

METHOD, APPARATUS, AND DEVICE FOR CONFIGURING AND EXECUTING PAYMENT PROCESS

TECHNICAL FIELD

[0001] The present specification relates to the field of information technology, and
5 in particular, to methods, apparatuses, and devices for configuring and executing a
payment process.

BACKGROUND

[0002] With the popularity of electronic payment, more and more users no longer
make cash payments, but instead make payments through an online electronic
10 payment system. In practice, a user sends a payment request to the electronic payment
system, which can trigger the electronic payment system to execute a predetermined
payment process.

[0003] FIG. 1 illustrates a typical payment process. In FIG. 1, the payment process
includes eight steps S100 to S114, where steps S106, S108, and S112 are actually
15 identical steps. When the electronic payment system determines that a payment
abnormality has occurred (for example, the account balance of the payer is
insufficient, the deduction has failed, or the payment has failed), the payment process
is terminated, and a failure result is returned to the payer (for example, step S106,
S108, or S112 is performed).

[0004] It is well known that the payment process is pre-configured by technical
staff by means of programming, and that execution of the payment process by the
electronic payment system is essentially to execute code logic that corresponds to the
payment process sequentially from beginning to end. Therefore, even if the operations
of steps S106, S108, and S112 shown in FIG. 1 are the same, the technical staff needs
25 to write code for each of the above three steps, that is, to write three code segments.

[0005] In view of the existing technology, there is a need for a more efficient
method of configuring a payment process.

SUMMARY

[0006] Implementations of the present specification provide methods, apparatuses, and devices for configuring and executing a payment process, to resolve a problem of inefficiency in an existing method for configuring a payment process.

5 [0007] To resolve the previously described technical problem, the implementations of the present specification provide the following solutions:

[0008] An implementation of the present specification provides a method for configuring a payment process, including: determining, based on operations included in a predetermined payment process, states corresponding to the operations,
10 respectively, where with respect to each operation, a state corresponding to the operation is a state of a payment service when the operation is performed; determining a transition relationship between the states based on an association relationship between the operations included in the payment process, and determining a condition for transition between the states based on a trigger condition corresponding to each
15 operation in the payment process; and configuring a finite state machine corresponding to the payment process based on the states, the transition relationship between the states, and the condition for transition between the states, so as to execute the payment process through the finite state machine.

[0009] An implementation of the present specification provides a method for
20 executing a payment process, including: determining, by using a finite state machine, a first state of a payment service, the finite state machine being pre-configured using the method for configuring a payment process; and in response to determining that a transition condition for triggering a transition from the first state to the second state is met, performing the operation corresponding to the second state, and transitioning the
25 payment service from the first state to the second state using the finite state machine.

[0010] An implementation of the present specification provides an apparatus for configuring a payment process, including: a first determining module, configured to determine, based on operations included in a predetermined payment process, states corresponding to the operations, respectively, where with respect to each operation, a
30 state corresponding to the operation is a state of a payment service when the operation is performed; a second determining module, configured to determine a transition relationship between the states based on an association relationship between the operations included in the payment process, and determine a condition for transition

between the states based on a trigger condition corresponding to each operation in the payment process; and a configuration module, configured to configure a finite state machine corresponding to the payment process based on the states, the transition relationship between the states, and the condition for transition between the states, so as to execute the payment process through the finite state machine.

[0011] An implementation of the present specification provides an apparatus for executing a payment process, including: a determining module, configured to determine, by using a finite state machine, a first state of a payment service, the finite state machine being pre-configured using the method according to any one of claims 1 to 4; and a processing module, configured to: in response to determining that a transition condition for triggering a transition from the first state to the second state is met, perform the operation corresponding to the second state, and transition the payment service from the first state to the second state using the finite state machine.

[0012] An implementation of the present specification provides a device for configuring a payment process, including one or more processors and a memory, where the memory stores a program executable by the one or more processors to perform the following steps: determining, based on operations included in a predetermined payment process, states corresponding to the operations, respectively, where with respect to each operation, a state corresponding to the operation is a state of a payment service when the operation is performed; determining a transition relationship between the states based on an association relationship between the operations included in the payment process, and determining a condition for transition between the states based on a trigger condition corresponding to each operation in the payment process; and configuring a finite state machine corresponding to the payment process based on the states, the transition relationship between the states, and the condition for transition between the states, so as to execute the payment process through the finite state machine.

[0013] An implementation of the present specification provides a device for executing a payment process, including one or more processors and a memory, where the memory stores a program executable by the one or more processors to perform the following steps: determining, by using a finite state machine, a first state of a payment service, the finite state machine being pre-configured using the method for configuring a payment process; and in response to determining that a transition

condition for triggering a transition from the first state to the second state is met, performing the operation corresponding to the second state, and transitioning the payment service from the first state to the second state using the finite state machine.

[0014] As can be seen from the technical solutions provided in the above

5 implementations of the present specification, in the implementations of the present specification, a state corresponding to each operation is determined based on each operation included in a predetermined payment process, a transition relationship between the states is determined based on an association relationship between operations included in the payment process, and a condition for transition between the
10 states is determined based on a trigger condition corresponding to each operation in the payment process. Thus, the payment process can be represented as a finite state machine, and the payment process can be executed using the finite state machine. In the implementations of the present specification, each operation included in the payment process is represented as a state in the finite state machine, even if the
15 payment process includes a plurality of steps of the same operation, which actually correspond to the same state, and the technical staff only needs to write code for the state, which improves the efficiency of configuring the payment process.

BRIEF DESCRIPTION OF DRAWINGS

[0015] To describe the technical solutions in the implementations of the present
20 specification or in the existing technology more clearly, the following outlines the accompanying drawings for illustrating such technical solutions. Clearly, the accompanying drawings outlined below are some implementations of the present specification and a person skilled in the art can derive other drawings from such accompanying drawings without creative efforts.

25 **[0016]** FIG. 1 is a schematic diagram illustrating a payment process;

[0017] FIG. 2 is a flowchart illustrating a method for configuring a payment process, according to an implementation of the present specification;

[0018] FIG. 3 is a flowchart illustrating a method for executing a payment process, according to an implementation of the present specification;

30 **[0019]** FIG. 4 is a flowchart illustrating an apparatus for configuring a payment process, according to an implementation of the present specification;

[0020] FIG. 5 is a flowchart illustrating an apparatus for executing a payment process, according to an implementation of the present specification;

[0021] FIG. 6 is a flowchart illustrating a device for configuring a payment process, according to an implementation of the present specification; and

5 [0022] FIG. 7 is a flowchart illustrating a device for executing a payment process, according to an implementation of the present specification.

DESCRIPTION OF IMPLEMENTATIONS

[0023] In the existing technology, when technical staff configures a payment process, for a plurality of steps of the same operation included in the payment process,
10 the technical staff needs to write a plurality of identical code segments that correspond to the plurality of steps of the same operation, which results in inefficiency of the existing configuration payment process.

[0024] However, in the implementations of the present specification, the payment process is represented as a finite state machine, and each operation included in the
15 payment process is represented as a state in the finite state machine, so that a plurality of steps of the same operation included in the payment process may correspond to a same state, and staff only need to write a code segment for the state, which improves efficiency of configuring the payment process.

[0025] To make a person skilled in the art better understand the technical solutions in the present specification, the following clearly describes the technical solutions in the implementations of the present specification with reference to the accompanying drawings in the implementations of the present specification. Clearly, the described implementations are merely some but not all of the implementations of the present specification. Based on the implementations of the present specification, a person
20 skilled in the art can obtain other implementations without creative efforts, which all fall within the scope of the present specification.

[0026] The following describes in detail the implementations of the present specification.

[0027] FIG. 2 is a flowchart illustrating a method for configuring a payment
30 process, according to an implementation of the present specification. The method includes the following steps:

[0028] S200: Determine, based on a plurality of operations included in a predetermined payment process, a state corresponding to each operation.

[0029] Generally, the electronic payment system is triggered by a payment request sent from a payer to execute a payment process. It is worthwhile to note that there can be a plurality of payment processes depending on different service scenarios. For example, in the case of payment with medical insurance, the payment process can be a process of paying medical expenses using one or more of a bank card and a medical insurance card.

[0030] The payment process includes a plurality of steps, and each step corresponds to an operation. When performing a step, the electronic payment system actually performs the operation corresponding to the step. For example, the operation corresponding to step S102 shown in FIG. 1 is to “determine whether the account balance of the payer is sufficient.” Clearly, the payment process can include a plurality of steps that have the same operation, (for example, steps S106, S108, and S112 shown in FIG. 1), but the operations included in the payment process are different from one another.

[0031] In this implementation of the present specification, each operation included in the predetermined payment process can be represented as a state. For each operation, the state corresponding to the operation is the state of the payment service when the electronic payment system performs the operation.

[0032] S202: Determine a transition relationship between the states based on an association relationship between operations included in the payment process, and determine a condition for transition between the states based on a trigger condition corresponding to each operation in the payment process.

[0033] There is usually an association between the operations included in the payment process. It is worthwhile to note that the association between two operations means that after one of the operations is performed, it is possible that the other operation is performed immediately after the operation. For two operations with an association relationship, the operation that is performed previously can be referred to as a “first operation,” and the operation that is performed subsequently can be referred to as a “second operation.”

[0034] For example, there is an association relationship between the operation corresponding to step S102 shown in FIG. 1 and the operation corresponding to step

S104. The operation corresponding to step S102 is a first operation, and the operation corresponding to step S104 is a second operation. It is also worthwhile to note that sometimes an operation has an association with itself, for example, step S100 in FIG. 1.

5 **[0035]** Clearly, if each operation included in the payment process is represented as a state, the process of performing the operations can be represented as a process of transitioning between the states; that is, a transition between every two operations with an association relationship can be viewed/ represented as a transition between the states corresponding to the two operations. Therefore, the transition relationship
10 between the states can be determined based on the association relationship between the operations. There is usually a transition relationship between the states corresponding to two operations having an association relationship therebetween.

[0036] Further, the condition for transition between the states can be determined based on the trigger condition corresponding to each operation in the payment process.
15 Specifically, for each operation, the trigger condition corresponding to the operation is generally that a first operation associated with the operation has obtained a specific execution result, and the operation is a second operation. For example, in FIG. 1, for an operation corresponding to step S102 and an operation corresponding to step S104 that have an association relationship, an execution result “yes” of the operation
20 corresponding to step S102 is a trigger condition of the operation corresponding to step S104.

[0037] S204: Configure a finite state machine corresponding to the payment process based on the states, the transition relationship between the states, and the condition for transition between the states, so as to execute the payment process
25 through the finite state machine.

[0038] In steps S200 to S202, each state of a finite state machine, a transition relationship between the states, and a condition for transition between the states are obtained. The finite state machine can be configured based on the states of the finite state machine, the transition relationship between the states, and the condition for
30 transition between the states and can be used for executing the payment process.

[0039] Further, a state transition table corresponding to the finite state machine can be generated based on the states, the transition relationship between the states, and the condition for transition between the states, for the electronic payment system to query

the state transition table in executing the payment process. Table 1 shows a state transition table according to implementations of the present specification.

Current state	Transition condition	State after transition
State 1 (operation A)	Operation A succeeds.	State 2 (operation B)
State 2 (operation B)	Operation B fails.	State 3 (operation C)
State 1 (operation A)	Operation A fails.	State 4 (operation D)
...	...	...

Table 1

5 [0040] It is worthwhile to note that the state transition table can be written to the configuration file of the electronic payment system, for the electronic payment system to retrieve the state transition table from the configuration file of the electronic payment system at a startup; or the state transition table can be written to the cache of the electronic payment system, for the electronic payment system to retrieve the state transition table from the cache of the electronic payment system at a runtime. Of course, the state transition table can be written to both the configuration file and the cache.

10 [0041] According to the method for configuring a payment process shown in FIG. 2, a state corresponding to each operation is determined based on each operation included in a predetermined payment process, a transition relationship between the states is determined based on an association relationship between operations included in the payment process, and a condition for transition between the states is determined based on a trigger condition corresponding to each operation in the payment process. Thus, the payment process can be represented as a finite state machine, and the payment process can be executed using the finite state machine. In the implementations of the present specification, each operation included in the payment process is represented as a state in the finite state machine. Even if a payment process includes a plurality of steps that have a same operation, which actually correspond to a same state, the technical staff only needs to write program codes for the state, which improves the efficiency of configuring the payment process.

20 [0042] In addition, for a more complex payment process (for example, the process includes a large number of steps, or each step can be transitioned to a large number of other steps), abstracting the payment process as a finite state machine also helps

simplify the programming work of a technical staff. The technical staff abstracts each operation included in the payment process into a state, and only needs to configure the condition for transition between the states (that is, configure the state transition table).

[0043] FIG. 3 is a flowchart illustrating a method for executing a payment process,
5 according to an implementation of the present specification. The method includes the following steps:

[0044] S300: Determine, by using a finite state machine, a first state of a payment service.

[0045] S302: In response to determining that a transition condition for triggering a
10 transition from the first state to the second state is met, perform the operation corresponding to the second state, and transition the payment service from the first state to the second state using the finite state machine.

[0046] The execution body of the method can be an electronic payment system, which can be specifically a server or a server cluster for processing electronic
15 payment services.

[0047] It is worthwhile to note that in the method shown in FIG. 3, the finite state machine is pre-configured using the method for configuring a payment process shown in FIG. 2. Specifically, the state transition table that corresponds to the payment process and that is generated using the method shown in FIG. 2 can be written to the
20 configuration file of the electronic payment system in advance, for the electronic payment system to retrieve the state transition table from the configuration file of the electronic payment system at a startup.

[0048] In this implementation of the present specification, the first state represents the current state of the payment service that is determined using the finite state
25 machine, and the second state represents a state to which the payment service is subsequently transitioned from the first state. The first state and the second state can be the same state.

[0049] After detecting, from the state transition table, that the transition condition for triggering the transition from the first state to the second state is met, the
30 electronic payment system can, on the one hand, perform the operation corresponding to the second state and, on the other hand, transition the payment service from the first state to the second state using the finite state machine.

[0050] Specifically, the state transition table can be queried; and in response to determining that a transition condition obtained from the state transition table is met, it is determined that a transition condition for triggering a transition from the first state to the second state is met.

5 [0051] Further, before querying the state transition table, the electronic payment system retrieves the state transition table from a configuration file of the electronic payment system at a startup; or the electronic payment system retrieves the state transition table from a cache of the electronic payment system at a runtime.

[0052] In addition, sometimes the electronic payment system can be a distributed
10 system; that is, a plurality of electronic payment subsystems cooperate with each other to perform the payment process. In this case, the electronic payment subsystems need to invoke each other (either synchronously or asynchronously) to execute the payment process. In addition, to ensure that the data in the databases of all electronic payment subsystems is consistent, the work of each electronic payment subsystem is divided
15 into a service acceptance stage and a service processing stage. At the service acceptance stage, the electronic payment subsystem guarantees the idempotent for the invocation requests received from other electronic payment subsystems (that is, a plurality of identical invocation requests that are repeatedly received are considered as one invocation request). If the electronic payment subsystem fails to handle the
20 service at the service handling stage, the electronic payment subsystem cannot obtain the corresponding service data at the service processing stage.

[0053] The electronic payment subsystem not only accepts the invocation request successfully at the service acceptance stage, but also processes the accepted invocation request successfully at the service processing stage before performing state
25 transition using the finite state machine. If the electronic payment subsystem fails to accept the invocation request at the service acceptance stage or fails to process the invocation request at the service processing stage, compensation and/or retry for the service acceptance or the service processing needs to be triggered, so that the electronic payment subsystem successfully accepts the invocation request at the
30 service acceptance stage and successfully processes the invocation request at the service processing stage, thereby enabling the state transition to be performed by the finite state machine.

[0054] Based on the method for configuring a payment process shown in FIG. 2, an implementation of the present specification further provides an apparatus for configuring a payment process. As shown in FIG. 4, the apparatus includes: a first determining module 401, configured to determine, based on operations included in a predetermined payment process, states corresponding to the operations, respectively, where with respect to each operation, a state corresponding to the operation is a state of a payment service when the operation is performed; a second determining module 402, configured to determine a transition relationship between the states based on an association relationship between the operations included in the payment process, and determine a condition for transition between the states based on a trigger condition corresponding to each operation in the payment process; and a configuration module 403, configured to configure a finite state machine corresponding to the payment process based on the states, the transition relationship between the states, and the condition for transition between the states, so as to execute the payment process through the finite state machine.

[0055] The payment process specifically includes a process for paying medical expenses using one or more of a bank card and a medical insurance card.

[0056] The configuration module 403 generates a state transition table corresponding to the finite state machine based on the states, a transition relationship between the states, and a condition for transition between the states, for the electronic payment system to query the state transition table in executing the payment process.

[0057] The apparatus further includes a writing module 404, configured to: write the state transition table to a configuration file of the electronic payment system, for the electronic payment system to retrieve the state transition table from the configuration file of the electronic payment system at a startup; and/or write the state transition table into a cache of the electronic payment system, for the electronic payment system to retrieve the state transition table from the cache of the electronic payment system at a runtime.

[0058] Based on the method for executing a payment process shown in FIG. 3, an implementation of the present specification further provides an apparatus for configuring a payment process. As shown in FIG. 5, the apparatus includes: a determining module 501, configured to determine, by using a finite state machine, a first state of a payment service, the finite state machine being pre-configured using the

method shown in FIG. 2; and a processing module 502, configured to: in response to determining that a transition condition for triggering a transition from the first state to the second state is met, perform the operation corresponding to the second state, and transition the payment service from the first state to the second state using the finite
5 state machine.

[0059] The processing module 502 queries a state transition table; and in response to determining that a transition condition obtained from the state transition table is met, determines that the transition condition for triggering the transition from the first state to the second state is met.

10 **[0060]** The processing module 502 retrieves the state transition table from a configuration file of the device when the device is started before querying the state transition table; or retrieves the state transition table from a cache of the device when the device is running.

[0061] Based on the method for configuring a payment process shown in FIG. 2, an
15 implementation of the present specification further provides a device for configuring a payment process. As shown in FIG. 6, the device includes one or more processors and a memory, where the memory stores a program executable by the one or more processors to perform the following steps: determining, based on operations included in a predetermined payment process, states corresponding to the operations,
20 respectively, where with respect to each operation, a state corresponding to the operation is a state of a payment service when the operation is performed; determining a transition relationship between the states based on an association relationship between the operations included in the payment process, and determining a condition for transition between the states based on a trigger condition corresponding to each
25 operation in the payment process; and configuring a finite state machine corresponding to the payment process based on the states, the transition relationship between the states, and the condition for transition between the states, so as to execute the payment process through the finite state machine.

[0062] Based on the method for executing a payment process shown in FIG. 3, an
30 implementation of the present specification further provides a device for executing a payment process. As shown in FIG. 7, the device includes one or more processors and a memory, where the memory stores a program executable by the one or more processors to perform the following steps: determining, by using a finite state machine,

a first state of a payment service, the finite state machine being pre-configured using the method shown in FIG. 2; and in response to determining that a transition condition for triggering a transition from the first state to the second state is met, performing the operation corresponding to the second state, and transitioning the payment service
5 from the first state to the second state using the finite state machine.

[0063] The implementations of the present specification are described in a progressive way. For same or similar parts of the implementations, mutual references can be made to the implementations. Each implementation focuses on a difference from the other implementations. Particularly, the device implementations shown in
10 FIG. 6 and FIG. 7 are basically similar to the method implementations, and therefore are described briefly. For related parts, references can be made to related descriptions in the method implementations.

[0064] In the 1990s, whether technology improvement is hardware improvement (for example, improvement of a circuit structure, such as a diode, a transistor, or a switch) or software improvement (improvement of a method procedure) can be clearly
15 distinguished. However, as technologies develop, the current improvement for many method procedures can be considered as a direct improvement of a hardware circuit structure. A designer usually programs an improved method procedure to a hardware circuit, to obtain a corresponding hardware circuit structure. Therefore, a method
20 procedure can be improved using a hardware entity module. For example, a programmable logic device (PLD) (for example, a field programmable gate array (FPGA)) is such an integrated circuit, and a logical function of the programmable logic device is determined by a user through device programming. The designer performs programming to “integrate” a digital system to a PLD without requesting a
25 chip manufacturer to design and produce an application-specific integrated circuit chip. In addition, at present, instead of manually manufacturing an integrated chip, this type of programming is mostly implemented using “logic compiler” software. The programming is similar to a software compiler used to develop and write a program. Original code needs to be written in a particular programming language for
30 compilation. The language is referred to as a hardware description language (HDL). There are many HDLs, such as the Advanced Boolean Expression Language (ABEL), the Altera Hardware Description Language (AHDL), Confluence, the Cornell University Programming Language (CUPL), HDCal, the Java Hardware Description

Language (JHDL), Lava, Lola, MyHDL, PALASM, and the Ruby Hardware Description Language (RHDL). The very-high-speed integrated circuit hardware description language (VHDL) and Verilog are most commonly used. A person skilled in the art should also understand that a hardware circuit that implements a logical method procedure can be readily obtained once the method procedure is logically programmed using the several described hardware description languages and is programmed into an integrated circuit.

[0065] A controller can be implemented using any appropriate method. For example, the controller can be a microprocessor or a processor, or a computer-readable medium that stores computer readable program code (such as software or firmware) that can be executed by the microprocessor or the processor, a logic gate, a switch, an application-specific integrated circuit (ASIC), a programmable logic controller, or a built-in microprocessor. Examples of the controller include but are not limited to the following microprocessors: ARC 625D, Atmel AT91SAM, Microchip PIC18F26K20, and Silicon Labs C8051F320. The memory controller can also be implemented as a part of the control logic of the memory. A person skilled in the art also knows that, in addition to implementing the controller by using the computer readable program code, logic programming can be performed on method steps to allow the controller to implement the same function in forms of the logic gate, the switch, the application-specific integrated circuit, the programmable logic controller, and the built-in microcontroller. Therefore, the controller can be considered as a hardware component, and a device configured to implement various functions in the controller can also be considered as a structure in the hardware component. Alternatively, the device configured to implement various functions can even be considered as both a software module implementing the method and a structure in the hardware component.

[0066] The system, device, module, or unit illustrated in the previous implementations can be implemented using a computer chip or an entity, or can be implemented using a product having a certain function. A typical implementation device is a computer. A specific form of the computer can be a personal computer, a laptop computer, a cellular phone, a camera phone, an intelligent phone, a personal digital assistant, a media player, a navigation device, an email transceiver device, a game console, a tablet computer, a wearable device, or any combination thereof.

[0067] For convenience of description, the above devices are described separately in terms of their functions. Certainly, functions of the units can be implemented in the same or different software or hardware when the present specification is implemented.

[0068] A person skilled in the art should understand that the implementations of the present specification can be provided as methods, systems, or computer program products. Therefore, the present specification can take a form of complete hardware implementations, complete software implementations, or implementations combining software and hardware. Further, the present specification can take a form of a computer program product implemented on one or more computer-usable storage media (including but not limited to disk storage, CD-ROM, and optical storage) containing computer-usable program code.

[0069] The present specification is described with reference to the flowcharts and/or block diagrams of the method, the device (system), and the computer program product according to the implementations of the present specification. It is worthwhile to note that computer program instructions can be used to implement each process and/or each block in the flowcharts and/or the block diagrams and a combination of a process and/or a block in the flowcharts and/or the block diagrams. These computer program instructions can be provided for a general-purpose computer, a dedicated computer, an embedded processor, or a processor of another programmable data processing device to generate a machine, so the instructions executed by the computer or the processor of the another programmable data processing device generate a device for implementing a specific function in one or more processes in the flowcharts and/or in one or more blocks in the block diagrams.

[0070] These computer program instructions can be stored in a computer readable memory that can instruct the computer or the another programmable data processing device to work in a specific way, so the instructions stored in the computer readable memory generate an artifact that includes an instruction device. The instruction device implements a specific function in one or more processes in the flowcharts and/or in one or more blocks in the block diagrams.

[0071] These computer program instructions can be loaded onto the computer or another programmable data processing device, so that a series of operations and steps are performed on the computer or the other programmable device, thereby generating computer-implemented processing. Therefore, the instructions executed on the

computer or the other programmable device provide steps for implementing a specific function in one or more processes in the flowcharts and/or in one or more blocks in the block diagrams.

5 [0072] In a typical configuration, a computing device includes one or more processors (CPUs), an input/output interface, a network interface, and a memory.

[0073] The memory can include a non-persistent memory, a random access memory (RAM), a non-volatile memory, and/or another form that are in a computer readable medium, for example, a read-only memory (ROM) or a flash memory (flash RAM). The memory is an example of the computer readable medium.

10 [0074] The computer readable medium includes persistent, non-persistent, movable, and unmovable media that can store information using any method or technology. The information can be a computer readable instruction, a data structure, a program module, or other data. Examples of the computer storage medium include but are not limited to a phase change random access memory (PRAM), a static random access
15 memory (SRAM), a dynamic random access memory (DRAM), another type of RAM, a ROM, an electrically erasable programmable read-only memory (EEPROM), a flash memory or another memory technology, a compact disc read-only memory (CD-ROM), a digital versatile disc (DVD) or another optical storage, a cassette magnetic tape, a magnetic tape/magnetic disk storage, another magnetic storage
20 device, or any other non-transmission medium. The computer storage medium can be used to store information accessible by a computer device. Based on the definition in the present specification, the computer readable medium does not include transitory media such as a modulated data signal and carrier.

[0075] It is also worthwhile to note that terms “include,” “include” or any other
25 variant is intended to cover non-exclusive inclusion, so that processes, methods, commodities or devices that include a series of elements include not only those elements but also other elements that are not explicitly listed, or elements inherent in such processes, methods, commodities or devices. An element described by “includes a ...” further includes, without more constraints, another identical element in the
30 process, method, product, or device that includes the element.

[0076] The present specification can be described in the general context of computer executable instructions executed by a computer, for example, a program module. Generally, the program module includes a routine, a program, an object, a

component, a data structure, etc. executing a specific task or implementing a specific abstract data type. The present specification can also be practiced in distributed computing environments. In the distributed computing environments, tasks are performed by remote processing devices connected through a communications
5 network. In a distributed computing environment, the program module can be located in both local and remote computer storage media including storage devices.

[0077] The above descriptions are merely examples of the present specification and are not intended to limit the present specification. For a person skilled in the art, the present specification can be subject to various modifications and variations. Any
10 modification, equivalent replacement, or improvement made without departing from the spirit and principle of the present specification shall fall within the scope of the claims.

CLAIMS

What is claimed is:

1. A method, comprising:
 - identifying a payment process corresponding to a payment service, the payment
 - 5 process including a logic flow of steps each corresponding to a functional operation;
 - compiling each of a plurality of programming language-based descriptions into a
 - distinct computer-executable code segment of a plurality of code segments, wherein
 - the plurality of programming language-based descriptions each describe a distinct
 - functional operation of a plurality of functional operations usable to implement the
 - 10 payment process and, wherein the plurality of code segments correspond to a plurality
 - of states of a finite state machine;
 - determining a transition relationship between states of the plurality of states
 - based, at least in part, on the logic flow of steps of the payment process;
 - generating a state transition table based, at least in part, on the plurality of states
 - 15 of the finite state machine and the transition relationship between the states;
 - storing the state transition table in a cache of a computer-based payment system
 - that implements the finite state machine; and
 - executing the payment process based, at least in part, on configuring the finite
 - state machine by retrieving the state transition table from the cache at a runtime of the
 - 20 computer-based payment system.
2. The method according to claim 1, wherein the plurality of states include at
- least one state that corresponds to two or more steps of the logic flow.
- 25 3. The method according to claim 1, further comprising:
 - writing the state transition table to a configuration file of the payment system for
 - the payment system to retrieve the state transition table from the configuration file of
 - the electronic payment system at a startup.
- 30 4. The method of claim 1, further comprising:
 - determining, through the finite state machine, a first state of the payment service;
 - determining that a first transition condition for triggering a transition from the
 - first state to a second state is met;

in response to the determining that the first transition condition is met, transitioning the payment service from the first state to the second state using the finite state machine; and

performing a functional operation corresponding to the second state.

5

5. The method according to claim 4, wherein the determining that the first transition condition is met includes querying the state transition table to retrieve the first transition condition.

10

6. The method according to claim 5, wherein the determining that the first transition condition is met includes:

retrieving, by the payment system, the state transition table from a configuration file of the electronic payment system at a startup.

15

7. A device, comprising one or more processors and a memory storing executable instructions of a program executable by the one or more processors to perform acts including:

identifying a payment process corresponding to a payment service, the payment process including a logic flow of steps each corresponding to a functional operation;

20

compiling each of a plurality of programming language-based descriptions into a distinct computer-executable code segment of a plurality of code segments, wherein the plurality of programming language-based descriptions each describe a distinct functional operation of a plurality of functional operations usable to implement the payment process and, wherein the plurality of code segments correspond to a plurality

25

of states of a finite state machine;

determining a transition relationship between states of the plurality of states based, at least in part, on the logic flow of steps of the payment process;

generating a state transition table based, at least in part, on the plurality of states of the finite state machine and the transition relationship between the states;

30

storing the state transition table in a cache of a computer-based payment system that implements the finite state machine; and

executing the payment process based, at least in part, on configuring the finite state machine by retrieving the state transition table from the cache at a runtime of the

computer-based payment system.

8. The device according to claim 7, wherein the plurality of states include at least one state that corresponds to two or more steps of the logic flow.

5

9. The device according to claim 7, wherein the executing the payment process is further based on conditions for transition between the states.

10. The device of claim 7, wherein the acts further include:
10 determining, through the finite state machine, a first state of the payment service;
determining that a first transition condition for triggering a transition from the first state to a second state is met;
in response to the determining that the first transition condition is met, transitioning the payment service from the first state to the second state using the
15 finite state machine; and
performing a functional operation corresponding to the second state.

11. A non-transitory computer-readable medium storing contents that, when executed by one or more processors, cause the one or more processors to perform
20 actions comprising:

identifying a payment process corresponding to a payment service, the payment process including a logic flow of steps each corresponding to a functional operation;
compiling each of a plurality of programming language-based descriptions into a distinct computer-executable code segment of a plurality of code segments, wherein
25 the plurality of programming language-based descriptions each describe a distinct functional operation of a plurality of functional operations usable to implement the payment process and, wherein the plurality of code segments correspond to a plurality of states of a finite state machine;
determining a transition relationship between states of the plurality of states
30 based, at least in part, on the logic flow of steps of the payment process;
generating a state transition table based, at least in part, on the plurality of states of the finite state machine and the transition relationship between the states;
storing the state transition table in a cache of a computer-based payment system

that implements the finite state machine; and

executing the payment process based, at least in part, on configuring the finite state machine by retrieving the state transition table from the cache at a runtime of the computer-based payment system.

5

12. The computer-readable medium according to claim 11, wherein the plurality of states include at least one state that corresponds to two or more steps of the logic flow.

10

13. The computer-readable medium according to claim 11, further comprising:
writing the state transition table to a configuration file of the payment system for the payment system to retrieve the state transition table from the configuration file of the electronic payment system at a startup.

15

14. The computer-readable medium of claim 11, wherein in the actions further comprise:

determining, through the finite state machine, a first state of the payment service;

determining that a first transition condition for triggering a transition from the first state to a second state is met;

20

in response to the determining that the first transition condition is met, transitioning the payment service from the first state to the second state using the finite state machine; and

performing a functional operation corresponding to the second state.

25

15. The computer-readable medium according to claim 14, wherein the determining that the first transition condition is met includes querying the state transition table to retrieve the first transition condition.

30

16. The computer-readable medium according to claim 15, wherein the determining that the first transition condition is met includes:

retrieving, by the payment system, the state transition table from a configuration file of the electronic payment system at a startup.