

TITLE OF INVENTION: SOFTWARE MULTILINGUALIZATION METHOD, DEVICE, EQUIPMENT, STORAGE MEDIUM AND PRODUCT

TECHNICAL FIELD

[0001] The present application relates to the technical field of software, and in particular to a software multilingualization method, a software multilingualization device, a software multilingualization equipment, a storage medium and a product.

BACKGROUND

[0002] With the acceleration of the globalization process, the demand for internationalization of software products is growing. The traditional software internationalization process (that is, translating the text content in the software product into foreign language types) often relies on a large number of code modifications, that is, modifying the initial text content in the software into the translated content, but this method is time-consuming and labor-intensive, and the efficiency of software internationalization is low, which prolongs the cycle of software internationalization.

[0003] The above content is only used to assist in understanding the technical solution of the present application, and does not mean that the above content is recognized as prior art.

SUMMARY

[0004] The main purpose of the present application is to provide a software multilingualization method, a software multilingualization device, a software multilingualization equipment, a storage medium and a product, aiming to solve the technical problem of low efficiency of the current software internationalization method.

[0005] To achieve the above purpose, the present application proposes a software multilingualization method including:

[0006] constructing a design-time interactive interface of a software in response to a user's interface construction operation, and generating a design-time metadata set of the design-time interactive interface based on an interface element on the design-time interactive interface;

[0007] obtaining a target language text data packet of the software based on a target language type of the software; and

[0008] generating a runtime interactive interface corresponding to the design-time interactive interface based on the design-time metadata set and the target language text data packet in response to a user's interface access operation.

[0009] In an embodiment, before the obtaining the target language text data packet of the software based on the target language type of the software, the method includes:

[0010] inputting a preset translation task prompt template and an original language type text data packet of the software into a preset large language model, so as to guide the preset large language model to translate the original language type text data packet into the target language text data packet conforming to a target language type by the translation task prompt template; the translation task prompt template includes software industry prompt words and translation consistency prompt words, the original language type text data packet at least includes a first original language type text data of a software corresponding interface element, and the preset large language model is obtained by fine-tuning an initial large language model by a preset industry terminology library.

[0011] In an embodiment, after the inputting the preset translation task prompt template and the original language type text data packet of the software into the preset large language model, so as to guide the preset large language model to translate the original language type text data packet into the target language text data packet conforming to the target language type, the method further includes:

[0012] for any original text segment in the original language type text data packet, determining an actual translation result of the preset keyword in the original text segment corresponding to a target language text segment in response to that a preset keyword exists in the original text segment; the preset keyword is a professional term in an original language type in the preset industry terminology library; and

[0013] outputting a prompt message associated with the preset keyword, or using a standard translation result to replace the actual translation result in response to that the actual translation result is different from the standard translation result of the preset keyword in the preset industry terminology library.

[0014] In an embodiment, the original language type text data packet further includes a second original language type text data of a software corresponding supporting resources, and the software corresponding supporting resources include a initialization script of the software, a workflow of the software, a report template of the software and an image of the software; the

software multilingualization method further includes:

[0015] extracting a text in an original language type from the initialization script, the workflow and the report template, and obtaining a first non-interface text data;

[0016] recognizing a text in the image to obtain an image text, extracting the text in the original language type from the image text, and obtaining a second non-interface text data; and

[0017] taking the first non-interface text data and the second non-interface text data as the second original language type text data.

[0018] In an embodiment, the generating the runtime interactive interface corresponding to the design-time interactive interface based on the design-time metadata set and the target language text data packet in response to the user's interface access operation includes:

[0019] taking the design-time metadata set as a runtime metadata set in response to the user's interface access operation; and

[0020] rendering and obtaining the runtime interactive interface based on the runtime metadata set and the target language text data packet.

[0021] In an embodiment, after the rendering and obtaining the runtime interactive interface based on the runtime metadata set and the target language text data packet, the method includes:

[0022] copying the runtime metadata set and obtaining a candidate design-time metadata set in response to a user's first tense switching operation;

[0023] modifying the candidate design-time metadata set and obtaining a new design-time metadata set, and/or modifying the target language text data package and obtaining a new target language text data package in response to a user's modification operation;

[0024] taking the candidate design-time metadata set or the new design-time metadata set as a new runtime metadata set in response to a user's second tense switching operation; and

[0025] performing the rendering and obtaining the runtime interactive interface based on the runtime metadata set and the target language text data packet based on the new runtime metadata set and/or the new target language text data package.

[0026] In addition, to achieve the above-mentioned purpose, the present application further proposes a software multilingualization device including:

[0027] a construction module, configured for constructing a design-time interactive interface of a software in response to a user's interface construction operation, and generating a design-time metadata set of the design-time interactive interface based on an interface element on the design-time interactive interface;

[0028] an obtaining module, configured for obtaining a target language text data packet of the software based on a target language type of the software; and

[0029] a generation module, configured for generating a runtime interactive interface corresponding to the design-time interactive interface based on the design-time metadata set and the target language text data packet in response to a user's interface access operation.

[0030] In addition, to achieve the above-mentioned purpose, the present application further proposes an electronic equipment by including: a memory, a processor, and a computer program stored on the memory and executable on the processor, and the computer program being configured to implement the software multilingualization method as described above.

[0031] In addition, to achieve the above-mentioned purpose, the present application further proposes a storage medium, the storage medium is a computer-readable storage medium, the storage medium stores a computer program, and the software multilingualization method as described above are implemented when the computer program is executed by a processor.

[0032] In addition, to achieve the above-mentioned purpose, the present application further proposes a computer program product including a computer program, and the software multilingualization method as described above are implemented when the computer program is executed by a processor.

[0033] One or more technical solutions proposed in the present application have at least the following technical effects.

[0034] In this embodiment, it will construct a design-time interactive interface of a software in response to a user's interface construction operation, and generate a design-time metadata set of the design-time interactive interface based on an interface element on the design-time interactive interface; obtain a target language text data packet of the software based on a target language type of the software; and generate a runtime interactive interface corresponding to the design-time interactive interface based on the design-time metadata set and the target language text data packet in response to a user's interface access operation. That is, the software development system can complete the construction of the design-time interactive interface in response to the user's interface construction operation, and generate the design-time metadata set for the interface element based on the interface element on the design-time interactive interface. Then, according to the target language type of the software, obtaining the target language text data packet of the software. After performing the interface access operation, generating the runtime interactive interface conforming to the target language type by the design-time metadata set and the target language text data packet. It can be understood that since the language text data packet of the software is separated from the construction process of the software design-time interactive interface, the language text data packet may not contain the code part, so the target language text data packet can be independently completed by the translator when making the

target language text data packet, so as to reduce the workload of the translator and the developer in the communication or cooperation during the software internationalization, thereby accelerating the process of the software internationalization, and improving the efficiency of the software internationalization.

BRIEF DESCRIPTION OF THE DRAWINGS

[0035] The accompanying drawings herein are incorporated into the specification and constitute a part of the specification, showing embodiments consistent with the present application, and together with the specification, are used to explain the principles of the present application.

[0036] In order to explain the embodiments of the present application or the technical solutions in the existing technology more clearly, the accompanying drawings needed to be used in the description of the embodiments or the existing technology will be briefly introduced below. Obviously, the accompanying drawings in the following description are only some embodiments of the present application, other accompanying drawings can be obtained based on the provided accompanying drawings without exerting creative efforts for those skilled in the art.

[0037] FIG. 1 is a schematic flow chart of a software multilingualization method according to a first embodiment of the present application.

[0038] FIG. 2 is a framework schematic view of a software development system in a software multilingualization method according to the present application.

[0039] FIG. 3 is a rendering schematic flow chart in a software multilingualization method according to the present application.

[0040] FIG. 4 is a schematic flow chart of a software multilingualization method according to a second embodiment of the present application.

[0041] FIG. 5 is a schematic flow chart of a software multilingualization method according to a third embodiment of the present application.

[0042] FIG. 6 is a schematic flow chart of a software multilingualization method according to a fourth embodiment of the present application.

[0043] FIG. 7 is a schematic view of tense changes in a software multilingualization method according to the present application.

[0044] FIG. 8 is a structural schematic view of a software multilingualization device in a software multilingualization method according to the present application.

[0045] FIG. 9 is a structural schematic view of an equipment of a hardware operating

environment involved in a software multilingualization method according to the present application.

[0046] The realization of the purpose, functional features and advantages of the present application will be further described in conjunction with the embodiments and with reference to the accompanying drawings.

DETAILED DESCRIPTION OF THE EMBODIMENTS

[0047] It should be understood that the specific embodiments described herein are only used to explain the technical solution of the present application and are not used to limit the present application.

[0048] In order to better understand the technical solution of the present application, the following will be described in detail in conjunction with the drawings and specific implementation methods of the specification.

[0049] With the acceleration of the globalization process, the demand for internationalization of software products is growing. The traditional software internationalization process (that is, translating the text content in the software product into foreign language types) often relies on a large number of code modifications, that is, modifying the initial text content in the software into the translated content, but this method is time-consuming and labor-intensive, and the efficiency of software internationalization is low, which prolongs the cycle of software internationalization.

[0050] In the embodiment of the present application, it will construct a design-time interactive interface of a software in response to a user's interface construction operation, and generate a design-time metadata set of the design-time interactive interface based on an interface element on the design-time interactive interface; obtain a target language text data packet of the software based on a target language type of the software; and generate a runtime interactive interface corresponding to the design-time interactive interface based on the design-time metadata set and the target language text data packet in response to a user's interface access operation.

[0051] That is, the software development system can complete the construction of the design-time interactive interface in response to the user's interface construction operation, and generate the design-time metadata set for the interface element based on the interface element on the design-time interactive interface. Then, according to the target language type of the software, obtaining the target language text data packet of the software. After performing the interface access operation, generating the runtime interactive interface conforming to the target language

type by the design-time metadata set and the target language text data packet. It can be understood that since the language text data packet of the software is separated from the construction process of the software design-time interactive interface, the language text data packet may not contain the code part, so the target language text data packet can be independently completed by the translator when making the target language text data packet, so as to reduce the workload of the translator and the developer in the communication or cooperation during the software internationalization, thereby accelerating the process of the software internationalization, and improving the efficiency of the software internationalization.

[0052] It should be noted that the execution subject of this embodiment can be computing service equipment with data processing, network communication and program running functions, such as a cloud server, a computer, a mobile phone, etc., or electronic equipment that can realize the above functions.

[0053] Based on this, an embodiment of the present application provides a software multilingualization method, referring to FIG. 1, which is a schematic flow chart of a software multilingualization method according to a first embodiment of the present application.

[0054] In this embodiment, the software multilingualization method includes step S10 to step S30.

[0055] Step S10, constructing a design-time interactive interface of a software in response to a user's interface construction operation, and generating a design-time metadata set of the design-time interactive interface based on an interface element on the design-time interactive interface.

[0056] It should be noted that in this embodiment, it mainly changes the language of the text displayed in the software product to text of other different language types, so as to internationalize the software (that is, multilingualization). It is worth noting that the interface of the software for multilingualization in this embodiment is constructed by visual modeling technology, that is, it allows developers to build software interfaces by dragging and dropping.

[0057] Exemplarily, the software development system will respond to the user's interface construction operation, and construct the design-time interactive interface of the software. The user's interface construction operation can be operations such as dragging and dropping performed by the user on the software interface construction page of the software development system. For example, the software interface construction page can provide some packaged interface elements, such as buttons, text boxes, drop-down menus, switches, sliders and so on. Correspondingly, each interface element usually has a related text description, such as the text "Confirm" will be displayed on the confirm button. Correspondingly, converting the text

"Confirm" into a text of other language types is one of the contents of multilingualization in the present application. In addition, the above-mentioned design-time interactive interface is an interactive interface of the software in the design stage. After completing the construction of the design-time interactive interface of the software, the design-time metadata set of the design-time interactive interface will be generated by the interface elements on the design-time interactive interface. That is, users can generate software interfaces in the software development system by dragging and dropping, and the software development system will generate an eXtensible Markup Language (XML for short) format file to express the page content and structure. Such a file is called a metadata file in the modeling platform (that is, the software development system), as shown in the following code:

```
[0058]    <?xml version="1.0" encoding="utf-8"?>;
[0059]    <functionPage id="" name="" application="7050">;
[0060]        <page> -- page information;
[0061]            <controls /> -- control information;
[0062]            <pageparams /> -- parameter information;
[0063]            <libraries /> -- resource file information;
[0064]            <events /> -- event information;
[0065]            <wizards /> -- wizard page information;
[0066]        </page>;
[0067]        <pageLayout> -- page layout information;
[0068]            <cells /> -- cell information;
[0069]            <relations /> -- relationship information;
[0070]            <rule /> -- rule information;
[0071]        </pageLayout>;
[0072]    </functionPage>;
```

[0073] The data in the metadata set may include page information, control information, number information, resource file information, event information, wizard page information, page layout information, cell information, relationship information, and rule information, etc. The specific content may be determined by the actual business needs, so it will not be repeated here.

[0074] Step S20, obtaining a target language text data packet of the software based on a target language type of the software.

[0075] Exemplarily, the target language type of the software may be set by the technician according to the needs, or may be determined based on the option selected by the user. Correspondingly, the target language type may be Chinese, English, Russian, Japanese, Korean,

etc. The target language text data packet of the software corresponding to the target language type may be obtained from the pre-built text database. The target language text data packet may be in the form of an Excel table, and the target language text data packet may be prepared in advance by a translator. It is understandable that, since the language text data packet of the software is separated from the construction process of the design-time interactive interface of the software, the language text data packet may not contain codes, so the translator may independently complete the production of the target language text data packet, so as to reduce the workload of the translator and the developer in the communication or cooperation during the software internationalization, thereby speeding up the process of software internationalization, and improving the efficiency of software internationalization.

[0076] Step S30, generating a runtime interactive interface corresponding to the design-time interactive interface based on the design-time metadata set and the target language text data packet in response to a user's interface access operation.

[0077] Exemplarily, the above user's interface access operation can be an operation of the user opening the software, or a switching operation of the user switching the software from design time to runtime. Correspondingly, the software development system will respond to the user's interface access operation, based on the design-time metadata set and the target language text data packet, to perform real-time rendering to obtain a design-time interactive interface that conforms to the target language type and corresponds to the design-time interactive interface. Since the target language type can be a different language type determined by demand, different target language text data packets can be selected according to demand to generate runtime interactive interfaces of different language types, thereby completing the multilingualization of the software (that is, the internationalization of the software).

[0078] In this embodiment, it will construct a design-time interactive interface of a software in response to a user's interface construction operation, and generate a design-time metadata set of the design-time interactive interface based on an interface element on the design-time interactive interface; obtain a target language text data packet of the software based on a target language type of the software; and generate a runtime interactive interface corresponding to the design-time interactive interface based on the design-time metadata set and the target language text data packet in response to a user's interface access operation. That is, the software development system can complete the construction of the design-time interactive interface in response to the user's interface construction operation, and generate the design-time metadata set based on the interface element on the design-time interactive interface. Then, according to the target language type of the software, obtaining the target language text data packet of the

software. After the user performs the interface access operation, generating a runtime interactive interface conforming to the target language type by the design-time metadata set and the target language text data packet. It is understandable that, since the language text data packet of the software is separated from the construction process of the design-time interactive interface of the software, the language text data packet may not include the code part, so the target language text data packet may be independently produced by the translator, so as to reduce the workload of the translator and the developer in the communication or cooperation in the software internationalization, thereby accelerating the process of software internationalization, and improving the efficiency of software internationalization.

[0079] In a feasible implementation, before the step of the obtaining the target language text data packet of the software based on the target language type of the software, the method includes step S200.

[0080] Step S200, inputting a preset translation task prompt template and an original language type text data packet of the software into a preset large language model, so as to guide the preset large language model to translate the original language type text data packet into the target language text data packet conforming to a target language type by the translation task prompt template; the translation task prompt template includes software industry prompt words and translation consistency prompt words, the original language type text data packet at least includes a first original language type text data of a software corresponding interface element, and the preset large language model is obtained by fine-tuning an initial large language model by a preset industry terminology library.

[0081] It should be noted that, at present, many software products rely on machine translation for preliminary language conversion during the internationalization process. However, although machine translation is fast, it often lacks accurate understanding of industry terms and grasp of context, resulting in stiff, inaccurate, and even ambiguous translation results, that is, the internationalization effect is poor. In response to the above problems, in the present application, a preset large language model will be introduced to perform translation to ensure the accuracy of the translation results.

[0082] Exemplarily, a user may select a translation task prompt template according to a requirement, and a translation workbench built into the software development system may input a preset translation task prompt template and an original language type text data packet of the software into a preset large language model, and the preset translation task prompt template is configured to inform (guide) the preset large language model, and the original language type text data packet needs to be translated into a target language type text data packet that conforms to

the target language type. For example, if the software of the Chinese version needs to be internationalized into the software of the English version, then Chinese is the original language type and English is the target language type. The original language type text data packet includes at least the first initial text data of the software corresponding interface element. For example, if the interface element is a button and the button plays a confirmation function, the first initial text data includes the text “Confirm”. In addition, it should be further explained that the translation task prompt template may further include software industry prompt words and translation consistency prompt words. The software industry prompt words are configured to inform the preset large language model of the linguistic context of the translation at this time. For example, if the software is used in the real estate industry, the software industry prompt words can be software industry prompt words, so that the preset large language model is informed that the translation result needs to conform to the professional terminology description of the real estate industry. The translation consistency prompt words are configured to inform the preset large language model that for the same content, its translation results should also be identical.

[0083] In addition, it is further necessary to explain that, in this embodiment, the initial large language model can be fine-tuned using a pre-built industry terminology library to obtain the above-mentioned preset large language model, and the initial large language model can be a Generative Pre-trained Transformer (GPT for short) series model. The industry terminology library may include the translation relationship between Chinese professional terms and professional terms of other language types. That is, the data in the industry terminology library is constructed into training samples, and some model parameters in the initial large language model are selected to be solidified, and then the training samples are configured to train the initial large language model to update the model parameters in the initial large language model to obtain the preset large language model, and these solidified model parameters will not be updated during the training process. It can be understood that after fine-tuning, the large language model can obtain translation results that are more in line with industry terminology descriptions.

[0084] In a feasible implementation, the step of the generating the runtime interactive interface corresponding to the design-time interactive interface based on the design-time metadata set and the target language text data packet in response to the user's interface access operation includes steps S31 and S32.

[0085] Step S31, taking the design-time metadata set as a runtime metadata set in response to the user's interface access operation.

[0086] Step S32, rendering and obtaining the runtime interactive interface based on the runtime metadata set and the target language text data packet.

[0087] Exemplarily, when the user performs the operation of opening the software, the software development system can directly take the design-time metadata set as the runtime metadata set. Then based on the runtime metadata set and the target language text data packet, rendering and obtaining the runtime interactive interface, and presenting to the user.

[0088] It should be noted that, referring to FIG. 2, which is a framework schematic view of a software development system in the present application, the function of generating the interface in this embodiment is the page modeling part. In addition, its actual background further has multiple functions such as data modeling and module modeling. The metadata content generated under each function is also different, so it will not be repeated here. Definition of metadata: being configured to systematically describe the content, source, format, storage location of the data, and the management mode and usage mode of the data. Different functional scenarios are described in different metadata. The interface is uniformly rendered dynamically by page metadata (that is, the design-time metadata set generated based on interface elements on the design-time interactive interface) and multilingual metadata (that is, different target language text data packets) to achieve multilingual translation.

[0089] In addition, referring to FIG. 3, and FIG. 3 is a rendering schematic flow chart in the present application. It should be noted that the page in FIG. 3 is also an interface. When rendering, the Uniform Resource Locator (URL) of the metadata set is first determined. If it is a standard page, the page metadata is obtained, and the FunctionPage metadata can be obtained by page name. Parsing the page information, and obtaining layout information, space information, resource information, style information, and so on, and the resource information may include relevant information of the target language text data packet. Finally, performing the page rendering, and the page rendering may include layout rendering, space rendering, etc., which will not be repeated here. After completing the rendering, outputting the HyperText Markup Language (HTML) document of the page for page display.

[0090] Referring to FIG. 4, which is a flow diagram of the second embodiment of the present application proposed based on the first embodiment of the present application. In this embodiment, the same or similar content as the above embodiment can refer to the above content, so it is not repeated here. After the step of the inputting the preset translation task prompt template and the original language type text data packet of the software into the preset large language model, so as to guide the preset large language model to translate the original language type text data packet into the target language text data packet conforming to the target language type, the method includes steps S210 to S220.

[0091] Step S210, for any original text segment in the original language type text data

packet, determining an actual translation result of the preset keyword in the original text segment corresponding to a target language text segment in response to that a preset keyword exists in the original text segment. The preset keyword is a professional term in an original language type in the preset industry terminology library.

[0092] Step S220, outputting a prompt message associated with the preset keyword, or using a standard translation result to replace the actual translation result in response to that the actual translation result is different from the standard translation result of the preset keyword in the preset industry terminology library.

[0093] It should be noted that, at present, in the traditional software internationalization process, manual proofreading is still an indispensable part, which is configured to correct the errors and deficiencies of machine translation. Similarly, in the traditional manual calibration method, manual proofreading of the translation results is required, which is labor-intensive and inefficient. In this regard, the embodiment will automatically perform proofreading.

[0094] Exemplarily, an original text segment refers to a sentence composed of multiple words and texts of the original language type. When a fine-tuned preset large language model is used for translation, for individual professional term vocabulary, the translation is usually accurate. However, when the professional term vocabulary is combined with other vocabulary to form a sentence, that is, the above-mentioned original text segment, the professionalism of the translation may be reduced. Therefore, in this embodiment, the key words of the professional term will be proofread. Since the process of proofreading each original text segment is basically the same, in this embodiment, one of the original text segments will be used as an example for explanation. For example, for any original text segment in the original language type text data packet, performing word segmentation on the original text segment, and then comparing each segmented word with a professional term in a preset industry terminology library. If the segmented word is the same as a professional term in the preset industry terminology library, the segmented word can be used as a preset keyword. Correspondingly, when the preset keyword exists in the original text segment, determining the actual translation result of the preset keyword in the original text segment corresponding to the target language text segment. Then comparing the actual translation result with the standard translation result of the preset keyword in the preset industry terminology library. If the actual translation result is different from the standard result, outputting a prompt message associated with the preset keyword, and the prompt message is configured to prompt relevant personnel that the translation result may require manual intervention. In addition, when the actual translation result is different from the standard result, the standard translation result can also be directly configured to replace the actual translation

result. In addition, it should be further explained that after performing word segmentation, the frequency of occurrence of a single vocabulary can be counted, and the vocabulary with a frequency greater than the preset word frequency can be selected to construct a calibration table for providing to translators for proofreading. That is, translators do not need to proofread the entire text, which can greatly reduce the workload of translators.

[0095] Referring to FIG. 5, which is a flow chart of the third embodiment of the present application proposed based on the first embodiment and the second embodiment of the present application. The original language type text data packet further includes a second original language type text data of software corresponding supporting resources, and the software corresponding supporting resources include an initialization script of the software, a workflow of the software, a report template of the software and an image of the software. The software multilingualization method further includes step S01 to step S03.

[0096] Step S01, extracting a text in an original language type from the initialization script, the workflow and the report template, and obtaining a first non-interface text data.

[0097] Step S02, recognizing a text in the image to obtain an image text, extracting the text in the original language type from the image text, and obtaining a second non-interface text data.

[0098] Step S03, taking the first non-interface text data and the second non-interface text data as the second original language type text data.

[0099] It should be noted that in the traditional software internationalization method, only the interface text (such as the text description corresponding to the interface element) in the software is often translated, while the non-interface resources in the software, such as the initialization script, workflow, report template, text in the picture, etc. are ignored, which will lead to the incomplete software internationalization obtained by the traditional software internationalization method, that is, part of the content in the software is translated, and part of the content is not translated. Therefore, in this embodiment, the software corresponding supporting resources are further translated.

[0100] Exemplarily, the original language type text data packet further includes the second original language type text data of software corresponding supporting resources. The software corresponding supporting resources may include the initialization script of the software, the workflow of the software, the report template of the software, the image of the software and so on. Therefore, in this embodiment, the software multilingualization method further includes extracting a text in an original language type from the initialization script of the software, the workflow of the software and the report template of the software, and obtaining a first non-interface text data. If the original language type is Chinese, the text in Chinese in the

initialization script of the software, the workflow of the software and the report template of the software can be extracted to form a text table, that is, the first non-interface text data. In addition, for supporting resources of the image class, image text recognition technology, such as Optical Character Recognition (OCR), can be used to recognize the text in the image to obtain the image text. Similarly, extracting the text of the original language type in the image text, and obtaining the second non-interface text data. The first non-interface text data and the second non-interface text data can constitute the second original language text data. The second original language text data will be translated by the preset large language model together with the first original language text data. It can be understood that since this embodiment will translate the second original language text data of the software corresponding supporting resources, the software internationalization can be made more comprehensive, and it is also convenient for the applicable party of the software to maintain the software.

[0101] Referring to FIG. 6, which is a flow chart of the fourth embodiment of the present application proposed based on the first embodiment, the second embodiment, and the third embodiment of the present application. After the step of rendering and obtaining the runtime interactive interface based on the runtime metadata set and the target language text data packet, the method includes steps A10 to A40.

[0102] Step A10, copying the runtime metadata set and obtaining a candidate design-time metadata set in response to a user's first tense switching operation.

[0103] Step A20, modifying the candidate design-time metadata set and obtaining a new design-time metadata set, and/or modifying the target language text data package and obtaining a new target language text data package in response to a user's modification operation.

[0104] Step A30, taking the candidate design-time metadata set or the new design-time metadata set as a new runtime metadata set in response to a user's second tense switching operation.

[0105] Step A40, performing the rendering and obtaining the runtime interactive interface based on the runtime metadata set and the target language text data packet based on the new runtime metadata set and/or the new target language text data package.

[0106] Exemplarily, there is a tense switching option in the interactive interface of the software. When the user clicks the switching option on the runtime interactive interface, it can be determined as a first tense switching operation, and the first tense switching operation is configured to switch the runtime to the design time. Accordingly, the software development system can copy the runtime metadata set in response to the first tense switching operation to obtain a candidate design-time metadata set. At this time, the user can modify the content in the

candidate design-time metadata set according to the needs. Accordingly, the software development system can modify the candidate design-time metadata set in response to the user's modification operation to obtain a new design-time metadata set, and/or modify the target language text data packet to obtain a new target language text data packet. That is, the user can modify only the candidate design-time metadata set, can modify only the target language text data packet, or can modify both the candidate design-time metadata set and the target language text data packet.

[0107] After completing the modification, the user can click the tense switching option to run the software. For example, if the user clicks the switching option on the design-time interactive interface, it can be determined as a second tense switching operation, and the second tense switching operation is configured to switch the design time to the runtime. Correspondingly, the software development system can respond to the second tense switching operation, if the user does not modify the candidate design-time metadata set, the candidate design-time metadata set can be used as a new runtime metadata set, if the user modifies the candidate design-time metadata set, the obtained new design-time metadata set can be used as a new runtime metadata set. Similarly, if obtaining a new runtime metadata set, the step of the rendering and obtaining the runtime interactive interface based on the runtime metadata set and the target language text data packet may be performed based on the new runtime metadata set. If obtaining a new target language text data packet, the step of rendering based on the runtime metadata set and the target language text data packet to obtain the runtime interactive interface may be performed based on the new target language text data packet. If obtaining a new runtime metadata set and a new target language text data packet, the step of rendering based on the runtime metadata set and the target language text data packet to obtain the runtime interactive interface may be performed based on the new target language text data packet and the new target language text data packet.

[0108] It can be understood that in this embodiment, the runtime environment and design-time environment are separated, which allows the business calibration group to directly modify the translation text (that is, the target language text data packet) online and preview the modification effect in real time without restarting the software. Metadata sets, such as text replacement and style adjustment, can also be made during design to ensure that the translated interface conforms to language habits and maintains the original visual style.

[0109] In addition, referring to FIG. 7, and FIG. 7 is a schematic view of tense changes in the present application. As shown in FIG. 7, adding refers to creating the interactive interface of the software. Accordingly, only metadata at design time exists. Publishing is the process of rendering

the interactive interface. In this process, metadata at design time can be used as metadata at runtime, that is, metadata at design time can be cut to runtime, thereby obtaining metadata at runtime. Modification refers to the user modifying the metadata, so the metadata at runtime is copied to the design time to obtain metadata at design time, so that the user can modify the metadata, and after the modification is completed, the modified metadata can be used to overwrite the original metadata at runtime.

[0110] The present application provides a software multilingualization device. Referring to FIG. 8, the software multilingualization device includes a construction module 10, an obtaining module 20 and a generation module 30.

[0111] The construction module 10 is configured for constructing a design-time interactive interface of software in response to a user's interface construction operation, and generating a design-time metadata set of the design-time interactive interface based on an interface element on the design-time interactive interface.

[0112] The obtaining module 20 is configured for obtaining a target language text data packet of the software based on a target language type of the software.

[0113] The generation module 30 is configured for generating a runtime interactive interface corresponding to the design-time interactive interface based on the design-time metadata set and the target language text data packet in response to a user's interface access operation.

[0114] In an embodiment, the software multilingualization device further includes a translation module 40, and the translation module 40 is further configured for:

[0115] inputting a preset translation task prompt template and an original language type text data packet of the software into a preset large language model, so as to guide the preset large language model to translate the original language type text data packet into the target language text data packet conforming to a target language type by the translation task prompt template. The translation task prompt template includes software industry prompt words and translation consistency prompt words, the original language type text data packet at least includes a first original language type text data of software corresponding interface element, and the preset large language model is obtained by fine-tuning an initial large language model by a preset industry terminology library.

[0116] In an embodiment, the translation module 40 is further configured for:

[0117] for any original text segment in the original language type text data packet, determining an actual translation result of the preset keyword in the original text segment corresponding to a target language text segment in response to that a preset keyword exists in the original text segment; the preset keyword is a professional term in an original language type in

the preset industry terminology library; and

[0118] outputting a prompt message associated with the preset keyword, or using a standard translation result to replace the actual translation result in response to that the actual translation result is different from the standard translation result of the preset keyword in the preset industry terminology library.

[0119] In an embodiment, the original language type text data packet further includes a second original language type text data of software corresponding supporting resources, and the software corresponding supporting resources include an initialization script of the software, a workflow of the software, a report template of the software and an image of the software. The software multilingualization device further includes an extraction module 50, and the extraction module 50 is further configured for:

[0120] extracting a text in an original language type from the initialization script, the workflow and the report template, and obtaining a first non-interface text data;

[0121] recognizing a text in the image to obtain an image text, extracting the text in the original language type from the image text, and obtaining a second non-interface text data; and

[0122] taking the first non-interface text data and the second non-interface text data as the second original language type text data.

[0123] In an embodiment, the generation module 30 is further configured for:

[0124] taking the design-time metadata set as a runtime metadata set in response to the user's interface access operation; and

[0125] rendering and obtaining the runtime interactive interface based on the runtime metadata set and the target language text data packet.

[0126] In an embodiment, the generation module 30 is further configured for:

[0127] copying the runtime metadata set and obtaining a candidate design-time metadata set in response to a user's first tense switching operation;

[0128] modifying the candidate design-time metadata set and obtaining a new design-time metadata set, and/or modifying the target language text data package and obtaining a new target language text data package in response to a user's modification operation;

[0129] taking the candidate design-time metadata set or the new design-time metadata set as a new runtime metadata set in response to a user's second tense switching operation; and

[0130] performing the rendering and obtaining the runtime interactive interface based on the runtime metadata set and the target language text data packet based on the new runtime metadata set and/or the new target language text data package.

[0131] The software multilingualization device provided by the present application adopts

the software multilingualization method in the above embodiment, aiming to solve the technical problem of low efficiency of the current software internationalization method. Compared with the prior art, the beneficial effects of the software multilingualization device provided by the embodiment of the present application are the same as the beneficial effects of the software multilingualization method provided by the above first embodiment, and other technical features in the software multilingualization device are the same as the features disclosed in the above embodiment method, which are not repeated here.

[0132] The present application provides electronic equipment, and the electronic equipment includes: at least one processor, and a memory being in communication connection with the at least one processor. The memory stores instructions executable by the at least one processor, and the instructions are executed by the at least one processor so that the at least one processor can execute the software multilingualization method in the above first embodiment.

[0133] Reference to FIG. 9 below, which shows a structural schematic view of electronic equipment suitable for implementing the embodiment of the present application. The electronic equipment in the embodiment of the present application may include but is not limited to mobile terminals such as laptop computers, Personal Digital Assistants (PDAs), Portable Application Descriptions (PADs), etc., and fixed terminals such as desktop computers, etc. The electronic equipment shown in FIG. 5 is only an example and should not bring any limitation to the functions and scope of use of the embodiment of the present application.

[0134] As shown in FIG. 9, the electronic equipment may include a processing device 1001 (such as a central processing unit, a graphics processor, etc.), which can perform various appropriate actions and processes according to a program stored in a read-only memory (ROM) 1002 or a program loaded from a storage device 1003 to a random access memory (RAM) 1004. In RAM 1004, various programs and data required for the operation of the electronic equipment are further stored. The processing device 1001, ROM 1002 and RAM 1004 are connected to each other by a bus 1005. An input/output (I/O) interface 1006 is also connected to the bus. Typically, the following systems can be connected to the I/O interface 1006: an input device 1007 including, for example, a touch screen, a touchpad, a keyboard, a mouse, an image sensor, a microphone, an accelerometer, a gyroscope, etc.; an output device 1008 including, for example, a liquid crystal display (LCD), a speaker, a vibrator, etc.; a storage device 1003 including, for example, a magnetic tape, a hard disk, etc.; and a communication device 1009. The communication device 1009 can allow an electronic equipment to communicate with other equipment wirelessly or by wire to exchange data. Although the figure shows electronic equipment with various systems, it should be understood that it is not required to implement nor

have all the systems shown. More or fewer systems may be implemented or have alternatively.

[0135] In particular, according to the embodiments disclosed in the present application, the process described above with reference to the flowchart can be implemented as a computer software program. For example, the embodiments disclosed in the present application include a computer program product, which includes a computer program carried on a computer-readable medium, and the computer program contains program code for executing the method shown in the flowchart. In such an embodiment, the computer program can be downloaded and installed from the network by the communication device, or installed from the storage device 1003, or installed from the ROM 1002. When the computer program is executed by the processing device 1001, the above functions defined in the method of the embodiment disclosed in the present application are executed.

[0136] The electronic equipment provided in the present application adopts the software multilingualization method in the above embodiment to solve the technical problem that the longitudinal control algorithm cannot provide better control effect while being easy to apply the real-time control of the vehicle. Compared with the prior art, the beneficial effects of the electronic equipment provided in the present application are the same as the beneficial effects of the software multilingualization method provided in the above embodiment, and the other technical features in the electronic equipment are the same as the features disclosed in the method of the previous embodiment, which will not be repeated here.

[0137] It should be understood that the various parts disclosed in the present application can be implemented by hardware, software, firmware or a combination thereof. In the description of the above implementation, specific features, structures, materials or characteristics can be combined in any one or more embodiments or examples in a suitable manner.

[0138] The above is only a specific implementation of the present application, but the scope of the present application is not limited thereto. Any person skilled in the art who is familiar with the present technical field can easily think of changes or substitutions within the technical scope disclosed in the present application, which should be included in the scope of the present application. Therefore, the scope of the present application should be based on the scope of the claims.

[0139] The present application provides a computer-readable storage medium having computer-readable program instructions (that is, computer programs) stored thereon, and the computer-readable program instructions are configured to execute the software multilingualization method in the above-mentioned embodiment.

[0140] The computer-readable storage medium provided in the present application may be,

for example, a Universal Serial Bus (USB) flash drive, but is not limited to electrical, magnetic, optical, electromagnetic, infrared, or semiconductor systems, systems or devices, or any combination thereof. More specific examples of computer-readable storage medium may include, but are not limited to: an electrical connection with one or more wires, a portable computer disk, a hard disk, a random access memory (RAM), a read-only memory (ROM), an erasable programmable read-only memory (EPROM), an optical fiber, a portable compact disk read-only memory (CD-ROM), an optical storage device, a magnetic storage device, or any suitable combination thereof. In the present embodiment, the computer-readable storage medium may be any tangible medium containing or storing a program, which may be used by or in combination with an instruction execution system, system or device. The program code contained in the computer-readable storage medium can be transmitted by any appropriate medium, including but not limited to: wires, optical cables, Radio Frequency (RF), etc., or any suitable combination of the above.

[0141] The above-mentioned computer-readable storage medium can be contained in electronic equipment; or it can exist independently without being assembled into the electronic equipment.

[0142] The computer-readable storage medium carries one or more programs. When the one or more programs are executed by the electronic equipment, the electronic equipment is caused to: construct a design-time interactive interface of a software in response to a user's interface construction operation, and generate a design-time metadata set of the design-time interactive interface based on an interface element on the design-time interactive interface; obtain a target language text data packet of the software based on a target language type of the software; and generate a runtime interactive interface corresponding to the design-time interactive interface based on the design-time metadata set and the target language text data packet in response to a user's interface access operation.

[0143] Computer program code for performing the operations of the present application may be written in one or more programming languages or a combination thereof. The programming languages include object-oriented programming languages such as Python, Java, Smalltalk, C++, and further includes conventional procedural programming languages such as "C" languages or similar programming languages. The program code may be executed entirely on the user's computer, executed partially on the user's computer, executed as a separate software package, partially executed on the user's computer and partially executed on a remote computer, or entirely executed on a remote computer or server. In situations involving remote computers, the remote computer may be connected to the user's computer by any type of network, which

includes a local area network (LAN) or a wide area network (WAN), or may be connected to an external computer (e.g., use an Internet Service Provider to connect via the Internet).

[0144] The flowcharts and block diagrams in the accompanying drawings illustrate the possible architecture, functions and operations of the systems, methods and computer program products according to various embodiments of the present application. In this regard, each box in the flowchart or block diagram may represent a module, a program segment, or a portion of a code, which contains one or more executable instructions for implementing the specified logical functions. It should also be noted that in some alternative implementations, the functions marked in the box may also occur in a different order than that marked in the accompanying drawings. For example, two boxes represented in succession can actually be executed substantially in parallel, and they can sometimes be executed in the opposite order, depending on the functions involved. It should also be noted that each box in the block diagram and/or flowchart, and the combination of boxes in the block diagram and/or flowchart, can be implemented by a dedicated hardware-based system that performs the specified functions or operations, or can be implemented by a combination of dedicated hardware and computer instructions.

[0145] The modules involved in the embodiments described in the present application can be implemented by software or by hardware. The name of the module does not constitute a limitation on the unit itself in some cases.

[0146] The readable storage medium provided in the present application is a computer-readable storage medium, which stores computer-readable program instructions (that is, computer programs) for executing the software multilingualization method, and can solve the technical problem that the existing data display method is difficult to intuitively and clearly complete data visualization. Compared with the prior art, the beneficial effects of the computer-readable storage medium provided in the present application are the same as the beneficial effects of the software multilingualization method provided in the above embodiment, which will not be repeated here.

[0147] The present application further provides a computer program product including a computer program, and the computer program implements the steps of the software multilingualization method as described above when executed by a processor.

[0148] The computer program product provided in the present application can solve the technical problem of software multilingualization. Compared with the prior art, the beneficial effects of the computer program product provided in the present application are the same as the beneficial effects of the software multilingualization method provided in the above embodiment, which will not be repeated here.

[0149] The above descriptions are only some embodiments of the present application, and are not intended to limit the scope of the present application. All equivalent structural changes made using the contents of the present application specification and drawings under the technical concept of the present application, or direct/indirect applications in other related technical fields are included in the scope of the present application.

CLAIMS

What is claimed is:

[Claim 1] A software multilingualization method, comprising:

constructing a design-time interactive interface of a software in response to a user's interface construction operation, and generating a design-time metadata set of the design-time interactive interface based on an interface element on the design-time interactive interface;

obtaining a target language text data packet of the software based on a target language type of the software; and

generating a runtime interactive interface corresponding to the design-time interactive interface based on the design-time metadata set and the target language text data packet in response to a user's interface access operation.

[Claim 2] The software multilingualization method according to claim 1, wherein before the obtaining the target language text data packet of the software based on the target language type of the software, the method comprises:

inputting a preset translation task prompt template and an original language type text data packet of the software into a preset large language model, so as to guide the preset large language model to translate the original language type text data packet into the target language text data packet conforming to a target language type by the translation task prompt template; wherein the translation task prompt template comprises software industry prompt words and translation consistency prompt words, the original language type text data packet at least comprises a first original language type text data of a software corresponding interface element, and the preset large language model is obtained by fine-tuning an initial large language model by a preset industry terminology library.

[Claim 3] The software multilingualization method according to claim 2, wherein after the inputting the preset translation task prompt template and the original language type text data packet of the software into the preset large language model, so as to guide the preset large

language model to translate the original language type text data packet into the target language text data packet conforming to the target language type, the method further comprises:

for any original text segment in the original language type text data packet, determining an actual translation result of the preset keyword in the original text segment corresponding to a target language text segment in response to that a preset keyword exists in the original text segment; wherein the preset keyword is a professional term in an original language type in the preset industry terminology library; and

outputting a prompt message associated with the preset keyword, or using a standard translation result to replace the actual translation result in response to that the actual translation result is different from the standard translation result of the preset keyword in the preset industry terminology library.

[Claim 4] The software multilingualization method according to claim 2, wherein the original language type text data packet further comprises a second original language type text data of a software corresponding supporting resources, and the software corresponding supporting resources comprise a initialization script of the software, a workflow of the software, a report template of the software and an image of the software; the software multilingualization method further comprises:

extracting a text in an original language type from the initialization script, the workflow and the report template, and obtaining a first non-interface text data;

recognizing a text in the image to obtain an image text, extracting the text in the original language type from the image text, and obtaining a second non-interface text data; and

taking the first non-interface text data and the second non-interface text data as the second original language type text data.

[Claim 5] The software multilingualization method according to claim 1, wherein the generating the runtime interactive interface corresponding to the design-time interactive interface based on the design-time metadata set and the target language text data packet in response to the user's interface access operation comprises:

taking the design-time metadata set as a runtime metadata set in response to the user's

interface access operation; and

rendering and obtaining the runtime interactive interface based on the runtime metadata set and the target language text data packet.

[Claim 6] The software multilingualization method according to claim 5, wherein after the rendering and obtaining the runtime interactive interface based on the runtime metadata set and the target language text data packet, the method comprises:

copying the runtime metadata set and obtaining a candidate design-time metadata set in response to a user's first tense switching operation;

modifying the candidate design-time metadata set and obtaining a new design-time metadata set, and/or modifying the target language text data package and obtaining a new target language text data package in response to a user's modification operation;

taking the candidate design-time metadata set or the new design-time metadata set as a new runtime metadata set in response to a user's second tense switching operation; and

performing the rendering and obtaining the runtime interactive interface based on the runtime metadata set and the target language text data packet based on the new runtime metadata set and/or the new target language text data package.

[Claim 7] A software multilingualization device, comprising:

a construction module, configured for constructing a design-time interactive interface of a software in response to a user's interface construction operation, and generating a design-time metadata set of the design-time interactive interface based on an interface element on the design-time interactive interface;

an obtaining module, configured for obtaining a target language text data packet of the software based on a target language type of the software; and

a generation module, configured for generating a runtime interactive interface corresponding to the design-time interactive interface based on the design-time metadata set and the target language text data packet in response to a user's interface access operation.

[Claim 8] An electronic equipment, comprising: a memory, a processor, and a computer

program stored on the memory and executable on the processor, and the computer program being configured to implement the software multilingualization method according to any one of claims 1 to 6.

[Claim 9] A storage medium, wherein the storage medium is a computer-readable storage medium, the storage medium stores a computer program, and the software multilingualization method according to any one of claims 1 to 6 are implemented when the computer program is executed by a processor.

[Claim 10] A computer program product, comprising:
a computer program, wherein the software multilingualization method according to any one of claims 1 to 6 are implemented when the computer program is executed by a processor.