



- (51) **International Patent Classification:**
G06Q 20/10 (2012.01) *H04L 29/06* (2006.01)
- (21) **International Application Number:**
PCT/US2020/026036
- (22) **International Filing Date:**
31 March 2020 (31.03.2020)
- (25) **Filing Language:** English
- (26) **Publication Language:** English
- (30) **Priority Data:**
16/428,611 31 May 2019 (31.05.2019) US
- (71) **Applicant:** MICROSOFT TECHNOLOGY LICENSING, LLC [US/US]; One Microsoft Way, Redmond, Washington 98052-6399 (US).
- (72) **Inventors:** TURGMAN, Ami; Microsoft Technology Licensing, LLC, One Microsoft Way, Redmond, Washington 98052-6399 (US). DINERSTEIN, Yosef; Microsoft Technology Licensing, LLC, One Microsoft Way, Redmond, Washington 98052-6399 (US). PODDAR, Ashwarya; Microsoft Technology Licensing, LLC, One Microsoft Way, Redmond, Washington 98052-6399 (US).
- (74) **Agent:** SWAIN, Cassandra T. et al.; Microsoft Technology Licensing, LLC, One Microsoft Way, Redmond, Washington 98052-6399 (US).
- (81) **Designated States** (*unless otherwise indicated, for every kind of national protection available*): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.
- (84) **Designated States** (*unless otherwise indicated, for every kind of regional protection available*): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

(54) **Title:** SMART CONTRACT TEMPLATE META-PROGRAMMING SYSTEM AND METHOD

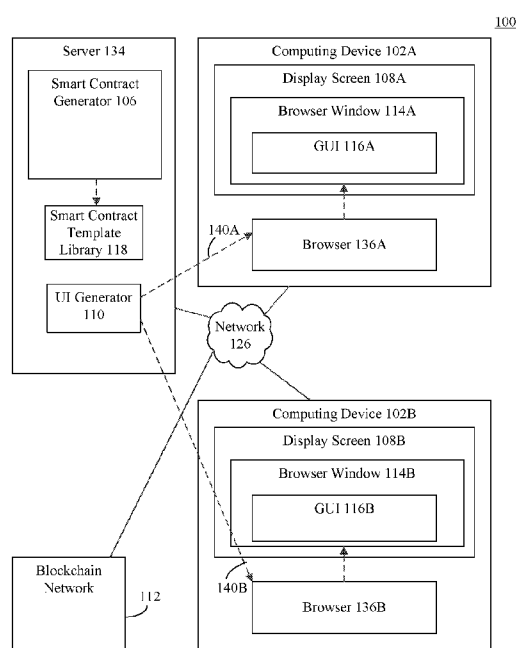


FIG. 1

(57) **Abstract:** Embodiments described herein are configured to generate a smart contract deployed to a distributed ledger platform. In particular, a first party and a second party select a smart contract template and a user interface user interface is thereafter presented to the parties for inclusion of negotiable contract elements. Upon agreement, a smart contract is thereafter generated by augmenting smart contract template programming code to include the included contract elements, and the code is compiled and deployed to a smart contract platform such as, for example, a blockchain. Embodiments include meta-instructions in the smart contract programming code that at least in part determines how the specified contract elements are included in the smart contract programming code. In embodiments, the meta-instructions are embedded in comments in the smart contract programming code.

Declarations under Rule 4.17:

- *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*
- *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

Published:

- *with international search report (Art. 21(3))*

SMART CONTRACT TEMPLATE META-PROGRAMMING SYSTEM AND METHOD

BACKGROUND

5 **[0001]** Mobile device technology has evolved to the point where useful business applications may be deployed to smartphones. For example, “gig economy” companies such as Uber and Lyft offer smartphone apps that connect drivers with riders. Similarly, companies like Postmates and DoorDash offer similar apps for providing food ordering and delivery to consumers. In each case, a consumer seeking goods or services uses an app to
10 order the goods or services they desire. The order is thereafter relayed by the gig company to an independent worker, or ‘freelancer’, who uses the app provided by the gig company to receive the gig (i.e., the task or project).

[0002] With these apps, gig companies make it easy for freelancers to find a quick, short term job. Freelancers enjoy the flexibility of the arrangement which can provide them with
15 greater control of their time and finances. Gig companies derive certain benefits as well. First, such companies take a percentage of the overall transaction cost, or impose a fixed fee for putting the freelancer and consumer together. Gig companies also enjoy operating without the overhead of a large number of employees.

[0003] The gig economy is not, however, without its problems. Freelancers are under
20 increasing financial pressure as companies continually seek a larger share of each transaction. Consumers of course also bear such pressures in the form of higher prices. Eliminating the middleman (i.e., the gig company) and putting service providers and consumers directly in touch eliminates these issues.

[0004] Recently, with the advent of large social networks, it is possible for consumers to
25 easily find service providers without needing an app or other middleman-provided service. For example, one may search on social networks such as LinkedIn for people having the required skill set, and then initiate an engagement with such people directly. For example, one could find a person who specializes in translating documents from one language to another, and then hire them for a one-time document translation project. As another
30 example, social networks such as Upwork™ and Fiverr® may be used to connect consumers with service providers.

[0005] These methods of distributing gig work are also not without problems. For one, there is an issue of how a service provider will get paid, and whether the service provider can trust they will, in fact, get paid when the work is complete. Likewise, consumers would

like an easy way to pay for gigs, and be certain they will receive their services and not be in danger of being scammed. Moreover, conflicts between the parties may arise without a ready means for resolving the dispute.

[0006] Cryptocurrency offers a decentralized means for making direct payments between one or more parties. Distributed ledgers often form the underlying technology of many cryptocurrencies. A blockchain is one type of distributed ledger, and Ethereum is one type of cryptocurrency based thereon. The implementation of Ethereum, and other cryptocurrencies, also provides for so-called 'smart contracts' to operate on and through the blockchain. A smart contract is computer code running on top of the blockchain. The computer code includes rules established by the contracting parties that govern how the transaction will progress and be completed.

SUMMARY

[0007] This Summary is provided to introduce a selection of concepts in a simplified form that are further described below in the Detailed Description. This Summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended to be used to limit the scope of the claimed subject matter.

[0008] Methods, systems, apparatuses, and computer-readable storage mediums described herein are configured to generate and allow interaction with a smart contract deployed to a distributed ledger platform. In embodiments, a first party and a second party initiate an engagement and negotiate the terms of the engagement. The first party and second party may also jointly choose a mediator from amongst mediators available on a social networking platform. Thereafter, a smart contract template may be chosen based on the terms, and a user interface is presented to the first and second party for inclusion of negotiable elements. Upon agreement, the first and second party may accept the agreement and a smart contract is thereafter generated and deployed to a smart contract platform such as, for example, a blockchain. In conjunction with deployment, embodiments permit the escrow of funds to the smart contract platform. Embodiments of the deployed smart contract enable the first and second party to indicate completion or take other actions with respect to the engagement, or to initiate a dispute. After a dispute is initiated, the previously chosen mediator is thereafter enabled by the smart contract to mediate the dispute through one or more smart contract actions made available to the mediator. Such actions may comprise, for example, determining how escrowed funds ought to be disbursed between the parties.

[0009] In one example implementation, a selection of a smart contract template is accepted from one or more parties of a plurality of parties, the parties engaging in a

transaction and choosing the selected smart contract template based at least in part on transaction terms negotiated between the parties. Transaction terms are accepted from the parties corresponding to fillable elements of the smart contract template. The smart contract template is augmented to include the accepted transaction terms to provide an augmented smart contract template. A smart contract is generated corresponding to the augmented smart contract template. The smart contract is deployed to a blockchain.

[0010] Further features and advantages, as well as the structure and operation of various example embodiments, are described in detail below with reference to the accompanying drawings. It is noted that the example implementations are not limited to the specific embodiments described herein. Such example embodiments are presented herein for illustrative purposes only. Additional implementations will be apparent to persons skilled in the relevant art(s) based on the teachings contained herein.

BRIEF DESCRIPTION OF THE DRAWINGS/FIGURES

[0011] The accompanying drawings, which are incorporated herein and form a part of the specification, illustrate example embodiments of the present application and, together with the description, further serve to explain the principles of the example embodiments and to enable a person skilled in the pertinent art to make and use the example embodiments.

[0012] FIG. 1 shows a block diagram of a system for generating a smart contract, in accordance with an example embodiment.

[0013] FIG. 2 is an example smart contract template configuration file, in accordance with an embodiment.

[0014] FIG. 3 is an example code fragment of a smart contract template source code file, in accordance with an embodiment.

[0015] FIG. 4 is an example GUI screen that depicts a human-readable rendering of a smart contract template, in accordance with an embodiment.

[0016] FIG. 5 shows a flowchart of a method for deploying a smart contract to a blockchain maintained by a computing platform, in accordance with an example embodiment.

[0017] FIG. 6 shows a block diagram of a system for deploying a smart contract to a blockchain maintained by a computing platform, in accordance with an example embodiment.

[0018] FIG. 7 shows an example GUI screen in which an immutable version of an agreed upon smart contract is shown, in accordance with an example embodiment.

[0019] FIG. 8 shows a block diagram of a system for presenting a user interface

representation of an action a user is enabled to take with respect to a smart contract, in accordance with an example embodiment.

[0020] FIG. 9A is an example GUI screen in which user interface elements are displayed to a service provider based on querying a smart contract on behalf of the service provider, in accordance with an example embodiment.

[0021] FIG. 9B is an example GUI screen in which user interface elements are displayed to a consumer based on querying a smart contract on behalf of the consumer, in accordance with an example embodiment.

[0022] FIG. 10 is another example GUI screen in which user interface elements are displayed to a service provider based on querying a smart contract after the service provider has signed the smart contract, in accordance with an example embodiment.

[0023] FIG. 11 is yet another example GUI screen in which user interface elements are displayed to a consumer based on querying a smart contract after the service provider has signed the smart contract, in accordance with an example embodiment.

[0024] FIG. 12 is an example GUI screen in which user interface elements are displayed to a consumer based on querying a smart contract after all parties have signed the smart contract, in accordance with an example embodiment.

[0025] FIG. 13 is an example GUI screen in which user interface elements are displayed to a mediator based on querying a smart contract after a party has initiated a dispute, in accordance with an example embodiment.

[0026] FIG. 14 is an example GUI screen in which user interface elements are displayed to a consumer based upon querying an alternative smart contract, in accordance with an example embodiment.

[0027] FIG. 15 is an example GUI screen that depicts a conversation between two users, in accordance with an example embodiment.

[0028] FIG. 16 shows a flowchart of a method for generating a smart contract from a smart contract template, in accordance with an example embodiment.

[0029] FIG. 17 shows a flowchart of a method for presenting a user interface enabling a smart contract action to be performed, in accordance with an example embodiment.

[0030] FIG. 18 is a block diagram of an exemplary mobile device that may implement embodiments described herein.

[0031] FIG. 19 is a block diagram of an example processor-based computer system that may be used to implement various embodiments.

[0032] The features and advantages of the implementations described herein will become

more apparent from the detailed description set forth below when taken in conjunction with the drawings, in which like reference characters identify corresponding elements throughout. In the drawings, like reference numbers generally indicate identical, functionally similar, and/or structurally similar elements. The drawing in which an element first appears is indicated by the leftmost digit(s) in the corresponding reference number.

DETAILED DESCRIPTION

I. Introduction

[0033] The present specification and accompanying drawings disclose numerous example implementations. The scope of the present application is not limited to the disclosed implementations, but also encompasses combinations of the disclosed implementations, as well as modifications to the disclosed implementations. References in the specification to “one implementation,” “an implementation,” “an example embodiment,” “example implementation,” or the like, indicate that the implementation described may include a particular feature, structure, or characteristic, but every implementation may not necessarily include the particular feature, structure, or characteristic. Moreover, such phrases are not necessarily referring to the same implementation. Further, when a particular feature, structure, or characteristic is described in connection with an implementation, it is submitted that it is within the knowledge of persons skilled in the relevant art(s) to implement such feature, structure, or characteristic in connection with other implementations whether or not explicitly described.

[0034] In the discussion, unless otherwise stated, adjectives such as “substantially” and “about” modifying a condition or relationship characteristic of a feature or features of an implementation of the disclosure, should be understood to mean that the condition or characteristic is defined to within tolerances that are acceptable for operation of the implementation for an application for which it is intended.

[0035] Furthermore, it should be understood that spatial descriptions (e.g., “above,” “below,” “up,” “left,” “right,” “down,” “top,” “bottom,” “vertical,” “horizontal,” etc.) used herein are for purposes of illustration only, and that practical implementations of the structures described herein can be spatially arranged in any orientation or manner.

[0036] Numerous example embodiments are described as follows. It is noted that any section/subsection headings provided herein are not intended to be limiting. Implementations are described throughout this document, and any type of implementation may be included under any section/subsection. Furthermore, implementations disclosed in any section/subsection may be combined with any other implementations described in the

same section/subsection and/or a different section/subsection in any manner.

II. Example Implementations

[0037] Embodiments described herein provide methods and systems for generating and interacting with a smart contract deployed to a distributed ledger such as, for example, a blockchain. A user interface (“UI”) is custom-generated for each party to the smart contract. In particular, a UI is generated that permits a party to select a smart contract template suitable for a given engagement, for suitable contract terms to be provided by one or more parties, and for a smart contract to thereafter be generated and deployed to a distributed ledger. Embodiments also enable parties to specify a mediator to resolve any disputes that may arise during the course of the engagement, and to generate a smart contract including smart contract actions that only the mediator is enabled to perform on behalf of the parties.

[0038] In embodiments, a UI representation of the current state of the smart contract may be provided to each party (including a mediator if included in the transaction). Hereinafter, it should be understood that the term “party” or “parties” may also include a mediator, and should not be construed to mean only the principals to the engagement. Furthermore, although embodiments described herein are often couched in terms of two parties to the smart contract, it should be understood that such embodiments are merely exemplary, and other embodiments may in principle include any number of parties. The UI representation provided to a particular party reflects actions that party is enabled to perform and/or has already taken with respect to the smart contract, and likewise may reflect actions taken by other parties (e.g., an action taken by a mediator). In an embodiment, a user experience generator may query the smart contract to determine the current state of the smart contract, and to derive therefrom which UI representations are to be generated and provided to each party via his respective user interface. For instance, the smart contract may comprise functions and/or user experience (UX) annotations that indicate to the user experience generator which user interface representations should be provided to a particular user based on at least one of the identity of the party and a state in which the smart contract is in. Each time the state of the smart contract changes, the smart contract may provide an event notification to the user experience generator, which prompts the user experience generator to query the smart contract to determine the allowed actions for a particular user.

[0039] By incorporating the functions and/or UX annotations within the smart contract itself, the user experience generator is not required to internally maintain a state machine that tracks the state of the smart contract. Instead, the user interface leverages the state maintained by the smart contract itself, and simply queries the smart contract to determine

whether a particular action is allowable for a particular party to the smart contract whenever the state of the smart contract changes. This advantageously removes any state synchronization required between a user interface and the smart contract. Accordingly, the functioning of computers on which the user experience generator executes is improved. In particular, processor cycles, memory, etc. of such computers are conserved as such computers do not need to maintain the state machine and process the state changes of the state machine. Instead, as described herein, the user experience generator simply queries the smart contract itself to determine its state. Moreover, because the manner in which user interface elements to be rendered are determined is greatly simplified, such user interface elements are rendered faster.

[0040] FIG. 1 shows a system 100 for generating a smart contract, according to an example embodiment. As shown in FIG. 1, system 100 includes a computing device 102A, a computing device 102B, a server 134, and a blockchain network 112. Server 134 includes a smart contract generator 106, a smart contract template library 118 (e.g., in storage), and a user interface (UI) generator 110. Computing device 102A includes a display screen 108A and a browser 136A, and computing device 102B includes a display screen 108B and a browser 136B. System 100 is described as follows. It is noted that while system 100 includes two computing devices 102A and 102B, system 100 may include any number of computing devices. For example, one such computing device may be used by each party to a smart contract, including a mediator if specified.

[0041] Computing devices 102A and 102B may be any type of stationary or mobile computing device, including a mobile computer or mobile computing device (e.g., a Microsoft® Surface® device, a laptop computer, a notebook computer, a tablet computer such as an Apple iPad™, a netbook, etc.), a wearable computing device (e.g., a head-mounted device including smart glasses such as Google® Glass™, etc.), or a stationary computing device such as a desktop computer or PC (personal computer). Server 134 may include one or more server devices and/or other computing devices. For example, server 124 may include a Web server, an application server, and/or a database.

[0042] Computing devices 102A and 102B and server 134 may each include at least one network interface that enables communications with each other over network 126. Examples of such a network interface, wired or wireless, include an IEEE 802.11 wireless LAN (WLAN) wireless interface, a Worldwide Interoperability for Microwave Access (Wi-MAX) interface, an Ethernet interface, a Universal Serial Bus (USB) interface, a cellular network interface, a Bluetooth™ interface, a near field communication (NFC) interface, etc.

Further examples of network interfaces are described elsewhere herein. Examples of network 126 include a local area network (LAN), a wide area network (WAN), a personal area network (PAN), and/or a combination of communication networks, such as the Internet.

[0043] Blockchain network 112 may comprise a decentralized, distributed blockchain computing platform maintained by a plurality of nodes (e.g., computing devices). Blockchain network 112 enables developers to build and execute decentralized applications, such as smart contracts, that run on blockchain technology. Any type of distributed ledger platform that offers smart contracts may be utilized for blockchain network 112. An example of blockchain network 112 includes, but is not limited to, Ethereum. Blockchain network 112 may be communicatively coupled to server 134 and/or computing devices 102A and 102B via network 126.

[0044] Smart contract generator 106 is configured to be operated/interacted with to create a smart contract between at least two users (e.g., a service provider and a consumer), wherein the smart contract is configured to be executed on blockchain network 112. For instance, a first user (e.g., the service provider) may access smart contract generator 106 by interacting with an application at computing device 102A capable of accessing a network-based application, such as browser 136A, and a second user (e.g., the consumer) may access smart contract generator 106 by interacting with an application at computing device 102B capable of accessing a network-based application, such as browser 136B. The first user may use browser 136A to traverse a network address (e.g., a uniform resource locator) to smart contract generator 106, which invokes a GUI 116A (e.g., a web page) in a browser window 114A. Similarly, the second user may use browser 136B to traverse the network address to smart contract generator 106, which invokes GUI 116B in a browser window 114B. It is noted that any number of users may participate in creating a smart contract via smart contract generator 106. For example, and as will be described in more detail below, the service provider and consumer may elect to include mediation terms in the smart contract, including identification of a designated mediator. In such instances, the selected mediator may likewise participate in creating the smart contract inasmuch as the mediator must signal his/her agreement to mediate any disputes per the express terms of the contract (e.g., by signing the contract).

[0045] UI generator 110 is configured to transmit GUI information 140A (e.g., one or more web pages, image content, user interface elements, code (e.g., JavaScript) etc.) to browser 136A to be displayed and/or executed as GUI 116A in browser window 114A of display screen 108A and transmit GUI information 140B to browser 136B to be displayed

as GUI 116B in browser window 114B of display screen 108B. GUI 116A and GUI 116B may be interacted with by users to draft a smart contract as described in further detail herein below.

[0046] In embodiments, users may draft a smart contract based on a smart contract
5 template selected by one or more of the users from a plurality of smart contract templates stored in smart contract template library 118. Smart contract templates may be stored as one or more files, in an embodiment. For example, a first file may specify template language of the contract (e.g., various predetermined terms, conditions, clauses, etc.) and is generally referred to herein as a definition file. In embodiments, a second file such as a source code
10 file may also be included in a smart contract template. Examples of such files are described in turn herein below.

[0047] There are numerous ways to construct a definition file according to embodiments. Consider, for example, the configuration file 200 as shown in FIG. 2. Configuration file 200 is a text file in JavaScript Object Notation (JSON) format, and includes attributes 202-206
15 and JSON object 208. Attributes 202-206 include 'title', 'description' and 'doc' attribute/value pairs, respectively. The title attribute 202 specifies a title or common name for the smart contract template, whereas the description attribute 204 includes a short description of the scope of the smart contract template. The doc attribute 206 includes a human-readable form of the smart contract corresponding to the smart contract template,
20 and itself includes user-fillable or definable fields such as, for example, {provider.displayName}. For the sake of brevity, doc attribute 206 as depicted in FIG. 2 is truncated, and does not include the full text of the human-readable smart contract template. An embodiment that employs a user-readable version of the smart contract template, along with user-fillable fields included therein, will be described in detail below with respect to
25 the example GUI screen 400 of FIG. 4 which includes the full, non-truncated version of doc attribute 206. Configuration file 200 also includes a params object 208 which includes various predetermined terms used to display the smart contract template, and to compile and deploy a smart contract as will be described in greater detail below.

[0048] Although configuration file 200 is described above in terms of a text file in JSON
30 format, it should be understood this is but one example embodiment, and other formats, whether text or binary, are possible in other embodiments. For example, the definition files may be formatted in accordance with an Extensible Markup Language (XML) format or any other format suitable for organizing and/or transmitting data.

[0049] A second file associated with a smart contract template may comprise a source

code template of a smart contract. The second file may be referred to as a source code template file. In an embodiment, the source code template file specifies a set of rules, in the form of smart contract programming code, under which the parties to the smart contract agree to interact with each other. If and when the rules are met, the agreement is automatically enforced. The smart contract code facilitates, verifies, and enforces the negotiation or performance of the agreement. The source code may be written in a programming language suitable for compilation and execution by a virtual machine executing in blockchain network 112. For instance, in an embodiment in which blockchain network 112 is an Ethereum blockchain network, the source code may be written in the Solidity programming language, although the embodiments described herein are not so limited. The source code may reference the data and/or variables specified by the definition file.

[0050] A source code template file may also include further code, instructions or meta-programming that specifies how the smart contract programming code included therein may be augmented or otherwise modified to include variable terms. For example, and as will be described in further detail below, during deployment of the smart contract to blockchain network 112, meta-instructions included in a source code template file may be used to determine at least in part how the data and/or variables corresponding to the user-fillable fields described above should be inserted in placeholders within the smart contract programming code.

[0051] Numerous types exist of smart contract programming code and techniques for embedding meta-code therein. Consider, the example code fragment 300 of FIG. 3. Code fragment 300 of FIG. 3 comprises an excerpt from a source code template file corresponding to an example smart contract template. Code fragment 300 is comprised of Solidity programming code, and further includes meta-language markup to permit transformation of the smart contract template code into customized Solidity code that reflects the parties to the contract, as well as the negotiated contract terms. Code fragment 300 includes declaration blocks 302 and 304, an example smart contract function `sign()` function 306 and its corresponding `cancall()` function, `cancall_sign()` function 308. Each of `sign()` function 306 and `cancall_sign()` function 308 will be described in further detail below. Discussion now turns to declaration blocks 302 and 304.

[0052] Definition block 302 includes declarations of the Ethereum address for each of a consumer, provider and mediator. For example, the consumer Ethereum address is defined as:

address constant public consumer = **/*<ph>consumer.publicAddress*/0/*</ph>*/;**

[0053] There are two things to note in this declaration shown above. First, the declaration includes inline comments (shown **underlined and bold**) that will be ignored by a Solidity compiler. Accordingly, a Solidity compiler would see the following equivalent declaration

5 if the bare smart contract template were provided to the compiler:

address constant public consumer = 0;

[0054] Second, since code fragment 300 corresponds to a smart contract template rather than a specific instance of smart contract code, the declaration does not include an actual address. Instead, and as described more detail herein below, after users select a smart

10 contract template and negotiate the contract terms, embodiments are configured to process the smart contract template and substitute known terms for the placeholders delimited by inline comments as shown above. For example, embodiments may be configured to perform a meta-compilation operation whereby the smart contract template code is scanned for inline comments that themselves include specific identifiers, and to substitute the actual data or

15 other information there. Thus, in this example, once the consumer is known, and further supposing the consumer's Ethereum address is, for example, 0x1C039c9cb1965113065f679B55C429Cc77FA664e, the meta-compiler may detect the consumer.publicAddress field in between the <ph> </ph> tags in the embedded comments, and substitute the known address to yield the following line of Solidity code:

20 address constant public consumer = 0x1C039c9cb1965113065f679B55C429Cc77FA664e;

[0055] Substitution of other configurable/declarable contract terms into the smart contract template code is readily accomplished by the same meta-compilation process. For example, consider declaration block 304 that includes declarations for a reward, refund, deposit and mediator fee rate. For example, the smart contract template code declaration for the payment

25 reward for the smart contract (and again omitted the inline comments) is as follows:

uint constant public reward = 0;

[0056] After negotiation of a public reward of 100 finney (e.g., as described below in conjunction with FIGS. 4-7, the meta-compiler may produce Solidity code as follows:

uint constant public reward = 100 finney;

30 [0057] With continued reference to system 100 of FIG. 1, the selected contract template may be rendered via GUI 116A and/or GUI 116B, UI generator 110 may provide information 140A and 140B to GUI 116A and GUI 116B, respectively, that causes GUI 116A and GUI 116B to render the human-readable portions of a smart contract template in accordance with the associated definition file. For instance, the smart contract template

terms may be displayed via GUI 116A and/or GUI 116B as human-readable text, and the variables may be displayed as user-interactive elements (e.g., text boxes, pull-down menus, etc.) that enable the users to specify user-specifiable terms of the smart contract.

[0058] Any number of users may be enabled to interact with contract generator 106 via their respective GUIs to cooperatively negotiate contract terms of the smart contract. For instance, users may be presented with a smart contract template, as selected by at least one of the users in their respective GUI (i.e., GUI 116A, GUI 116B, etc.). For example, FIG. 4 is an example GUI screen 400 that depicts a human-readable rendering of a smart contract template 402, according to an embodiment. As shown in FIG. 4, a user, using a browser (e.g., browser 136A), has traversed to a network address (e.g., www.example.com) corresponding to smart contract generator 106 and has selected template contract 402. GUI 116A renders smart contract template 402 in browser window 114A in accordance with the smart contract template's associated definition file. The same smart contract template may be displayed for a second user that agreed to collaboratively draft a contract with the first user. For instance, the second user, using browser 136B may traverse to the network address corresponding to smart contract generator 106 and view smart template contract 402 via browser window 114B.

[0059] As shown in FIG. 4, smart contract template 402 comprises a human-readable text and a plurality of user interface elements 404, 406, 408, 410, 412, 414, 416, 418, and 420. In embodiments, smart contract template 402 and the abovementioned user interface elements may correspond to doc attribute 206 as described above in conjunction with FIG. 2. The human-readable text represents template language in the associated definition file. Each of user interface elements 404, 406, 408, 410, 412, 414, 416, 418, and 420 represents a variable in the definition file. Each of user interface elements 404, 406, 408, 410, 412, 414, 416, and 418 may be text boxes that enable a user to specify certain terms of the contract. For example, user interface element 404 enables a user to specify the name of the provider, user interface element 406 enables a user to specify the type of service to be rendered, user interface 408 enables a user to specify the name of the consumer, user interface element 410 enables a user to specify the amount of funds to be paid to the provider, user interface element 412 enables a user specify the a refund amount to be paid by the consumer, user interface element 414 enables a user to specify a deposit amount to be paid by the provider, user interface element 416 enables a user to specify a percentage of the reward that a mediator is given upon dispute resolution, and user interface element 418 enables a user to specify the name of the mediator.

[0060] In an embodiment, user interface element 418 may instead comprise a drop down pre-populated with a list of available mediators. The list may reflect, for example, user preferences or mediators that a user has previously employed. Alternatively, the list may comprise a curated list of mediators, with such mediators being included base on, for example, a third-party vetting process. Alternatively, the list of mediators may include a reputation score or some other indicia of reputation associated with the mediator. For example, embodiments disclosed herein may be hosted by a social network platform, and mediators may be selected from among the mediators with a profile on the platform, and based at least in part on the reputation of the mediator (e.g., as reflected by upvotes or reviews of some type).

[0061] User interface element 420 may be a pulldown menu that enables a user to specify the type of currency to be used for the payment amount (e.g., cryptocurrency (or a unit thereof) supported by blockchain network 112). It is noted that the usage of text boxes for user interface elements 404, 406, 408, 410, 412, 414, 416, 418 and the usage of a pulldown menu for user interface 420 is purely exemplary and that other types of user interface elements may be utilized to enable a user to specify contract terms.

[0062] Users may collaboratively fill out smart contract template 402 via a communication session (e.g., via phone, chat, video chat, etc.), where each user is enabled to specify the terms of smart contract template 402 based on the conversation between the users. When a particular user makes a modification to a contract term, the modification is also reflected on the other users' respective GUI. For instance, if a first user specifies a provider name via user interface 404 using GUI 116A, UI generator 110 provides the specified provider name to the second user's GUI (e.g., GUI 116B), which is updated to reflect the specified value.

[0063] After the users finalize the terms of the smart contract, the smart contract may be deployed to blockchain network 112 for execution thereby. For instance, a user may interact with a user interface element 422, which, when activated, deploys the finalized contract to blockchain network 112.

[0064] FIG. 5 shows a flowchart 500 of a method for deploying a smart contract to a blockchain maintained by a computing platform in accordance with an example embodiment. In an embodiment, flowchart 500 may be implemented by a system 600 shown in FIG. 6, although the method is not limited to that implementation. FIG. 6 shows a block diagram of system 600 for deploying a smart contract to a blockchain maintained by a computing platform in accordance with an example embodiment. System 600 is an example

of system 100, as described above with reference to FIG. 1. As shown in FIG. 6, system 600 comprises a server 634, which is an example of server 134. System 600 further includes, computing device 102A and blockchain network 112, as described above with reference to FIG. 1. Computing device 102B is not shown for brevity. Server 634, computing device 5 102A and blockchain network 112 are communicatively coupled via network 126. Network 626 is an example of network 126, as described above with reference to FIG. 1. As further shown in FIG. 4, server 634 comprises a smart contract generator 606 and a UI generator 610. Smart contract generator 606 comprises a variable replacer 602 and a compiler 604. As also shown in FIG. 6, browser 136A is configured to execute a deployer 608. Smart contract 10 generator 606 and UI generator 610 are examples of contract generator 106 and UI generator 110, as described above with reference to FIG. 1. Other structural and operational embodiments will be apparent to persons skilled in the relevant art(s) based on the discussion regarding flowchart 500 and system 600 of FIG. 6.

[0065] Flowchart 500 begins with step 502. In step 502, variable(s) of the smart contract 15 are replaced with one or more user-specified parameters. For example, with reference to FIG. 6, after the user-specified parameters of the smart contract being negotiated are finalized via smart content generator GUI 116A, the user-specified terms (shown as user-specified parameters 616) are provided to smart contract generator 106 via browser 136A. Variable replacer 602 of smart contract generator 606 may be configured to replace the 20 variables in the smart contract source code file (shown as source file 612) associated with the smart contract template with user-specified parameters 616. For example, and as described in greater detail above with respect to smart contract code fragment 302 of FIG. 3, variable replacer 602 may be configured to perform a meta-compilation of smart contract template code to augment such code per meta-instructions embedded therein.

[0066] In addition to replacing variables with user-specified terms, meta-compilation may 25 also process UX annotations. For example, and with reference to code fragment 300 of FIG. 3 described above, sign() function 306 may include the following comment:

/// Sign the contract. Transfer {value_money_/*<ph>unit*//*/} upon signing
Embodiments process this UX annotation in two ways. First, supposing that the parties have 30 agreed that the currency unit will be in finney, during meta-compilation 'finney' will be substituted into the comment using the process described above with respect to FIG. 3 to yield the following:

/// Sign the contract. Transfer {value_money_finney} upon signing

Second, the comment as modified is processed to an artifacts file as described herein below

at step 504, and which results in UX operations being performed with the specified units (i.e., finney in this example). Thus, comments may serve as a tool for UX meta-programming in addition to enabling linkage of contract template parameters.

[0067] In step 504, the contract is compiled with the user-specified parameters into a format suitable for the blockchain. For example, with reference to FIG. 6, compiler 604 may be configured to compile source code file 612 with the user-specified parameters 616 into a format suitable for the blockchain maintained by blockchain network 112. For instance, compiler 604 may compile source code file 612 into byte code 614. In an embodiment in which source code file 612 is written in the Solidity programming language, compiler 604 may be a Solidity-based compiler. In an embodiment in which blockchain network 112 is an Ethereum blockchain network, byte code 614 may be Ethereum Virtual Machine (EVM) byte code. The compilation process may also generate a compilation artifacts file 618. Compilation artifacts file 618 may comprise a description of smart contract methods and UX annotations included in the smart contract (which are further described below with reference to FIG. 8). Compilation artifacts file 618 may be stored in storage (e.g., a database) communicatively coupled to server 634. Compilation artifacts file 618 may be formatted in accordance with the JSON format; however, the embodiments described herein are not so limited.

[0068] In step 506, the compiled smart contract is caused to be deployed to the blockchain for execution thereby. For example, with reference to FIG. 6, smart contract generator 606 may provide byte code 614 to browser 136A. Deployer 608 executing in browser 136A may be configured to deploy the compiled smart contract (e.g., byte code 614) to the blockchain maintained by blockchain network 112 for execution thereby. For instance, a virtual machine (e.g., an Ethereum virtual machine (EVM) of blockchain network 112) may execute byte code 614. Smart contract generator 606 may provide byte code 614 to browser 136A via network 126 using an application programming interface (API), such as, but not limited to a Representational State Transfer (REST) API. Deployer 608 may provide byte code 614 to blockchain network 112 via network 126 using an API, such as a Metamask API.

[0069] After the smart contract has been deployed to blockchain network 112, the smart contract becomes immutable (i.e., the contract language and contract terms are no longer editable and are read-only). In accordance with an embodiment, the immutable version of the smart contract (shown as text 702) is displayed to all the parties to the smart contract (e.g., via GUI 116A, GUI 116B, etc.). For example, FIG. 7 shows an example GUI screen

700 in which an immutable version of the agreed upon smart contract is shown. As shown in FIG. 7, user interface elements 404, 406, 408, 410, 412, 414, 416, 418, and 420 (shown in FIG. 4) are no longer displayed. Instead, the parameters specified by a user for each of user interface elements 404, 406, 408, 410, 412, 414, 416, 418, and 420 are displayed in place thereof, and a noneditable, finalized version of the smart contract is shown.

[0070] In accordance with an embodiment, certain user interface representations may be generated and presented to the user after the smart contract is deployed to blockchain network 112 that provides a status with respect to the smart contract and/or enable an action to be taken with respect to the smart contract. Such actions include, but are not limited to, depositing funds (into escrow or otherwise), withdrawing funds, executing the smart contract, initiating a dispute, etc. The user interface elements that are displayed may vary depending on the user viewing the smart contract and/or the state in which the smart contract is in. The foregoing techniques are described below.

[0071] For instance, FIG. 8 shows a block diagram of system 800 for presenting a user interface representation of an action a user is enabled to take with respect to the smart contract in accordance with an example embodiment. System 800 is an example of system 100, as described above with reference to FIG. 1, and system 600, as described above with reference to FIG. 6. As shown in FIG. 8, system 800 comprises a server 834, which is an example of server 134, as described above with reference to FIG. 1, and server 634, as described above with reference to FIG. 6. System 800 also includes computing device 102A, computing device 102B and blockchain network 112, as described above with reference to FIGS. 1 and 6. Server 834, computing device 102A, computing device 102, and blockchain network 112 are communicatively coupled via a network 126. As further shown in FIG. 8, server 834 comprises a smart contract generator 806, which is an example of smart contract generator 106 and smart contract generator 606, as respectively described above with reference to FIGS. 1 and 6, and a UI generator 810, which is an example of UI generator 110 and 610, as described above with reference to FIGS. 1 and 6. As further shown in FIG. 8, a smart contract 804 is deployed to blockchain network 112 (in accordance with the description provided above with reference to FIGS. 5 and 6). For instance, smart contract 804 may correspond to byte code 614 described above with reference to FIG. 6. As also shown in FIG. 8, a UX generator 802A is shown executing in browser 136A and a UX generator 802B is shown executing in browser 136B.

[0072] UX generator 802A and 802B may be configured to determine which UI elements are to be presented to different users of smart contract generator 806 (e.g., via GUI 116A

and GUI 116B, respectively). UX generator 802 may determine the UI elements to be presented based on a state in which smart contract 804 is in and/or the identity of the user viewing smart contract 804.

[0073] The state in which smart contract 804 is in may be determined based on event notifications that are provided by smart contract 804 to UX generator 802A and/or 802B. For instance, each time an event occurs with respect to smart contract 804, smart contract 804 may provide an event notification 826 to UX generator 802A and 802B that identifies that an event has occurred (e.g., a state change has occurred, a user has signed smart contract 804, a user has deposited funds associated with smart contract 804, etc.). Examples of states include, but are not limited to a pending state, a sealed state, a disputed state, or a completed state. Smart contract 804 is in a pending state when at least one of the parties to smart contract 804 has not signed smart contract 804. Smart contract 804 is in a sealed state after each of the parties to smart contract 804 have signed smart contract 804. Smart contract 804 is in a disputed state when at least one of the parties have initiated a dispute with respect to smart contract 804. Smart contract 804 is in a completed state after all the obligations and/or duties of each party to smart contract 804 is completed. It is noted that that the states described above are purely exemplary and that smart contract 804 may comprise any number and/or types of states.

[0074] UX generator 802A and/or 802B may determine the identity of the user based on a user profile associated with the user. For instance, server 834 may maintain user profiles 808 for different users that utilize smart contract generator 806. A user may be required to login to a website that provides smart contract generator 806, for example, using a username and/or password. Smart contract generator 806 may maintain the login information when a user logs into the website and may associate the login information with the user's profile. The user profile may include information associated with the user (e.g., name, address, interests, Ethereum ID, etc.). UX generator 802A and/or 802B may access the user's profile upon the user logging into the website and determine the identity of the user based on the information included in that user's profile. Alternatively, browser 136A may store an identifier (e.g., a username) that identifies the user utilizing GUI 116A, and browser 136B may store an identifier that identifies the user utilizing GUI 116B. UX generator 802A may provide a query to browser 136A and UX generator 802A may provide a query to browser 136B and obtain the user identifier therefrom.

[0075] The code of smart contract 804 may comprise functions (e.g., procedures or methods) that, when executed perform a particular action with respect to the smart contract.

Such actions include, but are not limited to, signing the smart contract, depositing funds, withdrawing funds, executing the smart contract, initiating a dispute, etc. For example, sign() function 306 as shown in FIG. 3 is an example smart contract action that may invoked when a user signs the smart contract. Such functions may be referred to variously as smart contract methods, functions or actions. In other embodiments, however, smart contract procedures and/or methods need not perform actions, but may instead be invoked for other purposes (e.g., to obtain information from the contract). For example, and as will be described in more detail below, a smart contract method may query the smart contract to determine the names of each party for the purposes of generating and rendering GUI 116A or GUI 116B.

[0076] In embodiments, each of the smart contract methods may be associated with a complementary function that is also included in the code. Each complementary function may be utilized by UX generator 802A and/or 802B to determine whether or not a user interface element associated with the action of its corresponding smart contract method is to be provided to a particular user, thereby enabling that user to perform that action. For instance, when coding smart contract 804 (e.g., using the Solidity language), the programmer may code a complementary function for each smart contract method. Each complementary function may be configured to receive at least the state of the smart contract and/or the identity of the user as inputs. Each complementary function may further specify one or more condition(s) that are required for the user interface element to be provided to the user smart contract method to be called (e.g., the contract must be in a signed state and the identity of the user must either be a service provider and/or a consumer). For example, cancell_sign() function 308 as depicted in FIG. 3 is one such complementary function. Briefly, cancell_sign() function 308 determines whether signAmount() is zero (i.e., the required amount(s) have not yet been specified) and whether the participant has already signed the contract. If signAmount() returns zero, or the participant has already signed, then cancell_sign() returns FALSE, and a user interface element that enables the user to sign the contract will thereby not be enabled. The selective enablement of user interface elements are described in further detail herein below. Note also, and as mentioned above, such functions may also be used by UX generator 802A and/or UX generator 802B to query the smart contract for data used to populate one or more text fields of the rendered GUI, and need not be restricted to smart contract actions or GUI rendering related thereto.

[0077] Upon receiving event notification 826 from smart contract 804 during the execution thereof, UX generator 802A and 802B may query server 834 for information

stored via compilation artifacts file 618 to obtain a description of each of the smart contract method calls, the complementary smart contract method calls and/or the UX annotations associated with smart contract 804. For instance, UX generator 802A may provide a query 830A to server 834 for information stored via compilation artifacts file 618, and UX generator 802B may provide a query 830B to server 834 for information stored via compilation artifacts file 618. In response, server 834 may provide a listing of the smart contract methods, complementary smart contract methods, and/or UX annotations included in smart contract 804 to UX generator 802A and/or 802B.

[0078] Using the listing, UX generator 802A and 802B may query each complementary function via one or more queries 816A and 816B, respectively, on behalf each user associated with smart contract 804 to determine whether a corresponding UI element should be provided to the user. For instance, when querying smart contract 804 on behalf of a first user, query 816A may include an identifier of the first user. When querying smart contract 804 on behalf of a second user, query 816B may include an identifier of the second user. If the condition(s) of a particular complementary function are met, the complementary function returns a positive response 820A and/or 820B to UX generator 802A and/or 802B, respectively, that indicates that a corresponding user interface element with respect to the associated smart contract method should be provided to the user. In response, UX generator 802A provides a representation of the user interface element to GUI 116A via browser 136A and/or UX generator 802B provides a representation of the user interface element to GUI 116B via browser 136B for displaying thereby. If the condition(s) are not met, the complementary function returns a negative response 822A and/or 822B to UX generator 802A and/or UX generator 802A, respectively, that indicates that no user interface element should be provided to the user with respect to the associated smart contract method.

[0079] The user interface element may be a user-interactive element, such as a button, that is displayed via GUI 116A and/or contact generator GUI 116B. The user interface element may also be a text string that, for example, provides a status update with reference to an action taken with respect to smart contract 804. The text string may be based on an annotation included in the code of smart contract 804. Such annotations may be referred to as user experience (UX) annotations, as they affect the user interface (e.g., GUI 116A and/or GUI 116B) utilized by the user. The annotation may be included in the code by a programmer. The annotations may appear as comments to the compiler that compiles the source code of smart contract 804 (e.g., compiler 604, as shown in FIG. 6). UX annotations may comprise a textual string. UX annotations may also comprise one or more variables

utilized by smart contract 804 and that are populated during execution of smart contract 804. The variables may be representative of user identities, fund amounts, etc. UX generator 802A and/or 802B may be configured to utilize such UX annotations to generate user interface elements (e.g., text strings). The UX annotations may be associated with the complementary functions described above. For instance, such UX annotations may be included within such complementary functions or included proximate to such complementary functions (e.g., located before after such complementary functions). Compilation artifacts file 618 may indicate which UX annotations are associated with a particular complementary function. If a complementary function provides a positive response, UX generator 802A and/or 802B may utilize the information received for compilation artifacts file 618 to determine which UX annotations are associated with the complementary functions that provided positive response 820A and/or 820B. UX generator 802A and/or 802B may display the associated UX annotations as a text string. Any variables included in such UX annotations are replaced by the values for such variables, as determined by smart contract 804.

[0080] Certain UX annotations may be provided in the code of smart contract 804 that are independent of a complementary function. For instance, certain UX annotations may provide the current state of smart contract 804. Such UX annotations may also be specified via compilation artifacts file 618. Each time smart contract 804 provides an event notification (e.g., event notification 826A and/or 826B), UX generator 802A and/or 802B may display such UX annotations as a text string utilizing the information received for compilation artifacts file 618. Any variables included in such UX annotations are replaced by the values for such variables, as determined by smart contract 804.

[0081] Because certain user interface elements may be conditioned based on the identity of the user, the user interface generated may differ from user to user for any given smart contract. Moreover, because certain user interface elements may be conditioned on the state of the smart contract, the user interface generated may dynamically change when the state of smart contract changes. For example, suppose a service provider named in smart contract 804 utilizes computing device 102A to view and/or interact with a representation of smart contract 804 via GUI 116A, and a consumer named in smart contract 804 utilizes computing device 102B to view and/or interact with a representation of smart contract via GUI 116B. After smart contract 804 is deployed, UX generator 802A and/or 802B may query each complementary function via one or more queries 816A and/or 816B, respectively, and/or query server 834 (via queries 830A and/or 830B, respectively) to determine whether a user

interface element associated with each complementary function is to be rendered via GUI 116A and/or GUI 116A and/or to determine whether any UX annotations are to be displayed via a text string to be rendered via GUI 116A and/or GUI 116A.

[0082] At this stage, smart contract 804 is in the pending state, as neither content provider
5 nor consumer have signed the contract. Accordingly, the complementary functions associated with smart contract methods for executing (i.e., signing) the contract may return a positive result and cause a user interface element to be rendered on both of GUI 116A and GUI 116A. For example, FIG. 9A is an example GUI screen 900A in which user interface elements are displayed to a service provider based on querying smart contract 804 on behalf
10 of the service provider in accordance with an example embodiment, and FIG. 9B is an example GUI screen 900B in which user interface elements are displayed to a consumer based on querying smart contract 804 on behalf of the consumer in accordance with an example embodiment. As shown in FIGS. 9A and 9B, text 902 representing an immutable version of a smart contract 804 is provided via a GUI (e.g., GUI 116A and GUI 116B).

[0083] In FIG. 9A, below text 902 are various interface elements 904, 906, 908, 910, and 912 that are displayed based on querying smart contract 804 on behalf of the service provider. For instance, after deploying smart contract 804 to blockchain network 112, smart contract 804 may provide event notification 826 to UX generator 802A that indicates that smart contract 804 has changed state (i.e., smart contract 804 is in a pending state). In
15 response, UX generator 802A may query each of the complementary functions of smart contract 804 on behalf of the service provider of smart contract 804. In the example shown in FIG. 9A, a first complementary function associated with a smart contract method for signing the smart contract and a second complementary function associated with a smart contract method associated with depositing funds may all provide a positive result.
20 Accordingly, user interface elements 904, 906, and 912 are displayed. For example, the conditions for displaying user interface elements 904 and 912 may be that smart contract 804 is in a pending state and the identity of the user (as indicated in the query to smart contract 804) is a service provider associated with smart contract 804. The conditions for displaying user interface elements 906 and 908 may be that smart contract 804 is in a
25 pending state, the identity of the user is a service provider or a consumer, and that the service provider and the consumer have not deposited funds.
30

[0084] User interface elements 904, 906, and 908 are text strings corresponding to UX annotations included in the code of smart contract 804 and/or compilation artifacts file 618, and user interface element 912 is a button. The UX annotation represented by user interface

element 904 may include a variable representative of the amount that the service provider is to deposit with respect to smart contract 804. Variable is populated by a value specified by the parties during smart contract negotiation. In the example shown in FIG. 9, because smart contract 804 is queried on behalf of the service provider, the value “20” populates the variable, as it represents the amount that the service provider is to deposit at the time of signing. User interface element 910 may be a UX annotation that is independent of a complementary function. user interface element 910 may be displayed as a result of UX generator 802A and/or 802B analyzing the information received for compilation artifacts file 618. For instance, the information may specify that the UX annotation is to be responsive to receiving displayed regardless of any complementary function returning a positive response.

[0085] In FIG. 9B, below text 902 are interface elements 904', 906, 908, 910, and 912 that are displayed based on querying smart contract 804 on behalf of the consumer. For instance, after deploying smart contract 804 to blockchain network 112, smart contract 804 may provide event notification 826 to UX generator 802B that indicates that smart contract 804 has changed state (i.e., smart contract 804 is in a pending state). In response, UX generator 802B may query each of the complementary functions of smart contract 804 on behalf of the consumer of smart contract 804. In the example shown in FIG. 9B, the same complementary functions described above with reference to FIG. 9A provide a positive result. Accordingly, user interface elements 906 and 912 are displayed. User interface element 910 is displayed in a similar manner as described above with reference to FIG. 9A. However, with reference to user element interface 904,' because the first complementary function is queried on behalf of the consumer, the variable included in the UX annotation is populated by the agreed-upon amount that the consumer is to deposit upon signing the contract (i.e., “110 Finney”).

[0086] Referring again to FIG. 8, now suppose that the service provider signs smart contract 804 by activating user interface element 912 of FIG. 9A (e.g., by pressing user interface element 912 via an input device, such as a mouse). Upon activating user interface element 912, GUI 116A may send a command to UX generator 802A via browser 136A that causes UX generator 802A to call the smart contract method for signing smart contract 804. After smart contract 804 is successfully signed by the service provider and/or the required funds are deposited, smart contract 804 may provide event notification 826 to UX generator 802A that indicates smart contract 804 has been signed by a party. In response, UX generator 802A queries each of the complementary functions of smart contract 804 via one

or more queries 816 (as specified in the information associated with compilation artifacts file 618) to determine which actions that the service provider and the consumer are allowed to perform and the user interface elements corresponding to such actions that are to be provided to the users. the GUI provided to service provider will differ from the GUI
5 provided to consumer. In particular, user interface elements 904 and 912 will no longer be provided to the service provider (via GUI 116A), as he has already signed the contract. However, user interface elements 904 and 912 will still be provided to the consumer's GUI (e.g., GUI 116B) because he has not yet signed the contract. Examples of such GUI screens are provided below with reference to FIGS. 10 and 11.

10 **[0087]** For example, FIG. 10 is an example GUI screen 1000 in which user interface elements are displayed based on querying smart contract 804 on behalf of the service provider after the service provider has signed smart contract 804 in accordance with an example embodiment. FIG. 11 is an example GUI screen 1100 in which user interface
15 elements are displayed based on querying smart contract 804 on behalf of the consumer after the service provider has signed smart contract 804 in accordance with an example embodiment. As shown in FIG. 10, when smart contract 804 is queried on behalf of the service provider, user interfaced elements 904 and 912 are no longer displayed as the conditions for displaying are no longer met (i.e., service provider has signed smart contract 804). Moreover, user interface element 908 has been updated to reflect the amount of funds
20 deposited by the service provider (shown as user interface element 908'). Finally, new user interface element 1002 and 1004 are displayed. User interface 1002 may be a button that enables the service provider to revert the signing and receive his funds back. User interface element 1004 provides a text description of the functionality provided by user interface element 1002. User interface elements 1002 and 1004 may be displayed upon a
25 corresponding complementary function associated with a smart contract method for reverting the signing returning a positive result. User interface element 1004 may be displayed based on a UX annotation included in the code of smart contract 804 and/or compilation artifacts file 618 that is associated with a smart contract method for reverting the signing of smart contract 804, thereby returning the contract to the previous state where
30 neither party has yet signed smart contract 804.

[0088] As shown in FIG. 11, user interface elements 904' and 912 are still displayed for the consumer, as he has yet to sign the contract (i.e., the condition required for not displaying user interface element 912 has not been met). Moreover, user interface element 908 has been updated (shown as user interface element 908') to reflect the amount of funds deposited by

the service provider.

[0089] The user interface elements presented to the service provider and consumer (and/or other parties to the smart contract, such as a mediator) continue to change through all the different phases of the smart contract depending on the actions taken by the parties and/or the state in which smart contract 804 is in. For example, FIG. 12 is an example GUI screen 1200 in which user interface elements are displayed based on querying smart contract 804 on behalf of the consumer after the service provider has signed smart contract 804, and after the consumer has signed smart contract 804, in accordance with an example embodiment. As shown in FIG. 12, user interface element 908' is unchanged since the deposit made by the provider is unaffected by the consumer signing smart contract 804. However, GUI screen 1200 now includes user interface element 906' which reflects the total deposit made by the consumer when they signed smart contract 804. Likewise, GUI screen 1200 also includes user interface element 910' which reflects the fact that the contract is now changed to the 'sealed' state (for the purposes of GUI screen 1200, please assume that mediator David Doe has also signed the contract).

[0090] GUI screen 1200 also includes new user interface elements 1202 and 1204. User interface element 1202 and 1204 are each, like user interface element 912, buttons that trigger actions. Whereas pressing user interface element 912 will trigger a user's signing smart contract 804, pressing user interface element 1202 shall signal the consumer wishes to complete the contract. Assuming that the service provider has likewise clicked on an analogous 'Complete Contract' user interface element in their GUI, the contract will complete and funds will be released to the service provider per the terms of the contract.

[0091] Upon completion of the contract, whether by action of user interface element 1202 as described herein just above or by action of a mediator as described herein below, smart contract 804 may be configured to delete itself from the blockchain. For example, embodiments implemented using the Ethereum blockchain may elect to execute a `selfdestruct()` at the conclusion of smart contract 804. Such a `selfdestruct()` call may be configured to send any remaining fund balance to a specified address which has the benefit of costing less than execution of an `address.send()` invocation. Moreover, such an operation frees up space on the blockchain resulting in lowering of overall fees due at the end of the contract. It should be noted, however, that deletion of the smart contract from the blockchain will thereafter prevent rendering a GUI as described herein since the underlying smart contract state no longer exists. Since the smart contract was terminated, however, the inability to render the state of the contract is generally acceptable.

[0092] User interface element 1204, on the other hand, may be used by the consumer to initiate a dispute. Of course, it should be understood that an analogous user interface element is made available to the provider in their GUI (not shown here), and which thereby permits the provider to initiate a dispute. When user interface element 1204 is pressed, the contract will enter the disputed state, and a GUI available to the mediator will be enabled to allow the mediator to take appropriate action.

[0093] For example, FIG. 13 is an example GUI screen 1300 in which user interface elements are displayed based on querying smart contract 804 on behalf of the mediator after either the service provider or consumer has initiated a dispute, in accordance with an example embodiment. The user interface elements of GUI screen 1300 reflect the fact that smart contract 804 is now being disputed by one or both parties. User interface elements 908' and 906' reflect no change from the previous state inasmuch as such funds continue to be on deposit (i.e. escrowed in the smart contract). User interface element 910'', on the other hand reflects the fact that the contract has entered the 'disputed' state. Once smart contract 804 enters the disputed state, only the mediator may perform smart contract actions pursuant to the terms of the contract. In practice, the mediator will get in contact with the parties to gather facts and discuss the nature of the dispute. The mediator may thereafter perform whichever of one or more actions that may be enabled in GUI screen 1300 consistent with the terms of the contract.

[0094] Thus, in an embodiment, GUI screen 1300 is enabled to provide user interface elements 1302 and 1304.. In an embodiment, user interface elements 1302 permits the mediator to select a percentage of the escrowed reward amount (after the mediation fee is deducted) to release to the provider, with the remainder to be returned to the consumer. In alternate embodiments, the mediator may instead be able to select an amount instead of a percentage. Upon satisfactory apportionment of the funds per user interface element 1302, the mediator may click user element 1304 to resolve the dispute and release the funds as specified. Upon invocation of the 'resolve dispute' method associated with user interface element 1304, the smart contract will disburse the specified percentage of the funds to the specified party, and the balance of the funds (if any) to the other party. In embodiments, the disbursement of funds will ordinarily end smart contract 804, and the mediator's fee will likewise be disbursed to him/her. In the running example of smart contract 804 described herein, the mediator will be rewarded 5% of the reward amount, or 5 finney.

[0095] Note, the actions GUI screen 1300 enables the mediator to undertake are merely exemplary. In other embodiments, example GUI screen 1300 may enable other actions such

as rescinding the contract, modifying the contract, or such other actions as may be agreed upon by the parties during negotiation.

5 **[0096]** UX generator 802A and/or 802B may be configured to provide query(ies) 816A and/or 816B and/or receive event notification 826, positive response 820A and/or 820B and/or negative response 822A and/or 822B via an API that interfaces blockchain network 112 and UX generator 802A and/or 802B. In an embodiment in which blockchain network 112 is an Ethereum-based, the API may be a Metamask API. UX generator 802A and/or 802B may be configured to provide query(ies) 830A and/or 830B to server 834 via a REST API.

10 **[0097]** The above described example smart contract 804, and associated GUI screens 9A-9B and 10-14, is but one relatively simple smart contract example. In principle, smart contracts (and associated GUI screens) may be arbitrarily complex and reflect any scenario capable of being coded on the underlying smart contract platform. For example, FIG. 14 depicts an example GUI screen 1400 in which user interface elements are displayed to a consumer based upon querying an alternative smart contract, in an embodiment. GUI screen 1400 includes smart contract text 1402 that represents an immutable version of an alternative version of smart contract 804. GUI screen 1400 also includes various interface elements 1404, 1406, 1408, 1410, 1412, 1414 and 1416 that are displayed based on querying smart contract 804 on behalf of the consumer, Charlie Clark, in the manner described above.

15 **[0098]** Smart contract text 1402 reflects a more complex smart contract 804 that enables the parties to negotiate and agree to alternative payment terms after the contract has been sealed, and without requiring the involvement of, or payment to, a mediator. The smart contract embodied by smart contract text 1402 reduces the chance that a mediator will be required since it offers the parties the chance to resolve a conflict amongst themselves. Such a smart contract may be beneficial since on average, mediation costs are lowered. Example GUI 1400 may best be understood in terms of an example scenario.

20 **[0099]** Suppose that the consumer, Charlie Clark, is not pleased with the work provided by Bob Brown and refuses to pay full and previously agreed upon sum of 100 finney. GUI screen 1400 depicts the smart contract 804 as seen by Charlie Clark. GUI screen 1400 includes user interface elements 1404, 1406 and 1408 which are analogous to those described above in conjunction with FIGS. 9A-9B and 10-14. In particular, user interface elements 1404 and 1406 indicate the amount of funds deposited by each party. User interface element 1408 indicates that all parties have signed the contract since the contract is in the 'sealed' state.

[00100] Thus, GUI screen 1400 of FIG. 12 is analogous that GUI screen 1200 of FIG. 12, except that the underlying smart contract associated with each is different. Recall that GUI screen 1200 exposed only two smart contract actions to the consumer after the contract was sealed: complete the contract via user interface element 1202, or initiate a dispute via user interface element 1204. GUI screen 1400 of FIG. 14 enables the consumer to perform these same actions via user interface elements 1412 and 1414, respectively. GUI screen 1400 differs, however, in that the consumer is enabled to offer a lesser amount to the consumer in order to complete the contract. In particular GUI screen 1400 includes user interface elements 1410 and 1416. User interface element 1410 comprises a text entry box that permits the consumer to enter the proposed settlement amount to thereafter be sent to the provider by pressing the user interface element 1416.

[00101] In the previous example described above, the consumer could only dispute the entire contract, thereby committing to paying a mediator 5% of the entire amount. In this example, however, instead of initiating the dispute as previously described, this contract allows parties to revise their agreement after the work done. Suppose, for example, that consumer Charlie believes that the work provided by Bob Brown was incomplete or low quality. In that instance, Charlie may propose, for example, to pay Bob 30 finney by entering such in user interface element 1410 and clicking on user interface element 1416.

[00102] Once Charlie sends his settlement offer, a notification may be generated and sent to Bob as described above, and wherein Bob's GUI screen (not shown) may be updated to reflect the new state of the contract and enable Bob additional actions to be performed consistent with the above described embodiments. For example, an analogous GUI screen rendered for Bob may enable him to accept the offered settlement, to enter and send a counteroffer, or to initiate a dispute of the full contract amount. Continuing with our example, suppose that although amenable to compromise, Bob feels that 30 finney is too low a settlement amount, and instead proposes that the appropriate amount is 70 finney. Bob may enter that amount in his GUI (not shown) and transmit the counteroffer to Charlie for consideration.

[00103] In the event that Charlie agrees with the counteroffer (or Bob agrees with Charlie's initial offer), the parties may end the contract on the revised terms without involving a mediator or paying a fee. However, if a disagreement still exists, the mediator may be called upon to resolve the remaining dispute. In this example, Charlie has offered 30 finney and Bob is asking for 70 finney, thus bringing the amount in dispute to the difference, 40 finney. Here, David Doe will be enabled via his GUI (e.g., a GUI analogous to that shown in FIG.

13 but also not shown herein) to determine an appropriate amount based on the facts and his judgment, and resolve the dispute accordingly. This type of smart contract structure may be beneficial since even where there is a dispute, the mediator is paid his percentage on a lower disputed amount (in this example, 5% of the 40 finney dispute amount, or 2 finney) instead
5 of the value of the entire reward amount (i.e., where David Doe would instead be owed 5 finney).

[00104] It is noted that while the embodiments described above describe that users may negotiate and generate a smart contract via manual selection of an appropriate smart contract template, the embodiments described herein are not so limited. For instance, in an
10 embodiment, the language and/or terms of the smart contract may be generated based on information obtained from a communication session between a plurality of different users. For instance, users may engage in a chat session via a website (e.g., a social media-based website, such as, but not limited to, Facebook, LinkedIn, etc.). UI generator 810 (as described above with reference to FIG. 8) may be configured to analyze the communication
15 session between the users (e.g., using machine learning techniques) and generate the language and/or terms based on the context of the messages communicated between the users. Such analysis of the communication session may be performed with various types of natural language processing.

[00105] For instance, FIG. 15 is an example GUI screen 1500 that depicts a conversation
20 1502 between two users in accordance with an example embodiment. As shown in FIG. 15, conversation 1502 is with regards to establishing a work agreement between John and Sara; in particular, a work agreement for translating a document into Spanish. UI generator 810 may be configured to analyze the messages and determine that John and Sara are negotiating a contract and determine the terms of the contract. The terms determined in this example
25 would be the service provider (“Sara”), the consumer (“John”), the amount to be paid for the work by the consumer (“\$150”), and a description of the work (e.g., document translation to Spanish).

[00106] After determining the terms of the contract, UI generator 810 may cause user interface elements to be rendered in the chat window utilized by each of John and Sara that
30 may be interacted with to generate the smart contract or to reject the agreement therebetween. For instance, as shown in FIG. 15, UI generator 810 causes user interface elements 1504 and 1506 to be rendered. User interface element 1504, when activated by the user, causes the smart contract to be generated. User interface element 1506, when activated by the user, enables the user to reject the agreement, and therefore, not causing the smart

contract to be generated.

[00107] Upon each of John and Sara activating their respective user interface element 1504 indicating their willingness to have a smart contract generated, UI generator 810 may generate a smart contract based on the above-determined information, or alternatively, determine a smart contract template from which the smart contract will be based. In the latter situation, the above-determined information may be used to populate variables of the smart contract template during compilation of the smart contract in a manner as described above with reference to FIGS. 5 and 6.

[00108] After generating and compiling the smart contract, the smart contract may be deployed to blockchain network 112 as described above with reference to FIGS. 5 and 6. Thereafter, UI representation determination engine 802 may query the smart contract (e.g., smart contract 804) to determine which actions each of John and Sara are allowed to perform with respect to smart contract 804 in a similar manner as described above with reference to FIGS. 8-14. For example, with reference to FIG. 15, UI representation determination engine 802 outputs a text string 1508 (which e.g., may be determined via UX annotations included in smart contract 804) and user interface elements 1510 and 1512 (which e.g., may be determined based on particular complementary functions returning positive indications 822). Text string 1508 indicates the state of smart contract 804, user interface element 1510 enables the user (i.e., John or Sara) to sign smart contract 804, and user interface element 1512 enables the user to reject smart contract 804. The user interface elements displayed to a particular user may differ based on the identity of the user and/or the state in which smart contract 804 is in, as described above with reference to FIGS. 8-14.

[00109] FIG. 16 shows a flowchart 1600 of a method for a method for generating a smart contract from a smart contract template, according to an embodiment. In an embodiment, flowchart 1600 may be implemented by systems 100, 600, and/or 800 as shown in FIGS. 1, 6, and 8, respectively, although the method is not limited to those implementations. Accordingly, flowchart 1600 will be described with continued reference to FIGS. 1, 6, and 8. Other structural and operational embodiments will be apparent to persons skilled in the relevant art(s) based on the discussion regarding flowchart 1600 and systems 100, 600 and 800 of FIGS. 1, 6 and 8.

[00110] Flowchart 1600 begins with step 1602. In step 1602, a selection of a smart contract template is accepted from one or more parties of a plurality of parties, the plurality of parties engaging in a transaction and choosing the selected smart contract template based at least in part on transaction terms negotiated between the plurality of parties. For example, with

reference to FIG. 1, and as described above, UI generator 110 may be configured to transmit GUI information 140A and/or 140B to each of browser 136A and browser 136B, respectively, where such GUI information may enable a user to select a smart contract template from among those stored in smart contract template library 118. In an embodiment, GUI 116A and/or GUI 116B may enable users to browse the smart contract template library 118 for a suitable smart contract template that most closely matches the negotiated transaction. UI generator 110 may thereafter accept selection from one or both users of one of the smart contract templates from the smart contract template library 118. Flowchart 1600 continues at step 1604.

10 **[00111]** At step 1604, transaction terms corresponding to fillable elements of the smart contract template are accepted from the plurality of parties. For example, with reference to FIGS. 1, 6 and 8, the selected contract template may be rendered via GUI 116A and/or GUI 116B by UI generator 110 to provide information 140A and 140B to GUI 116A and GUI 116B, respectively. The rendered UI may be displayed via GUI 116A and/or GUI 116B as human-readable text, and the variables may be displayed as user-interactive elements (e.g., text boxes, pull-down menus, etc.) that enable the users to specify user-specifiable terms of the smart contract. For example, user interface element 418 of FIG. 4 enables a user to specify a mediator in a number of ways as described above, which is thereafter accepted by, for example, system 100, 600 or 800 of FIGS. 1, 6 or 8, respectively.

20 **[00112]** At step 1606, the smart contract template is augmented to include the accepted transaction terms to provide an augmented smart contract template. For example, and with continued reference to systems 100, 600 and 800 of FIGS. 1, 6 and 8, respectively, variable replacer 602 of smart contract generator 606 may be configured to replace the variables in the smart contract source code file (such as source file 612) associated with the smart contract template with user-specified parameters 616.

25 **[00113]** At step 1608, a smart contract corresponding to the augmented smart contract template is generated. For example, and with continued reference to systems 100, 600 and 800 of FIGS. 1, 6 and 8, respectively, compiler 604 may be configured to compile source code file 612 with the user-specified parameters 616 into a format suitable for the blockchain maintained by blockchain network 112. For instance, compiler 604 may compile source code file 612 into byte code 614. In an embodiment in which source code file 612 is written in the Solidity programming language, compiler 604 may be a Solidity-based compiler. In an embodiment in which blockchain network 112 is an Ethereum blockchain network, byte code 614 may be Ethereum Virtual Machine (EVM) byte code.

[00114] Flowchart 1600 concludes at step 1610. At step 1610, the smart contract is deployed to a blockchain. With continued reference to systems 100, 600 and 800 of FIGS. 1, 6 and 8, respectively, smart contract generator 606 may provide byte code 614 to browser 136A. Deployer 608 executing in browser 136A may be configured to deploy the compiled smart contract (e.g., byte code 614) to the blockchain maintained by blockchain network 112 for execution thereby. For instance, a virtual machine (e.g., an Ethereum virtual machine (EVM) of blockchain network 112) may execute byte code 614.

[00115] FIG. 17 shows a flowchart 1700 of a method for selecting a smart contract template for generating a smart contract, in accordance with an example embodiment. In an embodiment, flowchart 1700 may be implemented by systems 100 and/or 800 as shown in FIGS. 1 and 8, although the method is not limited to those implementations. Accordingly, flowchart 1700 will be described with continued reference to FIGS. 1 and 8. Other structural and operational embodiments will be apparent to persons skilled in the relevant art(s) based on the discussion regarding flowchart 1700 and systems 100 and 800 of FIGS. 1 and 8, respectively.

[00116] Flowchart 1700 begins with step 1702. In step 1702, at least one of a plurality of functions associated with the smart contract is called to determine whether at least one party of the plurality of parties should be enabled to perform a smart contract action. For example, and as described above in conjunction with code fragment 300 of FIG. 3 and embodiments of system 800, embodiments may be enabled to call the complementary `cancall()` function associated with a smart contract action to determine whether the user is enabled to perform such an action, and therefore, render user interface elements that enable such actions to be performed.

[00117] Flowchart 1700 concludes at step 1704. At step 1704, a user interface representation is presented to the at least one party of the plurality of parties, the user interface representation enabling the at least one party of the plurality of parties to perform the smart contract action. For example, and as described above with respect to code fragment 300 of FIG. 3 and system 800 of FIG. 8, where the complementary `cancall()` function associated with a smart contract action returns `TRUE`, embodiments are enabled to render user interface representations that enable the user to perform such actions.

III. Example Mobile Device Implementation

[00118] FIG. 18 is a block diagram of an exemplary mobile device 1802 that may implement embodiments described herein. For example, mobile device 1802 may be used to implement server 134, computing device 102A, computing device 102B, smart contract

generator 106, UI generator 110, smart contract template library 118, GUI 116A, browser 136A, browser window 114A, display screen 108A, GUI 116B, browser 136B, browser window 114B, and display screen 108B of FIG. 1, server 634, smart contract generator 606, UI generator 610, variable replacer 602, compiler 604, and deployer 608 of FIG. 6, server 5 834, smart contract generator 806, UX generator 802, and user profiles 808 of FIG. 8, and/or any of the components respectively described therein and/or any of the steps of any of the flowcharts of FIGS. 5, 16, and 17. As shown in FIG. 18, mobile device 1802 includes a variety of optional hardware and software components. Any component in mobile device 1802 can communicate with any other component, although not all connections are shown 10 for ease of illustration. Mobile device 1802 can be any of a variety of computing devices (e.g., cell phone, smart phone, handheld computer, Personal Digital Assistant (PDA), etc.) and can allow wireless two-way communications with one or more mobile communications networks 1804, such as a cellular or satellite network, or with a local area or wide area network. Mobile device 1802 can also be any of a variety of wearable computing device 15 (e.g., a smart watch, an augmented reality headset, etc.).

[00119] Mobile device 1802 can include a controller or processor 1810 (e.g., signal processor, microprocessor, ASIC, or other control and processing logic circuitry) for performing such tasks as signal coding, data processing, input/output processing, power control, and/or other functions. An operating system 1812 can control the allocation and 20 usage of the components of mobile device 1802 and provide support for one or more application programs 1814 (also referred to as “applications” or “apps”). Application programs 1814 may include common mobile computing applications (e.g., e-mail applications, calendars, contact managers, web browsers, messaging applications) and any other computing applications (e.g., word processing applications, mapping applications, 25 media player applications).

[00120] Mobile device 1802 can include memory 1820. Memory 1820 can include non-removable memory 1822 and/or removable memory 1824. Non-removable memory 1822 can include RAM, ROM, flash memory, a hard disk, or other well-known memory devices or technologies. Removable memory 1824 can include flash memory or a Subscriber 30 Identity Module (SIM) card, which is well known in GSM communication systems, or other well-known memory devices or technologies, such as “smart cards.” Memory 1820 can be used for storing data and/or code for running operating system 1812 and application programs 1814. Example data can include web pages, text, images, sound files, video data, or other data to be sent to and/or received from one or more network servers or other devices

via one or more wired or wireless networks. Memory 1820 can be used to store a subscriber identifier, such as an International Mobile Subscriber Identity (IMSI), and an equipment identifier, such as an International Mobile Equipment Identifier (IMEI). Such identifiers can be transmitted to a network server to identify users and equipment.

5 **[00121]** Mobile device 1802 can support one or more input devices 1830, such as a touch screen 1832, a microphone 1834, a camera 1836, a physical keyboard 1838 and/or a trackball 1840 and one or more output devices 1850, such as a speaker 1852 and a display 1854. Other possible output devices (not shown) can include piezoelectric or other haptic output devices. Some devices can serve more than one input/output function. For example,
10 touch screen 1832 and display 1854 can be combined in a single input/output device. Input devices 1830 can include a Natural User Interface (NUI).

[00122] Wireless modem(s) 1860 can be coupled to antenna(s) (not shown) and can support two-way communications between processor 1810 and external devices, as is well understood in the art. Modem(s) 1860 are shown generically and can include a cellular
15 modem 1866 for communicating with the mobile communication network 1804 and/or other radio-based modems (e.g., Bluetooth 1864 and/or Wi-Fi 1862). At least one of wireless modem(s) 1860 is typically configured for communication with one or more cellular networks, such as a GSM network for data and voice communications within a single cellular network, between cellular networks, or between the mobile device and a public
20 switched telephone network (PSTN).

[00123] Mobile device 1802 can further include at least one input/output port 1880, a power supply 1882, a satellite navigation system receiver 1884, such as a Global Positioning System (GPS) receiver, an accelerometer 1886, and/or a physical connector 1890, which can be a USB port, IEEE 1594 (FireWire) port, and/or RS-232 port. The illustrated
25 components of mobile device 1802 are not required or all-inclusive, as any components can be deleted and other components can be added as would be recognized by one skilled in the art.

[00124] In an embodiment, mobile device 1802 is configured to implement any of the above-described features of server 134, computing device 102A, computing device 102B, smart contract generator 106, UI generator 110, smart contract template library 118, GUI
30 116A, browser 136A, browser window 114A, display screen 108A, GUI 116B, browser 136B, browser window 114B, and display screen 108B of FIG. 1, server 634, smart contract generator 606, UI generator 610, variable replacer 602, compiler 604, and deployer 608 of FIG. 6, server 834, smart contract generator 806, UX generator 802, and user profiles 808

of FIG. 8, and/or any of the components respectively described therein and/or any of the steps of any of the flowcharts of FIGS. 5, 16, and 17. Computer program logic for performing the functions of these devices may be stored in memory 1820 and executed by processor 1810.

5 IV. Example Computer System Implementation

[00125] FIG. 19 depicts an example processor-based computer system 1900 that may be used to implement various embodiments described herein. For example, system 1900 may be used to implement server 134, computing device 102A, computing device 102B, blockchain network 112, smart contract generator 106, UI generator 110, smart contract
10 template library 118, GUI 116A, browser 136A, browser window 114A, display screen 108A, GUI 116B, browser 136B, browser window 114B, and display screen 108B of FIG. 1, server 634, smart contract generator 606, UI generator 610, variable replacer 602, compiler 604, and deployer 608 of FIG. 6, server 834, smart contract generator 806, UX generator 802, and user profiles 808 of FIG. 8, and/or any of the components respectively
15 described therein. System 1900 may also be used to implement any of the steps of any of the flowcharts of FIGS. 5, 16, and 17, as described above. The description of system 1900 provided herein is provided for purposes of illustration, and is not intended to be limiting. Embodiments may be implemented in further types of computer systems, as would be known to persons skilled in the relevant art(s).

20 [00126] As shown in FIG. 19, system 1900 includes a processing unit 1902, a system memory 1904, and a bus 1906 that couples various system components including system memory 1904 to processing unit 1902. Processing unit 1902 may comprise one or more circuits, microprocessors or microprocessor cores. Bus 1906 represents one or more of any of several types of bus structures, including a memory bus or memory controller, a
25 peripheral bus, an accelerated graphics port, and a processor or local bus using any of a variety of bus architectures. System memory 1904 includes read only memory (ROM) 1908 and random access memory (RAM) 1910. A basic input/output system 1912 (BIOS) is stored in ROM 1908.

[00127] System 1900 also has one or more of the following drives: a hard disk drive 1914
30 for reading from and writing to a hard disk, a magnetic disk drive 1916 for reading from or writing to a removable magnetic disk 1918, and an optical disk drive 1920 for reading from or writing to a removable optical disk 1922 such as a CD ROM, DVD ROM, BLU-RAY™ disk or other optical media. Hard disk drive 1914, magnetic disk drive 1916, and optical disk drive 1920 are connected to bus 1906 by a hard disk drive interface 1924, a magnetic

disk drive interface 1926, and an optical drive interface 1928, respectively. The drives and their associated computer-readable media provide nonvolatile storage of computer-readable instructions, data structures, program modules and other data for the computer. Although a hard disk, a removable magnetic disk and a removable optical disk are described, other types of computer-readable memory devices and storage structures can be used to store data, such as flash memory cards, digital video disks, random access memories (RAMs), read only memories (ROM), and the like.

[00128] A number of program modules may be stored on the hard disk, magnetic disk, optical disk, ROM, or RAM. These program modules include an operating system 1930, one or more application programs 1932, other program modules 1934, and program data 1936. In accordance with various embodiments, the program modules may include computer program logic that is executable by processing unit 1902 to perform any or all of the functions and features of server 134, computing device 102A, computing device 102B, blockchain network 112, smart contract generator 106, UI generator 110, smart contract template library 118, GUI 116A, browser 136A, browser window 114A, display screen 108A, GUI 116B, browser 136B, browser window 114B, and display screen 108B of FIG. 1, server 634, smart contract generator 606, UI generator 610, variable replacer 602, compiler 604, and deployer 608 of FIG. 6, server 834, smart contract generator 806, UX generator 802, and user profiles 808 of FIG. 8, and/or any of the components respectively described therein, and/or any of the steps of any of the flowcharts of FIGS. 5, 16, and 17, as described above. The program modules may also include computer program logic that, when executed by processing unit 1902, causes processing unit 1902 to perform any of the steps of any of the flowcharts of FIGS. 5, 16, and 17 as described above.

[00129] A user may enter commands and information into system 1900 through input devices such as a keyboard 1938 and a pointing device 1940 (e.g., a mouse). Other input devices (not shown) may include a microphone, joystick, game controller, scanner, or the like. In one embodiment, a touch screen is provided in conjunction with a display 1944 to allow a user to provide user input via the application of a touch (as by a finger or stylus for example) to one or more points on the touch screen. These and other input devices are often connected to processing unit 1902 through a serial port interface 1942 that is coupled to bus 1906, but may be connected by other interfaces, such as a parallel port, game port, or a universal serial bus (USB). Such interfaces may be wired or wireless interfaces.

[00130] Display 1944 is connected to bus 1906 via an interface, such as a video adapter 1946. In addition to display 1944, system 1900 may include other peripheral output devices

(not shown) such as speakers and printers.

[00131] System 1900 is connected to a network 1948 (e.g., a local area network or wide area network such as the Internet) through a network interface 1950, a modem 1952, or other suitable means for establishing communications over the network. Modem 1952, which may
5 be internal or external, is connected to bus 1906 via serial port interface 1942.

[00132] As used herein, the terms "computer program medium," "computer-readable medium," and "computer-readable storage medium" are used to generally refer to memory devices or storage structures such as the hard disk associated with hard disk drive 1914, removable magnetic disk 1918, removable optical disk 1922, as well as other memory
10 devices or storage structures such as flash memory cards, digital video disks, random access memories (RAMs), read only memories (ROM), and the like. Such computer-readable storage media are distinguished from and non-overlapping with communication media (do not include communication media or modulated data signals). Communication media typically embodies computer-readable instructions, data structures, program modules or
15 other data in a modulated data signal such as a carrier wave. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wireless media such as acoustic, RF, infrared and other wireless media. Embodiments are also directed to such communication media.

[00133] As noted above, computer programs and modules (including application programs 1932 and other program modules 1934) may be stored on the hard disk, magnetic disk, optical disk, ROM, or RAM. Such computer programs may also be received via network interface 1950, serial port interface 1942, or any other interface type. Such computer programs, when executed or loaded by an application, enable system 1900 to implement
20 features of embodiments described herein. Accordingly, such computer programs represent controllers of the system 1900. Embodiments are also directed to computer program products comprising software stored on any computer useable medium. Such software, when executed in one or more data processing devices, causes a data processing device(s) to operate as described herein. Embodiments may employ any computer-useable or
25 computer-readable medium, known now or in the future. Examples of computer-readable mediums include, but are not limited to memory devices and storage structures such as RAM, hard drives, floppy disks, CD ROMs, DVD ROMs, zip disks, tapes, magnetic storage devices, optical storage devices, MEMs, nanotechnology-based storage devices, and the like.

[00134] In alternative implementations, system 1900 may be implemented as hardware logic/electrical circuitry or firmware. In accordance with further embodiments, one or more of these components may be implemented in a system-on-chip (SoC). The SoC may include an integrated circuit chip that includes one or more of a processor (e.g., a microcontroller, microprocessor, digital signal processor (DSP), etc.), memory, one or more communication interfaces, and/or further circuits and/or embedded firmware to perform its functions.

V. Further Example Embodiments

[00135] A computer-implemented method is described herein. The method includes: accepting a selection of a smart contract template from one or more parties of a plurality of parties, the plurality of parties engaging in a transaction and choosing the selected smart contract template based at least in part on transaction terms negotiated between the plurality of parties; accepting from the plurality of parties transaction terms corresponding to fillable elements of the smart contract template; augmenting the smart contract template to include the accepted transaction terms to provide an augmented smart contract template; generating a smart contract corresponding to the augmented smart contract template; and deploying the smart contract to a blockchain.

[00136] In another implementation of the foregoing method, the smart contract template comprises smart contract programming code including meta-instructions configured to specify at least in part how the smart contract template should be augmented to include the accepted transaction terms.

[00137] In another implementation of the foregoing method, the smart contract programming code comprises code suitable for compilation to bytecode that is executable on a blockchain virtual machine.

[00138] In another implementation of the foregoing method, the meta-instructions are embedded as comments in the smart contract programming code.

[00139] In another implementation of the foregoing method, the meta-instructions are further configured to cause generation of an artifacts file that specifies a plurality of functions associated with the smart contract.

[00140] In another implementation of the foregoing method, the method further comprises calling at least one of the plurality of functions associated with the smart contract to determine whether at least one party of the plurality of parties should be enabled to perform a smart contract action.

[00141] In another implementation of the foregoing method, the method further comprises presenting a user interface representation to the at least one party of the plurality of parties,

the user interface representation enabling the at least one party of the plurality of parties to perform the smart contract action.

[00142] A smart contract generation system is described herein. The system includes: one or more processors and one or more memory devices connected thereto, the one or more
5 memory devices storing computer program logic for execution by the one or more processors, the computer program logic configured to: accept a selection of a smart contract template from one or more parties of a plurality of parties, the plurality of parties engaging in a transaction and choosing the selected smart contract template based at least in part on transaction terms negotiated between the plurality of parties; accept from the plurality of
10 parties transaction terms corresponding to fillable elements of the smart contract template; augment the smart contract template to include the accepted transaction terms to provide an augmented smart contract template; generate a smart contract corresponding to the augmented smart contract template; and deploy the smart contract to a blockchain.

[00143] In one implementation of the foregoing system, the smart contract template
15 comprises smart contract programming code including meta-instructions configured to specify at least in part how the smart contract template should be augmented to include the accepted transaction terms.

[00144] In one implementation of the foregoing system, the smart contract programming code comprises code suitable for compilation to bytecode that is executable on a blockchain
20 virtual machine.

[00145] In one implementation of the foregoing system, the meta-instructions are embedded as comments in the smart contract programming code.

[00146] In one implementation of the foregoing system, the meta-instructions are further configured to cause generation of an artifacts file that specifies a plurality of functions
25 associated with the smart contract.

[00147] In one implementation of the foregoing system, the computer program logic is further configured to call at least one of the plurality of functions associated with the smart contract to determine whether at least one party of the plurality of parties should be enabled to perform a smart contract action.

[00148] In one implementation of the foregoing system, the computer program logic is further configured to present a user interface representation to the at least one party of the plurality of parties, the user interface representation enabling the at least one party of the plurality of parties to perform the smart contract action.

[00149] A computer-readable storage medium having program instructions recorded

thereon that, when executed by at least one processor, perform a method is disclosed herein. The method includes: accepting a selection of a smart contract template from one or more parties of a plurality of parties, the plurality of parties engaging in a transaction and choosing the selected smart contract template based at least in part on transaction terms negotiated
5 between the plurality of parties; accepting from the plurality of parties transaction terms corresponding to fillable elements of the smart contract template; augmenting the smart contract template to include the accepted transaction terms to provide an augmented smart contract template; generating a smart contract corresponding to the augmented smart contract template; and deploying the smart contract to a blockchain.

10 **[00150]** In another implementation of the foregoing computer-readable storage medium, the smart contract template comprises smart contract programming code including meta-instructions configured to specify at least in part how the smart contract template should be augmented to include the accepted transaction terms.

[00151] In another implementation of the foregoing computer-readable storage medium the
15 meta-instructions are embedded as comments in the smart contract programming code.

[00152] In another implementation of the foregoing computer-readable storage medium, the meta-instructions are further configured to cause generation of an artifacts file that specifies a plurality of functions associated with the smart contract.

[00153] In another implementation of the foregoing computer-readable storage medium,
20 the method further comprises calling at least one of the plurality of functions associated with the smart contract to determine whether at least one party of the plurality of parties should be enabled to perform a smart contract action.

[00154] In another implementation of the foregoing computer-readable storage medium, the method further comprises presenting a user interface representation to the at least one
25 party of the plurality of parties, the user interface representation enabling the at least one party of the plurality of parties to perform the smart contract action.

VI. Conclusion

[00155] While various example embodiments have been described above, it should be understood that they have been presented by way of example only, and not limitation. It will
30 be understood by those skilled in the relevant art(s) that various changes in form and details may be made therein without departing from the spirit and scope of the embodiments as defined in the appended claims. Accordingly, the breadth and scope of the disclosure should not be limited by any of the above-described example embodiments, but should be defined only in accordance with the following claims and their equivalents.

CLAIMS

1. A computer-implemented method, comprising:
 - accepting a selection of a smart contract template from one or more parties of a plurality of parties, the plurality of parties engaging in a transaction and choosing the selected smart contract template based at least in part on transaction terms negotiated between the plurality of parties;
 - accepting from the plurality of parties transaction terms corresponding to fillable elements of the smart contract template;
 - augmenting the smart contract template to include the accepted transaction terms to provide an augmented smart contract template;
 - generating a smart contract corresponding to the augmented smart contract template; and
 - deploying the smart contract to a blockchain.
2. The computer-implemented method of claim 1, wherein the smart contract template comprises smart contract programming code including meta-instructions configured to specify at least in part how the smart contract template should be augmented to include the accepted transaction terms.
3. The computer-implemented method of claim 2, wherein the smart contract programming code comprises code suitable for compilation to bytecode that is executable on a blockchain virtual machine.
4. The computer-implemented method of claim 2, wherein the meta-instructions are embedded as comments in the smart contract programming code.
5. The computer-implemented method of claim 2, wherein the meta-instructions are further configured to cause generation of an artifacts file that specifies a plurality of functions associated with the smart contract.
6. The computer-implemented method of claim 5, further comprising calling at least one of the plurality of functions associated with the smart contract to determine whether at least one party of the plurality of parties should be enabled to perform a smart contract action.
7. The computer-implemented method of claim 6, further comprising presenting a user interface representation to the at least one party of the plurality of parties, the user interface representation enabling the at least one party of the plurality of parties to perform the smart contract action.
8. A smart contract generation system, comprising:

one or more processors and one or more memory devices connected thereto, the one or more memory devices storing computer program logic for execution by the one or more processors, the computer program logic configured to:

accept a selection of a smart contract template from one or more parties of a plurality of parties, the plurality of parties engaging in a transaction and choosing the selected smart contract template based at least in part on transaction terms negotiated between the plurality of parties;

accept from the plurality of parties transaction terms corresponding to fillable elements of the smart contract template;

augment the smart contract template to include the accepted transaction terms to provide an augmented smart contract template;

generate a smart contract corresponding to the augmented smart contract template; and

deploy the smart contract to a blockchain.

9. The smart contract generation system of claim 8, wherein the smart contract template comprises smart contract programming code including meta-instructions configured to specify at least in part how the smart contract template should be augmented to include the accepted transaction terms.

10. The smart contract generation system of claim 9, wherein the smart contract programming code comprises code suitable for compilation to bytecode that is executable on a blockchain virtual machine.

11. The smart contract generation system of claim 9, wherein the meta-instructions are embedded as comments in the smart contract programming code.

12. The smart contract generation system of claim 9, wherein the meta-instructions are further configured to cause generation of an artifacts file that specifies a plurality of functions associated with the smart contract.

13. The smart contract generation system of claim 12 wherein the computer program logic is further configured to call at least one of the plurality of functions associated with the smart contract to determine whether at least one party of the plurality of parties should be enabled to perform a smart contract action.

14. The smart contract generation system of claim 13 wherein the computer program logic is further configured to present a user interface representation to the at least one party of the plurality of parties, the user interface representation enabling the at least one party of the plurality of parties to perform the smart contract action.

15. A computer program product comprising a computer-readable memory device having computer program logic recorded thereon, comprising:
- computer program logic for enabling a processor to perform any of the steps of claims 1-7.

1/20

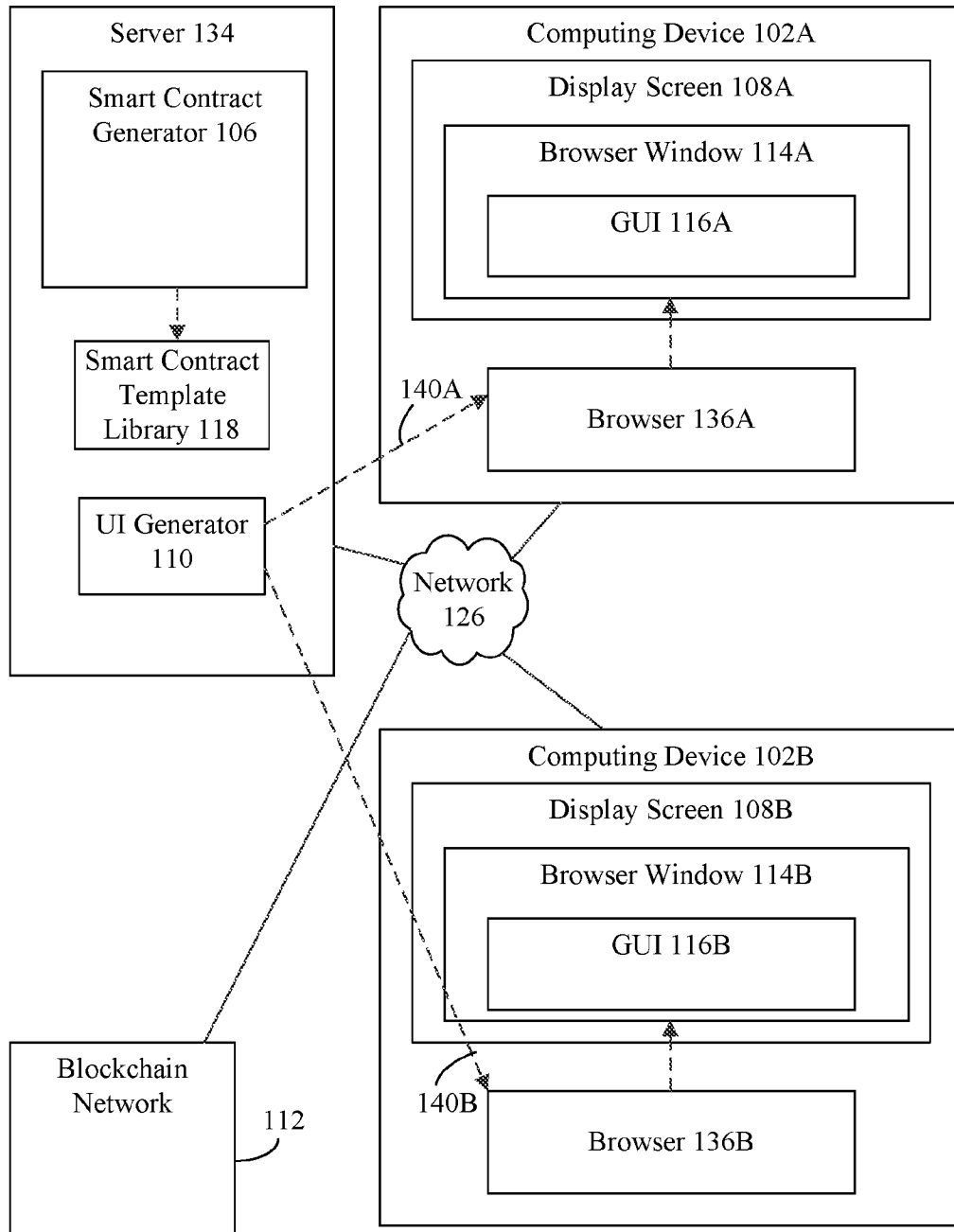
100

FIG. 1

2/20

```

202 {
204   "title": "Basic Escrow",
206   "description": "Basic escrow contract with mediator dispute resolution",
208   "doc": "We agree that provider {provider.displayName} will perform {jobName} for
        consumer {consumer.displayName} for the following amount: [...]",
    "params": {
        "jobName": {
            "type": "string",
            "title": "The work title"
        },
        "jobDescription": {
            "type": "multiline",
            "title": "The work descriptoin"
        },
        "consumer": {
            "type": "user",
            "title": "Consumer"
        },
        "provider": {
            "type": "user",
            "title": "Provider"
        },
        "mediator": {
            "type": "user",
            "title": "Mediator"
        },
        "unit": {
            "type": "currency",
            "title": "The name of the money unit (e.g., ether, finne, etc.)"
        },
        "reward": {
            "type": "number",
            "title": "The reward paid by consumer to provider"
        },
        "refund": {
            "type": "number",
            "title": "The refund that returns to consumer upon contract completion"
        },
        "deposit": {
            "type": "number",
            "title": "The deposit that returns to provider upon contract completion"
        },
        "mediationFeeRate": {
            "type": "number",
            "title": "The fee for mediation paid to the mediator (if needed)"
        }
    }
  }
}

```

200

FIG. 2

3/20

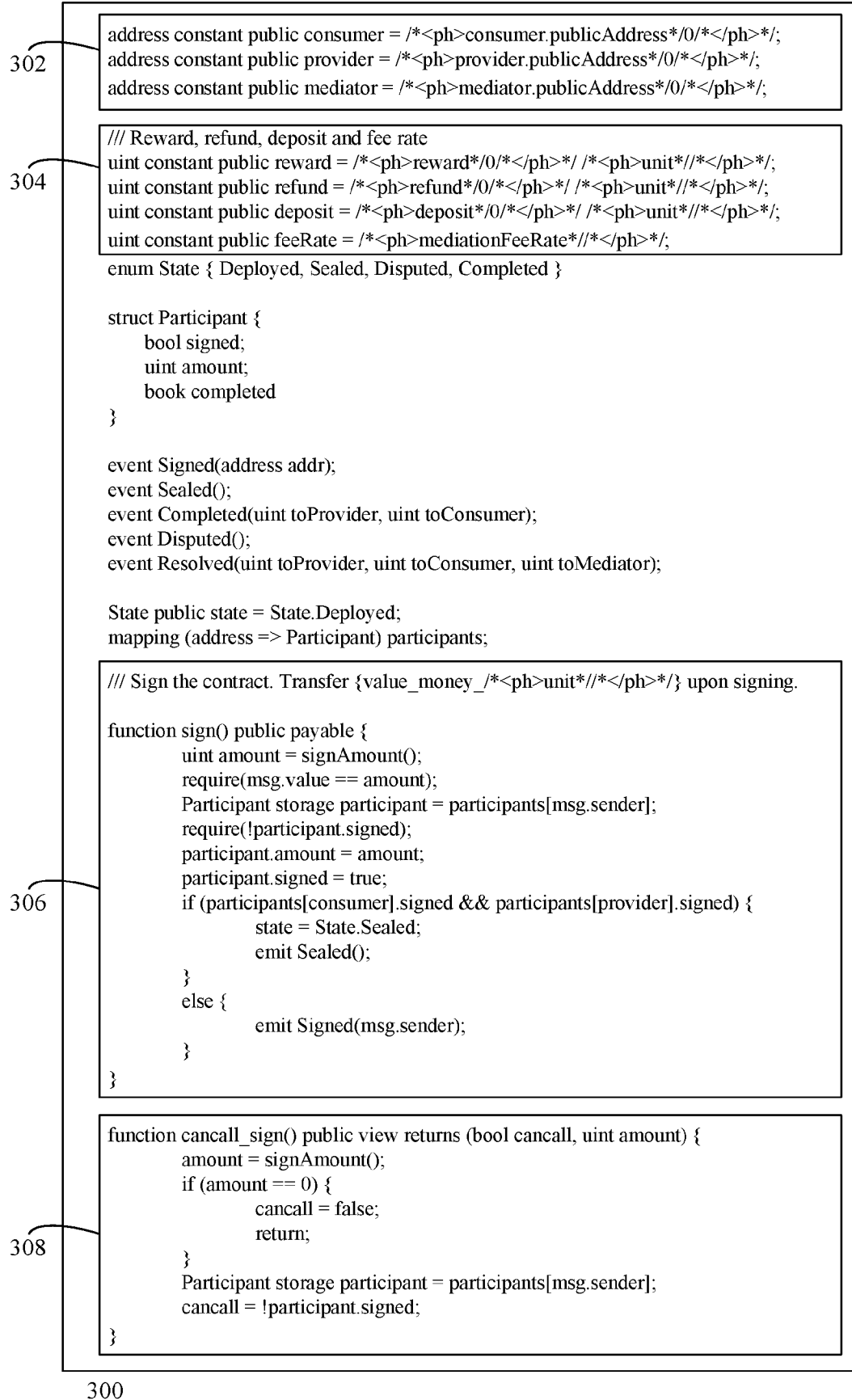


FIG. 3

400

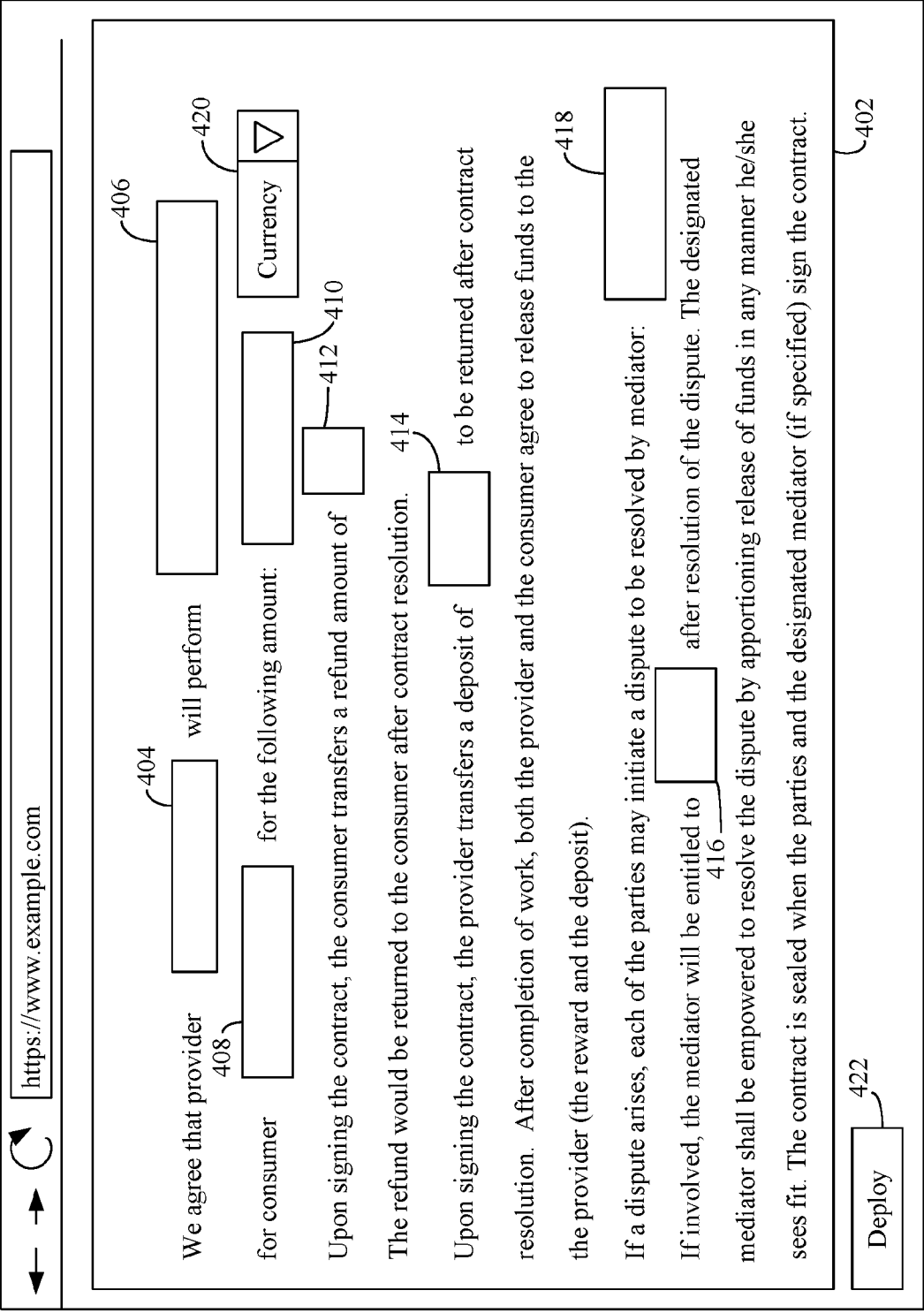


FIG. 4

5/20

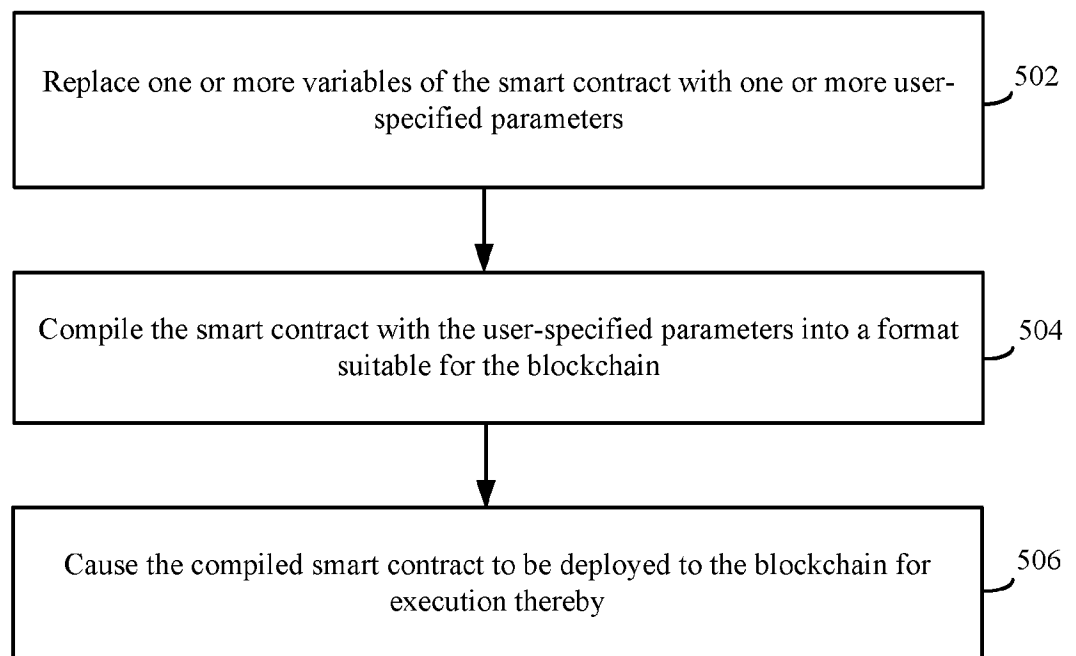


FIG. 5

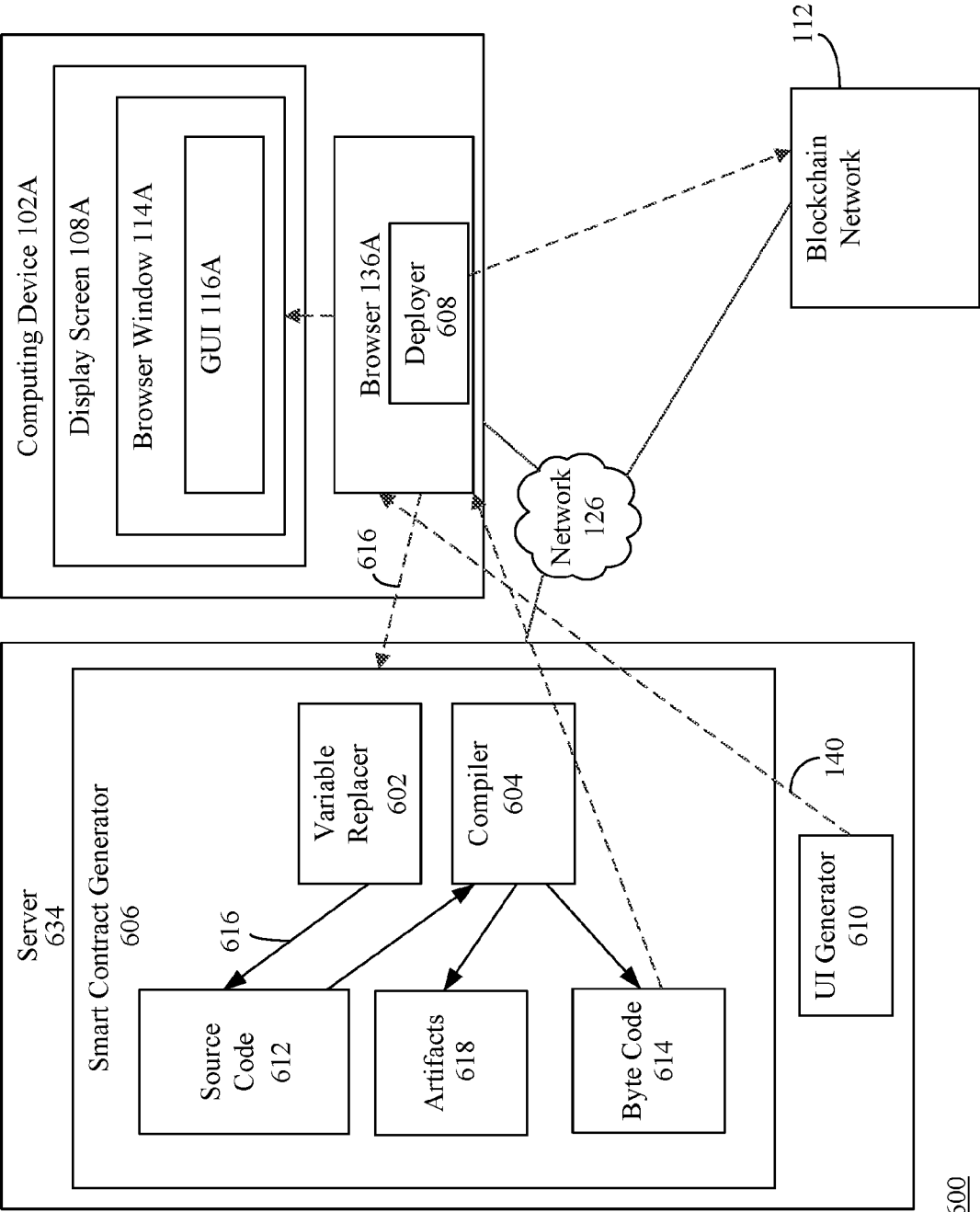


FIG. 6

600

7/20

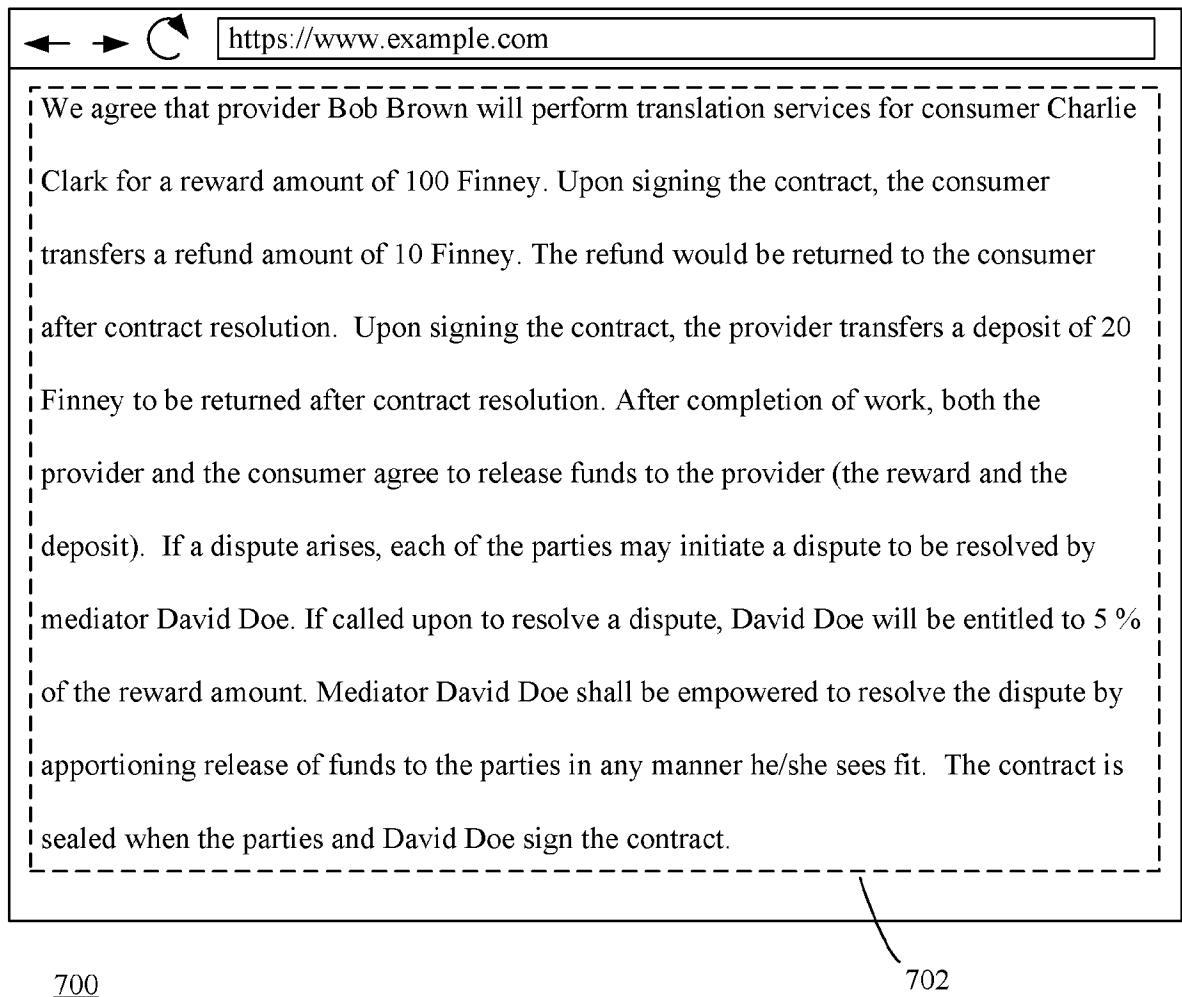
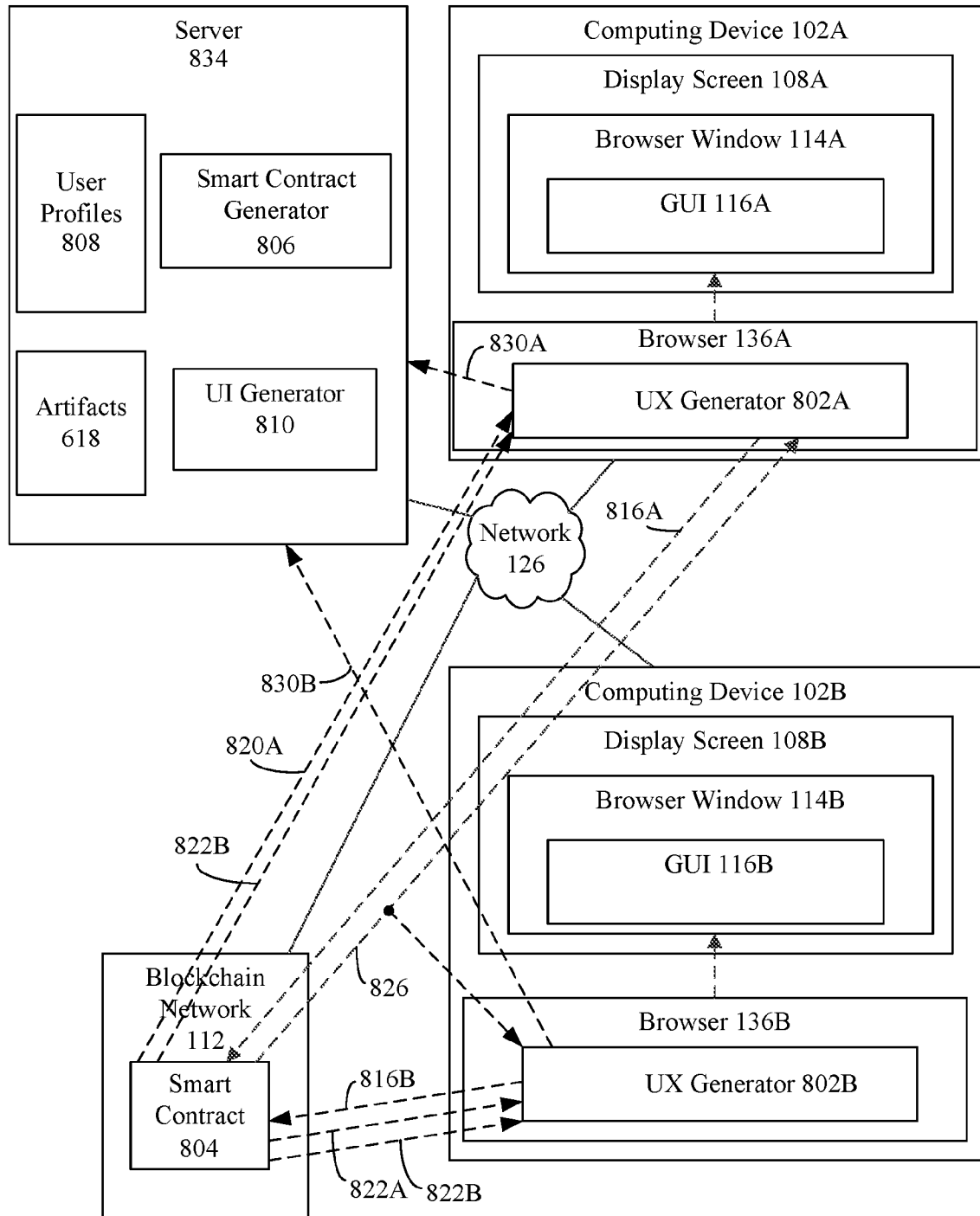


FIG. 7

8/20



800

FIG. 8

9/20

← → ↻ <https://www.example.com>

We agree that provider Bob Brown will perform translation services for consumer Charlie Clark for a reward amount of 100 Finney. Upon signing the contract, the consumer transfers a refund amount of 10 Finney. The refund would be returned to the consumer after contract resolution. Upon signing the contract, the provider transfers a deposit of 20 Finney to be returned after contract resolution. After completion of work, both the provider and the consumer agree to release funds to the provider (the reward and the deposit). If a dispute arises, each of the parties may initiate a dispute to be resolved by mediator David Doe. If called upon to resolve a dispute, David Doe will be entitled to 5 % of the reward amount. Mediator David Doe shall be empowered to resolve the dispute by apportioning release of funds to the parties in any manner he/she sees fit. The contract is sealed when the parties and David Doe sign the contract.

902

Provider (Bob Brown) deposited 0 Finney 908

Consumer (Charlie Clark) deposited 0 Finney 906

The state of the contract is "Pending". 910

Sign the contract. Transfer 20 Finney upon singing. 904

912

900A

FIG. 9A

10/20

← → ↻ <https://www.example.com>

We agree that provider Bob Brown will perform translation services for consumer Charlie Clark for a reward amount of 100 Finney. Upon signing the contract, the consumer transfers a refund amount of 10 Finney. The refund would be returned to the consumer after contract resolution. Upon signing the contract, the provider transfers a deposit of 20 Finney to be returned after contract resolution. After completion of work, both the provider and the consumer agree to release funds to the provider (the reward and the deposit). If a dispute arises, each of the parties may initiate a dispute to be resolved by mediator David Doe. If called upon to resolve a dispute, David Doe will be entitled to 5 % of the reward amount. Mediator David Doe shall be empowered to resolve the dispute by apportioning release of funds to the parties in any manner he/she sees fit. The contract is sealed when the parties and David Doe sign the contract.

Provider (Bob Brown) deposited 0 Finney ⁹⁰⁸

Consumer (Charlie Clark) deposited 0 Finney ⁹⁰⁶

The state of the contract is "Pending". ⁹¹⁰

Sign the contract. Transfer 110 Finney upon signing. ^{904'} ⁹¹²

900B

FIG. 9B

11/20

← → ↺
https://www.example.com

We agree that provider Bob Brown will perform translation services for consumer Charlie Clark for a reward amount of 100 Finney. Upon signing the contract, the consumer transfers a refund amount of 10 Finney. The refund would be returned to the consumer after contract resolution. Upon signing the contract, the provider transfers a deposit of 20 Finney to be returned after contract resolution. After completion of work, both the provider and the consumer agree to release funds to the provider (the reward and the deposit). If a dispute arises, each of the parties may initiate a dispute to be resolved by mediator David Doe. If called upon to resolve a dispute, David Doe will be entitled to 5 % of the reward amount. Mediator David Doe shall be empowered to resolve the dispute by apportioning release of funds to the parties in any manner he/she sees fit. The contract is sealed when the parties and David Doe sign the contract.

902

Provider (Bob Brown) deposited 20 Finney

908'

906

Consumer (Charlie Clark) deposited 0 Finney

906

910

The state of the contract is "Pending".

910

Revert signing and receive the submitted funds back

1004

1002

Revert

1000

FIG. 10

12/20

← → ↻ <https://www.example.com>

We agree that provider Bob Brown will perform translation services for consumer Charlie Clark for a reward amount of 100 Finney. Upon signing the contract, the consumer transfers a refund amount of 10 Finney. The refund would be returned to the consumer after contract resolution. Upon signing the contract, the provider transfers a deposit of 20 Finney to be returned after contract resolution. After completion of work, both the provider and the consumer agree to release funds to the provider (the reward and the deposit). If a dispute arises, each of the parties may initiate a dispute to be resolved by mediator David Doe. If called upon to resolve a dispute, David Doe will be entitled to 5 % of the reward amount. Mediator David Doe shall be empowered to resolve the dispute by apportioning release of funds to the parties in any manner he/she sees fit. The contract is sealed when the parties and David Doe sign the contract.

902

Provider (Bob Brown) deposited 20 Finney 908'

Consumer (Charlie Clark) deposited 0 Finney 906

The state of the contract is "Pending". 910

Sign the contract. Transfer 110 Finney upon signing. 904' 912

Sign Contract

1100

FIG. 11

13/20

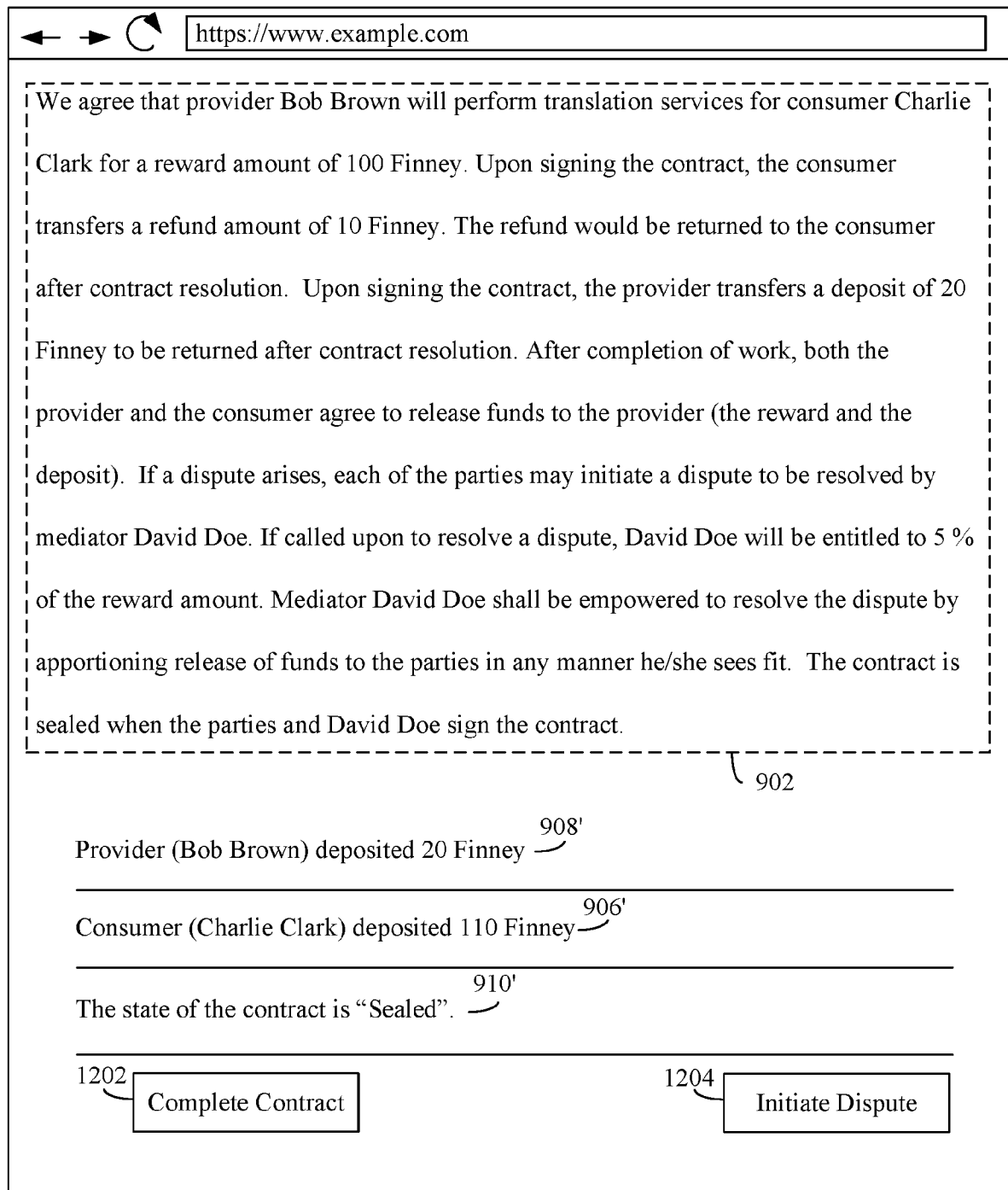


FIG. 12

1200

14/20

← → ↻ <https://www.example.com>

We agree that provider Bob Brown will perform translation services for consumer Charlie Clark for a reward amount of 100 Finney. Upon signing the contract, the consumer transfers a refund amount of 10 Finney. The refund would be returned to the consumer after contract resolution. Upon signing the contract, the provider transfers a deposit of 20 Finney to be returned after contract resolution. After completion of work, both the provider and the consumer agree to release funds to the provider (the reward and the deposit). If a dispute arises, each of the parties may initiate a dispute to be resolved by mediator David Doe. If called upon to resolve a dispute, David Doe will be entitled to 5 % of the reward amount. Mediator David Doe shall be empowered to resolve the dispute by apportioning release of funds to the parties in any manner he/she sees fit. The contract is sealed when the parties and David Doe sign the contract.

902

Provider (Bob Brown) deposited 20 Finney 908'

Consumer (Charlie Clark) deposited 110 Finney 906'

The state of the contract is "Disputed". 910"

1302

Resolve dispute by releasing ☐ 100% of the reward amount (less mediation fee) to the provider. The remainder returns to the consumer.

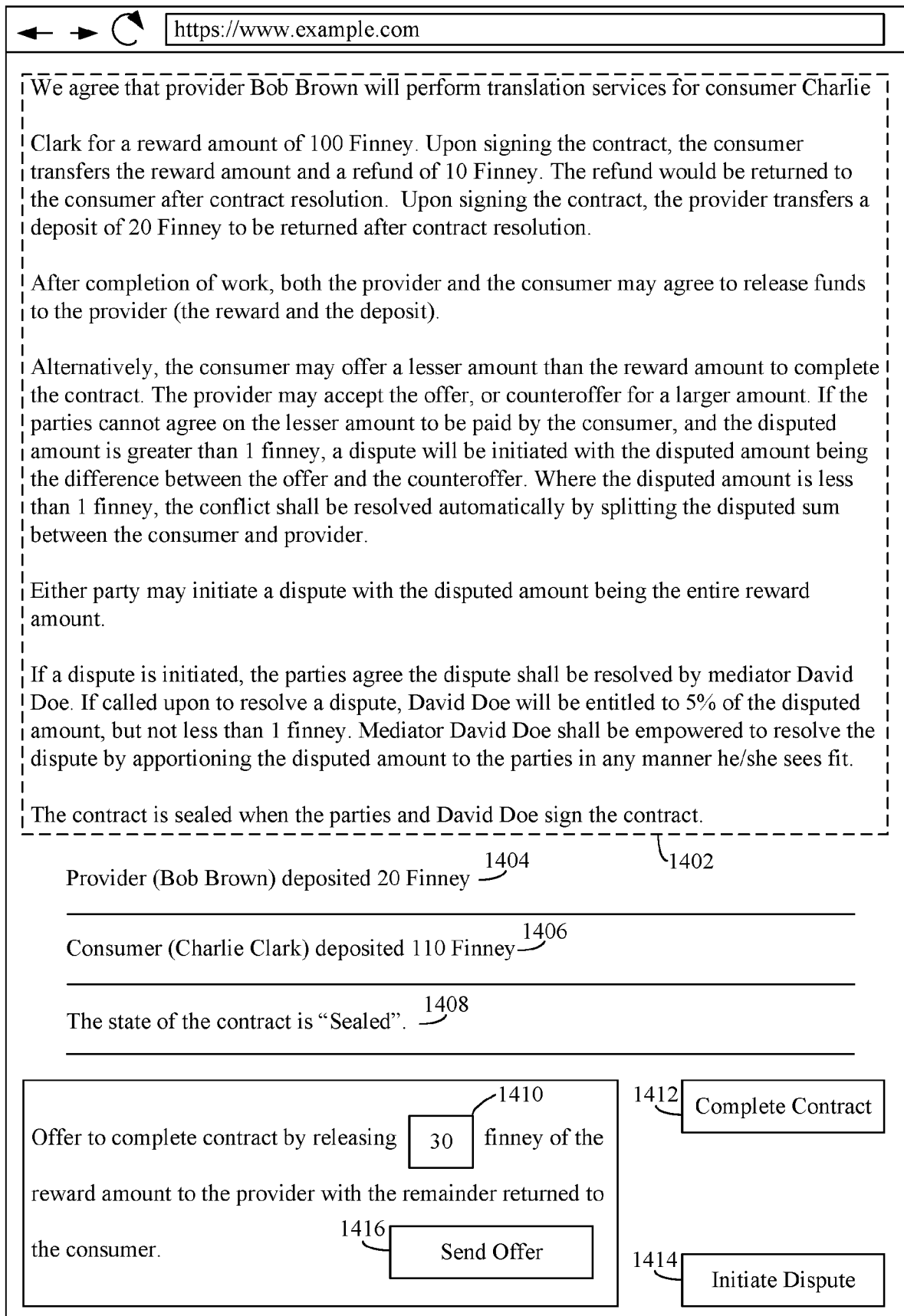
1304

Resolve Dispute

1300

FIG. 13

15/20



1400

FIG. 14

16/20

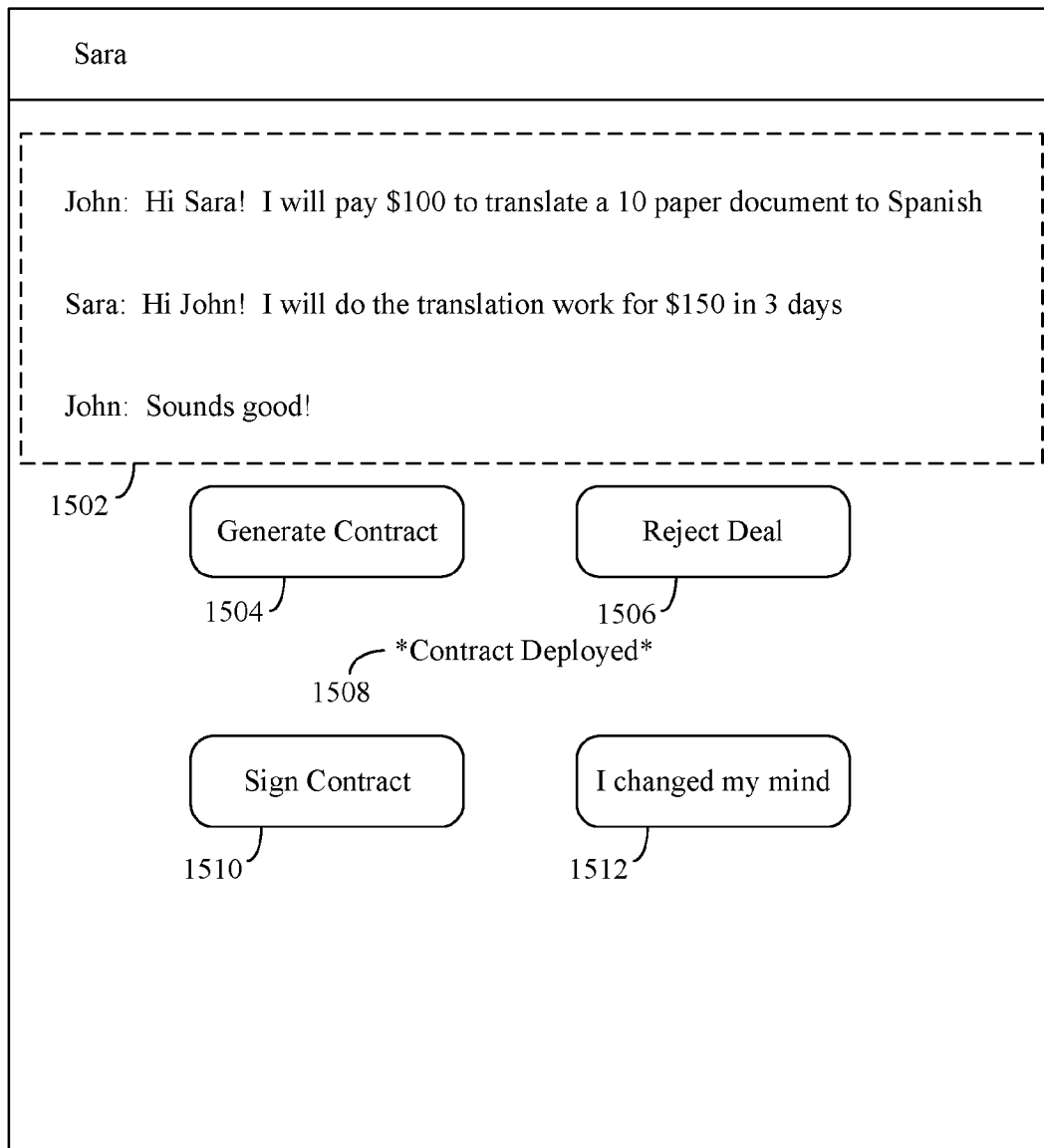
1500

FIG. 15

17/20

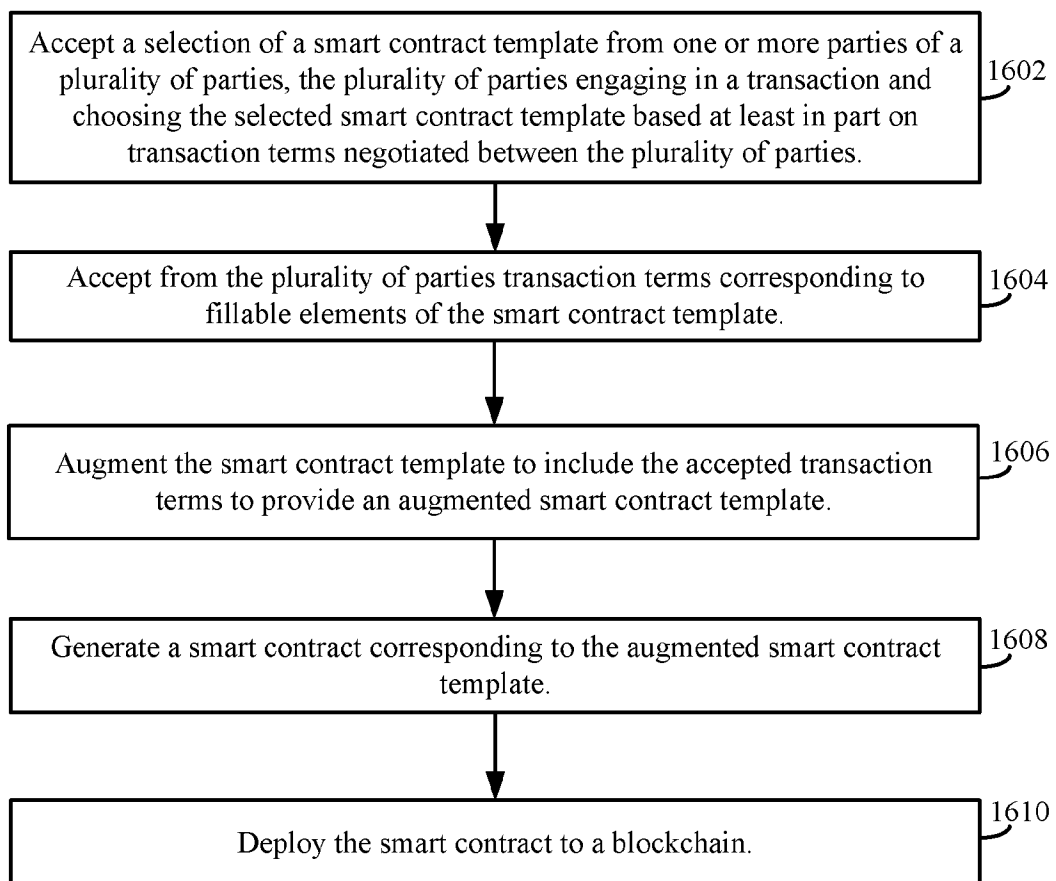
1600

FIG. 16

18/20

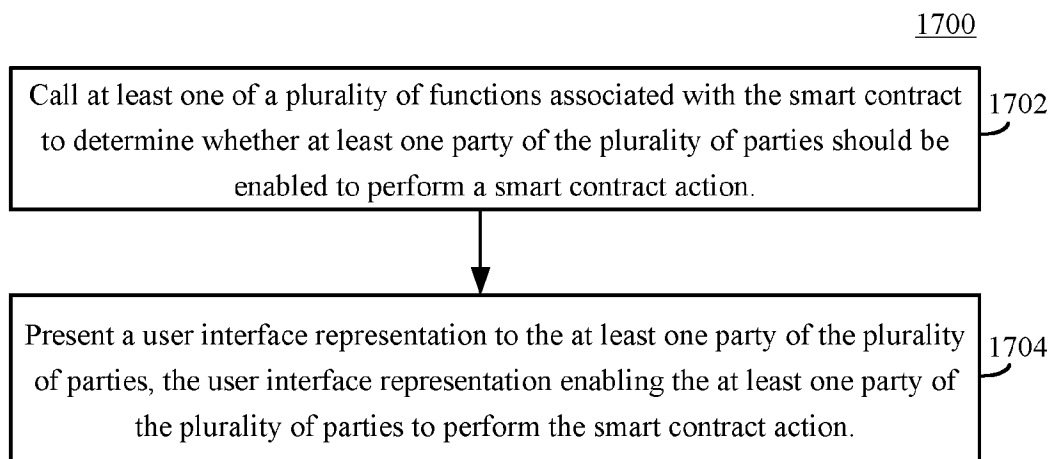


FIG. 17

19/20

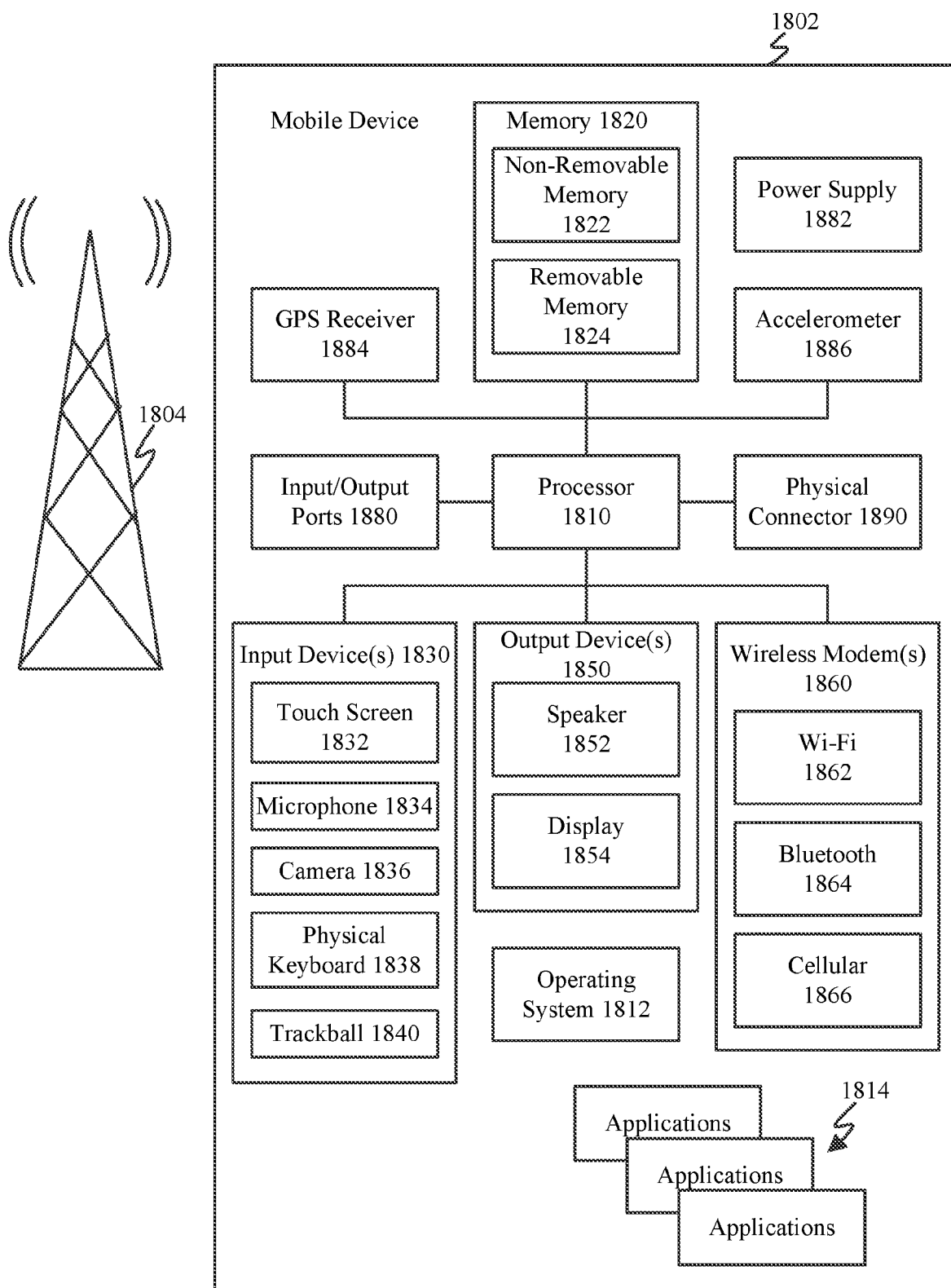


FIG. 18

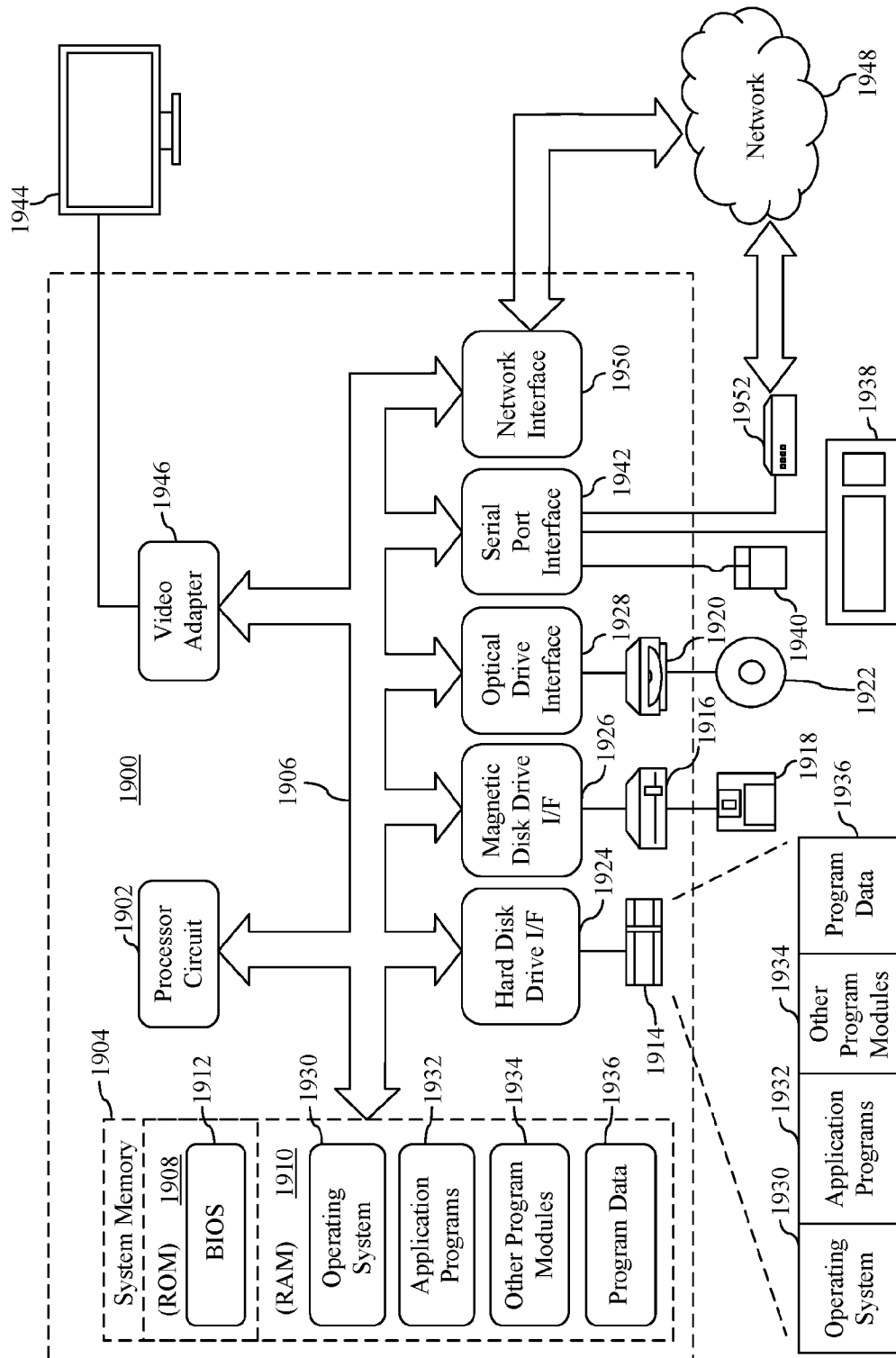


FIG. 19

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2020/026036

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06Q20/10 H04L29/06
ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06Q G07G H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2019/034402 A1 (ANDERSON RYAN [US] ET AL) 31 January 2019 (2019-01-31) figures 1-4 paragraph [0017] paragraph [0018] paragraph [0019] paragraph [0020] paragraph [0021] paragraph [0025] paragraph [0026] claims 1-21 ----- -/-	1-15



Further documents are listed in the continuation of Box C.



See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search

5 May 2020

Date of mailing of the international search report

19/05/2020

Name and mailing address of the ISA/

European Patent Office, P.B. 5818 Patentlaan 2
NL - 2280 HV Rijswijk
Tel. (+31-70) 340-2040,
Fax: (+31-70) 340-3016

Authorized officer

Mennerun, Steeve

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2020/026036

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2019/108140 A1 (DESHPANDE PRALHAD D [SG] ET AL) 11 April 2019 (2019-04-11) paragraph [0024] paragraph [0025] paragraph [0031] paragraph [0035] paragraph [0036] figures 1-8 -----	1-15
A	US 2017/232300 A1 (TRAN BAO [US] ET AL) 17 August 2017 (2017-08-17) paragraph [0123] paragraph [0124] figure 13a -----	1-15

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No

PCT/US2020/026036

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2019034402 A1	31-01-2019	US 2019034402 A1	31-01-2019
		US 2019034404 A1	31-01-2019
US 2019108140 A1	11-04-2019	NONE	
US 2017232300 A1	17-08-2017	NONE	