- “User visits the website and is greeted with a captivating landing page showcasing various musical instruments;
User navigates through different instrument categories, filtering by string, percussion and wind types; and
5 User logs in securely to access their account and personalized preferences.”.

Preferably, the reference requirements of the first reference template 32 is derived from the reference descriptors, while the reference input is derived from one or both of the reference requirements and the reference descriptors. In various implementations, this is done by
10 reverse-prompting where in a process of building the reference template 32, the reference descriptors, for example in the form of user stories and user journey statements, are provided as prompts to a language model such as the first language model 30, to derive the reference requirements. The reference requirements, for example in the form of user requirements, are then used to derive the reference input.

15 Further, each of the reference input and the user input 28 comprises at least one of a project descriptor, project requirements and project details. An example of the project descriptor comprises “Build a complete field sales application”. An example of the project requirements comprises “Develop a comprehensive field sales application to streamline
20 sales process and enhance efficiency of field sales”. Examples of the project details comprises: “Create an engaging user experience”, “Improve accessibility, be responsive”, and “More beautiful UI”.

Additionally or alternatively, the user input 28 can further comprise data that may be
25 relevant, in part or in whole, be relevant only to subsequent language modules or LLM agents. Therefore, the user input 28 can further comprise API documentation that may be relevant for subsequent code generation steps and data provided as answers and comments to a questionnaire. Examples of data provided as answers and comments to a questionnaire are:

30

1. General Information

- *Name/Organization:* Field Sales App
- *Primary Objectives or Use Cases:*
 - To assist sales representatives in efficiently managing sales activities directly in the field.
 - Facilitate order creation, management of returns, handling collections, and access to invoices and sales reports.
- *Target Users:* Field sales representatives.
- *Primary Goal:* To streamline field sales operations through a comprehensive mobile application integrated with BC365.

2. Functional Requirements

- *Core Features:*
 - Order creation, returns management, collections, accessing invoices, and generating sales reports.
- *Additional Functional Requirements:*
 - High responsiveness with seamless BC365 backend integration.
 - User-friendly interface designed for field use.

3. User Management

- *Multi-user Application:* Yes, designed for multiple user roles within the sales team.
- *User Login Required:* Yes, secure authentication via backend system credentials.

4. Navigation and User Experience

- *Preferred Navigation Structure:* Main menu with icon-based navigation for quick access to all functionalities.
- *Preferred Layout Style:* User-friendly and minimalistic to facilitate quick operations.
- *Other Requests for Navigation and Layout:* Responsive design to ensure usability across various devices and screen sizes.

5. Page Structure

Splash Page

- *Overview:* Initial loading page which determines the next navigation step based on settings.
- *Functional Logic:*
 - Checks the 'Setting' table for existing configurations and navigates accordingly to either the setting page or the login page.
- *Key Elements:* App logo, possibly a loading indicator.

Menu Page

- *Overview:* Main navigation hub displaying various functionality icons and user details.
- *Functional Logic:*

- Simple navigational clicks to different functional areas like settings, customers, orders, etc.
- *Key Elements:* Icons for settings, connection, logout, customers, orders, returns, collections, invoices, reports.

5

Customer List Page

- *Overview:* Displays a searchable list of customers with details such as code, name, address, and balance.
- *Web Services:* "Customer List" API for fetching customer data.
- *Tables Used:* BC – Customer list BC table
- *Functional Logic:*
 - Retrieves customers upon page load.
 - Provides a search functionality to filter customers by name or code.
- *Key Elements:* Search bar, list of customers, navigation popup on customer selection.

10

15

Cart Page

- *Overview:* Manages and displays items added to the shopping cart.
- *Functional Logic:*
 - Displays items from a local 'Cart' table.
 - Allows modification or deletion of cart items.
- *Key Elements:* Item details (quantity, price, discount, total), delete all items option, individual item removal, add more items button, checkout navigation.

20

25

7. Functional Logic

- *Functional Modules on Each Page:*
 - Detailed functional logic for user authentication, customer and item selection, order processing, and collections management.
- *Interactive or Dynamic Features:*
 - Real-time interactions with BC365 via APIs for immediate updates and validations.

30

8. Backend Integration

- *Integration with Third-party Services:* Directly integrates with BC365 using specific APIs for each functionality.
- *API Endpoint Requirements:* Outlined per page, detailing interactions like data retrieval, user authentication, and order processing.

35

9. Security

- *Important Security Measures:*
 - Use of HTTPS for secure data transmission.
 - Data encryption and secure handling of sensitive information.”

40

At least a portion of the first artifacts 34 forms a portion of the developmental requirements 21. In a step 113a, the first artifacts 34 are further processable by at least one additional language model 36 using at least one additional reference template 38 for generating additional artifacts 40. The first artifacts 34 and the additional artifacts 40 constitute the developmental requirements 21 to be processed by a code language model 42 for generating at least a portion of codes 43 of the software application therefrom in a step 113b. When viewed from an LLM agent implementation perspective, the additional artifacts 40 are generated by another LLM agent comprising the additional language model 36 and the additional reference template 38, while the at least a portion of codes 43 of the software application is generated by a further LLM agent comprising the code language model 42.

Each of the first language model 30 and the additional language model 36 adopts a corresponding one of a plurality of software development roles. Each of the plurality of software development roles having role-specific objectives, capabilities and behavioral pattern. In some implementations, the first artifacts 34 are processed directly by the code language model 42, without further processing by the additional language model 36, for generating the at least a portion of codes 43 of the software application therefrom. In other implementations, the first artifacts 34 are processed by more than one of the additional language model 36, each adopting a unique one of the plurality of software development roles, before being eventually processed by the code language model 42 for generating the at least a portion of codes 43 of the software application therefrom.

The method 100 further comprises a step 114 of generating output analysis of the first artifacts 34 by the first language model 30 based on pre-defined descriptors associated with one of the plurality of software development roles. The plurality of software development roles can comprise a business analyst role, a technical lead role and a project manager role.

In addition, the method 100 further comprises a step 116 of creating an additional software development role with additional descriptors associated therewith based one of the user input 28 and the first artifacts 34. In some implementations, the additional software

development roles and the associated additional descriptors can go beyond the defined business analyst role, technical lead role and project manager role based on needs and/or deficiencies in workflow as well as the sufficiency, suitability and quality of one or more of the first artifacts 34 and the additional artifacts 40. This is akin to creating new LLM agents and associated agent roles. Additionally, the creation of the additional software development roles can be based on the outcome of the output analysis of the first artifacts 34 in the step 114.

The first reference template 32 further comprises reference explainers for arriving at the plurality of reference requirements and the first artifacts 34 further comprising output explainers for arriving at the plurality of output requirements. The output explainers enable complex reasoning capabilities through intermediate reasoning steps. This approach is also often adopted in chain-of-thought prompting.

The method 100 further comprises a step 120 of presenting at least a portion of the first artifacts 34 for review by a user, a step 122 of receiving feedback input from the user, and a step 124 of processing the user input 28 based on the feedback input by the first language model 30 for updating the first artifacts 34 generated thereby. Alternatively, the step 120 of presenting at least a portion of the first artifacts 34 for review by a user and the step 122 of receiving feedback input from the user 34 may be performed by a standalone review language model, constituting a review LLM agent, in a step 124. The review language model learns the developmental and review preferences of the user and performs review of the first artifacts 34 as a proxy of the user. The review LLM agent can be pre-configured with structured or unstructured rule-based constraints, have review requirements generated based on the needs or demands of the review, or be based on review requirements further constituting the reference descriptors of the first reference template 32 or a reference template pre-defined or generated for the review LLM agent.

When utilizing the additional language model 36, the method 100 further comprises a step 130 of presenting at least a portion of the additional artifacts 40 for review by a user, a step

132 of receiving feedback input from the user, and a step 134 of processing the first artifacts 34 based on the feedback input by the additional language model 36 for updating the additional artifacts 40 generated thereby. Alternatively, the step 130 of presenting at least a portion of the additional artifacts 40 for review by a user and the step 132 of receiving
 5 feedback input from the user 34 may be performed by a standalone review language model 50, constituting a review LLM agent, in a step 134. Similar to the step 124, the review language model 50 learns the developmental and review preferences of the user and performs review of the additional artifacts 40 as a proxy of the user. The review LLM agent can be pre-configured with structured or unstructured rule-based constraints, have review
 10 requirements generated based on the needs or demands of the review, or be based on review requirements further constituting the reference descriptors of the additional reference template 38 or a reference template pre-defined or generated for the review LLM agent.

Further, in a corresponding one of a step 140 and a step 142, each of the first language
 15 model 30 and the additional language model 36 checks for data sufficiency of and normalizes the semantics of the respective the user input 28 and the first artifacts 34 to thereby generate the corresponding one of the first artifacts 34 and the additional artifacts 40.

Each of the output requirements of at least one of the first artifacts 34 and the additional
 20 artifacts 40, and the reference requirements comprising a class diagram being descriptive of class attributes comprising data structures and interfaces. The class diagram can further comprise instructions on the data structure, the interfaces and the implementations, for example, “annotations, clearly mark the relationships between classes” and “Use mermaid classDiagram code syntax”. The class diagram may further define constraints based on pre-
 25 defined plurality of data items. 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. Each of the output descriptors of at least one of the first artifacts 34 and the additional artifacts 40, and the reference descriptors comprising a sequence diagram being descriptive of a call flow graph. The sequence diagram can further comprise
 30 instructions, for example, “Ensure clarity and adherence to PEP8 standards”.

The first artifacts 34 and the additional artifacts 40 are preferably generated as JSON formatted data items for provision to the code language model 42.

- 5 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 receiving user input 28 comprising input user requirements for a software application for meeting a business objective, developmental requirements 21 for the software application to be generatable from the user input 28, processing the user input
- 10 28 by a first language model 30 using a first reference template 32 for generating first artifacts 34 therefrom, the first reference template 32 comprising reference input, a plurality of reference requirements and reference descriptors and the first artifacts 34 comprising output requirements and output descriptors substantially correlated with the reference requirements and reference descriptors. Wherein at least a portion of the first artifacts 34
- 15 forming a portion of the developmental requirements 21, the first artifacts 34 are further processable by at least one additional language model 36 using at least one additional reference template 38 for generating additional artifacts 40, the first artifacts 34 and the additional artifacts 40 constituting the developmental requirements 21 to be processed by a code language model 42 for generating at least a portion of codes 43 of the software
- 20 application therefrom, each of the first language model 30 and the additional language model 36 adopting a corresponding one of a plurality of software development roles, each of the plurality of software development roles having role-specific objectives, capabilities and behavioral pattern.
- 25 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 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
- 30 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 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
5 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 generating of developmental requirements for a software application for achieving a business objective comprising:

5 receiving user input comprising input user requirements for meeting the business objective; and

processing the user input by a first language model using a first reference template for generating first artifacts therefrom, the first reference template comprising reference input, a plurality of reference requirements and reference descriptors and the first artifacts comprising output requirements and output descriptors substantially correlated with the reference requirements and reference descriptors,

10

wherein at least a portion of the first artifacts forming a portion of the developmental requirements, the first artifacts are further processable by at least one additional language model using at least one additional reference template for generating additional artifacts, the first artifacts and the additional artifacts constituting the developmental requirements to be processed by a code language model for generating at least a portion of codes of the software application therefrom, each of the first language model and the additional language model adopting a corresponding one of a plurality of software development roles, each of the plurality of software development roles having role-specific objectives, capabilities and behavioral pattern.

15

20

2. The method as in claim 1, further comprising:

generating output analysis of the first artifacts by the first language model based on pre-defined descriptors associated with one of the plurality of software development roles.

25

3. The method as in claim 2, further comprising:

creating an additional software development role with additional descriptors associated therewith based one of the user input and the first artifacts.

30

4. The method as in claim 1, the first reference template further comprising reference explainers for arriving at the plurality of reference requirements and the first artifacts further comprising output explainers for arriving at the plurality of output requirements.
5
5. The method as in claim 1, further comprising:
 presenting at least a portion of the first artifacts for review by a user;
 receiving feedback input from the user; and
10 processing the user input based on the feedback input by the first language model for updating the first artifacts generated thereby.
6. The method as in claim 1, the plurality of software development roles comprising a business analyst role, a technical lead role and a project manager role.
15
7. The method as in claim 1, the reference requirements being derived from the reference descriptors, and the reference input being derived from one or both of the reference requirements and the reference descriptors.
- 20 8. The method as in claim 1, each of the reference input and the user input comprises at least one of a project descriptor, project requirements and project details.
9. The method as in claim 8, each of the output requirements and the reference requirements comprising a plurality of business requirements, and each of the output
25 descriptors and the reference descriptors comprising at least one of a plurality of user journeys and a plurality of user stories.
10. The method as in claim 9, each of the output descriptors and the reference descriptors further comprises a project descriptor, project requirements, project details and project
30 concerns.

11. The method as in claim 1, each of the first language model and the additional language model checks for data sufficiency of and normalizes the semantics of the respective the user input and the first artifacts to thereby generate the corresponding one of the first artifacts and the additional artifacts.

12. The method as in claim 1, each of the output requirements and the reference requirements comprising a class diagram being descriptive of class attributes comprising data structures and interfaces; and each of the output descriptors and the reference descriptors comprising a sequence diagram being descriptive of a call flow graph.

13. A system for use in managing generating of developmental requirements for a software application for achieving a business objective, the system comprising:

cloud-based storage circuitry storing a first language model and a first reference template;

cloud-based control circuitry configured to:

receive user input comprising input user requirements for meeting the business objective; and

process the user input by the first language model using the first reference template for generating first artifacts therefrom, the first reference template comprising reference input, a plurality of reference requirements and reference descriptors and the first artifacts comprising output requirements and output descriptors substantially correlated with the reference requirements and reference descriptors; and

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

present at least a portion of the first artifacts on the UI for review by a user;

receive feedback input from the user via the UI; and

process the user input based on the feedback input by the first language model for updating the first artifacts generated thereby,

wherein at least a portion of the first artifacts forming a portion of the developmental requirements, the first artifacts are further processable by at least one additional language model using at least one additional reference template for generating additional artifacts, the first artifacts and the additional artifacts constituting the developmental requirements to be processed by a code language model for generating at least a portion of codes of the software application therefrom, each of the first language model and the additional language model adopting a corresponding one of a plurality of software development roles, each of the plurality of software development roles having role-specific objectives, capabilities and behavioral pattern.

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

receiving user input comprising input user requirements for a software application for meeting a business objective, developmental requirements for the software application to be generatable from the user input; and

processing the user input by a first language model using a first reference template for generating first artifacts therefrom, the first reference template comprising reference input, a plurality of reference requirements and reference descriptors and the first artifacts comprising output requirements and output descriptors substantially correlated with the reference requirements and reference descriptors,

wherein at least a portion of the first artifacts forming a portion of the developmental requirements, the first artifacts are further processable by at least one additional language model using at least one additional reference template for generating additional artifacts, the first artifacts and the additional artifacts constituting the developmental requirements to be processed by a code language model for generating at least a portion of codes of the software application therefrom,

each of the first language model and the additional language model adopting a corresponding one of a plurality of software development roles, each of the plurality of software development roles having role-specific objectives, capabilities and behavioral pattern.

5

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

generating output analysis of the first artifacts by the first language model based on pre-defined descriptors associated with one of the plurality of software development roles; and

10

creating an additional software development role with additional descriptors associated therewith based one of the user input and the first artifacts,

wherein the first reference template further comprising reference explainers for arriving at the plurality of reference requirements and the first artifacts further comprising output explainers for arriving at the plurality of output requirements.

15

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

presenting at least a portion of the first artifacts for review by a user;

20

receiving feedback input from the user; and

processing the user input based on the feedback input by the first language model for updating the first artifacts generated thereby.

25

17. The non-transitory computer readable medium of claim 14, the plurality of software development roles comprising a business analyst role, a technical lead role and a project manager role,

wherein the reference requirements being derived from the reference descriptors, and the reference input being derived from one or both of the reference requirements and the reference descriptors.

30

18. The non-transitory computer readable medium of claim 14, each of the reference input and the user input comprises at least one of a project descriptor, project requirements and project details, each of the output requirements and the reference requirements comprising a plurality of business requirements, each of the output descriptors and the reference descriptors comprising at least one of a plurality of user journeys and a plurality of user stories, and each of the output descriptors and the reference descriptors further comprises a project descriptor, project requirements, project details and project concerns.

19. The non-transitory computer readable medium of claim 14, each of the first language model and the additional language model checks for data sufficiency of and normalizes the semantics of the respective the user input and the first artifacts to thereby generate the corresponding one of the first artifacts and the additional artifacts, and each of the output requirements and the reference requirements comprising a class diagram being descriptive of class attributes comprising data structures and interfaces; and each of the output descriptors and the reference descriptors comprising a sequence diagram being descriptive of a call flow graph.