

SOFTWARE APPLICATION DEVELOPMENT METHOD

TECHNICAL FIELD

5 This invention relates generally to a computer-implemented method for software application development.

Background

Development of software applications take time and know-how. When one is unfamiliar with coding approaches and workflow, development of software applications will may be a
10 daunting task and takes away from the focus of the functional requirements of the software application, limiting the developer to only what his skills are capable of. This is exacerbated by the need to understand efficacy of different programming framework catering to different needs and requirements of the eventual software application based on the platform it is intended to operate on. For example, we might find ourselves not able to share React Native
15 code when targeting different platforms. This happens when we would rather have user interface elements behave natively to their specific platform, in turn giving the user a better experience and a more native feel to the app. It is not uncommon to start developing a pure web application, later bundle it as a Cordova application, and reuse as much code as possible for all the (mobile) platforms we want to target. However, unfamiliarity with the
20 myriad of programming frameworks available makes such transitions, from one platform to another, difficult. There is therefore a need for an improved software application development method and system/platform.

25 Summary

In accordance with a first aspect of the invention, there is disclosed a software application development method comprising receiving user input from a user being indicative of modification to a schema of a target software application defined by a first set of code, the first set of code being based on a source programming framework, modifying the first set of
30 code based on the received modification to the schema of the target software application,

and converting the first set of code into the target software application by a selected one of at least one converter, the selected one of the at least one converter being associated with the one of at least one target programming framework selected by the user, the target software application based on the one of at least one target programming framework selected by the user.

In accordance with a second aspect of the invention, there is disclosed machine-readable medium having stored therein a plurality of programming instructions, which when executed, the instructions cause the machine to receive user input from a user being indicative of modification to a schema of a target software application defined by a first set of code, the first set of code being based on a source programming framework, modify the first set of code based on the received modification to the schema of the target software application, and convert the first set of code into the target software application by a selected one of at least one converter, the selected one of the at least one converter being associated with the one of at least one target programming framework selected by the user, the target software application based on the one of at least one target programming framework selected by the user.

Brief Description of the Drawings

FIG. 1 shows a system diagram of a software application development method in accordance with an aspect of the invention;

FIG. 2 shows a process flow diagram of a software application development system in accordance with an aspect of the invention for implementing the software application development method of FIG. 1;

FIG. 3 shows an exemplary graphical representation 46 of a schema of a target software application presented on the GUI 50 of the software application development system 20 of FIG. 2;

FIG. 4 shows an exemplary graphical representation 46 of a schema of a target software application presented on the GUI 50 of the software application development system 20 of FIG. 2 with functionality of the target software application being previewable in a preview window;

5

FIG. 5 shows an exemplary graphical representation 46 of a schema of a target software application presented on the GUI 50 of the software application development system 20 of FIG. 2 with a build window for selection of a converter to generate a second set of code in a corresponding one of the at least one target programming framework; and

10

FIG. 6 shows an exemplary graphical representation 46 of a schema of a target software application presented on the GUI 50 of the software application development system 20 of FIG. 2 with the target software application 36, in the target programming framework, being presented for use in a program execution window.

15

Detailed Description

An exemplary embodiment of the present invention, a software application development method 100 (“development method 100”) utilising a software application development system 20 (“development system 20”), is described hereinafter with reference to FIG. 1 to FIG. 6.

The development method 100 comprises receiving user input 32 from a user being indicative of modification to a schema 34 of a target software application 36 defined by a first set of code 38 in a step 120. The first set of code 38 is based on a source programming framework 40. The development method 20 further comprises modifying the first set of code 38 based on the received modification to the schema 34 of the target software application 36 in a step 122, and converting the first set of code 38 into the target software application 36 by a selected one of at least one converter 41 in a step 124. The selected one of the at least one converter 41 is associated with the one of at least one target programming framework 42 selected by the user with the target software application 36 being based on the one of at least one target programming framework 42 selected by the user.

The target software application 36 is for running and operating on electronic devices which may be a mobile device such as a smartphone or a tablet, with a touch panel. This is in no way limiting as the present teaching would work for any electronic devices such as lap tops, desktops and the likes, using user input provided for instance with a mouse.

The schema 34 can comprise a plan, a model and cognitive structures of the target software application 36 which may comprise conceptual, physical and external schemas and further define or abstract interactions, including user interaction, with the target software application 36.

The first set of code 38 can be in the form of a single segment or be segregated into multiple segments, for example, contained in multiple files in accordance to the requirement of the source programming framework 40. When segregated into multiple segments, one or more

of the segments of the first set of code 48 may be compiled with the other segments containing textual codes.

5 Preferably, the step 124 of converting the first set of code 38 into the target software application 36 comprises translating the first set of code 38 into a second set of code 44 associated with the selected one of at least one target programming framework 42 in a step 130 and incorporating operating components 45 with the second set of code 44 based on operating requirements of the second set of code 44 in a step 132 to obtain the target software application 36.

10

Similar to the first set of code 38, the second set of code 44 can be in the form of a single segment or be segregated into multiple segments, for example, contained in multiple files in accordance to the requirement of the target programming framework 42. When segregated into multiple segments, one or more of the segments of the second set of code 44 may be
15 compiled with the other segments containing textual codes.

Preferably, the source programming framework 40 is one of JSON, UML and XML, and the at least one target programming framework 42 comprises Cordova, React, React Native, Ionic, Flutter, Java and Javascript.

20

Therefore, in some instances, the step 124 of converting the first set of code 38 into the target software application 36 further comprises compiling at least a portion of the second set of code 44 into machine code. Further, in other instances, the first set of code 38 and the second set of code 44, or corresponding segments between the first set of code 38 and the
25 second set of code 44, have differing syntax. This being said, the target software application 36 can be a web application, a native application or a hybrid application.

In such an instance where the target programming framework 44 is, for example, React Native, the corresponding at least one converter 41 will take advantage of React Native's

features, for example its own packaging and transpiling capability, without need for reconfiguring required tools for the second set of code 42.

5 The step 132 of incorporating operating components 45 with the second set of code 44 comprises identifying at least one component, plugin, typography, dependency, mobile theme and library requirements from the first set of code 38 in a step 140, and incorporating the identified at least one component, plugin, typography, dependency, mobile theme and library with the second set of code 44 in a step 142. Preferably, the library comprises native components for at least one of iOS and Android platforms, interactive paradigms and UI
10 components.

Having a software application that it isn't able to adapt will take users out of the experience. Hence, incorporating, for example, Cordova plugins or Microsoft's Ace library that connect us to the native side of things is helpful when once the required operating components 45
15 are identified.

A module on the iOS platform may be created using Objective-C with the same to be said for creating a module on the Android platform using Java. However, there is a need to link native plugins within the project files for each platform. With iOS, for example, this means
20 we must link the compiled native part with our target software application 36 and add the corresponding header files as part of the operating components 45 required. This can be a long process, especially if there are a lot of native modules. Fortunately, this is significantly simplified via pre-configured at least one converter 41 that enables utilization of the features in the form of components of the respective target programming framework 42. For
25 example with React Native, we have access to native controls and interaction out of the box. Components such as Text, TextInput or Slider map to its native counterparts. While some components are available for all platforms, other components only work on specific platforms. The closer we want our application to have a native look and feel, the more we need to use components that are only available for this specific platform and so the more our
30 codebase diverges. The selection and use of the at least one converter 41 for identifying

requirements for and mapping of operating component 45, which may be platform specific, enables the use of an initial first set of code 38 in the source programming framework 40 to operate close to platform requirement based on the features of the target programming framework 44 without the user having to worry about having differing codebase.

- 5 Mind touch interactions and gestures are part of React Native as well.

By utilizing the at least one converter 41, the features of target programming frameworks 44 such as Cordova and React Native, for example the ability to create and use plugins that connect to the native side of things, can be utilised.

10

The step 120 of receiving user input 32 from a user being indicative of modification to a schema 34 of a target software application 36 comprises receiving at least one of a drag input, a selection input and a data input modification to a graphical representation 46 representative of the schema of the target software application 36 through an interface of a computing device. Preferably, the graphical representation 46 of the schema 34 of the target software application 36 being presented as a plurality of objects 48 presented on a graphical user interface 50 (“GUI 50”) wherein the relations and properties 52, of and between the plurality of objects 48 are further presented by and modifiable through the GUI 50.

15

- 20 The software application development system 20 and the software application development method 100 may be used on a device, for example a laptop or a desktop computer having the GUI 50 for a user, for example a software developer, to interact with the software application development system 20.

- 25 The GUI 50 comprises a screen, which may be a touch-screen for receiving touch input in addition to the keyboard, keypad, track pad or mouse of the device, for rendering visual and textual images which describe various visual metaphors, for example the schema 34 of the target software application 36

Furthermore, GUIs can represent programs, files and operational functions with graphical images, objects, or vector representations. The graphical images can include windows, fields, dialog boxes, menus, icons, buttons, cursors, scroll bars, maps, etc. Such images can be arranged in predefined layouts, or can be created dynamically (by the device itself or by a web-based server as mentioned before) to serve the specific actions being taken by a user. In general, the user can select and/or activate various graphical images in order to initiate functions and tasks, i.e. controls, associated therewith.

Additionally, the graphical representation 46 is further indicative and descriptive of the functional aspect and the datalogical aspect of the target software application 36. The development method 100 further comprises capturing infological and essential information being descriptive of the user input 22 in a step 126 with the essential information comprising a user identifier and date and time of the user input.

The functional aspect of the target software application 36 not only models the deployed functionalities of the target software application, but may also abstract a class diagram to model a solution, classes, interfaces, properties and operations. The functional aspect of the target software application 36 aims to classify and associate classes into Essential interfaces, Infological interfaces and Datalogical interfaces. This allows the respective at least one converter 41 to transform the first set of code 38 in the source programming framework 40, for example a UML model, into the target programming framework 44. The Essential, infological and datalogical data, descriptors and interfaces functions as a guide and reference for enabling translation or conversion between the source programming framework 40 and the target programming framework 42 by the selected converter 41.

In the case of use of the UML model, the UML model is responsible for populating dependencies according to relationships usage and association at each particular class. By applying the respective converter 41 this pseudocode, we take parameters, operations and attributes to generate a structure composed by the attributes and pointers at the associated classes.

The Essential interface comprises interfaces for enabling the relevant mobile platform to receive the final build phase of target software application 36, enabling the process for generating the final the quantity of code, for example the second set of code 44, for the target software application 36, enabling the transmission process of property events between devices and device owners, and enabling acceptance of ownership of a mobile project, for example the project for developing the target software application 36, from the address of a given point of origin. To order and catch the pre/post state of an address to real-time, we separate the transition between its origin and destination.

The context of the target software application 36 is constructed at the datalogic level where it is created, operated and mapped to a targeted mobile app platform. Hence, the datalogic is a profile that provides a standardized extension mechanism for customizing the target programming framework 42, for example, UML models. The datalogics, which enables indication by the datalogic interface for conversion or translation by the selected at least one converter 41 from the source programming framework 40 to the target programming framework 42, comprises the following:

- Operation interfaces: Applicable interfaces with this element are:(1) Event, (2) Function, (3) Modifier and (4) Constructor interfaces. All these interfaces relate to the target programming language, for example the second set of code 44, that orchestrate, manage, and manipulate the mobile app code, for example the target software application 36, in the targeted mobile app platform.
- Class model element manages objects and features classifications as formal template parameter. Applicable interfaces are: (1) Struct, (2) Contract and (3) Library. These interfaces are characterized from its mobile application target language.
- Constraint element to manage the semantics of elements by applying ParameterModifier interface.

- Enumeration element import all non-private members belonging to a package from the target mobile app platform by applying FunctionType interface
- PackageImport element connects with different kind of packages by applying import interface and importFilter tagged values.
- 5 • Parameter element passes information from/to invocation of behavioralFeature by applying the indexed interface.
- Property element changes the semantics of how the property behaves. It also exposes by overriding the getter/setter methods. By applying Getters and setters interface with property element, we can allow different access levels of public properties while still being protected.
- 10 • ValueMapping element represents key and value from the various target platform by applying Mapping and KeyMapping interfaces.
- AppInterface element connects with APIs of mobile app platforms (i.e Android API, iOS API) by applying the DatalogicalI interface.
- 15 • StandardizedLogicAPI element manage the semantics of API by generalizing AppInterface interface.
- StandardizedIOSAPI element manage the semantics of iOS standard API by applying StandardIOSModifier interface.
- StandardizediAndroidAPI element manage the semantics of elements by applying
- 20 StandardAndroidModifier interface.
- VisualComponent element manage the semantics of elements by applying VisualComponentModifier interface.
- Container element manage the semantics of mobile app container by applying ContainerParameterModifier interface.
- 25 • Dataset element manage the semantics of mobile app dataset by applying DatasetParameterModifier interface.

Preferably, the step 124 of converting the first set of code 38 into the target software
 30 application 36 comprises establishing data transfer connection between the target software

application 36 and server processing resources defined in the second set of code 44. The server processing resources defines how server resources are managed, allocated and consumed by the target software application 36.

- 5 The software application development method 100 can be further implemented as a machine-readable medium having stored therein a plurality of programming instructions, which when executed, the instructions cause the machine to receive user input from a user being indicative of modification to a schema of a target software application defined by a first set of code, the first set of code being based on a source programming framework,
- 10 modify the first set of code based on the received modification to the schema of the target software application, and convert the first set of code into the target software application by a selected one of at least one converter, the selected one of the at least one converter being associated with the one of at least one target programming framework selected by the user, the target software application based on the one of at least one target programming
- 15 framework selected by the user.

- In an example of implementation of the software application development method 100, FIG. 3 shows an exemplary graphical representation 46 of the schema of the target software application 36 presented on the GUI 50 of the software application development system 20.
- 20 The GUI 50 shows a dialog box 60 being added to the graphical representation 46 with the dialog box 60 including a button component 62 to be presented as part of the target software application 36. The relations and properties 52 of the button component 62, constituting the plurality of objects 48 presented on the GUI 50 is editable through a properties window 64 presented on the GUI 50. Through the user input 32 via the GUI 50, the graphical
- 25 representation 46 and the relations and properties of the plurality of objects 48, including that of the button component 62, may be edited and modified while keeping the generated first set of code 38 in the source programming framework 40. The functionality of the target software application 36, which still in the source programming framework 40, may be previewed in an exemplary preview window 66, as shown in FIG. 4, to assess functionality
- 30 thereof.

Selection of one of the at least one converter 41 to generate the second set of code 44 in the corresponding one of the at least one target programming framework 42, including the defining and incorporating operating components 45, can be done via an exemplary build window 68 as shown in FIG. 5. The first set of code 38 in the source programming framework 40 is then converted into the second set of code 44 in the target programming framework 42 to thereby obtain the target software application 36 which is presented for use in a program execution window 70 as shown in FIG. 6.

Aspects of particular embodiments of the present disclosure address at least one aspect, problem, limitation, and/or disadvantage associated with existing software application development 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 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 ordinary skill in the art within the scope of the present disclosure, which is limited only by the following claims.

Claims

1. A software application development method comprising:
 - receiving user input from a user being indicative of modification to a schema of a target software application defined by a first set of code, the first set of code being
 - 5 based on a source programming framework;
 - modifying the first set of code based on the received modification to the schema of the target software application; and
 - converting the first set of code into the target software application by a selected one of at least one converter, the selected one of the at least one converter being
 - 10 associated with the one of at least one target programming framework selected by the user, the target software application based on the one of at least one target programming framework selected by the user.
2. The software application development method as in claim 1, converting the first set of
- 15 code into a target software application comprising:
 - translating the first set of code into a second set of code associated with the selected one of at least one target programming framework; and
 - incorporating operating components with the second set of code based on operating requirements of the second set of code to obtain the target software
 - 20 application.
3. The software application development method as in claim 2, the first set of code and the second set of code having differing syntax.
- 25 4. The software application development method as in claim 2, converting the first set of code into a target software application further comprising:
 - compiling at least a portion of the second set of code into machine code.
5. The software application development method as in claim 2, incorporating operating
- 30 components with the second set of code comprising:

identifying at least one component, plugin, typography, dependency, mobile theme and library requirements from the first set of code; and

incorporating the identified at least one component, plugin, typography, dependency, mobile theme and library with the second set of code.

5

6. The software application development method as in claim 4, the library comprises native components for at least one of iOS and Android platforms, interactive paradigms and UI components.

10 7. The method as in claim 1, the source programming framework being one of JSON, UML and XML, and the at least one target programming framework comprising cordova, react, react native, ionic and flutter.

15 8. The software application development method as in claim 1, receiving user input from a user being indicative of modification to a schema of a target software application:
receiving at least one of a drag input, a selection input and a data input modification to a graphical representation representative of the schema of the target software application through an interface of a computing device.

20 9. The software application development method as in claim 8, the graphical representation of the schema of the target software application being presented as a plurality of objects on a graphical user interface (GUI) wherein the relations between the plurality of objects are further presented by and modifiable through the GUI.

25 10. The software application development method as in claim 8, the graphical representation further being functionally and datalogically indicative of the target software application.

11. The software application development method as in claim 8, further comprising:

capturing infological and essential information being descriptive of the user input, the essential information comprising a user identifier and date and time of the user input.

- 5 12. The software application development method as in claim 1, converting the first set of code into the target software application by a selected one of at least one converter comprising:

establishing data transfer connection between the target software application and server processing resources defined in the second set of code.

10

13. A machine-readable medium having stored therein a plurality of programming instructions, which when executed, the instructions cause the machine to:

receive user input from a user being indicative of modification to a schema of a target software application defined by a first set of code, the first set of code being based on a source programming framework;

15

modify the first set of code based on the received modification to the schema of the target software application; and

convert the first set of code into the target software application by a selected one of at least one converter, the selected one of the at least one converter being associated with the one of at least one target programming framework selected by the user, the target software application based on the one of at least one target programming framework selected by the user.

20

14. The machine-readable medium as in claim 13, wherein the plurality of programming instructions, which when executed, the instructions further cause the machine to:

25

translate the first set of code into a second set of code associated with the selected one of at least one target programming framework;

incorporate operating components with the second set of code based on operating requirements of the second set of code to obtain the target software application;

30

identify at least one component, plugin, typography, dependency, mobile theme and library requirements from the first set of code; and

incorporate the identified at least one component, plugin, typography, dependency, mobile theme and library with the second set of code,

5 wherein the library comprises native components for at least one of iOS and Android platforms, interactive paradigms and UI components, the source programming framework being one of JSON, UML and XML, and the at least one target programming framework comprising cordova, react, react native, ionic and flutter.

10

15. The machine-readable medium as in claim 14, wherein the plurality of programming instructions, which when executed, the instructions further cause the machine to:

receive at least one of a drag input, a selection input and a data input modification to a graphical representation representative of the schema of the target software application through an interface of a computing device,

15

wherein the graphical representation of the schema of the target software application being presented as a plurality of objects on a graphical user interface (GUI) wherein the relations between the plurality of objects are further presented by and modifiable through the GUI, and the graphical representation further being functionally and datalogically indicative of the target software application.

20

25

30